Владимир Дронов

bhv

Macromedia Flash Professional 8

графика и анимация

Наиболее полное руководство

В ПОДЛИННИКЕ®

Владимир Дронов

Macromedia Flash Professional 8
графика и анимация

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06
ББК 32.973.26-018.2
Д75

Дронов В. А.

Д75 Macromedia Flash Professional 8. Графика и анимация. — СПб.: БХВ-Петербург, 2006. — 656 с.: ил.

ISBN 5-94157-907-1

Подробно, доступно и с юмором рассматривается создание современной интернет-графики и анимации с помощью пакета Macromedia Flash Professional 8. Описываются основные инструменты рисования и правки графики, использование слоев, образцов и библиотек, работа с текстом, импорт графики, видео и звука. Рассмотрен процесс производства фильмов стандартными, и весьма богатыми, средствами Flash. Также приводятся краткое описание встроенного языка программирования Flash — ActionScript — и методика его применения для разработки интерактивных элементов. Рассказывается о публикации готового изображения или фильма для распространения через Интернет.

Для Web-дизайнеров

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Игорь Цырульников*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн серии	*Игоря Цырульникова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.04.06.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 52,89.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-907-1

© Дронов В. А., 2006
© Оформление, издательство "БХВ-Петербург", 2006

Оглавление

Введение **1**

Flash 8 — что нового? 2
Flash 8 и Flash 8 Professional 3
Типографские соглашения 4
Благодарности 5

Часть I. Введение во Flash **7**

Глава 1. Основы компьютерной графики **9**

Компьютерная графика: прошлое и настоящее 10
Статичная графика 12
Два вида статичной графики 13
Растровая графика 13
Векторная графика 17
Применение растровой и векторной графики 20
Гибридная графика 20
Форматы графики 21
Формат BMP 21
Формат GIF 21
Формат PNG 22
Формат JPEG 23
Анимация и видео 24
Два вида компьютерной анимации 24
Покадровая анимация 25
Трансформационная анимация 28
Применение разных видов анимации 30
Форматы видео 31
Формат QuickTime 31
Формат AVI 31
Формат MPEG 32
Формат WMV 32
Формат RealMedia 33
Формат "анимированный GIF" 33
Краткое введение во Flash 34
Недостатки языка HTML 34
Достоинства Flash 35

Универсальность ... 35
Компактность ... 37
Безопасность ... 37
Формат Shockwave/Flash ... 37
Проигрыватель Flash ... 38
Формат документов Flash ... 39
Конкуренты Flash ... 40
Что делают с помощью Flash ... 42
Что дальше? ... 43

Глава 2. Пользовательский интерфейс Flash ... 44

Рабочая среда Flash ... 44
Главное окно ... 45
Управление окнами и панелями Flash ... 52
Работа с документами в среде Flash ... 55
Создание нового документа ... 55
Создание нового пустого документа ... 55
Создание нового документа на основе шаблона ... 56
Параметры документа ... 58
Окно документа Flash и работа с ним ... 60
Окно документа ... 61
Основные принципы работы с графикой Flash ... 62
Управление окном документа ... 63
Средства позиционирования ... 66
Работа с файлами ... 71
Интерактивная справка Flash ... 72
Что дальше? ... 75

Часть II. Работа со статичной графикой ... 77

Глава 3. Простейшие операции рисования ... 79

Базовые средства управления цветом ... 80
Инструменты рисования ... 81
Инструмент *Line* (*"Линия"*) ... 81
Инструменты *Rectangle* (*"Прямоугольник"*) и *Oval* (*"Эллипс"*) ... 82
Инструмент *PolyStar* (*"Многоугольник"*) ... 84
Инструмент *Pencil* (*"Карандаш"*) ... 86
Инструмент *Pen* (*"Перо"*) ... 88
Инструмент *Paint Bucket* (*"Ведро с краской"*) ... 91
Инструмент *Brush* (*"Кисть"*) ... 92
Правка графики ... 97
Выделение графики ... 97
Фрагментация и слияние графики ... 100

Группировка фрагментов....103
Объектное рисование....104
Правка графики....104
Перемещение и удаление графики....105
"Приклеивание" графики....105
Работа с буфером обмена....108
Изменение цвета графики....110
Правка контура фигуры....110
Точная правка контуров....112
Сложное выделение. Инструмент *Lasso* (*"Лассо"*)....114
Частичное стирание графики. Инструмент *Eraser* (*"Ластик"*)....117
Дополнительные возможности работы с контурами....122
Отмена и повтор выполненных действий....128
Что дальше?....128

Глава 4. Трансформация графики....129

Простейшие трансформации....129
Выравнивание и распределение....129
Перемещение и изменение размеров....133
Зеркальное отражение....135
Управление порядком перекрытия....135
Объединение фрагментов изображения....136
Более сложные трансформации....139
Изменение размеров....139
Вращение и сдвиг....141
Искажение формы....142
Деформация....143
Свободная трансформация....144
Дополнительные возможности....145
Трансформация копии графического фрагмента....145
Сброс всех трансформаций, примененных к фрагменту....145
Блокировка фрагмента изображения....146
Что дальше?....147

Глава 5. Работа со слоями....148

Создание слоев и управление ими....148
Использование папок для организации слоев....153
Распределение графики по слоям....155
Что дальше?....155

Глава 6. Работа с цветом и стилем линий....156

Инструменты выбора цвета....156
Использование селектора цвета....157
Область *Colors* главного инструментария....159
Создание новых цветов. Смеситель....159

Задание цвета линий и заливок ... 162
Задание сплошного цвета или отсутствия цвета ... 163
Задание сложных цветов ... 163
Градиентные цвета ... 164
Графические цвета ... 167
Работа со сложными цветами ... 168
Трансформация сложных цветов ... 169
Фиксированные цвета ... 173
Работа с линиями ... 174
Задание стиля линий ... 174
Дополнительные параметры стиля линий ... 178
Инструмент *Ink Bottle* (*"Чернильница"*) ... 179
Инструмент *Eyedropper* (*"Пипетка"*) ... 179
Работа с палитрами ... 180
Что дальше? ... 182

Глава 7. Работа с текстом ... 183

Текстовые блоки и работа с ними ... 183
Работа с текстовыми блоками ... 184
Форматирование текста ... 187
Форматирование отдельных символов ... 187
Форматирование абзацев ... 189
Поддержка шрифтов во Flash ... 191
Параметры текстового блока ... 193
Трансформация текстовых блоков ... 195
Создание гиперссылок ... 196
Специальные текстовые блоки ... 197
Поля ввода ... 197
Динамические текстовые блоки ... 202
Подстановка шрифтов ... 203
Что дальше? ... 206

Глава 8. Образцы, экземпляры и библиотеки ... 207

Типы образцов Flash ... 208
Создание образцов ... 209
Создание экземпляров ... 212
Работа с экземплярами ... 213
Задание цветового эффекта ... 213
Трансформация экземпляров ... 216
Сброс всех трансформаций, примененных к экземпляру ... 216
Смена типа экземпляра ... 216
Смена породившего экземпляр образца ... 217
Разбиение экземпляра (преобразование в обычную графику) ... 218

Работа с образцами 218
Правка образцов 218
Дублирование образцов 221
Панель *Library* 222
Управление образцами 224
Использование папок 225
Совместное использование образцов и библиотек 228
Копирование образцов из документа в документ 229
Обновляемые образцы 231
Разделяемые образцы 233
Создание разделяемых образцов 234
Использование разделяемых образцов 236
Разделяемые образцы-шрифты 238
Превращение разделяемого образца в обычный 240
Библиотеки общего использования 241
Что дальше? 241

Глава 9. Использование фильтров и наложений 242

Фильтры 242
Создание фильтров 243
Фильтры, поставляемые в составе Flash 244
Отображение тени (*Drop Shadow*) 244
Размытие (*Blur*) 246
Эффект сияния (*Glow*) 247
Эффект вдавливания (*Bevel*) 249
Эффект градиентного сияния (*Gradient Glow*) 250
Эффект градиентного вдавливания (*Gradient Bevel*) 252
Настройка цвета (*Adjust Color*) 253
Управление фильтрами 254
Наложения 256
Что дальше? 258

Глава 10. Импорт графики 259

Какие форматы графики поддерживает Flash 260
Импорт графики 261
Особенности импорта разных графических форматов 262
Macromedia Fireworks 262
Macromedia Freehand 263
Adobe Illustrator и Adobe Acrobat 266
AutoDesk AutoCAD 267
Работа с импортированной растровой графикой 268
Особенности работы с импортированными образцами 268
Векторизация растровой графики 270

Разбиение растровой графики. Модификатор *Magic Wand* (*"Волшебная палочка"*) 271
Немного об авторских правах 273
Что дальше? 274

Часть III. Анимация и звук 275

Глава 11. Покадровая анимация 277

Создание покадровой анимации 278
Временная шкала 279
Создание кадров анимации 281
Использование растянутых кадров 282
Просмотр созданной анимации 283
Правка анимации 285
Работа с кадрами 285
Дополнительные возможности работы с кадрами 288
Дополнительные возможности временной шкалы 291
Использование сцен 294
Что дальше? 297

Глава 12. Трансформационная анимация 298

Два вида трансформационной анимации 299
Движение 300
Создание простейшего движения 300
Более сложное движение 302
Параметры движения 303
Дополнительные средства управления движением 306
Еще раз об "оборванной" анимации 309
Морфинг 310
Создание морфинга 310
Параметры морфинга 312
Маркеры морфинга и их использование 312
Что дальше? 316

Глава 13. Создание составной анимации 317

Создание составной анимации 318
Использование образцов-клипов 320
Параметры внутренней анимации 321
Что дальше? 323

Глава 14. Использование слоев для создания анимации 324

Многослойная анимация 325
Создание многослойной анимации 325

Создание фоновых изображений....325
Ускорение отображения фрагментов фона....326
Использование специальных слоев....327
Слои-направляющие....327
Маскирующие слои....330
Что дальше?....332

Глава 15. Использование эффектов и преобразований....333

Простейшие эффекты....334
Клонирование по сетке ("Copy to Grid")....334
Анимированное копирование ("Distributed Duplicate")....335
Более сложные эффекты....337
Анимированное размытие ("Blur")....337
Отображение тени ("Drop Shadow")....338
Расширение ("Expand")....339
"Взрыв" ("Explode")....341
Преобразования....342
Обычное движение ("Transform")....342
Проявление ("Transition")....344
Работа с эффектами и преобразованиями....345
Что дальше?....346

Глава 16. Импорт видео....347

Сжатие видео во Flash. Кодеки On2 VP6 и Sorenson Spark....348
Параметры цифрового видео....349
Полезные советы по подготовке видео для импорта....350
Какие форматы видео поддерживает Flash....351
Импорт видео....352
Использование импортированных клипов....360
Работа с образцами-импортированными клипами....362
Правка импортируемого клипа средствами Flash....364
Управление качеством импортируемого клипа....367
Macromedia Flash Video....370
Что дальше?....371

Глава 17. Работа со звуком....372

Введение в цифровой звук....373
Три способа кодирования звука....373
Прямая оцифровка....373
Командное кодирование....374
Командно-сэмплерное кодирование....375
Параметры цифрового звука....376
Полезные советы по подготовке звука для импорта....377

Какие форматы звука поддерживает Flash ... 378
Формат WAV ... 378
Формат MP3 ... 379
Импорт звука ... 379
Использование импортированного звука ... 380
Параметры звука ... 383
Работа с образцами-звуками ... 385
Что дальше? ... 387

Часть IV. Интерактивность ... 389

Глава 18. Основы программирования во Flash ... 391

Как работают сценарии. События ... 392
Привязка сценариев к кадрам анимации ... 393
Знакомство с панелью *Actions* ... 393
Написание сценария ... 395
Воспроизведение фильмов, кадры которых содержат сценарии ... 397
Привязка сценариев к клипам ... 398
Привязка сценариев к кнопкам ... 400
Создание кнопок ... 401
Воспроизведение фильмов, содержащих кнопки ... 402
Добавление звуков к кнопкам ... 403
Написание сценариев, привязанных к кнопкам ... 404
Работа с панелью *Actions* ... 404
Поведения Flash и их использование ... 408
Что дальше? ... 411

Глава 19. Язык ActionScript ... 412

Начала языка ActionScript ... 412
Основные понятия ActionScript ... 412
Данные ActionScript. Типы данных ... 414
Константы ... 416
Переменные ... 417
Именование переменных ... 417
Объявление переменных ... 418
Область видимости переменных ... 418
Операторы ... 420
Арифметические операторы ... 420
Оператор объединения строк ... 421
Операторы присваивания ... 421
Операторы сравнения ... 422
Логические операторы ... 423
Оператор *typeof* ... 424

Совместимость и преобразование типов данных....424
Приоритет операторов....426
Комментарии....428
Сложные выражения ActionScript....428
Блоки....428
Условные выражения....429
Выражения выбора....430
Циклы....432
Цикл со счетчиком....432
Цикл с постусловием....433
Цикл с предусловием....434
Прерывание и перезапуск цикла....435
Функции....435
Объявление функций....436
Вызов функций....437
Рекурсия....438
Массивы....439
Создание массивов и работа с ними....439
Ссылки....440
Объекты....441
Понятия объекта и экземпляра....441
Работа с объектами и их экземплярами....442
Несколько новых операторов и ключевых слов....444
Встроенные объекты ActionScript....445
Объект *String*....445
Объект *Number*....446
Объект *Boolean*....447
Объект *Date*....447
Объект *Array*....448
Объект *Math*....448
Объект *Object*....448
Работа с клипами....449
Доступ к нужному клипу....449
Управление воспроизведением анимации....452
Обработка событий, возникающих в клипах....453
Управление клипами....455
Создание новых клипов....455
Удаление клипа....458
Изменение параметров встроенных клипов....458
Получение служебной информации о клипе....460
Реализация drag'n'drop....461
Создание фигурного курсора мыши....463
Выявление совпадений и касаний....464
Работа с графикой....466
Программное рисование графики....466

Пример сценария, рисующего график функции 470
Программное управление масками....472
Работа с кнопками 473
Свойства и методы объекта *Button*....473
Обработка событий, возникающих в кнопках 475
Создание и удаление кнопок из сценариев....475
Работа с текстовыми блоками....476
Объект *TextField* и его свойства и методы 476
Обработка событий, возникающих в текстовых блоках 479
Использование перехватчиков событий 480
Создание и удаление текстовых блоков из сценариев....482
Использование объекта *Selection*....482
Обработка нажатий клавиш клавиатуры 484
Свойства и методы объекта *Key*....484
Обработка нажатий клавиш в клипах 485
Обработка нажатий клавиш в клипах 485
Обработка нажатий клавиш с помощью перехватчика 487
Взаимодействие с мышью 488
Управление звуковым сопровождением фильма 489
Использование таймеров....490
Загрузка внешних файлов 493
Загрузка и выгрузка фильмов Shockwave/Flash и изображений в формате JPEG 494
Вложение загружаемого фильма или изображения в основной фильм....494
Перекрывающиеся фильмы 495
Загрузка звуков 497
Загрузка видео 498
Загрузка Web-страниц....499
Ограничения на доступ к внешним файлам 500
Работа с внешними данными 501
Отправка данных серверной программе 502
Получение данных от серверной программы....503
Использование объекта *LoadVars* 505
Что дальше? 507

Часть V. Публикация и экспорт графики. Прочие возможности Flash 509

Глава 20. Подготовка графики Flash к экспорту 511

Оптимизация фильма....512
Использование профилировщика загрузки для выявления "узких мест" фильма....512

Как оптимизировать фильм 517
Оптимизация векторной графики....517
Задание параметров импортированной растровой графики....518
Задание параметров звука....520
Обеспечение доступности фильма 523
Технология "чтение с экрана" 523
Поддержка Flash технологии "чтение с экрана" 524
Дополнительные параметры доступности 525
Полезные советы по созданию доступных фильмов....526
Что дальше? 527

Глава 21. Публикация и экспорт готовой графики Flash 528

Публикация....529
Выбор формата публикации....529
Выбор формата публикации фильма....529
Выбор формата публикации изображения-замены 531
Задание состава публикации 531
Параметры форматов публикации....533
Вкладка *Flash*....533
Вкладка *HTML*....536
Вкладка *GIF*....541
Вкладка *JPEG*....545
Вкладка *PNG*....546
Вкладка *QuickTime*....549
Предварительный просмотр публикуемой графики....553
Создание карты-изображения....553
Использование профилей публикации....554
Экспорт 557
Форматы экспорта, поддерживаемые Flash....557
Как экспортировать изображение или фильм 558
Параметры форматов экспорта....559
Adobe Illustrator 559
Apple QuickTime....559
AutoDesk AutoCAD 560
AVI....560
GIF 562
"Анимированный GIF"....563
JPEG....564
Macromedia Flash Video....565
Macromedia Shockwave/Flash 565
PNG....565
WAV....567
Растровые файлы Windows (формат BMP)....568

Как графика Flash помещается на Web-страницы 569
Понятие о внедренных элементах 569
Теги <*OBJECT*> и <*EMBED*> 570
Тег <*OBJECT*> 570
Тег <*EMBED*> 571
Совместное использование тегов <*OBJECT*> и <*EMBED*> 572
Параметры встраиваемого проигрывателя Flash 573
Создание и правка шаблонов HTML 579
Что дальше? 582

Глава 22. Дополнительные возможности Flash 583

Панель *Movie Explorer* 583
Поиск и замена фрагментов фильма 588
Панель *Find and Replace* 588
Регулярные выражения и их использование 594
Печать 597
Печать документов Flash 598
Создание фильмов, предназначенных для печати 600
Что дальше? 601

Глава 23. Настройка Flash 602

Общие настройки 602
Категория *General* 602
Категория *ActionScript* 605
Категория *Clipboard* 607
Категория *Drawing* 609
Категория *Text* 611
Категория *Warnings* 612
Настройка комбинаций клавиш 614
Что дальше? 618

Заключение 619

Предметный указатель 621

Введение

Вы держите в руках книгу, посвященную последней на данный момент версии программного пакета Macromedia Flash — Flash 8. А это значит, что вы им заинтересовались. Так что же такое Macromedia Flash?

Flash поможет вам создать компьютерную графику и анимацию, предназначенную, в основном, для публикации в Интернете. Скажем больше: Flash также позволяет создавать настоящие программы, принимающие данные от пользователя и обрабатывающие их. Скажем проще: это средство создания мультиков и программ, которые вы можете выложить в Сеть. Скажем чистую правду: именно Flash принес в Интернет высококачественную и компактную анимацию. Скажем еще кое-что: Flash породил целый вид искусства, известный как "Flash-анимация" и "Flash-программирование", и целую касту деятелей этого искусства, известную как "Flash-аниматоры" и "Flash-программисты". Скажем банальность: Flash — это современно. Скажем пошлость: Flash — это модно.

На сегодняшний момент существует множество Web-сайтов, созданных с использованием технологии Flash. Есть также довольно много программ, использующих для тех или иных целей Flash-графику. (Например, замечательный проигрыватель мультимедийных файлов J. River Media Jukebox, который вы можете найти на сайте **http://www.musicex.com/mediajukebox/**. Этот проигрыватель не только воспроизводит Flash-фильмы, но и хранит некоторые части своего интерфейса в формате Flash.) Создано большое количество неплохих Flash-фильмов, которые вы можете увидеть на сайтах **http://www.mp4.com**, **http://www.hypnotic.com**, **http://atomfilms.shockwave.com** и др. Существует Дмитрий Дибров, показывавший в телепередаче "Ночная смена" (уже давно закрытой) потрясающие по своей невразумительности Flash-ролики. И, наконец, существует целое сообщество "флэшеров", в которое при желании можете влиться и вы.

Так для кого же эта книга? Для тех, кто хочет научиться рисовать и анимировать в среде Flash. Только и всего.

Flash 8 — что нового?

Печальная тенденция, сполна проявившаяся в последнее время, — новые версии программ становятся все больше и больше по размерам. Вот и дистрибутив нового Flash "распух" до 110 Мбайт. Что же он при таких размерах может предложить бывалым "флэшерам"?

А предложить он может довольно много. Хотя все эти нововведения, скорее, эволюционные, в отличие от предыдущей версии — Flash MX 2004, — которая произвела небольшую революцию в интернет-графике. Давайте перечислим все нововведения восьмой версии, и начнем с самых значительных.

- Новый режим рисования — объектный. Рисуемые в этом режиме примитивы не подвергаются ни фрагментации, ни слиянию. Часто это бывает полезно.
- Градиентные и графические цвета. Это те же самые градиентные и графические заливки, поддерживавшиеся в предыдущих версиях, но теперь их можно применять и к линиям. Полезная штуковина, которой так не хватало раньше.
- Новые средства для отображения текста. Теперь возможно задавать для текстовых блоков различные режимы сглаживания. Разработчики утверждают, что в новой версии текст отображается много качественнее, чем в предыдущих.
- Улучшения в панели **Actions**, предназначенные для начинающих Flash-программистов. Фактически в новую версию Flash вернули базовый режим, имевшийся во Flash MX, но убранный во Flash MX 2004.
- Теперь серое пространство в рабочей области растягивается, так что мы можем "запихать" в нее сколько угодно графики.
- Улучшенное диалоговое окно **Preferences**, "ведающее" настройками программы.
- Улучшения в панели **Library**. Теперь мы можем открыть сколько угодно этих панелей, и каждая из них будет отображать содержимое библиотеки каждого открытого документа.
- Новый режим отката. Теперь мы можем выбрать такое поведение Flash, при котором мы можем выполнять откат для каждого из присутствующих на рабочем листе фрагментов нашего изображения, не затрагивая другие фрагменты.
- Фильтры. Они позволяют нам, например, создать у любого клипа тень исключительно средствами Flash, не обращаясь к другим программам.
- Цветовые наложения. С их помощью мы можем смешать цвета накладывающихся друг на друга клипов и получить таким образом новый цвет.

- Управление течением анимации. Теперь мы можем, например, заставить анимированный элемент двигаться сначала медленно, потом — быстро, а в самом конце снова замедлить его движение.
- Новый видеокодек — On2 VP6. Утверждается, что он обеспечивает лучшее качество изображения и более сильное сжатие, но нагрузка на процессор при этом будет больше.
- Немного переделанная структура меню. В новой версии она несколько удобнее, чем в предыдущей.

А теперь поговорим о том, что осталось как было. К сожалению...

- На редкость неудобная панель **Help**, в которой отображается содержимое интерактивной справки Flash. Ну неужели нельзя сделать так, как в Macromedia Dreamweaver, — отдельное окно со всей справочной информацией! Так ведь намного удобнее: можно одновременно смотреть текст справки и работать в программе. Вдобавок интерактивная справка Flash сделана в виде набора Web-страниц с картинками и занимает очень много места на жестком диске.
- Старая беда всех продуктов фирмы Macromedia — большое количество ошибок. И еще кто-то ругает Microsoft за ошибки в продуктах... Почему никто не ругает Macromedia?!!

> ***НА ЗАМЕТКУ***
>
> Когда писалась эта книга, фирму Macromedia со всеми ее продуктами купила давний ее конкурент — фирма Adobe. Так что следующая версия Flash будет выпущена уже Adobe.

Flash 8 и Flash 8 Professional

Flash 8, как и предыдущая версия этой программы, распространяется в двух разных комплектах поставки: обычная (Flash 8) и профессиональная (Flash 8 Professional). Профессиональная версия включает больше возможностей, но и стоит дороже.

Вот что включает в себя Flash 8 Professional:

- средства управления анимацией;
- эффекты и цветовые наложения;
- средства управления сглаживанием текста и растровой графики;
- утилита Macromedia Flash 8 Video Encoder;
- расширенный набор компонентов;

- окно кода — мощный инструмент, предназначенный для создания и правки файлов сценариев;
- панель **Accessibility**, предназначенную для задания параметров доступности приложений;
- некоторые дополнительные элементы.

А теперь — внимание! В данной книге мы будем изучать именно Flash 8 Professional как более полную. Никакие пакеты обновления для Flash 8 фирма Macromedia на момент написания этой книги еще не выпустила.

Типографские соглашения

В этой книге будут приведены многочисленные примеры сценариев ActionScript. При написании примеров были использованы типографские соглашения, уже ставшие своего рода стандартами в компьютерном книгоиздании. Нам необходимо их знать.

- В угловые скобки (<>) заключаются названия параметров или фрагментов кода, набранные курсивом. В код реального сценария, разумеется, должен быть подставлен реальный параметр или реальный код. Например:

  ```
  gotoAndPlay(<Номер кадра>);
  ```

  Здесь вместо подстроки `Номер кадра` должно быть подставлено реальное значение номера кадра.

- В квадратные скобки ([]) заключаются необязательные фрагменты кода. Например:

  ```
  function <Имя>([<Набор параметров>]);
  ```

  Здесь `Набор параметров` может присутствовать, а может и не присутствовать.

- Если в какое-либо место сценария должна быть подставлена команда, выбираемая из некоего ограниченного набора, в этом месте приводятся все команды данного набора, разделенные символом `|` ("вертикальная черта"). Например:

  ```
  onClipEvent(mouseDown|mouseUp) { . . .
  ```

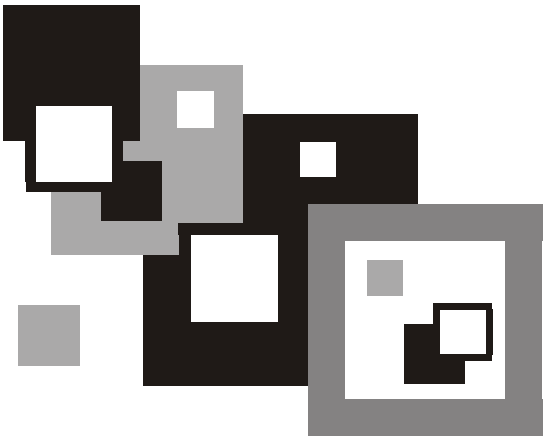
  Здесь в качестве параметра действия `onClipEvent` может присутствовать либо `mouseDown`, либо `mouseUp` (но не оба одновременно).

Весь остальной код сценария набирается в области редактирования кода панели **Actions** "как есть".

Благодарности

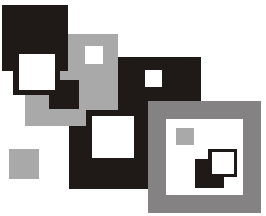
Автор приносит благодарности своим родителям, знакомым и коллегам по работе.

- Губине Наталье Анатольевне, начальнику отдела АСУ Волжского гуманитарного института (г. Волжский Волгоградской обл.), где работает автор, — за понимание и поддержку.
- Всем работникам отдела АСУ — за понимание и поддержку.
- Родителям — за терпение, понимание и поддержку.
- Архангельскому Дмитрию Борисовичу — за дружеское участие.
- Шапошникову Игорю Владимировичу — за содействие.
- Рыбакову Евгению Евгеньевичу, заместителю главного редактора издательства "БХВ-Петербург", — за неоднократные побуждения к работе, без которых автор давно бы обленился.
- Издательству "БХВ-Петербург" — за издание моих книг.
- Всем своим читателям и почитателям — за прекрасные отзывы о моих книгах.
- Всем, кого я забыл здесь перечислить, — за все хорошее.



ЧАСТЬ I

ВВЕДЕНИЕ ВО FLASH



Глава 1

Основы компьютерной графики

Изучать Flash мы начнем с теории. Да-да, куда же без нее!.. Усядемся поудобнее на диван или в кресло, выключим компьютер — пусть отдохнет! — и приступим к чтению этой главы. В ней описано все, что нам будет полезно знать о компьютерной графике и анимации. (Те, кто и так все о ней знают, может начать чтение книги сразу с *главы 2*.)

Сначала мы выясним, как возникла и развивалась компьютерная графика, а также что ждет ее в ближайшем будущем (на отдаленное будущее загадывать не будем — мы же не гадалки какие-нибудь, в конце концов). Историю знать никогда не вредно, а, наоборот, очень полезно.

Далее мы узнаем все о двух принципиально разных видах компьютерной графики, их достоинствах и недостатках, поговорим о различных способах сохранения графической информации в файлах (*форматах графики*). После этого мы выясним, какие форматы применяются для распространения графики в Интернете и какой формат в каком случае применить.

Ну и, наконец, узнаем, что анимация, или видео (с технической точки зрения эти слова — синонимы), тоже бывает двух видов, которые мы непременно рассмотрим. Также мы поговорим о форматах сохранения анимации, иначе говоря, видео в файлах (*форматах видео*).

Особый разговор пойдет о великолепном Macromedia Flash 8 (ведь именно этому пакету и посвящена данная книга). Пора, наконец, выяснить, чем же он так хорош.

Компьютерная графика: прошлое и настоящее

История компьютерной графики и видео неотделима от истории персональных компьютеров. В самом деле, во времена больших ЭВМ компьютерная графика если и существовала, то носила чисто утилитарный характер. В ее задачи могла входить, например, необходимость нарисовать зелеными линиями на черном фоне простейший график, основанный на результатах каких-либо расчетов, или вывести на экран только что спроектированную печатную плату. Таков был максимум возможностей, которые обеспечивал *видеоадаптер* (устройство для вывода информации на монитор) этих машин.

Компьютерного же видео в те времена вообще не существовало. И техника была не та, и машинное время стоило слишком дорого для того, чтобы тратить его на просмотр фильмов. Не забывайте, что компьютеры в те времена использовались исключительно для дела. А как поется в одной старой песне, "первым делом — самолеты"...

Появившийся в начале семидесятых персональный компьютер (ПК) произвел настоящую революцию в мире вычислительной техники. Изначально подразумевалось, что это чудо может использоваться его хозяином не только для дела, но и для потехи ("и делу время, и потехе час" — так в действительности гласит старая пословица). А для этого "персоналке" нужны хорошие возможности по выводу сложных графических изображений. Потому неудивительно, что даже на заре новой эпохи только самые дешевые ПК имели видеоадаптер, приспособленный исключительно для вывода текста (как говорят профессиональные компьютерщики, *алфавитно-цифровой*). Все более-менее приличные машины уже тогда были с графикой на "ты", так как имели *графический* видеоадаптер.

Первой потехой для пользователей ПК стали компьютерные игры. Индустрия игр появилась также вместе с ПК — на больших компьютерах игр просто не существовало (хотя, по слухам, первая компьютерная игра была написана именно для большой ЭВМ). И не просто появилась, а вскоре стала подлинным двигателем прогресса в компьютеростроении. Ведь зачем совершенствуются трехмерные ускорители, звуковые карты, зачем выпускаются умопомрачительные джойстики? Неужели для Microsoft Word или "1С:Бухгалтерии"? Отнюдь! Лишь для того, чтобы любимая игрушка "бегала" быстрее, монстры были страшнее, а игроку — сподручнее их отстреливать.

Компьютерная графика эволюционировала вместе с компьютерным "железом" и программным обеспечением. Сначала это были корявые картинки, выполненные в гамме шестнадцати цветов. С совершенствованием видеоадаптеров и мониторов изображение на них стало выглядеть более похожим на... гм... изображение и менее — на плохую мозаику. А по мере развития программного обеспечения качество изображения улучшалось, разме-

ры графического файла уменьшались, а компьютерные художники из сумасбродов-одиночек превратились в настоящих деятелей искусства.

Наконец, где-то в начале девяностых настало время компьютерного видео. Как раз в это время появились достаточно мощные видеоадаптеры, позволяющие выводить графику фотографического качества, емкие жесткие диски, CD-ROM и соответствующие программы.

Вероятно, самой первой программой для работы с видео была программа *QuickTime*, разработанная фирмой Apple для компьютеров Macintosh и позднее перенесенная на Microsoft Windows. Фирма Microsoft в ответ разработала программный пакет *Video for Windows*. Но это было еще не компьютерное видео, а лишь первый шаг к нему.

Несколькими годами позднее, когда мощности персональных компьютеров еще более возросли, а CD-ROM получили достаточно широкое распространение, появился формат записи фильмов на CD, который так и назывался — *VideoCD*. Это был первый и не слишком удачный опыт создания цифрового кино "для народа". На одном диске VideoCD помещалось только 74 минуты видео, поэтому полнометражные фильмы поставлялись на двух дисках. Качество "картинки" было невысоким; фильмы смотрели на компьютерах с помощью специальных программ, открывая их в маленьком окошке, или на особых проигрывателях, похожих на видеомагнитофоны.

Вероятно, вы видели эти диски — не могли не видеть. Бизнес по продаже VideoCD был развит довольно сильно: автор припоминает, что видел в Волгограде магазин, торгующий фирменными дисками с отечественным кино. Но все же VideoCD — один из тех многочисленных "блинов", что, будучи первыми, выходят комом.

Настоящую жизнь компьютерному кино дали появившиеся во второй половине 90-х диски DVD и соответствующий формат для записи на них видео — *DVD-Video*. На DVD помещается до 4,7 Гбайт информации, чего вполне хватает не только на полноразмерный (два часа) фильм высокого качества, но и на несколько звуковых дорожек, наборов субтитров, интерактивные меню и прочие бонусы. К несчастью, этот замечательный носитель очень долго пробивал себе дорогу и стал популярным только в самые последние годы.

Середина 90-х годов примечательна еще одним событием. Именно в то время появилось понятие *мультимедиа* (multimedia, "многосредность"), обозначавшее совокупность текстовых, цифровых, звуковых данных и видеоданных, объединенных в единое целое. А достаточно мощные компьютеры, обрабатывающие все эти данные, стали называться мультимедийными.

И тогда же, в 90-х годах прошлого века, разразился интернет-бум. Конечно же, он не мог не повлиять на развитие компьютерной графики и видео.

Ведь что такое Интернет? Нет, не так... Что такое Интернет (Всемирная Сеть или просто Сеть с большой буквы) для большинства его пользователей? Это

электронная почта (отметаем сразу, ибо не наш профиль), вирусы (лучше бы их не было), хакеры (аналогично) и *World Wide Web* (она же *WWW*, *Web* и Всемирная Паутина). Вот на WWW мы остановимся подробнее.

Что же такое WWW? Это *Web-страницы*, для просмотра которых используются особые программы — *Web-обозреватели* (или *Web-браузеры* — от английского *browser*, обозреватель). А что такое Web-страницы? Это текст, который можно читать, файлы, которые можно загрузить, музыка, которую можно слушать, графика и видео, которые можно смотреть. Вы слышите? Графика и видео!!!

Изначально, правда, WWW была полностью текстовой. Да-да, не удивляйтесь. Тим Бернерс-Ли, создавший в 1989 году язык *HTML* (HyperText Markup Language, язык гипертекстовой разметки), на котором и пишутся Web-страницы, не предусмотрел в нем поддержку графики. Лишь в дальнейшем, под нажимом общественности, консорциум *W3C* (или *WWWC*, World Wide Web Consortium — консорциум всемирно протянутой паутины... ну и название!..), занимающийся развитием языка HTML, внес в него соответствующие изменения. Так в Интернет пришла графика.

Сейчас WWW без графики представить очень сложно. Всевозможные картинки, фотографии, схемы, графические элементы оформления, баннеры, в конце концов, заполонили Web-страницы (зачастую от них просто рябит в глазах, что не есть хорошо). Интернет-графика сейчас — это и строгие новостные сайты, и шикарные развлекательные порталы, и концептуальные страницы с потрясающе красивым дизайном. Да что и говорить — наверняка вы не новички во Всемирной Сети и сами все это видели.

Забегая немного вперед, скажем, что и Flash был создан именно для работы с интернет-графикой. Так что ее развитие отнюдь не прекратилось.

Ну что ж, исторический экскурс можно считать законченным. Конечно, мы о многом не упомянули, обойдя и настольные издательства, и научную графику, и трехмерное моделирование. Но все это — узкоспециализированные решения, весьма далекие от среднестатистического компьютерщика, так что не будем подробно их рассматривать.

А теперь пора приступить к изучению компьютерной графики. Начнем со статичной, т. е. неподвижной графики, анимацией же займемся потом.

Статичная графика

Статичная графика — это неподвижные изображения: фотографии, рисунки, схемы, многие элементы оформления Web-страниц. Здесь будет идти речь именно о ней.

Два вида статичной графики

Сначала поговорим о двух принципиально разных видах статичной компьютерной графики. А именно, о растровой и векторной графике. Это нам очень пригодится в дальнейшем.

Растровая графика

Если хорошенько рассмотреть фотографию в электронном виде на мониторе компьютера при большом увеличении, то можно увидеть, что она состоит из множества точек квадратной формы. Если рассматривать такую иллюстрацию на некотором расстоянии, а не вблизи, как сделали мы, отдельные точки сливаются в единое, кажущееся цельным, изображение. Это классический пример *растровой* графики. Такая графика, как мы выяснили, состоит из множества разноцветных точек — *пикселов*. Поэтому растровую графику иногда называют *пиксельной*.

На рис. 1.1 показан небольшой пример растрового изображения — литера А, как она отображается на экране компьютера. Хорошо видно, что она состоит из множества разноцветных — белых, серых и черных — пикселов.

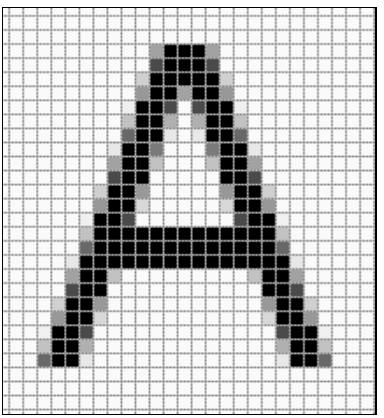


Рис. 1.1. Пример растровой графики — литера А

Вообще, все, что отображается на экране компьютера, суть растровая графика. Дело в том, что компьютерный экран сам представляет собой не что иное, как растр. Поэтому самые первые компьютерные графические форматы были именно растровыми.

В случае растровой графики в графическом файле сохраняется упорядоченный набор (опытные компьютерщики говорят — *массив*) значений цветов в

пикселах растра. Разумеется, где-то в начале файла, в его *заголовке*, должен быть записан размер изображения, например, 320×200 пикселов, иначе программное обеспечение не сможет правильно обработать файл. Также иногда в файл записываются дополнительные данные: сведения о создателе, о программе, в которой редактировался файл, и пр.

Для кодирования каждого пиксела растрового изображения отводится определенное количество битов, поэтому изображение может содержать только ограниченное количество цветов, называемое *цветностью*. Понятно, что чем больше выделяется битов на кодирование одного пиксела, тем бóльшее количество цветов может быть использовано в изображении. В табл. 1.1 приведены используемые в настоящее время значения цветности изображений.

Таблица 1.1. *Значения цветности, используемые в настоящее время*

Количество битов на пиксел	Цветность (количество цветов, используемых в изображении)	Общепринятое название изображений с такой цветностью
1	2 (черный и белый)	–
2	4	–
4	16	CGA
8	256	–
16	65 536	HiColor
24 (32)	16 776 576	TrueColor

Да, но каким образом представляются значения цветов? Для этого используются два способа, которые мы сейчас рассмотрим.

В случае графики с цветностью TrueColor (*фотореалистичной*, или *полноцветной*) все очень просто. Значение цвета пиксела представляет собой три числа, обозначающих доли красной, зеленой и синей составляющих соответственно. Причем каждое число занимает ровно восемь битов, т. е. один байт. Такой способ задания цвета называется *RGB* (от английского Red, Green, Blue — красный, зеленый, синий).

Если изображение содержит меньшее количество цветов, то все немного сложнее. Сначала создается *палитра* — особая таблица, в которой записаны все цвета, используемые в изображении, в формате RGB. А значение цвета каждого пиксела в этом случае — просто номер (*индекс*), указывающий на нужный цвет в палитре. Такие цвета называются *индексированными*, а сама графика — *графикой с палитрой*. Размер палитры зависит от количества битов, выделяемых на представление цвета; например, если выделено 4 бита (полубайт), то палитра может содержать 16 цветов.

Очень часто, особенно в последнее время, применяются полупрозрачные изображения, сквозь которые "просвечивает" то, что находится под ними. Вы, наверно, видели шикарные пиктограммы Windows XP, в которых полупрозрачность используется очень часто. В этом случае наряду со значением цвета каждого пиксела нужно хранить и степень его прозрачности. Для этого также используются два способа.

В случае полноцветной графики TrueColor все тоже довольно просто. Степень прозрачности пиксела задается с помощью дополнительных восьми битов ("Одного байта!" — кричат бывалые компьютерщики), добавляемых к уже имеющимся двадцати четырем (если 8 бит умножить на 3 цвета, получится как раз 24). Эти восемь битов называются *каналом прозрачности* или *альфа-каналом*, а сама цветность — *TrueColor с каналом прозрачности* или просто 32-битной.

Полноцветная графика позволяет художнику задать прозрачность отдельно для каждого пиксела. Графика же с палитрой таких вольностей не допускает. Здесь используется другой способ задания прозрачности: один из цветов палитры "в приказном порядке" объявляется прозрачным (*прозрачный цвет*). Обычно это цвет левого верхнего пиксела изображения.

Растровая графика имеет как достоинства, так и недостатки. Перечислим их, начав, разумеется, с достоинств.

- Простота вывода. В самом деле, для того чтобы вывести растровое изображение на экран монитора или принтер, не требуются сверхсложные вычисления. Отображение растровой графики не "нагружает" слишком сильно процессор компьютера, а значит, вывод изображения происходит очень быстро. Какая-либо дополнительная обработка при этом отсутствует, за исключением, может быть, подстройки цветов.
- Размер массива пикселов, а значит и графического растрового файла, зависит от геометрических размеров самого изображения и от его цветности (фактически — от количества битов на точку). Размер растрового изображения не зависит от его сложности. Это означает, что маленькие черно-белые изображения занимают меньше места, чем большие полноцветные. Это очень хорошо для Web-дизайна — там как раз используются, в основном, небольшие изображения.
- Высокая точность и достоверность передачи полутоновых изображений, например, сканированных картин и фотографий. В самом деле, если использовать достаточно большое разрешение и цветность TrueColor, то цифровая копия визуально не будет отличаться от оригинала.

Теперь рассмотрим недостатки растровой графики.

- Мы уже знаем, что размер массива пикселов зависит от геометрических размеров самого изображения и от его цветности. Иногда это выходит боком. Так, если мы сохраним в растровом формате простенькое, но

полноцветное и, вдобавок, огромное по размерам изображение, оно вполне может занять на диске десятки мегабайт. Что ж, очень часто недостаток является обратной стороной достоинства...

- Растровая графика зависит от разрешения устройства вывода: монитора или принтера. *Разрешение* — это максимальное количество пикселов по горизонтали и вертикали, которое может вывести устройство. В самом деле, если вывести изображение размером 640×480 пикселов на монитор с таким же разрешением, то этот рисунок займет весь экран целиком. Если же его вывести при разрешении 1024×768, то на экране отобразится только часть рисунка. Так что нам либо придется мириться с этим, либо выполнять *масштабирование* изображения — пропорциональное изменение его размеров, — чтобы "вписать" его в нужное нам разрешение.
- Качество растровых изображений ухудшается при сильном масштабировании.

Последний пункт нужно пояснить на примере. Предположим, что мы имеем небольшое растровое изображение, и у нас возникло желание его увеличить. Откроем его в программе графического редактора, выполним команду увеличения и... Получим результат, показанный на рис. 1.2.

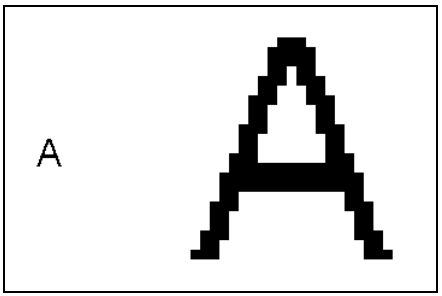


Рис. 1.2. Результат увеличения растрового изображения (слева — изначальное изображение, справа — увеличенное)

Слева на рис. 1.2 показано исходное изображение, справа — результат его увеличения. Видно, что каждый пиксел исходного изображения увеличился до размеров огромного "кирпича", в результате чего правое изображение сильно исказилось.

Как можно преодолеть этот недостаток?

Во-первых, по мере возможности не следует менять размеры растровых изображений. Лучше всего создавать их именно такого размера, какой нужен. В крайнем случае их можно уменьшить или совсем немного увеличить, чтобы точечная структура была незаметна.

Во-вторых, рекомендуется использовать достаточно мощные графические пакеты, например, последние версии Adobe Photoshop, для масштабирования растровой графики. Реализованные в них алгоритмы позволяют менять размеры изображений практически без потерь качества. Поставляемый в составе Microsoft Windows простейший графический редактор Paint этого не может.

Что касается первого недостатка растровой графики — прямой зависимости размера графического файла от геометрических размеров изображения — то он также практически преодолен. Дело в том, что подавляющее большинство графических форматов предоставляют возможность *сжатия* массива пикселов, в результате которого размер графического файла сильно уменьшается. Правда, такой подход чреват ростом затрат процессорного времени на распаковку изображения и риском потери данных при использовании слишком сильного сжатия.

Вот и все о растровой графике. Предоставим слово конкурирующей стороне.

Векторная графика

Рассказ о векторной графике мы начнем с небольшого допущения. Предположим, что любое, даже очень сложное графическое изображение можно разбить на простейшие элементы: прямые и кривые линии, эллипсы, прямоугольники и т. п. Эти простейшие элементы, называемые *примитивами*, описываются с помощью определенных формул. В результате мы получим набор параметров для этих формул, используя которые, можно точно воссоздать исходный набор примитивов, а значит и исходное изображение. Так вот, графика, состоящая из примитивов, и называется *векторной* графикой.

В качестве примера возьмем все ту же литеру А в векторном представлении. Если внимательно присмотреться к ней (рис. 1.3), можно увидеть, что она состоит из трех примитивов — прямых линий. (На рис. 1.3 они немного отделены друг от друга для лучшей наглядности.)

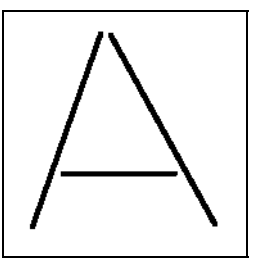


Рис. 1.3. Векторное представление литеры А

Но, спросите вы, как же компьютер выводит векторные изображения на экран? Ведь экран компьютера — это растр, и компьютер должен сначала преобразовать изображение в набор пикселов, т. е. растрировать его? Вы правы. Да, компьютер растрирует векторную графику, для чего дополнительно тратятся его системные ресурсы. Затраты системных ресурсов на растеризацию — один из главнейших недостатков векторной графики, но неоспоримые достоинства с лихвой его окупают.

Перечислим эти достоинства.

- Независимость размера файла векторного изображения от геометрических размеров самого изображения. Ведь в этом случае в файл записывается не огромный массив цветовых значений для всех пикселов, составляющих изображение, а только типы и параметры всех задействованных в нем примитивов, занимающие сравнительно небольшой объем.
- Прекрасная масштабируемость. В самом деле, для того чтобы изменить размеры изображения, нужно лишь умножить параметры размера всех формул примитивов на значение масштаба, вычислить их повторно и перерисовать изображение. Взглянем на рис. 1.4 — векторное изображение в любом масштабе выглядит идеально.
- Как следствие масштабируемости — независимость от разрешения устройства вывода: монитора или принтера.
- Исключительные возможности по обработке изображений. Векторные изображения можно поворачивать, искажать, отображать зеркально, перекрашивать, делать полупрозрачными и т. п. (рис. 1.5). Аналогичные манипуляции с растровыми изображениями потребуют много системных ресурсов.

Рис. 1.4. Векторное изображение прекрасно масштабируется

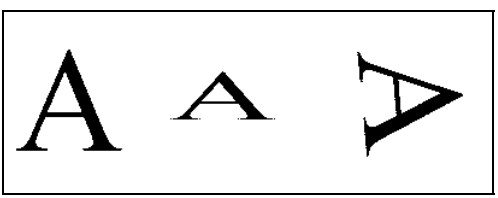


Рис. 1.5. Векторное изображение можно вращать и искажать

Кстати, знаете ли вы, что обычные компьютерные шрифты, используемые Windows, суть векторные изображения? (Здесь имеются в виду так называемые шрифты формата *TrueType*, файлы которых имеют расширение ttf.) Благодаря векторному представлению они исключительно хорошо масштабируются до любых размеров. Однако системные шрифты, используемые для вывода надписей на диалоговых окнах, заголовков окон, пунктов меню, хранятся все же в растровом виде, чтобы зря не расходовать системные ресурсы.

Теперь перечислим недостатки векторной графики и укажем пути их преодоления.

- Размер файла векторного изображения зависит от уровня его сложности. В самом деле, чем сложнее изображение, тем больше примитивов включает оно в себя и тем больше данных потребуется сохранить в файле.
- Вывод векторной графики (а именно, ее растеризация) требует больше времени и бóльших системных ресурсов. В этом смысле растровая графика "работает" быстрее.
- Практически невозможно преобразовать полутоновое растровое изображение TrueColor в векторное (выполнить *векторизацию*) без больших потерь его качества.

Первые два недостатка вполне преодолимы. Во-первых, не нужно без нужды создавать слишком сложные векторные изображения. Во-вторых, надо стараться комбинировать векторную и растровую графику — современные графические пакеты предоставляют такую возможность. В-третьих, чересчур сложную векторную графику для распространения ее среди потребителей (или поклонников) можно перевести в растровый вид (кстати, так часто и делают).

К сожалению, третий недостаток преодолеть практически невозможно. Все продукты изобразительного искусства, созданные людьми до появления компьютера и векторной графики, в большинстве случаев лучше передаются растровой графикой. Если же попытаться превратить растровое изображение в векторное с помощью программы векторизации, наступит момент, когда оно окажется слишком сложным для этой программы. Да и качество получившегося шедевра будет очень низким. Так что ваши фотографии, снятые во время последнего отпуска, пусть остаются в растровом формате.

Вместе с тем, векторная графика неплохо подходит для сохранения штриховых изображений. Так что если у вас случайно завалялись чертежи — почему бы не попробовать!

Вот и все — о векторной графике нам больше сказать нечего. Давайте же теперь еще раз сравним возможности растровой и векторной графики и выясним, какая из них и в каких случаях предпочтительнее.

Применение растровой и векторной графики

Главный козырь растровой графики — точность передачи сканированных изображений. При этом растровая графика занимает тем больший объем, чем больше само изображение, плохо масштабируется и медленно обрабатывается. Главный козырь векторной графики — наличие развитых средств обработки изображения, а главный ее недостаток — невозможность сохранения полутоновых изображений в близком к оригиналу виде. Исходя из этого, можно определить область применения для каждого из двух видов компьютерной графики.

Итак, растровая графика применяется:

- для хранения и обработки полутоновых изображений (сканированные или изначально созданные на компьютере картины, фотографии);
- в Web-дизайне. Применяемые на Web-страницах изображения, как правило, невелики, а вывод их на экран осуществляется самими Web-обозревателями без привлечения дополнительных программ.

Векторная графика лучше всего подойдет, если нужно:

- сохранить штриховые изображения (карты, чертежи, рисунки карандашом, гравюры) в электронном виде;
- создать небольшие изображения, которые в дальнейшем будут всячески обрабатываться при выводе. Хороший пример таких изображений — шрифты формата TrueType, которые при выводе на экран не только масштабируются, но и раскрашиваются в разные цвета, поворачиваются и т. п.

В остальных случаях можно использовать как векторную, так и растровую графику. Нужно только помнить о недостатках, присущих обоим этим видам, и, разумеется, об их преимуществах.

Осталось напомнить о том, что Flash — формат векторной графики. Точнее же будет сказать: гибридной.

Гибридная графика

Собственно, *гибридная* графика — это разновидность векторной графики, поддерживающая включение в изображение растровых фрагментов. Благодаря такому подходу часто удается преодолеть главнейшие недостатки и растровой, и векторной графики: слишком большой размер файла растрового изображения и невозможность точной передачи полутоновых изображений средствами векторной графики.

Все современные редакторы векторной графики, в том числе Flash, предоставляют возможность создания гибридной графики. Фактически их можно

назвать редакторами гибридной графики. Гибридную графику создают также настольные издательские программы.

Вот и все о двух видах графики: растровой и векторной. Далее мы рассмотрим конкретные форматы сохранения графики, поговорим об их достоинствах и недостатках и опишем область применения каждого формата.

Форматы графики

В этом разделе речь пойдет о самых распространенных на сегодняшний день форматах сохранения графики в файлах. Как правило, все эти форматы поддерживаются Flash, за исключением некоторых, особо оговоренных нами. Дополнительные, менее распространенные форматы, также поддерживаемые Flash, будут описаны в *главе 10*.

Формат BMP

Растровый формат *BMP* (BitMaP, битовая матрица) — простейший формат записи растровых изображений. Он также известен под названием *DIB* (Device Independent Bitmap, битовая матрица, независимая от устройства вывода). Разработан фирмой Microsoft в самом начале 90-х годов прошлого века для сохранения графики в операционной системе Windows и совместимых с ней программах. Поддержка файлов формата BMP встроена непосредственно в ядро Windows.

Графика сохраняется в файлах с расширением bmp или (крайне редко) dib. Может быть любой цветности. Графические данные могут быть сжаты с использованием простейшего алгоритма *RLE* (Run Length Encoding, кодирование с переменной длиной строки).

В настоящее время это один из самых распространенных графических форматов, поддерживаемый практически всеми графическими программами. Из-за своей простоты он требует для вывода очень мало системных ресурсов, поэтому основное его предназначение — хранение изображений, используемых как элементы пользовательского интерфейса операционной системы. В частности, именно в формате BMP хранятся системные "обои", заставки, пиктограммы и т. п.

Формат GIF

Растровый формат *GIF* (Graphic Interchange Format, формат обмена графикой) был разработан фирмой CompuServe в 1987 году для использования в собственной одноименной компьютерной сети. Наряду с форматом JPEG

(об этом формате будет рассказано ниже) получил огромное распространение в компьютерных сетях, в частности, в Интернете.

Графика хранится в файлах с расширением gif. Цветность — 256 цветов (т. е. используется палитра). Для сжатия графики используется алгоритм *LZW*, разработанный математиками Лемпелом, Зивом и Велчем. Графика может быть сохранена с *чередованием строк* (interleaving) — в этом случае изображение как бы постепенно "проявляется" строка за строкой по мере загрузки файла.

В 1989 году формат GIF был расширен; новая версия стандарта получила название *GIF89A*. Во-первых, была введена поддержка "прозрачного" цвета. Во-вторых, появилась возможность сохранять в одном файле несколько изображений, которые могли бы демонстрироваться последовательно, как кадры анимационного фильма (так называемые *анимированные GIF-файлы*).

Формат GIF прекрасно подходит для сохранения штриховой графики и вообще всех изображений с резкими цветовыми переходами. В частности, Web-дизайнеры создают в этом формате элементы оформления и рекламные баннеры для своих страниц, а поддержка прозрачности и анимации им только на руку. Иногда в этом формате создаются начальные заставки и графические элементы Windows-программ.

К несчастью, будущее формата GIF весьма туманно. Еще в середине 90-х годов фирма CompuServe хотела получать авторские отчисления с продажи каждой программы, поддерживающей формат GIF, но сетевому сообществу тогда удалось отстоять его бесплатность. Теперь же фирма Unisys, владеющая патентом на алгоритм LZW, тоже хочет получать авторские отчисления, ведь именно этот алгоритм применен в формате GIF для сжатия графики. Судя по всему, этот формат в покое не оставят. Кроме того, 256 цветов по сегодняшним временам — очень мало. Поэтому формату GIF уже прочат преемника — формат PNG, описанный ниже.

В настоящее время формат GIF поддерживается практически всеми графическими программами и Web-обозревателями. И, несмотря на все перипетии с авторскими правами, он все еще активно используется.

Формат PNG

Растровый формат *PNG* (Portable Network Graphic, переносимая сетевая графика) разработан сообществом независимых программистов в качестве замены устаревающего и грозящего перейти в разряд коммерческих продуктов формата GIF. Он поддерживается в настоящее время многими графическими пакетами, однако большой популярности в Интернете пока не снискал, хотя автору время от времени встречаются сайты с PNG-графикой.

Графика хранится в файлах с расширением png. Она может быть любой цветности. Для сжатия графики применяется очень мощный алгоритм *Deflate* (буквально — "усыхание"), обеспечивающий более сильное сжатие по сравнению с LZW. Графика может быть сохранена с чередованием, причем не только строк, но и столбцов; таким образом, изображение будет "проявляться" и по строкам, и по столбцам. Поддерживается 256 степеней прозрачности "прозрачного" цвета и автоматическая коррекция яркости.

Однако, по сравнению с GIF, формат PNG имеет и недостатки, правда, не принципиальные. Первый недостаток — PNG не поддерживает анимацию, что сейчас, в связи с повсеместным переходом Web-аниматоров на Flash, неактуально. Второй недостаток — файлы формата PNG больше, чем GIF-файлы, примерно на один килобайт из-за того, что в заголовке файла хранится больше служебной информации.

Пока что формат PNG используется для хранения графики, разрабатываемой в пакете Macromedia Fireworks, и на некоторых "продвинутых" сайтах. Как уже говорилось выше, его поддерживают практически все графические пакеты и Web-обозреватели.

Формат JPEG

Растровый формат *JPEG* (Joint Photographic Experts Group, Объединенная группа экспертов по фотографии) разработан одноименной группой программистов специально для распространения высококачественной графики в компьютерных сетях. Именно для этого он и используется в настоящее время.

Графика сохраняется в файлах с расширениями jpeg, jpe или jpg. Поддерживается только цветность TrueColor (24-битный цвет). Для сжатия графики используется исключительно мощный алгоритм под названием JPEG, фактически включающий в себя несколько алгоритмов сжатия, используемых в разных случаях. Этот алгоритм реализует *сжатие с потерями*, при котором из массива пикселов перед сжатием убирается некоторая часть информации, отчего его размер становится еще меньше. Во всех других форматах графики, использующих сжатие, применяются алгоритмы *сжатия без потерь*.

При использовании алгоритмов сжатия с потерями качество графики ухудшается. Чем сильнее сжатие, тем сильнее искажается изображение. Однако художник может регулировать процент сжатия, выбирая подходящее сочетание хорошего качества изображения с удовлетворительным размером результирующего JPEG-файла.

Формат JPEG, в отличие от GIF и PNG, не поддерживает ни анимацию, ни прозрачность. Однако существует разновидность формата JPEG, называемая *прогрессивным JPEG* (progressive JPEG, или p-JPEG), поддерживающая чередование строк.

Область применения формата JPEG достаточно узка — распространение высококачественной полутоновой графики в Интернете. Сканированные полутоновые изображения при использовании умеренного сжатия получаются очень даже неплохими. Формат JPEG поддерживается практически всеми современными графическими программами и Web-обозревателями.

На этом разговор о статичной компьютерной графике можно считать законченным. На очереди — анимация и видео!

Анимация и видео

Любой кино- или видеофильм можно представить как последовательный набор статичных изображений (*кадров*), которые очень быстро сменяют друг друга. Так как человеческий глаз не может уследить за сменой одного кадра другим, субъективно этот непрерывный "поток" изображений выглядит как одна движущаяся картинка. Для достижения такого эффекта скорость смены или, как говорят профессионалы, *частота кадров* должна быть достаточно велика. В табл. 1.2 приведены стандартные значения частоты кадров, применяемые в кино и на телевидении.

Таблица 1.2. *Стандартные значения частоты кадров*

Значение частоты, кадров/с	Где применяется
12	Web-анимация, японские мультфильмы "анимэ"
24	Обычное кино
25	Телевидение стандартов SECAM и PAL
30	Телевидение стандарта NTSC

В зависимости от способа хранения и представления отдельных кадров фильма, компьютерная анимация, так же как и графика, относится к одному из двух видов. Давайте их рассмотрим.

Два вида компьютерной анимации

Компьютерные анимация и видео (в дальнейшем мы будем употреблять эти термины как синонимы, так как особой разницы между ними нет) могут быть покадровыми или трансформационными.

Покадровая анимация

Покадровая анимация (ее еще называют *классической*) представляет собой набор кадров, хранящихся как отдельные изображения и сменяющих друг друга с большой скоростью. Это самый старый и самый надежный способ сохранения движущегося изображения на каком-либо носителе (пленке, бумаге, магнитной ленте, жестком диске, CD, DVD). Пример покадровой анимации из пяти кадров показан на рис. 1.6.

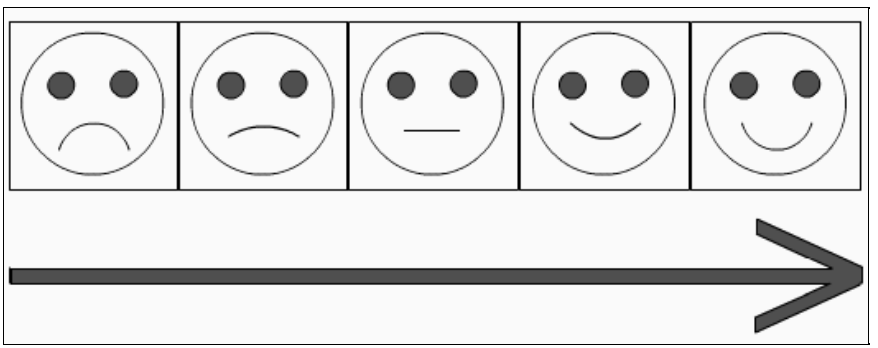


Рис. 1.6. Пример покадровой анимации из пяти кадров (стрелка указывает направление воспроизведения)

Абсолютно все фильмы, созданные к данному моменту времени трудолюбивым человечеством, представляют собой покадровую анимацию. Еще бы — ведь сам принцип действия киноаппарата основан на фиксации на светочувствительной пленке множества неподвижных изображений, каждое — через определенный промежуток времени. Двадцать четыре (стандартная частота кадров "большого" кино) раза в секунду киноаппарат приказывает: "Остановись, мгновенье". Из многих тысяч таких вот "остановившихся мгновений" и состоит любая кинолента.

Аналогичным образом работает и видеокамера. Правда, в этом случае процесс создания последовательности кадров не так очевиден: информация записывается в электронном виде на магнитный носитель, и невооруженным глазом ее не увидишь. Но, можете поверить, здесь все абсолютно так же, как в случае с кинокамерой.

А если взять рисованные и кукольные анимационные фильмы, то там покадровая анимация существует в чистом виде. Каждый кадр фильма рисуется или выстраивается на сцене, после чего кинокамерой делается один-единственный кадр. Затем готовится следующий кадр — и т. д., пока не будет готов весь фильм. Адская работа... Конечно, сейчас появилось множест-

во технических новинок, облегчающих труд аниматора, в том числе и компьютеры, но принцип остался тем же.

Чем же полюбилась человечеству покадровая анимация? Вместо ответа рассмотрим все ее преимущества.

- Относительная очевидность создания. В самом деле, для того чтобы изготовить анимационный фильм, нужно всего лишь нарисовать все входящие в него кадры и перевести их на какой-нибудь информационный носитель. Что ж, очевидно, но отнюдь не просто...
- Широкие возможности для творчества. Ну, тут уж и говорить не о чем!..

К несчастью, на этом преимущества покадровой анимации кончаются. И начинаются недостатки.

- Большая трудоемкость создания фильмов. Если каждый кадр рисуется вручную и при этом не применяются никакие технические средства, облегчающие работу, процесс создания фильма может затянуться на многие месяцы, а то и годы. (Обычные кинофильмы создаются значительно быстрее, так как для них не нужно рисовать кадры — оператор просто фиксирует реальную сцену.) Да и пресловутые технические средства ненамного ускоряют этот процесс.
- Большие проблемы, возникающие при сохранении покадровой анимации в цифровом виде.

Вот здесь давайте остановимся и поговорим о переводе фильмов в цифровой вид (*оцифровке*) и их хранении.

Каждый из множества кадров, составляющих фильм, занимает при хранении определенное пространство на диске. Предположим, что это пространство составляет 100 килобайт — для хранения полноцветного изображения высокого разрешения в формате JPEG этого даже маловато. Теперь предположим, что количество изображений составляет 100 000 — такой длинный у нас фильм. Умножив 100 на 100 000, получим 10 000 000, т. е. примерно 10 гигабайт (примерно, потому что гигабайт — это не 1 000 000 000, а 1 073 741 824 байта). Выходит, для хранения фильма нам нужен целый жесткий диск или примерно 2,5 диска DVD, а уж сколько для этого понадобится обычных CD, просто страшно подумать!

Что делать? Разумеется, сжать фильм посильнее! И заодно сжать звуковое сопровождение, если оно есть.

Для сжатия фильмов практически всегда используется сжатие с потерями. Как мы уже знаем, в этом случае какая-то часть информации, не очень нужная при воспроизведении, отбрасывается, за счет чего размер файла фильма становится заметно меньше. Более того, алгоритмы, реализующие сжатие именно фильмов, анализируют каждый кадр и сохраняют в резуль-

тирующем файле только данные о различиях между соседними кадрами. Это еще сильнее уменьшает размер сжатого фильма.

> **НА ЗАМЕТКУ**
>
> Совсем короткие, порядка нескольких секунд, фильмы либо вообще не сжимаются, либо сжимаются без потерь. В частности, такие вот несжатые фильмы используются в качестве элементов интерфейса Windows-программ (например, летящие листочки в диалоговом окне процесса копирования Проводника).

Перечислим самые популярные алгоритмы сжатия видео, применяемые в настоящее время.

- *Intel Indeo.* Разработан фирмой Intel в начале 90-х, на заре эры мультимедиа. Обеспечивает довольно слабое сжатие, но зато без проблем работает на старых компьютерах. Сейчас используется для сжатия совсем коротких, в несколько секунд, видеороликов, зачастую используемых в качестве элементов интерфейса Windows-программ.
- *MPEG I.* Самый первый из этого семейства алгоритмов, разработанный также в начале 90-х группой MPEG (Motion Picture Encoding Group, группа кодировки движущихся изображений) для записей дисков VideoCD. Обеспечивает среднюю степень сжатия и довольно высокое качество изображения. Существует также разновидность этого алгоритма, предназначенная для сжатия звука, — *MPEG I level 3* (*MP3*).
- *MPEG II.* Был разработан во второй половине 90-х для записи дисков DVD-Video. Обеспечивает более высокие качество и степень сжатия изображения, чем MPEG I.
- *MPEG IV.* Был разработан также во второй половине 90-х специально для распространения фильмов через Интернет. Обеспечивает более высокую степень сжатия, чем MPEG II, а также поддерживает различные дополнительные возможности, например, защиту от несанкционированного копирования и создание интерактивных элементов.
- *DivX.* Был разработан в самом конце 90-х группой независимых программистов как бесплатная альтернатива коммерциализированному MPEG IV. Использовался для распространения пиратских копий фильмов, но потом "вступил на честный путь" и в настоящее время сам стремительно коммерциализируется.

Современные алгоритмы сжатия, например, MPEG IV и DivX, позволяют поместить сжатый в неплохом качестве полноразмерный фильм на обычный компакт-диск, т. е. размер сжатого с их помощью видеофайла составляет примерно 700 мегабайт. Фактически именно эти два алгоритма и совершили "компьютерно-киношную" революцию, создав высококачественное цифровое кино "для народа".

Сжатие фильма выполняется с помощью особой программы, называемой *кодером*. Такой кодер реализует какой-либо из перечисленных выше алгоритмов сжатия.

Программа, воспроизводящая сжатое видео, должна иметь возможность распаковать его. Для распаковки фильма используется программа-*декодер*, которая также реализует один из алгоритмов сжатия. При открытии файла с фильмом программа-проигрыватель видео определяет по записанной в его заголовке информации, каким алгоритмом сжат фильм, и подключает соответствующий декодер.

Очень часто и кодер, и декодер объединяют в одну программу, называемую *кодеком* (кодером-декодером). Кодек часто носит название реализуемого им алгоритма сжатия: так, например, существуют кодеки MPEG II и DivX.

Но здесь возникает другая проблема. Сжатые с помощью алгоритмов MPEG IV и DivX фильмы могут "осилить" только достаточно мощные компьютеры. Если вы попробуете просмотреть фильм DivX на компьютере выпуска пятилетней давности, то увидите не нормальный фильм, а некое слайд-шоу. Это происходит потому, что маломощный процессор, не успевая распаковывать данные и выдавать их на экран, вынужден пропускать целые кадры. К счастью, никому в голову не приходит запускать цифровое кино на старых компьютерах.

Вот, собственно, и все о покадровой анимации. Теперь поговорим о ее конкуренте.

Трансформационная анимация

Давайте еще раз посмотрим на рис. 1.6 и предположим, что каждый кадр такой анимации хранится в векторном виде. (Анимация, изображенная на рис. 1.6, так и просится в векторный вид. Сами посмотрите — ведь это простейшая графика, одни только линии.) Далее, предположим, что мы можем описывать с помощью формул не только форму кривых линий и прочих графических примитивов, но и их поведение. Следовательно, мы можем изменить форму "рта", просто вызвав соответствующую формулу и подставив в нее нужные параметры. Что у нас получится?

А получится у нас *трансформационная анимация*. От покадровой она отличается тем, что не описывает каждый кадр последовательности отдельно, а сразу задает поведение того или иного примитива (рис. 1.7).

Так как же создается трансформационная анимация? Очень просто. Сначала мы создаем два кадра, определяющие начальное и конечное состояние нашего изображение. Давайте назовем эти два кадра, созданные нами, *ключевыми* — в дальнейшем этот термин будет применяться очень часто. Остальные же кадры (*промежуточные*; на рис. 1.7 они показаны серым цветом)

будут сформированы программой-проигрывателем на основе заданных нами ключевых кадров.

Рис. 1.7. Пример трансформационной анимации из пяти кадров (стрелка указывает направление воспроизведения, серые кадры формируются программой-проигрывателем)

Внимание!

Введенные нами два термина имеют смысл только в случае трансформационной анимации. В покадровой же анимации все кадры будут ключевыми, а промежуточных кадров не будет вовсе.

Понятно, что создать трансформационную анимацию проще всего на основе векторной графики. В этом случае, чтобы создать промежуточные кадры, программе-проигрывателю будет достаточно взять параметры примитивов, из которых состоят изображения на начальном и конечном ключевых кадрах, и создать на их основе параметры примитивов для всех промежуточных кадров. Растровую графику анимировать таким образом много сложнее.

Хоть векторная графика как способ представления изображений существует довольно давно, трансформационная анимация возникла только в последние годы. Фактически трансформационную анимацию создал пакет Flash. Если до него и существовали какие-то аналогичные разработки, то они остались неизвестными широкой публике.

Перечислим все достоинства и недостатки трансформационной анимации. Начнем, конечно же, с достоинств.

- Исключительная простота создания. Нам нужно всего лишь создать ключевые кадры анимации, задать ее длительность и некоторые дополнительные параметры, а остальное — дело техники (программы-проигрывателя). Нам не придется кропотливо вырисовывать все входящие в наш фильм кадры, как это требуется в случае покадровой анимации.

- Исключительная компактность получающегося массива данных. Как мы помним, векторная графика занимает меньше места, чем растровая — так и трансформационная анимация занимает меньше места, чем покадровая. Ведь согласитесь — для хранения нескольких параметров функции, задающей анимацию, нужно меньше места, чем для множества кадров, каждый из которых представляет собой растровое изображение.
- Легкость правки трансформационной анимации. Чтобы исправить что-то в классическом фильме, нам придется перерисовывать или переснимать целые сцены. В случае же трансформационной анимации нам во многих случаях нужно будет только изменить пару параметров функции, задающей анимацию.

Полюбовались мы трансформационной анимацией — и хватит! Пора и поругать ее за недостатки. Недостаток, правда, всего один, но зато какой!

Давайте еще раз посмотрим на рис. 1.6 и 1.7 и еще раз подумаем. Что мы можем делать с помощью покадровой и трансформационной анимации? С помощью покадровой — все. А с помощью трансформационной? Не так уж и много — только самые простейшие движения. Все богатство возможностей, предлагаемых покадровой анимацией, нам в этом случае недоступно.

В утешение скажем, что Flash позволяет создавать как покадровую, так и трансформационную анимацию. А это значит, что мы можем объединять достоинства и избавляться от недостатков этих двух видов анимации. Осталось только выяснить, какой вид анимации, а также — когда и где следует применять.

Применение разных видов анимации

Покадровая анимация незаменима при создании сложных фильмов с богатой графикой. Тут уж комментарии излишни.

Трансформационная анимация, наоборот, пригодна для создания простейших анимационных эффектов для Web-страниц. В виде трансформационной анимации также создаются простейшие фильмы рекламного, развлекательного и учебного назначения, например, пресловутые баннеры. Помимо этого, трансформационная анимация прекрасно подходит для создания на Flash пользовательских интерфейсов и целых программ — всех этих всплывающих меню и нажимающихся кнопок.

Ну и, конечно, никто не запрещает нам сочетать оба вида анимации в одном фильме, объединяя их преимущества и избегая недостатков. Так, для простейших случаев создания движений мы можем использовать трансформационную анимацию, для более сложных — покадровую. Опытные Flash-аниматоры так и поступают.

Форматы видео

Теперь поговорим о самых распространенных на сегодняшний день форматах хранения видео. Практически все эти форматы, кроме формата Shockwave/Flash, обеспечивают сохранение только покадровой анимации. Все они, за небольшим исключением, поддерживаются Flash.

Здесь нужно сделать небольшую оговорку. В отличие от графических форматов GIF, JPEG и PNG, ни один видеоформат не поддерживается Web-обозревателем напрямую (единственное исключение — анимированный GIF, но это, как бы, не совсем видео). Чтобы просмотреть помещенный на Web-страницу фильм, нам придется воспользоваться дополнительными программами. К счастью, эти программы уже включены в состав поставки всех современных Web-обозревателей.

Формат QuickTime

QuickTime — один из самых первых, если не самый первый, формат видеофайлов, получивший широкое распространение. Разработан фирмой Apple в конце 80-х годов; изначально предназначался для использования на компьютерах Macintosh, впоследствии был перенесен в операционную систему Microsoft Windows. Позднее подвергался неоднократным усовершенствованиям. На момент написания книги последней версией является 7.0.4, но на момент ее выхода, скорее всего, появится новая версия.

Позволяет хранить и видео-, и аудиоинформацию в одном файле с расширением mov. Для сжатия данных используется одноименный алгоритм. Степень сжатия довольно велика, но качество получающегося фильма не очень высоко по сравнению с качеством, обеспечиваемым алгоритмами группы MPEG и производными от них. Поэтому формат и алгоритм QuickTime в последнее время применяется не очень широко.

За прошедшие десять лет формат QuickTime приобрел довольно большую популярность, в основном, для распространения коротких музыкальных, рекламных или обзорных видеоклипов. Абсолютный лидер на платформе Macintosh. В Web-дизайне применяется крайне редко.

Формат AVI

Формат данных *AVI* (Audio and Video Interlaced, чередующиеся аудио и видео) был разработан фирмой Microsoft в начале 90-х годов для использования в пакете Microsoft Video for Windows — мультимедийном расширении операционной системы Windows (позже это расширение стало составной частью данной системы).

Позволяет хранить и видео-, и аудиоинформацию в одном файле с расширением avi. Для сжатия информации могут использоваться различные алгоритмы, что делает этот формат очень гибким. Существует также разновидность этого формата, служащая для хранения только звука; файлы такого формата имеют расширение wav (от WAVe — волна).

AVI — абсолютный лидер на платформе Windows. Широко используется для распространения самых разных фильмов: художественных, рекламных, учебных, специального назначения. Часто используется для сохранения небольших анимационных роликов, применяемых в программах (в частности, "летающие листочки" в окне копирования Windows хранятся именно в формате AVI). Иногда формат AVI также используется в Web-дизайне.

В настоящее время фирма Microsoft признала формат AVI устаревшим в силу некоторых ограничений (в частности, на размер: AVI-файл не может быть больше 4-х гигабайт). На смену ему подготовлен формат WMV, описанный ниже.

Формат MPEG

Формат *MPEG* был разработан в начале 90-х годов одноименной группой программистов для сохранения фильмов на компакт-дисках CD и DVD (т. е. создания VideoCD и DVD-Video).

Файл формата MPEG может хранить и видео-, и аудиоинформацию и имеет расширения dat, vob, mpg, mpe, mpeg, mp1, mp2 или mp4. Для сжатия данных чаще всего используются алгоритмы MPEG I, MPEG II или MPEG IV, иногда — DivX.

Формат MPEG используется только для распространения фильмов различной длительности и назначения на компакт-дисках CD и DVD. В Web-дизайне он практически не применяется.

Формат WMV

Формат *WMV* (Windows Media — Video) — это дальнейшее развитие формата AVI. Разработан фирмой Microsoft в самом конце 90-х годов как часть инициативы Windows Media (создание набора аппаратных и программных средств, направленных на улучшение мультимедийных возможностей современных Windows-совместимых компьютеров). В настоящее время постепенно приобретает популярность.

Файлы WMV могут хранить и видео-, и аудиоинформацию и имеют расширение wmv или asf. Для сжатия информации используется собственный кодек. По сравнению с AVI формат WMV имеет дополнительные возможности, в частности, средства защиты от несанкционированного копирова-

ния. Существует также особая разновидность формата, предназначенная для хранения звука; файлы такого формата имеют расширение wma (Windows Media — Audio).

Используется для распространения фильмов различной длительности и назначения. В Web-дизайне практически не применяется.

Формат RealMedia

Формат *RealMedia* был разработан фирмой RealNetwork в середине 90-х специально для распространения видео через Интернет. Вероятно, это первый видеоформат, ориентированный на использование в Сети. Подвергался неоднократным усовершенствованиям. В настоящее время последней версией является 10.0.

Позволяет хранить и видео-, и аудиоинформацию в одном файле с расширением rm или smil. Для сжатия данных используется одноименный алгоритм. Степень сжатия весьма велика, и достигаемое при сжатии качество фильма также достаточно высоко.

За время своего существования захватил довольно большую долю рынка "сетевого видео" и терять ее не собирается. В основном применяется для распространения фильмов различной длительности и назначения и для трансляции так называемого "интернет-телевидения". В Web-дизайне применяется нечасто.

> ***ВНИМАНИЕ!***
>
> Формат RealMedia не поддерживается Flash.

Формат "анимированный GIF"

Также для хранения анимации может быть использован анимированный GIF-файл. Как мы помним, формат GIF89A позволяет записывать в один файл несколько изображений, которые станут кадрами фильма.

К достоинству GIF-фильма можно отнести то, что он поддерживается Web-обозревателями напрямую, так что нам не понадобятся никакие дополнительные программы. Недостатки: трудность управления таким фильмом (скажем, невозможно остановить его воспроизведение, а потом запустить снова) и прямая зависимость размера файла от продолжительности фильма (вспомним, что в формате GIF применяется сжатие без потерь, а значит, не очень сильное).

В настоящее время формат "анимированный GIF" очень широко применяется в Web-дизайне. В нем создают рекламные баннеры и элементы оформ-

ления страниц. Иногда изображения в этом формате используют для оформления Windows-программ.

Краткое введение во Flash

А теперь настало время разобраться с Flash, выяснить, что это такое и чем он может нам помочь.

Flash — это пакет создания статичной графики и анимации, разработанный фирмой Macromedia. Графика, созданная в нем, распространяется в формате Shockwave/Flash, о котором мы потом поговорим отдельно.

То, что дает нам Flash, можно выразить тремя словами: универсальность, компактность, безопасность. Рассмотрим подробно, что же скрывается за этими словами. И начнем с перечисления недостатков старого доброго языка HTML, на котором пишутся Web-страницы.

Недостатки языка HTML

Из небольшого исторического экскурса, приведенного в начале этой главы, мы узнали, что изначально WWW была полностью текстовой, т. е. помещение на Web-страницы графики не поддерживалось. И это не удивительно: ведь WWW изначально была сетью для ученых, которые отнюдь не баловались графикой, и обычного текста им вполне хватало, чтобы донести друг до друга свои труды. Конечно, впоследствии, когда в Сеть пришел падкий на красивости обыватель, разработчики из W^3C добавили в HTML возможность помещения графики на Web-страницы, таким образом устранив его первый недостаток.

Да, устранили! Но давайте посмотрим, как реализована эта возможность. Сначала все графические изображения, которые *Web-дизайнер* (создатель Web-страниц) хочет поместить на свои творения, сохраняются в отдельных файлах. Потом в нужных местах HTML-кода вставляются особые ссылки на эти файлы. И, наконец, когда Web-обозреватель загрузит страницу, он считывает эти файлы и выводит хранящиеся в них изображения в отмеченных местах. Т. е. сама Web-страница и все входящие в ее состав изображения хранятся в разных файлах. Согласитесь — подход не из самых изящных...

Другой недостаток HTML — это его статичность. Web-страницы, уже загруженные и отображенные Web-обозревателем, невозможно изменить. Если, например, мы хотим, чтобы при наведении курсора на изображение оно заменялось другим, язык HTML ничем нам не поможет. А все потому, что он описывает внешний вид страниц, но никак не поведение отдельных их эле-

ментов в ответ на какие-либо события (в частности, на наведение курсора мыши).

Но и этот недостаток, в принципе, преодолен. Фирма Netscape разработала для своего Web-обозревателя Navigator особый язык программирования *JavaScript*. На этом языке пишутся особые программы (*Web-сценарии*), управляющие поведением различных элементов Web-страницы, которые вставляются прямо в HTML-код. В настоящее время JavaScript поддерживается практически всеми Web-обозревателями.

Вот только эта связка — HTML + JavaScript — работает из рук вон плохо. Дело в том, что различные программы Web-обозревателей поддерживают Web-сценарии (как и сам HTML) по-разному. В результате код, прекрасно работающий в Internet Explorer, может не функционировать в Navigator или Opera. Эта пресловутая несовместимость Web-обозревателей отравила жизнь многим Web-дизайнерам. И выхода — увы! — не предвидится...

Есть еще одна проблема, связанная с самим JavaScript. Если мы собираемся реализовать на этом языке какой-либо сложный алгоритм, являющийся "ноу-хау", будем готовы к тому, что его очень быстро украдут. Дело в том, что программу, написанную на JavaScript, можно просмотреть с помощью обычного Блокнота, поставляемого в составе Windows. А уж разобраться в том, что она делает, может любой школьник, знакомый с основными навыками JavaScript-программирования.

Что делать? Как решить все эти проблемы?

И вот тут под гром фанфар на сцене появляется великий и ужасный Macromedia Flash 8. Настал его час!

Достоинства Flash

Давайте перечислим все достоинства Flash, Короля Графики, Спасителя Всея Интернета и проч., и проч., и проч., и подробно их опишем. Ну и, конечно, расскажем о его недостатках и о том, как их можно обойти или преодолеть.

Универсальность

Представим себе двух Web-дизайнеров, которые хотят создать Web-страничку с анимированной графикой и различными эффектами, вроде изменяющихся картинок. При этом один из них пользуется классическими технологиями (HTML и JavaScript), а другой — Flash.

Первый Web-дизайнер пишет HTML-код, пользуясь Блокнотом или более мощным Web-редактором, наподобие Macromedia Dreamweaver. После этого

он переключается в графический пакет — Adobe Photoshop или Macromedia Fireworks, — рисует картинки и сохраняет их в файлах. Далее он правит HTML-код, помещая в него ссылки на нужные файлы. Результат его работы — файл Web-страницы и набор графических файлов, где хранятся нужные изображения.

Второй Web-дизайнер запускает Flash, создает новый документ, пишет текст, помещает изображения, и получает на выходе один-единственный файл Shockwave/Flash, содержащий данный документ, со всем текстом и графикой. И все это — не выходя из Flash!

Теперь нужно добавить на страницу анимацию. Первый Web-дизайнер мучительно чешет в затылке, пытаясь выбрать файловый формат, в котором сохранить анимацию. Потом запускает нужный пакет, делает фильм и сохраняет его в отдельном файле. А ведь ему еще нужно исправить HTML-код, чтобы поместить на страницу готовый фильм, — а это, уж поверьте автору, задачка не для слабонервных. И хорошо, если посетители его страницы смогут просмотреть этот фильм — а вдруг у них на компьютерах не установлено нужное программное обеспечение?..

Второй Web-дизайнер говорит: "Значит, анимация... Будет вам анимация!" После этого он запускает Flash, открывает документ и добавляет в него анимацию. И все!

Осталось обеспечить специальные эффекты. Первый Web-дизайнер, ругаясь последними словами, пишет Web-сценарии на JavaScript, отлаживает их, исправляет ошибки и уф!.. борется с несовместимостью Web-обозревателей. В результате получается нечто, кое-как работающее, да и то не во всех Web-обозревателях.

Второму Web-дизайнеру для этого достаточно открыть документ и написать нужные Flash-сценарии на встроенном языке *ActionScript*. И никаких проблем с совместимостью!

Вот что значит универсальность! Если первый Web-дизайнер был вынужден для решения поставленной задачи пользоваться сразу несколькими пакетами и несколькими интернет-технологиями, зачастую плохо работающими совместно друг с другом, то второму для этого было достаточно одного только Flash. Вдобавок, первому Web-дизайнеру придется заботиться о том, чтобы на компьютерах посетителей его сайта стояло программное обеспечение, необходимое для проигрывания анимации. Для просмотра же изображений Flash нужна только одна программа-проигрыватель, а ей, как говорят, оснащены сейчас 95% всех компьютеров мира. Но даже если где-нибудь в дремучем лесу найдется компьютер, где этой программы нет, то для того чтобы ее установить, достаточно зайти на нужную страницу сайта фирмы Macromedia и подождать несколько минут.

Компактность

Результат трудов первого Web-дизайнера — довольно большой набор файлов, занимающий немало места на диске и довольно долго загружающийся по сети. Результат трудов второго — один-единственный файл формата Shockwave/Flash размером зачастую всего в несколько килобайт. Есть разница?

Файлы, хранящие графику Shockwave/Flash, очень компактны. Во-первых, как мы помним, это векторная графика, и занимает она значительно меньше места, нежели растровая. Во-вторых, файлы Shockwave/Flash сжимаются с помощью довольно сильного алгоритма. В-третьих, Flash при сохранении в формат Shockwave/Flash оптимизирует графику, удаляя те ее части, которые гарантированно в любом случае не будут выводиться на экран.

Вот что такое компактность в случае Flash!

Безопасность

Когда шел разговор о языке JavaScript, было сказано, что исходный код этих написанных на нем сценариев можно просмотреть с помощью любого Web-обозревателя и любого текстового редактора. В самом деле, JavaScript-сценарий пишется прямо в HTML-коде Web-страницы, страница сохраняется в обычном текстовом файле с расширением htm[l], а просмотреть текстовый файл — вообще не задача для грамотного компьютерщика.

Посмотреть, как устроены файлы Shockwave/Flash, значительно труднее. После того как мы выполним сохранение изображения Flash в этом формате, оно превращается в "вещь в себе", которую можно только рассматривать, но никак не изменять. Это значит, что мы можем реализовать в нем какие-либо ноу-хау без боязни того, что их позаимствуют без спросу.

Так что и с безопасностью у Flash все в порядке. Во всяком случае, лучше, чем у связки HTML + JavaScript.

> ***НА ЗАМЕТКУ***
>
> Здесь нужно сказать, что существуют программы для просмотра содержимого файла Shockwave/Flash в удобном для пользователя виде. Однако эти программы мало распространены, зачастую имеют ограниченные возможности, а работа с ними требует определенной квалификации.

Формат Shockwave/Flash

Как мы уже выяснили, векторный формат *Shockwave/Flash* разработан фирмой Macromedia для распространения изображений и фильмов, созданных в пакете Flash. Настало время поговорить о нем.

НА ЗАМЕТКУ

В этом же самом формате сохраняется распространяемая графика и анимация, созданная в пакете Macromedia Shockwave — "старшем брате" Flash. Поэтому формат Shockwave/Flash и носит такое название.

Файл формата Shockwave/Flash хранит в себе и графику, и анимацию, и сценарии ActionScript, а также растровую графику, видео и звук, созданные в других программах и импортированные во Flash, причем хранится все это в оптимизированном и сжатом виде. Такие файлы имеют расширение swf.

Этот формат поддерживается очень многими графическими пакетами и программами для работы с видео. Но, как ни странно, он, хотя существует уже достаточно давно — несколько лет — и успел стать в Сети стандартом де-факто, до сих пор не поддерживается Web-обозревателями непосредственно. Поэтому для просмотра графики Shockwave/Flash требуется дополнительная программа — *проигрыватель Flash*, которая требует отдельного разговора.

Проигрыватель Flash

Существуют три разновидности проигрывателей Flash. По предоставляемым возможностям они абсолютно схожи: могут отображать как статичную, так и анимированную графику, воспроизводить видео и звук, а также выполнять сценарии ActionScript. Различаются они только принципами работы с пользователем.

Первая разновидность — это *встраиваемый* проигрыватель Flash. Он работает совместно с Web-обозревателем и занимается выводом изображений и фильмов Shockwave/Flash на Web-страницы. Встраиваемый проигрыватель управляется самим Web-обозревателем; сам же пользователь, как правило, непосредственно с ним дела не имеет.

Современные Web-обозреватели уже имеют в комплекте своей поставки встраиваемые проигрыватели Flash. Пользователям более старых программ придется загрузить и установить их самостоятельно. Для этого проще всего зайти на особую страницу Web-сайта фирмы Macromedia и подождать несколько минут. Web-обозреватель сам загрузит дистрибутивный комплект проигрывателя и сам же его установит.

Если на каком-то из доступных вам компьютеров установлен пакет Macromedia Flash, вы можете установить себе проигрыватель Flash вручную. Для этого откройте папку Players/Release, находящуюся в папке, где установлен сам пакет Flash. В этой папке находятся файлы Install Flash Player 8 ActiveX.msi и Install Flash Player 8 Plugin.msi — это и есть нужные дистрибутивы. Первый предназначен для пользователей Web-обозревателя Internet Ex-

plorer, второй — для поклонников Navigator, Mozilla, Firefox и Opera (впрочем, для Internet Explorer он тоже подойдет).

Вторая разновидность проигрывателя Flash называется *автономным проигрывателем*. Это обычная программа, которую можно запустить и открыть в ней файл с изображением Shockwave/Flash, после чего он начнет воспроизводиться. Автономный проигрыватель также поставляется в составе пакета Flash и хранится в той же папке Players/Release. Исполняемый файл этой программы называется SAFlashPlayer.exe.

Третья разновидность проигрывателя Flash не совсем обычна. Это так называемый *проектор* — исполняемый файл, содержащий не только уже знакомый нам автономный проигрыватель, но и изображение Shockwave/Flash. При запуске такого файла на экране появится окно автономного проигрывателя, в котором сразу же будет открыто включенное в состав проектора изображение.

Проекторы создаются в самом пакете Flash и очень полезны для распространения фильмов Shockwave/Flash — ведь в этом случае их может просмотреть любой пользователь, даже не имеющий на компьютере проигрывателя Flash. Единственный недостаток: при создании проектора размер файла с изображением Shockwave/Flash очень сильно возрастает — на целых 1544 килобайта (это размер исполняемого файла автономного проигрывателя Flash).

Формат документов Flash

Формат Shockwave/Flash используется только для распространения готовой графики и анимации Flash. Для хранения же графики, находящейся в данный момент в разработке, используется другой формат — так называемый *документ Flash*. После окончания работы над изображением необходимо выполнить операцию *публикации* или *экспорта*, чтобы сохранить его в формате Shockwave/Flash. (Публикация и экспорт графики Flash будут описаны в *главе 21*.)

Зачем так сделано? Почему бы не хранить рабочие файлы сразу в формате Shockwave/Flash? Тому есть всего одна, но существенная причина.

Дело в том, что документ Flash включает в себя много информации, нужной только художнику, работающему над изображением. Такой информацией могут быть импортированные растровые изображения, фильмы, звуковые клипы, тексты сценариев ActionScript, различные временные данные, графика, которая не должна отображаться на экране (например, наброски или фрагменты, оставленные про запас), и пр. Потребителю графики Shockwave/Flash все это совершенно не нужно. Более того, эта информация зачастую занимает очень много места, поэтому размер файла

документа Flash может достигать нескольких мегабайт. Попробуйте опубликовать его в Интернете!

При публикации или экспорте пакет Flash сохраняет в файле Shockwave/Flash только ту графику, что должна выводиться на экран. Кроме того, он ее оптимизирует и довольно сильно сжимает. В результате всего этого размер файла, хранящего изображение, может уменьшиться на два порядка.

> ***ВНИМАНИЕ!***
>
> Обязательно экспортируйте изображения Flash в формат Shockwave/Flash перед распространением.

Осталось только сказать, что файл, хранящий документ Flash, имеет расширение fla.

Конкуренты Flash

Наш разговор будет неполон без краткого перечисления конкурентов технологии Macromedia Flash, их достоинств и недостатков. А также перспектив — а то вдруг Flash уже безнадежно устарел, а Дронов все пишет о нем книги!

На данный момент главнейшими конкурентами Flash являются уже существующие технологии и стандарты. Это, прежде всего, языки HTML и JavaScript. Простота создания документов с помощью этих языков (и бесплатность — пакет Macromedia Flash дорого стоит!..), ориентированность на тексты, обилие поддерживающего эти языки программного обеспечения обеспечивают HTML и JavaScript большую популярность, которой Flash пока что похвастаться не может. О недостатках этого конкурента мы уже говорили: "открытость" кода документов и программ и различия в поддержке этих языков разными Web-обозревателями.

Для распространения статичной графики в Интернете используются три традиционных формата: GIF, JPEG и PNG. Они, в принципе, справляются со своими задачами, и менять их на что-то другое пока смысла нет. Кроме того, они широко поддерживаются современными графическими программами и Web-обозревателями. Прочие достоинства, равно как и недостатки этих форматов, — суть достоинства и недостатки растровой графики в целом.

В мире компьютерного видео сейчас безраздельно царят форматы AVI, QuickTime и MPEG версий I, II и IV. (И еще формат "анимированный GIF", но его возможностей хватает только на короткие видеоролики, в основном рекламные.) Они также неплохо справляются со своими задачами, широко поддерживаются программным обеспечением, и менять их на что-то другое потребности не возникает.

Итак, что же мы выяснили? Оказывается, для распространения текстов лучше всего подходит язык HTML, для распространения графики — традиционные растровые форматы GIF, JPEG и PNG, а для распространения видео — не менее традиционные AVI, QuickTime и различные версии MPEG. И особых причин менять существующее положение дел у интернет-сообщества нет.

А что Flash? Выходит, это бедный родственник, которого выгнали из-за стола? Ни в коем случае! Вспомним, какую его возможность мы изучили первой. *Универсальность!* Flash незаменим там, где нужно объединить текст, графику и анимацию в один компактный файл. Он — "игрок" совершенно другой "весовой категории", и противники ему нужны соответствующие. А противников таких на данный момент имеется три: апплеты Java, модули расширения Web-обозревателя и компоненты ActiveX.

Апплет Java (от английского *applet* — "приложеньице") — это небольшая программа, написанная на особом языке *Java*, созданном в середине 90-х годов фирмой Sun и внедренная в Web-страницу. Для обработки апплетов Java требуется особая программа — виртуальная машина Java, которую можно найти на сайте фирмы Sun (**http://www.sun.com**).

Преимущества апплетов Java: небольшой размер, межплатформенность (апплеты Java могут работать на любой компьютерной платформе, так как не привязаны к конкретной компьютерной архитектуре), безопасность (даже вредоносный апплет не способен серьезно набедокурить). Недостатки: относительная трудность создания, потребность в дополнительной программе (виртуальной машине Java) и очень большие требования к системным ресурсам.

Модуль расширения Web-обозревателя — это небольшая программа, работающая совместно с Web-обозревателем и расширяющая его возможности. В частности, модуль расширения может использоваться для обработки форматов данных, не поддерживаемых самим Web-обозревателем, как то: графики, видео, звука и пр. Достоинства модулей расширения: могут быть созданы с помощью любого языка программирования; большее, по сравнению с апплетами Java, быстродействие и меньшая требовательность к ресурсам, а также бóльшая универсальность. Недостатки: большой размер и некоторые проблемы с безопасностью.

Компонент ActiveX похож на апплет Java — это также небольшая программа, работающая совместно с Web-обозревателем и расширяющая его возможности. Его достоинства и недостатки те же, что у модуля расширения. К недостаткам также можно добавить привязанность к платформе Windows (что неудивительно, ведь технология ActiveX была разработана фирмой Microsoft).

На заметку

Кстати, встраиваемый проигрыватель Flash реализуется либо в виде модуля расширения Web-обозревателя, либо элемента ActiveX. Так что с этими технологиями мы, так или иначе, столкнемся, когда будем пользоваться встраиваемым проигрывателем Flash.

Из-за трудности создания апплеты Java и компоненты ActiveX не нашли широкого применения в Web-дизайне, заняв лишь очень узкую нишу в сфере корпоративных решений. В массовом же Web-дизайне царствует Flash, и конкурентов, достойных его, пока что не видно...

На заметку

Надо, правда, упомянуть о формате создания векторной графики и анимации SVG (Scalable Vector Graphics, масштабируемая векторная графика). Этот формат позиционируется как прямой конкурент Shockwave/Flash и постепенно набирает популярность.

Кроме того, в новой версии Microsoft Windows — Vista — будет подерживаться новый формат создания электронных документов под кодовым названием *Metro*. В этом формате будет возможно создавать документы, содержащие, кроме всего прочего, и анимацию. К сожалению, Windows Vista появится только в конце 2006 года, так что оценить возможности формата Metro сейчас не представляется возможным.

Что делают с помощью Flash

Напоследок давайте перечислим все виды графики, которые обычно создаются с использованием незаменимого Macromedia Flash.

- Рекламные баннеры и кнопки, как правило, содержащие анимацию.
- Интерфейсы для различных интернет-сервисов, например, почтовых серверов, интернет-магазинов или справочных баз данных.
- Учебные, рекламные и художественные фильмы, в том числе и помещаемые на Web-страницы. Классический пример — "Масяня".
- Небольшие программы, также часто помещаемые на Web-страницы. Чаще всего встречаются онлайновые игры или различные "приколы". (Автору однажды встретилась Flash-программа, сочиняющая название для рок-группы.)
- Полноценные Web-сайты, в основном, развлекательные и околокультурные. (Автор может привести в качестве примеров "стильный" сайт английской группы "Death in June" и потрясающий по красоте портал, посвященный владивостокскому року. К сожалению, автор потерял ссылки на эти сайты...)

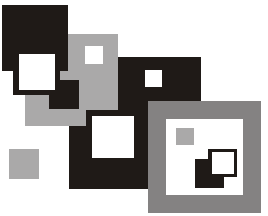
Осталось привести список всего того, что не делается в формате Flash.

- Обычный текст. Его проще всего оформить в формате HTML и выложить в Сеть для всеобщего обозрения. Если же вы хотите, чтобы ваши тексты прочитали только избранные, распространяйте их по электронной почте (почтовые рассылки).
- Обычная графика: схемы, фотографии, картины и т. п. Их распространяют в форматах GIF и JPEG. Штриховые рисунки — схемы, гравюры, карты — лучше распространять в формате GIF, а полутоновые — картины и фотографии — в формате JPEG.
- Любые более-менее сложные программы. Их проще написать на обычном языке программирования (C++, Pascal, Delphi, Visual Basic), откомпилировать и опубликовать в Сети в виде архивного файла или дистрибутива. Программу также можно оформить в виде апплета Java, модуля расширения Web-обозревателя или компонента ActiveX, в крайнем случае — как Web-сценарий, написанный на JavaScript.
- Документы, предназначенные для печати на бумаге. Ну, это было бы совсем глупо!

Хотя, в принципе, никто не запрещает создавать на Flash обычную статичную графику или мощные программные пакеты. Попробуйте — вдруг вы станете основоположником нового направления в программировании!

Что дальше?

А дальше начнется знакомство с самим пакетом Macromedia Flash 8. Самое первое и самое предварительное знакомство, но пренебрегать им не стоит. Мы изучим интерфейс этого пакета, научимся управлять его многочисленными окнами и панелями и пользоваться интерактивной справкой.



Глава 2

Пользовательский интерфейс Flash

В этой главе мы изучим основные принципы работы с пакетом Macromedia Flash 8, без знания которых не обойтись. А именно, познакомимся с Flash, что называется, "лицом к лицу", изучим его интерфейс, многочисленные окна и панели, вспомогательные, но отнюдь не второстепенные возможности. И, конечно же, выясним, чем все это богатство поможет нам в работе.

После рассмотрения интерфейса мы узнаем, как рисуются изображения, как создаются, открываются и сохраняются файлы, как вызывается интерактивная справка и пр.

Flash 8 — программа очень мощная, а это значит, что ее интерфейс весьма сложен. (Более того — на взгляд автора, он чересчур усложнен.) Так что у начинающих пользователей Flash первое время рябит в глазах — автор изведал это на своем опыте...

Но у нас не так много времени, чтобы тратить его на пустую болтовню! Вперед, и пусть Flash покорится нам!

Рабочая среда Flash

Сначала мы разберемся с самой рабочей средой Flash и всеми инструментами, которые она предоставляет в наше распоряжение.

Запустить программу в Windows проще простого! Нажмем хорошо знакомую нам кнопку **Start** (Пуск), выберем в меню пункт **Programs** (Программы), далее — пункт **Macromedia** и в появившемся подменю — пункт **Macromedia Flash 8**.

Главное окно

Сразу после запуска программы Flash 8 на экране появляется ее так называемое *главное* окно (рис. 2.1). Рассмотрим его подробнее.

Рис. 2.1. Главное окно Flash

Главное окно служит "вместилищем" для превеликого множества других окон, содержащих как открытые документы Flash, так и различные инструменты, предназначенные для работы с ними. Также в главном окне находится строка *главного меню*, с помощью которого мы сможем получить доступ ко всем инструментам, предлагаемым Flash.

Если во Flash еще не открыт ни один документ, становится видима так называемая *стартовая страница*, отображаемая прямо на фоне главного окна. С помощью стартовой страницы мы можем быстро, не залезая в меню, открыть нужный документ, создать новый либо перейти на сайт фирмы Macromedia (**http://www.macromedia.com**). После открытия хотя бы одного документа стартовая страница пропадет.

ВНИМАНИЕ!

Отображение стартовой страницы отнимает довольно много системных ресурсов, поэтому, если ваш компьютер недостаточно быстр, лучше ее отключить. Для этого найдите в ее левом нижнем углу флажок **Don't show again** и включите его. При следующем запуске Flash стартовая страница уже не будет выводиться.

Окно документа Flash служит для отображения открытого документа. Во Flash можно открыть сколько угодно таких окон. Поэтому говорят, что Flash — программа с *многодокументным интерфейсом*. (Существуют также программы с *однодокументным интерфейсом*, позволяющие открыть одновременно только один документ; это уже знакомый нам Блокнот, Microsoft Paint, Wordpad и др.)

Мы можем перемещать, свертывать и развертывать открытые окна документов, изменять их размеры — в общем, проделывать с ним те же манипуляции, что и с любым другим окном Windows. Единственное исключение: мы не сможем "вытащить" ни одно из этих окон за пределы главного окна программы.

Выше, ниже и правее окна документа находятся *группы панелей* — небольшие окна, которые могут быть либо "приклеены" к краю главного окна, либо "плавать" свободно. В верхней части каждой группы панелей имеется ее *заголовок* — "выпуклая" синяя полоса, на которой написано название панели.

Сами же *панели*, которые объединяют в себе группы, содержат различные инструменты, которыми мы будем пользоваться для работы с графикой Flash, и представлены в виде вкладок. Чтобы переключиться на нужную панель в группе (сделать ее *активной*), достаточно щелкнуть мышью вкладку, на которой написано название этой панели.

Изначально все группы панелей, имеющиеся на экране, "приклеены" к разным краям главного окна. Разработчики из фирмы Macromedia считают, что так будет удобно большинству пользователей Flash, и, похоже, они правы. Однако у нас может быть по этому поводу иное мнение. Поэтому есть возможность отделить какую-либо группу панелей от края главного окна и превратить ее в отдельное окно (рис. 2.2) (так называемая *плавающая* группа панелей). Для этого каждая из них имеет "ручку" для ее "переноски", находящуюся в левой части заголовка и выглядящую как набор мелких темных точек. Просто тащим нужную группу панелей подальше от края главного окна и оставляем ее там.

Ну и, конечно же, мы всегда можем присоединить группу панелей обратно к краю главного окна, чтобы она не загораживала редактируемый документ. Для этого, опять же, достаточно перетащить ее мышью за "ручку"; при этом толстая черная рамка покажет нам, в каком месте главного окна она будет

"приклеена". Кроме того, мы можем таким же образом присоединить одну плавающую группу панелей к другой (рис. 2.3).

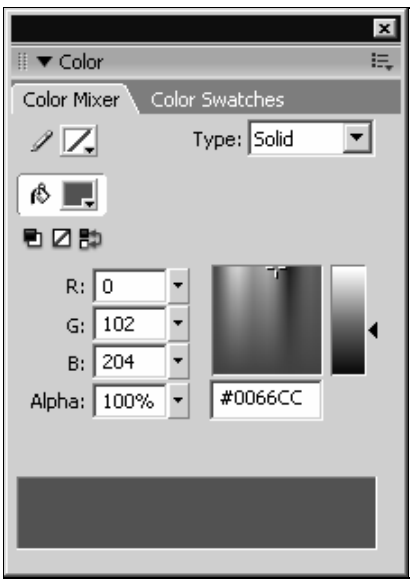


Рис. 2.2. Плавающая группа панелей

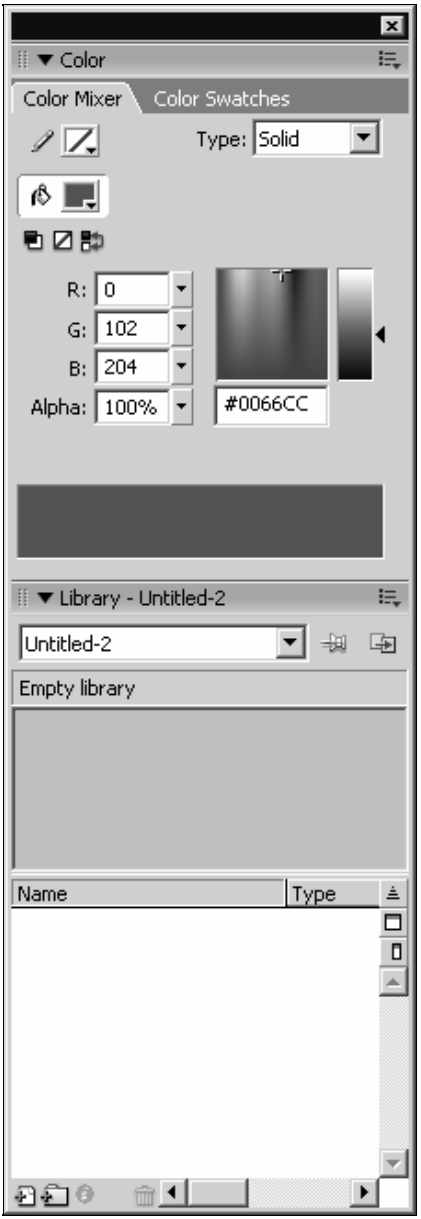


Рис. 2.3. Две соединенные вместе плавающие группы панелей

Мы также можем перемещать группу панелей за ее "ручку" или заголовок ее окна. В последнем случае, правда, не будет работать автоматическое "приклеивание" группы панелей к краю главного окна, что тоже иногда бывает полезно. Можно также временно отключить "приклеивание", удерживая нажатой клавишу <Ctrl> при перемещении группы панелей.

Мы можем изменять размеры плавающих групп панелей (некоторых, не всех). Для этого достаточно перетащить мышью нижний правый угол окна, в котором находится нужная группа.

Чтобы временно уменьшить площадь, занимаемую плавающей группой панелей, мы можем ее сжать, чтобы на экране остался только ее заголовок (рис. 2.4). Для этого нужно щелкнуть по названию группы панелей, находящемуся в ее заголовке. Чтобы развернуть группу до обычного состояния, снова щелкнем по ее заголовку.

Рис. 2.4. "Сжатая" группа панелей

Большинство групп панелей имеют *дополнительное меню*. Оно открывается при щелчке мышью по небольшой кнопке, расположенной в правом верхнем углу группы и имеющей изображение списка из трех позиций и небольшой стрелки, направленной вниз (рис. 2.5). (В сжатом состоянии эта кнопка не видна.) Это меню содержит пункты, предоставляющие доступ к дополнительным инструментам, связанным с активной панелью. В дальнейшем мы часто будем пользоваться этими меню.

Группы панелей всегда располагаются над окнами документов, даже если в данный момент неактивны. Это сделано для того, чтобы мы всегда имели к ним доступ, вне зависимости от того, какое окно сейчас активно. Чтобы убрать какую-либо из этих групп, мы можем "вынести" ее за пределы окна документа или вообще закрыть, раскрыв ее дополнительное меню и выбрав пункт **Close panel group**.

Как только главное окно Flash перестает быть активным (например, когда пользователь переключается в другую программу), все плавающие группы панелей временно скрываются. При активизации Flash они опять появляются на экране.

Теперь обратим внимание на правый край главного окна, туда, где находится множество всяческих групп панелей. Это так называемый *док* — область,

специально для них предназначенная. Док отделен от остального пространства главного окна толстой серой полосой, которую мы можем перетаскивать мышью, изменяя размеры дока.

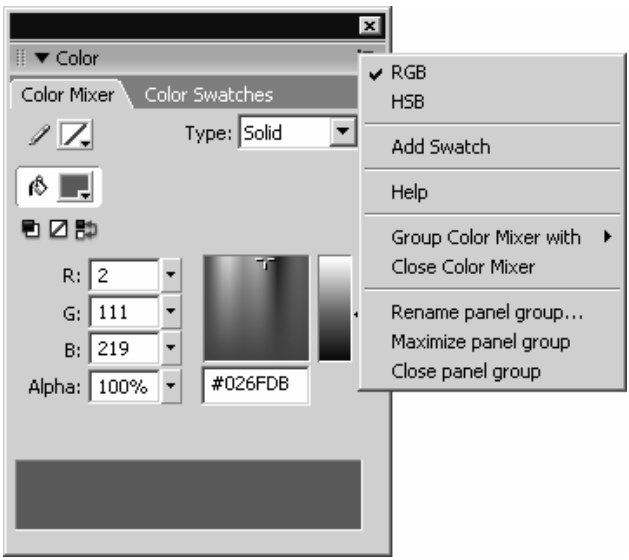


Рис. 2.5. Дополнительное меню группы панелей (открыто)

Мы также можем щелкнуть мышью по довольно приметной кнопке на доке (рис. 2.6), чтобы быстро скрыть его со всеми группами панелей, а потом щелкнуть еще раз и открыть его снова.

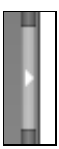


Рис. 2.6. Кнопка скрытия-раскрытия дока

Еще один док располагается в нижней части главного окна (см. рис. 2.1). В нем находится группа с единственной панелью **Properties**, о которой мы еще поговорим. Третий док, в данный момент скрытый, находится в верхней части главного окна; попробуйте переместить туда любую плавающую группу панелей — и вы его увидите.

Некоторые из панелей Flash достойны отдельного упоминания. Они так сильно отличаются от остальных панелей (в частности, имеют постоянные

размеры), что даже носят особое название — *инструментарии*. Таких инструментариев во Flash несколько, но мы пока упомянем три из них:

- главный инструментарий;
- инструментарий документа;
- стандартный инструментарий.

Главный инструментарий Flash (рис. 2.7) расположен вдоль левого края главного окна. Эта небольшая серая панель, вытянутая по вертикали, содержит в себе набор кнопок. Нажимая кнопки, мы можем выбирать те или иные *инструменты*, предлагаемые Flash для рисования или правки графики.

Рис. 2.7. Главный инструментарий

Если хорошенько присмотреться к инструментарию, можно увидеть, что он разделен на четыре *области*. Перечислим их в порядке сверху вниз.

1. *Область основных инструментов* (заголовок **Tools**). Здесь находятся кнопки, предоставляющие доступ ко всем инструментам, что предусмотрены во Flash для рисования и правки уже нарисованного.
2. *Область вспомогательных инструментов* (заголовок **View**). Здесь находятся всего две кнопки, которые мы рассмотрим далее в этой главе.
3. *Область задания цвета* (заголовок **Colors**). Здесь находятся элементы управления, позволяющие нам задавать цвет рисуемой или уже нарисованной графики.
4. *Область модификаторов* (заголовок **Options**). Здесь находятся кнопки, предоставляющие доступ к *модификаторам* — дополнительным режимам, предусмотренным в том или ином выбранном в данный момент инструменте.

Не всегда в данный момент времени в инструментарии присутствуют все четыре области. В частности, область модификаторов (**Options**) в некоторые моменты может быть пуста.

В верхней части главного инструментария находится его "ручка". С ее помощью мы можем перемещать этот инструментарий.

Инструментарий документа (рис. 2.8) позволяет задавать различные режимы отображения открытого в активном окне документа и получать доступ к его составным частям. Он также содержит набор кнопок.

Рис. 2.8. Инструментарий документа

Стандартный инструментарий (рис. 2.9), предоставляющий доступ к файловым операциям (создание, открытие и сохранение документов), операциям с буфером обмена и пр., по умолчанию не выводится на экран. Почему-то программисты из Macromedia сочли его не очень нужным.

Рис. 2.9. Стандартный инструментарий

Стандартный инструментарий не имеет "ручки". Переместить его можно, захватив мышью за пустое пространство, не занятое кнопками.

Управление окнами и панелями Flash

Как мы выяснили, Flash может вывести на экран сразу множество разнообразнейших окон. Как разобраться во всем этом многообразии?

Если в одной программе Flash открыто несколько документов, то переключаться между ними можно, щелкая мышью на заголовках соответствующих им окон. (В принципе, щелкать именно на заголовках необязательно — можно и в других местах области окна, — просто в случае заголовков мы ничего случайно в изображении не изменим.) Этот способ действует, только если ни одно окно документа не максимизировано.

Есть и другой способ переключения между окнами. Откроем меню **Window** и посмотрим в самый его низ. Там будут находиться пункты, имеющие имена, схожие с именами файлов открытых Web-страниц. Просто выбираем нужный пункт — и Flash выводит на передний план (делает *активным*) соответствующее окно документа.

Если же мы раскроем любое из окон документов на весь экран (точнее, на все главное окно), то переключаться между окнами станет еще проще. В этом случае все открытые окна документов будут перечислены в верхней части главного окна в виде вкладок (рис. 2.10) — нам останется только выбрать нужную.

Рис. 2.10. Вкладки, перечисляющие все открытые окна документов

Если нам понадобится держать на виду сразу два или больше окон документов, мы воспользуемся пунктами **Cascade**, **Tile Horizontally** и **Tile Vertically** меню **Window**. Первый из этих пунктов "выкладывает" все открытые окна документов в виде "стопки" в главном окне так, что мы сможем видеть их заголовки и часть содержимого. Второй и третий пункты "выкладывают" в главном окне "мозаику" из окон документов так, чтобы они не перекрывались. Причем второй пункт выкладывает "мозаику" по горизонтали, а третий — по вертикали.

Иногда нужно держать перед глазами несколько частей очень большого изображения. Для этого случая в меню **Window** предусмотрен пункт **Duplicate Window** и эквивалентная ему комбинация клавиш <Ctrl>+<Alt>+<K>. При выборе этого пункта Flash откроет еще одно окно документа, в котором покажет то же изображение, что было открыто в активном окне. Мы можем

воспользоваться пунктом **Tile** меню **Window**, чтобы держать оба этих окна перед глазами.

Теперь давайте откроем подменю **Toolbars** меню **View**. В нем находятся три пункта:

- пункт **Main** выводит стандартный инструментарий;
- пункт **Controller** выводит инструментарий проигрывателя (о нем будет подробно рассказано в *главе 11*);
- пункт **Edit Bar** выводит инструментарий документа.

Если слева от имени одного из этих пунктов стоит галочка, это значит, что соответствующий инструментарий выведен на экран (или, как еще говорят, соответствующий пункт меню "включен"). Чтобы убрать инструментарий, нужно выбрать соответствующий пункт еще раз — и инструментарий исчезнет вместе с галочкой. Такие пункты меню, меняющие свое состояние на противоположное при выборе, называют *пунктами-выключателями*.

Пункт-выключатель **Tools** меню **Window** служит для вывода на экран или скрытия главного инструментария. Вместо выбора этого пункта можно нажать комбинацию клавиш <Ctrl>+<F2>.

Еще мы можем скрыть на время все панели, чтобы без помех просмотреть открытую Web-страницу. Для этого достаточно выбрать пункт **Hide Panels** в меню **Window**, но проще всего, конечно, нажать клавишу <F4>. Чтобы снова вывести скрытые панели на экран, достаточно в меню **Windows** выбрать пункт **Show Panels** или, опять же, нажать клавишу <F4>.

Как мы уже знаем, каждая группа панелей имеет дополнительное меню. Пользуясь пунктами этого меню, мы можем выполнить над данной группой или панелью различные манипуляции:

- закрыть активную панель, выбрав пункт **Close <*название панели*>**;
- закрыть всю группу панелей, выбрав пункт **Close panel group**;
- увеличить размеры группы панелей так, чтобы она заняла весь экран компьютера по вертикали, выбрав пункт **Maximize panel group**;
- переместить выбранную в данный момент панель в другую группу, выбрав пункт **Group <*название панели*> with** и далее в появившемся на экране подменю — пункт, соответствующий названию нужной группы панелей;
- переместить выбранную панель в новую группу, выбрав пункт **Group <*название панели*> with** и далее в появившемся на экране подменю — пункт **New panel group**;
- переименовать группу, выбрав пункт **Rename panel group**. После этого нам остается ввести новое название группы панелей в поле ввода **Name** диа-

логового окна **Rename Panel Group** (рис. 2.11) и нажать кнопку **OK** для его сохранения или **Cancel** для отказа от переименования;

- получить справку по активной панели, выбрав пункт **Help**.

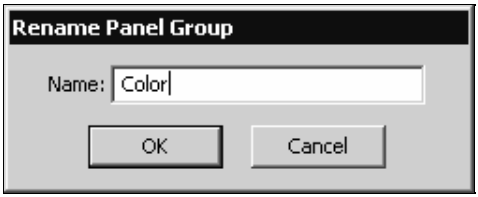


Рис. 2.11. Диалоговое окно **Rename Panel Group**

Мы также можем сохранить созданное нами расположение групп панелей в виде рабочего окружения Flash. Чтобы создать новое рабочее окружение, выберем пункт **Save Current** в подменю **Workspace Layout** меню **Window**. На экране появится диалоговое окно **Save Workspace Layout**, похожее на уже знакомое нам окно **Rename Panel Group** (см. рис. 2.11). Введем в поле ввода **Name** этого окна имя создаваемого рабочего окружения и нажмем кнопку **OK**.

Все созданные нами рабочие окружения присутствуют в виде пунктов в подменю **Workspace Layout** меню **Window**. Чтобы выбрать нужное рабочее окружение, нам будет достаточно выбрать в этом подменю нужный пункт. Изначально там присутствует только пункт **Default**, включающий рабочее окружение Flash по умолчанию.

Для управления рабочими окружениями (переименования и удаления их) служит диалоговое окно **Manage Workspace Layouts** (рис. 2.12). Это окно появится на экране после выбора пункта **Manage** подменю **Workspace Layout** меню **Window**.

Бóльшую часть этого окна занимает список, в котором представлены все созданные нами рабочие окружения. Обратим внимание на то, что рабочее окружение по умолчанию, вызываемое пунктом **Default**, там отсутствует; это значит, что мы не сможем ни переименовать его, ни удалить.

Чтобы переименовать рабочее окружение, нужно выбрать его в списке и нажать кнопку **Rename**. Dreamweaver выведет диалоговое окно **Rename Workspace Layout**, похожее на окно **Save Workspace Layout**. Вводим в поле ввода **Name** этого окна новое имя и нажимаем кнопку **OK**.

Для удаления выбранного в списке рабочего окружения достаточно нажать кнопку **Delete**. После этого на экране появится небольшое окно-предупреждение; нажмем кнопку **Да** для удаления или **Нет** для отказа от него.

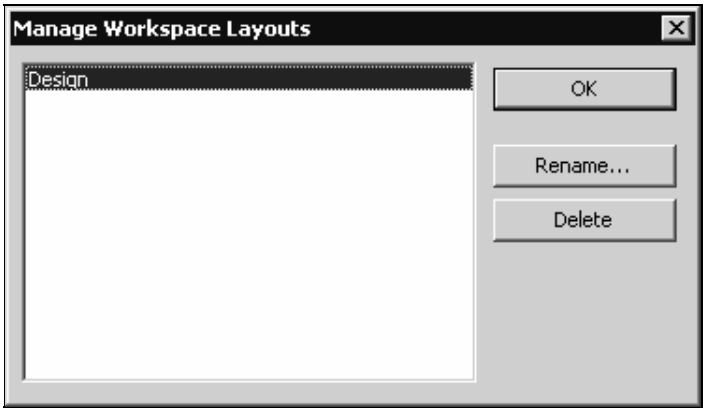


Рис. 2.12. Диалоговое окно **Manage Workspace Layouts**

Для закрытия окна **Manage Workspace Layouts** следует нажать кнопку **OK**. (Хотя по логике следовало бы назвать эту кнопку **Close**.)

Работа с документами в среде Flash

Теперь давайте поговорим о том, как во Flash выполняется работа с документами, а потом плавно перейдем к рассмотрению окна документа и основных принципов создания графики.

Создание нового документа

Перед тем как начать что-то рисовать, нам нужно создать новый документ. Сейчас мы выясним, как это сделать.

Создание нового пустого документа

Создать новый документ Flash, не содержащий никакой графики (пустой), очень просто. Для этого достаточно выбрать в меню **File** пункт **New** или нажать комбинацию клавиш <Ctrl>+<N>.

На заметку

Вполне возможно, что сразу же после запуска Flash новый документ будет создан автоматически. Это поведение зависит от настроек, которые мы рассмотрим в *главе 23*.

На экране появится диалоговое окно **New Document** (рис. 2.13), в котором будет активизирована вкладка **General**.

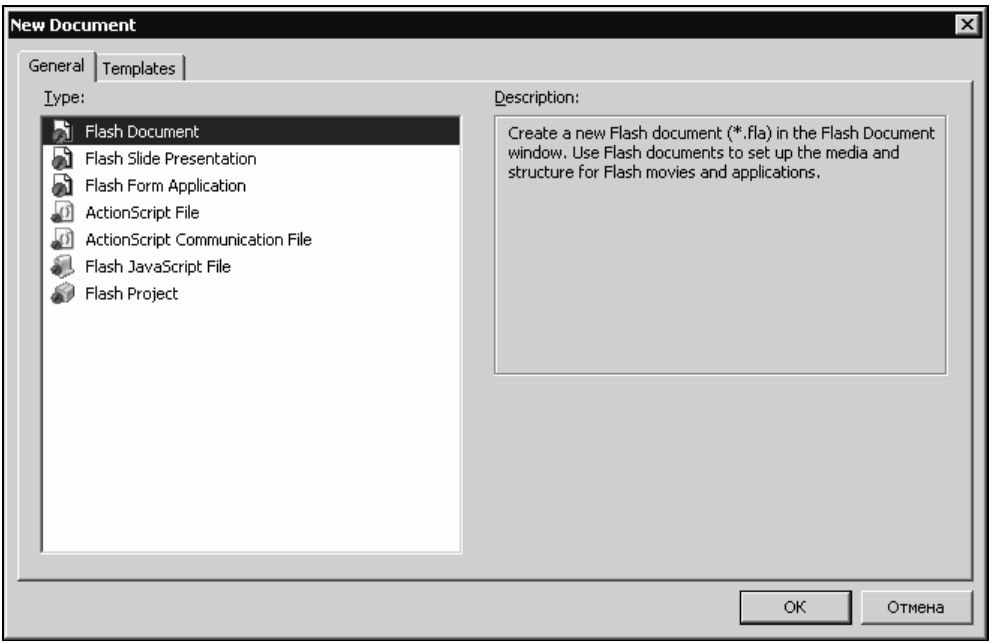


Рис. 2.13. Диалоговое окно **New Document** (вкладка **General**)

В списке **Type** выбирается тип документа, который поддерживается Flash. Первым в этом списке идет пункт **Flash Document**, представляющий обычное изображение или фильм Flash — это нам и нужно. Остальные же пункты создают другие, более специфические документы.

Как только мы выберем в списке **Type** какой-либо пункт, в текстовой области **Description**, расположенной правее списка, появится описание типа документа, за который "отвечает" выбранный пункт. Единственный недостаток текстовой области — она англоязычная. Впрочем, как и сам Flash...

Для создания документа выбранного типа нужно нажать кнопку **OK**. Чтобы отказаться от создания нового документа, следует нажать кнопку **Cancel**.

Создание нового документа на основе шаблона

Также пакет Flash позволяет создать новый документ на основе *шаблона* — "заготовки", созданной опытными дизайнерами по заказу Macromedia. Та-

кой шаблон уже имеет в своем составе некоторые фрагменты изображений, а нам остается только добавить к ним свою графику и текст.

Создать новый документ на основе шаблона так же просто, как и пустой документ. Для этого выберем все тот же пункт **New** в меню **File** или нажмем комбинацию клавиш <Ctrl>+<N> и в диалоговом окне **New Document** переключимся на вкладку **Templates** (рис. 2.14).

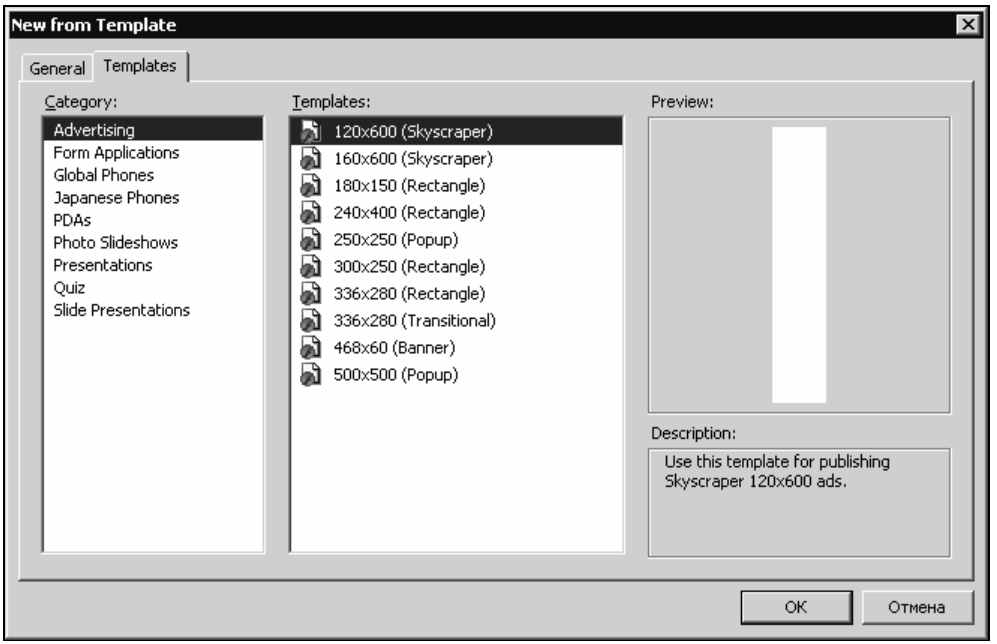


Рис. 2.14. Диалоговое окно **New Document** (вкладка **Templates**)

Все шаблоны Flash организованы в категории. Всего таких категорий восемь. Им соответствуют следующие пункты списка **Category**:

- **Advertising** — рекламные баннеры различных размеров;
- **Form Applications** — Flash-программы различных типов;
- **Global Phones** — изображения и фильмы, предназначенные для воспроизведения на мобильных телефонах;
- **Japanese Phones** — изображения и фильмы, предназначенные для воспроизведения на мобильных телефонах японских производителей;
- **PDAs** — изображения и фильмы, предназначенные для просмотра на карманных компьютерах;

- **Photo Slideshows** — фотогалереи, в которой одно изображение плавно сменяется другим (слайд-шоу);
- **Presentations** — презентации;
- **Quiz** — анкеты, опросные листы и викторины;
- **Slide Presentations** — другая разновидность презентаций.

После того как мы выберем нужную категорию из списка **Category**, в списке **Templates** появится перечень собственно шаблонов, относящихся к этой категории. Например, для категории **Ads** появится список различных форматов баннеров, а для категории **Presentations** — форматов презентаций. Выберем нужный шаблон, после чего в панели просмотра **Preview** появится изображение этого шаблона, а в текстовом поле **Description** — его краткое описание.

Чтобы создать новый документ на основе выбранного шаблона, нажмем кнопку **OK**. Чтобы отказаться от создания нового документа, нажмем кнопку **Cancel**.

Параметры документа

К сожалению, Flash 8 не позволит нам задать параметры (в частности, его размеры и цвет фона) документа прямо при его создании, как было в предыдущих версиях Flash. Для этого нам придется выбрать пункт **Document** в меню **Modify** или нажать комбинацию клавиш <Ctrl>+<J>. После этого на экране появится диалоговое окно **Document Properties** (рис. 2.15).

Поле ввода **Title** и область редактирования **Description** служат для ввода *заголовка* и *описания* документа Flash соответственно. Эти данные потом будут помещены в результирующий файл Shockwave/Flash. Зачем они нужны, непонятно. Так или иначе, вводить их не обязательно.

Поля ввода **width** и **height**, находящиеся в группе **Dimensions**, служат для указания размеров изображения — ширины и высоты соответственно. Значения задаются в текущих единицах измерения; по умолчанию это пикселы. Размеры изображения по умолчанию — 550×400, минимальный размер — 1×1, максимальный — 2880×2880 пикселов.

Ниже находится группа переключателей **Match**, позволяющая быстро задать размеры изображения. Переключатель **Printer** делает размер изображения равным текущему размеру бумаги, заданному в настройках текущего принтера. Переключатель **Contents** задает такой размер изображения, чтобы вся нарисованная графика помещалась в нем, оставляя минимум пустого места. А включенный по умолчанию переключатель **Default** задает размер изображения по умолчанию (см. выше).

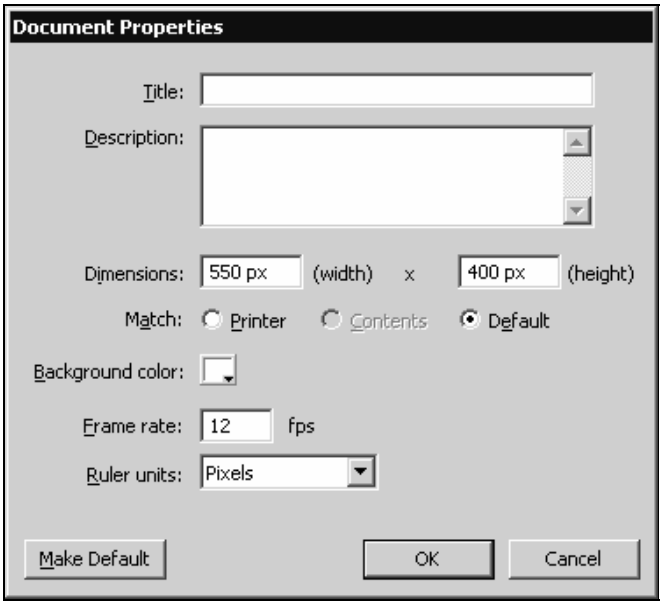


Рис. 2.15. Диалоговое окно **Document Properties**

Еще ниже находится *селектор цвета* **Background color**, позволяющий задать цвет фона изображения. Чтобы сделать это, нужно щелкнуть на селекторе цвета мышью. После этого он откроется, и на экране появится небольшое окно, содержащее набор цветов, доступных для выбора (рис. 2.16). Для выбора нужного цвета достаточно щелкнуть мышью на квадратике, содержащем нужный цвет; окно селектора при этом закроется автоматически.

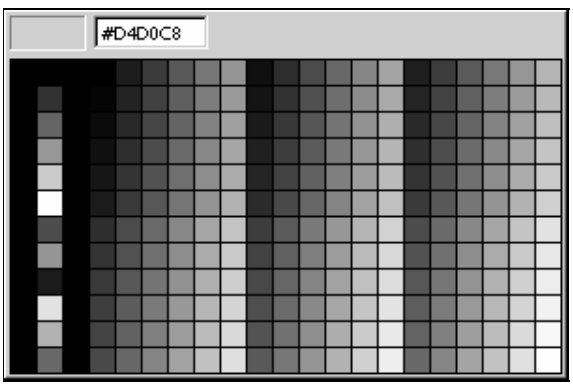


Рис. 2.16. Открытый селектор цвета

По умолчанию цвет фона белый. Рекомендуется его таким и оставить, так как цветные фоны смотрятся очень плохо. В конце концов, человечество уже много лет пишет черными чернилами по белой бумаге и не комплексует по этому поводу.

В поле ввода **Frame rate** задается частота кадров создаваемого фильма. Для фильмов, публикуемых в Интернете, обычно задается частота от 8 до 12 кадров в секунду. Значение по умолчанию — 12 кадров в секунду; рекомендуется его оставить, если только мы не хотим сделать свой фильм компактнее в ущерб качеству. Разумеется, в случае статичных изображений этот параметр роли не играет.

Раскрывающийся список **Ruler units** позволяет задать текущую единицу измерения. Эта единица будет потом использоваться везде, где нам будет нужно задать какие-либо размеры или расстояния, например, при настройке шага линий координатной сетки (о координатной сетке будет рассказано ниже). Этот список предоставляет для выбора шесть пунктов:

- **Inches** — дюймы;
- **Inches (decimal)** — десятичные дюймы;
- **Points** — пункты;
- **Centimeters** — сантиметры;
- **Millimeters** — миллиметры;
- **Pixels** — пикселы (значение по умолчанию).

Задав параметры фильма, нажмем кнопку **OK**, чтобы применить их. Кнопка **Cancel** позволит отказаться от применения новых параметров фильма. А чтобы заданные нами параметры использовались по умолчанию при создании последующих документов Flash, следует нажать кнопку **Make Default**.

Окно документа Flash и работа с ним

Теперь, после создания нового документа, рассмотрим работу в окне документа — самом главном окне Flash. Ведь, если подумать, все эти инструментарии и панели нужны только для обслуживания окна документа, точнее — самого документа, открытого в этом окне. Можно сказать, что окно документа — сердце Flash.

Сначала давайте внимательно посмотрим на окно документа и выясним, из каких частей оно состоит. Рисованием собственно графики и различными вспомогательными инструментами мы займемся чуть позже.

Окно документа

Окно документа Flash показано на рис. 2.17. Рассмотрим его подробнее.

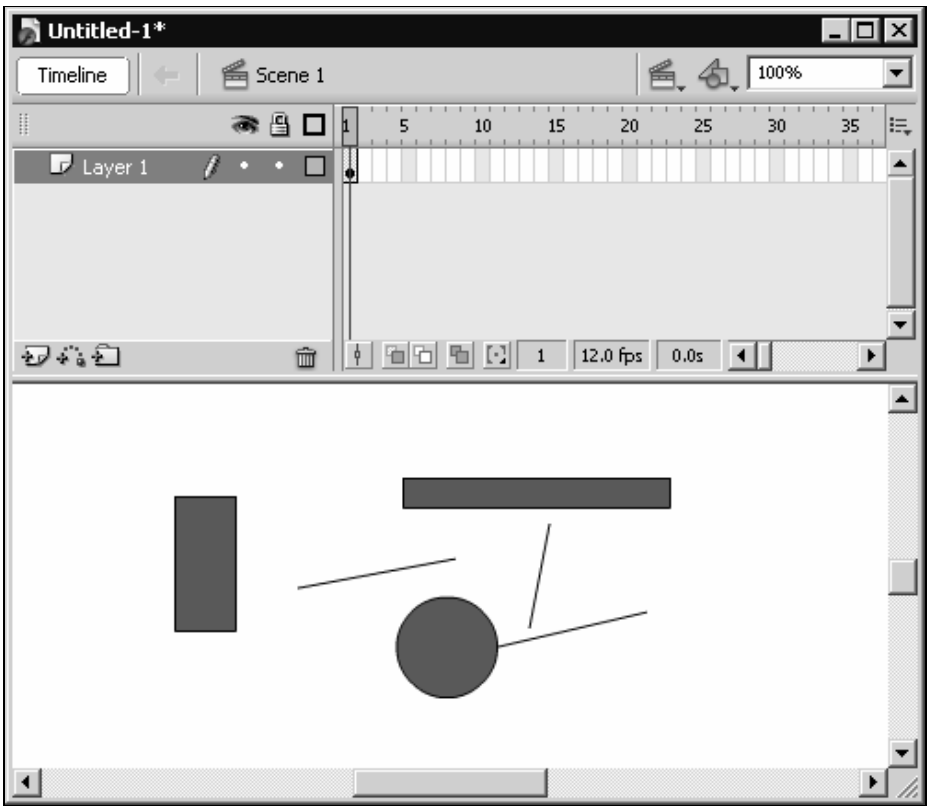


Рис. 2.17. Окно документа Flash

Прежде всего, заметим, что верхнюю половину окна документа занимает некая панель. Это хорошо заметно, если присмотреться к ее верхнему левому углу, — там видна уже знакомая нам "ручка". Так вот, панель эта называется **Timeline** и служит для работы со слоями (*см. главу 5*) и создания анимации (*см. часть III*). Чтобы она нам не мешала на первых порах, мы можем "вытащить" ее за пределы окна документа или вообще скрыть, для чего достаточно отключить пункт-выключатель **Timeline** меню **Window** или нажать комбинацию клавиш <Ctrl>+<Alt>+<T>.

Само же окно документа занято серым полем, в котором находится белый прямоугольник, ограниченный тонкими черными линиями. (На рис. 2.17 он

так велик, что не помещается в окно документа.) Этот прямоугольник схематически представляет наше изображение, и размеры его совпадают с размерами изображения, заданного в параметрах документа. Назовем этот прямоугольник *рабочим листом*, а всю серую область вместе с ним — *рабочей областью.*

Создаваемое изображение рисуется на этом самом рабочем листе (см. рис. 2.17). Конечно, какие-то его фрагменты можно нарисовать и на сером поле, но тогда они не попадут в результирующий файл Shockwave/Flash и, соответственно, не будут выведены на экран при воспроизведении. Тем не менее, в файле документа Flash будут сохранены все фрагменты изображения: и попадающие на рабочую область, и выходящие за ее пределы.

Опытные художники часто пользуются этой особенностью, помещая на серую часть рабочей области графику, которая не должна попасть в результирующее изображение Shockwave/Flash. Это могут быть наброски, варианты какого-либо рисунка и пр.

В верхней части окна документа находится уже знакомый нам инструментарий документа. Инструменты, предлагаемые этим инструментарием, мы рассмотрим далее в этой книге.

Основные принципы работы с графикой Flash

Теперь самое время дать некоторые базовые понятия в области создания графики в среде Flash. Они очень помогут нам в дальнейшем. (Подробнее работа со статичной графикой будет описана в *части II* данной книги.)

Графическое изображение Flash является векторным, а это значит, что оно состоит из примитивов — линий, заливок, фрагментов текста, импортированных растровых изображений и фильмов и пр. Создаются эти примитивы с помощью соответствующих инструментов Flash, кнопки для выбора которых находятся в области **Tools** главного инструментария. Как только мы нажмем нужную кнопку, она останется нажатой — это значит, что данный инструмент выбран.

Кнопки главного инструментария снабжены "говорящими" картинками, позволяющими точно узнать, какой инструмент выбирается той или иной кнопкой. В частности, инструмент "прямоугольник" скрывается за кнопкой с изображением прямоугольника, инструмент "эллипс" — за кнопкой с изображением эллипса и т. п. Есть и еще один способ выяснить, что за инструмент скрывается за кнопкой: достаточно задержать на ней курсор мыши — и на экране появится всплывающая подсказка, кратко описывающая соответствующий инструмент.

Многие простейшие примитивы рисуются следующим образом. Мы выбираем нужный инструмент щелчком по соответствующей кнопке, помещаем

курсор мыши в нужную точку рабочего листа, нажимаем левую кнопку, протаскиваем мышь по рабочей области и отпускаем кнопку. Так, в частности, рисуются прямые линии, прямоугольники и эллипсы, показанные на рис. 2.17.

Щелкая мышью на различных примитивах, составляющих изображение, мы можем *выделять* их. Целые группы примитивов и даже их фрагменты можно выделять протаскиванием мыши: ставим курсор мыши в нужную точку, нажимаем левую кнопку, протаскиваем мышь, пока не выделим все, что нужно, и отпускаем кнопку. Видно, операции выделения во Flash выполняются так же, как и в других программах.

Впоследствии над выделенными фрагментами изображения мы можем выполнить какие-либо действия. Например, пользуясь панелью **Properties**, можно задавать цвет выделенной линии, делать ее пунктирной или извилистой. Также можно перемещать выделенные фрагменты изображения и выполнять над ними различные преобразования. Все это мы изучим в *части II*.

А пока вы можете немного поэкспериментировать с рисованием различных примитивов. Когда закончите, мы поговорим о различных вспомогательных инструментах, любезно предоставляемых нам Flash.

Управление окном документа

Сначала мы выясним, как можно управлять видом изображения в рабочей области.

Мы уже знаем, как убрать панель **Timelines**, занимающую верхнюю часть окна документа. Для этого нужно отключить пункт-выключатель **Timeline** в меню **Window** или нажать комбинацию клавиш <Ctrl>+<Alt>+<T>.

Чтобы скрыть всю графику, которая оказывается за пределами рабочего листа (на сером поле), нужно отключить пункт-выключатель **Work Area** меню **View** или нажать комбинацию клавиш <Ctrl>+<Shift>+<W>. Нужно иметь в виду, что вся эта графика так и останется на своих местах, просто она не будет отображаться.

Для задания масштаба отображения открытого в окне документа изображения служит раскрывающийся список масштаба отображения, находящийся в правой части инструментария документа (рис. 2.18).

Этот список содержит ряд пунктов, с помощью которых задается тот или иной масштаб. Подробно описывать их нет нужды — названия этих пунктов говорят сами за себя (см. рис. 2.18). Пункт **Show Frame** показывает в окне документа весь рабочий лист, автоматически выбирая нужный масштаб. Пункт **Show All** показывает в окне документа только нарисованное на рабочем листе изображение. А пункт **Fit in Window** выполняет то же самое, что и

пункт **Show Frame**, но при этом подбирает такой масштаб, чтобы пользователю не пришлось прокручивать изображение в окне, чтобы рассмотреть его полностью.

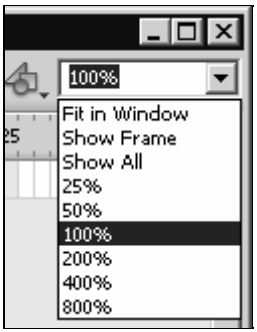


Рис. 2.18. Раскрывающийся список задания масштаба, находящийся в инструментарии документа

Другой способ управления масштабом — использование инструмента **Zoom** (Лупа). Чтобы выбрать инструмент "лупа", нужно щелкнуть кнопку , находящуюся в области **View** главного инструментария, или нажать клавишу <M> или <Z>. После этого курсор мыши должен принять вид небольшой лупы.

Для увеличения масштаба изображения на рабочем листе достаточно щелкнуть по этому изображению. А чтобы уменьшить масштаб, нужно щелкнуть по изображению, удерживая нажатой клавишу <Alt>. При увеличении изображения внутри небольшой лупы, форму которой примет курсор мыши, отображается знак "плюс", при уменьшении — знак "минус".

Для смены режимов увеличения и уменьшения мы также можем воспользоваться кнопками модификаторов **Enlarge** (Увеличение) ( ) и **Reduce** (Уменьшение) ( ). Это *кнопки-переключатели* — включение одной такой кнопки автоматически отключает другую.

Если мы хотим выбрать какой-либо фрагмент изображения, чтобы рассмотреть его поближе, сделаем следующее. Выберем инструмент "лупа", поместим курсор мыши в какую-либо точку на листе Flash, нажмем левую кнопку и, не отпуская ее, протащим мышь так, чтобы захватить в воображаемый прямоугольник выделения нужный фрагмент графики. После этого Flash автоматически рассчитает требуемый масштаб и так позиционирует рабочий лист в окне, чтобы показать выбранный нами фрагмент целиком.

Для увеличения и уменьшения масштаба мы также можем воспользоваться пунктами **Zoom In** и **Zoom Out** меню **View** или комбинациями клавиш <Ctrl>+<=> и <Ctrl>+<-> соответственно. При этом увеличение и уменьшение масштаба производится ступенчато, по пунктам списка масштабов (см. рис. 2.18).

Еще одна возможность изменения масштаба изображения — подменю **Magnification** меню **View**. Оно имеет те же пункты, что и знакомый нам список задания масштаба. Комбинация клавиш <Ctrl>+<1> быстро устанавливает масштаб 100%, <Ctrl>+<4> — 400%, <Ctrl>+<8> — 800%; <Ctrl>+<2> действует аналогично пункту **Show Frame**, а <Ctrl>+<3> — аналогично пункту **Show All**.

Если в окне документа помещается не весь рабочий лист, а только небольшой его фрагмент, мы можем воспользоваться инструментом **Hand** (Рука), чтобы перемещать рабочий лист при просмотре всего его содержимого. Инструмент "рука" включается кнопкой , находящейся в области **View** главного инструментария, или клавишей <H>. После его включения курсор мыши принимает форму руки. Теперь, чтобы переместить рабочий лист, мы просто "захватим" его мышью и перетащим.

Чтобы временно переключиться на другой инструмент, когда включена "рука", нажмем на клавиатуре клавишу пробела и, удерживая ее нажатой, щелкнем нужную кнопку главного инструментария. Выбранный нами инструмент будет активным до тех пор, пока мы не отпустим клавишу пробела. Как только мы это сделаем, Flash автоматически выберет инструмент "рука".

Мы можем задать качество отображения графики в окне документа. По умолчанию она выводится с самым высоким качеством. Однако если наш компьютер не справляется с выводом сложной высококачественной графики, мы можем понизить качество ее вывода. Для этого служит набор из пяти пунктов подменю **Preview Mode** меню **View**. При выборе любого из этих пунктов левее его названия появляется большая черная точка, говорящая о том, что он включен; при этом бывший ранее включенным пункт отключается. Такие пункты называются *пунктами-переключателями*.

Перечислим по порядку все пункты подменю **Preview Mode** меню **View** и опишем режимы вывода изображения, за которые они "отвечают", в порядке повышения качества графики.

1. Режим вывода контуров. Включается выбором пункта **Outlines** или комбинацией клавиш <Ctrl>+<Shift>+<Alt>+<O>. Все примитивы выводятся в виде тонких контуров без всякого *сглаживания* (скрытия "зернистости") и без заливок. В этом режиме графика выводится очень быстро.
2. "Быстрый" режим. Включается выбором пункта **Fast** или комбинацией клавиш <Ctrl>+<Shift>+<Alt>+<F>. Если выбран этот режим, Flash выводит графику полностью, но не выполняет сглаживания графики. Если

вы работаете на маломощном компьютере, выводящем на экран одновременно максимум 256 цветов, используйте этот режим.

3. Режим сглаживания графики. Включается выбором пункта **Antialias** или комбинацией клавиш <Ctrl>+<Shift>+<Alt>+<A>. Если выбран этот режим, Flash выполняет сглаживание контуров графики, но не символов текста. В этом режиме качество вывода графики очень высоко, но символы текста получаются грубыми, зернистыми. Используйте этот режим, если видеоадаптер вашего компьютера может работать в режимах HiColor и TrueColor, в противном случае переключитесь на "быстрый" режим.
4. Режим сглаживания и графики, и символов текста. Включается выбором пункта **Antialias Text** или комбинацией клавиш <Ctrl>+<Shift>+<Alt>+ +<T>. Включен по умолчанию. Качество вывода графики и текста очень высокое. Всегда используйте этот режим, если имеете достаточно мощный компьютер.
5. Режим высококачественной графики. Включается выбором пункта **Full**.

Перечисленные выше режимы вывода графики задают только качество вывода ее на экран. Сама же нарисованная нами графика при этом никак не меняется, и качество ее не искажается.

Средства позиционирования

Средства позиционирования Flash помогут нам точно разместить графические фрагменты на листе. Давайте их рассмотрим.

Самое простое из этих средств — *координатные линейки*. Чтобы вывести их на экран, выберем пункт-выключатель **Rulers** в меню **View** или нажмем комбинацию клавиш <Ctrl>+<Alt>+<Shift>+<R>. Результат этих действий показан на рис. 2.19.

Координатные линейки отображаются вдоль верхней и левой сторон рабочей области Flash, позволяя точно определить соответственно горизонтальную (*X*) и вертикальную (*Y*) координаты. Начало координат находится в верхнем левом углу рабочего листа. По умолчанию линейки проградуированы в пикселах, но мы можем задать другую единицу измерения. Для этого вызовем диалоговое окно **Document Properties** (см. рис. 2.15), выберем нужный пункт в раскрывающемся списке **Ruler unit** и нажмем кнопку **OK**.

Если мы уже включили координатные линейки, то можем воспользоваться такой полезной возможностью, как *направляющие*. Это тонкие линии, которые удобно использовать, например, для точного выравнивания графики на рабочей области или для создания и соблюдения границ вокруг изображения. Эти линии сохраняются в файле документа Flash, но не переносятся в результирующий файл Shockwave/Flash.

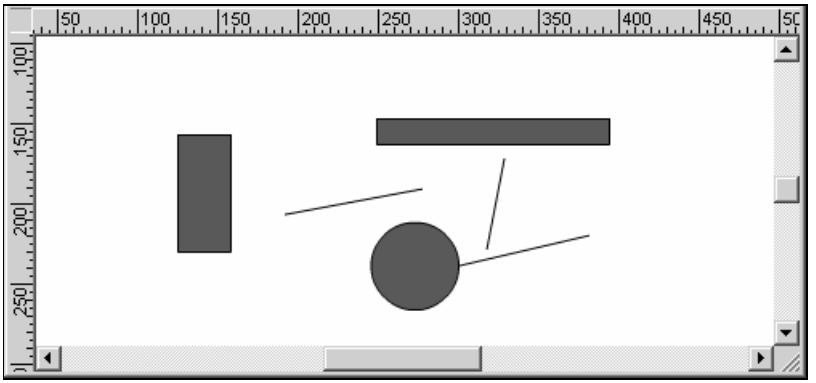


Рис. 2.19. Координатные линейки

Чтобы создать направляющую, нам нужно, прежде всего, вывести на экран координатные линейки. Как это сделать, мы уже знаем. Затем поместим курсор мыши на горизонтальную или вертикальную линейку, в зависимости от того, горизонтальную или вертикальную направляющую мы хотим создать. Далее нажмем левую кнопку мыши и, не отпуская ее, "вытащим" направляющую на рабочую область, после чего отпустим кнопку. Созданные нами направляющие имеют вид тонких линий; две из них — горизонтальная и вертикальная — показаны на рис. 2.20.

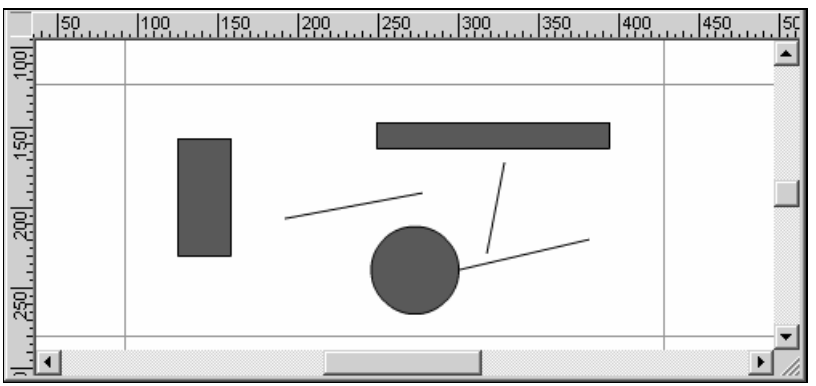


Рис. 2.20. Направляющие (две горизонтальные и две вертикальные)

Мы можем перемещать направляющие мышью и удалять их. Чтобы удалить направляющую, достаточно перетащить ее обратно на линейку. Мы можем временно сделать направляющие неперемещаемыми, *заблокировав* их. Для этого нужно включить пункт-выключатель **Lock Guides** в подменю **Guides**

меню **View** или нажать комбинацию клавиш <Ctrl>+<Alt>+<;>. Существует также возможность временно скрыть все направляющие, отключив пункт-выключатель **Show Guides** в подменю **Guides** меню **View** или нажав комбинацию клавиш <Ctrl>+<;>.

Flash поддерживает такую интересную возможность, как автоматическое *"приклеивание"* рисуемых или перетаскиваемых мышью примитивов к направляющим. Это случается, если "поднести" рисуемый или перемещаемый примитив слишком близко к направляющей. Конечно, это очень удобно, но иногда мешает. Чтобы временно отменить "приклеивание", мы отключим пункт-выключатель **Snap to Guides** в подменю **Snapping** меню **View** или нажмем комбинацию клавиш <Ctrl>+<Shift>+<;>.

Выбор пункта **Edit Guides** подменю **Guides** меню **View** или комбинация клавиш <Ctrl>+<Alt>+<Shift>+<G> выводят на экране диалоговое окно **Guides** (рис. 2.21). С помощью этого диалогового окна мы можем задать некоторые параметры направляющих и выполнить над ними различные действия.

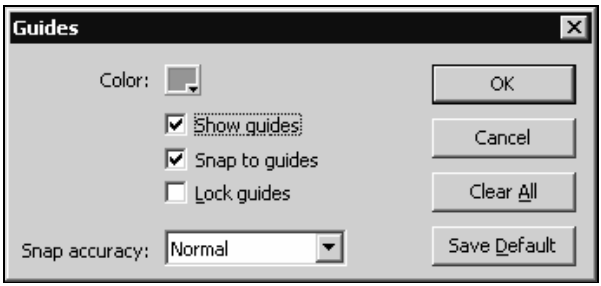


Рис. 2.21. Диалоговое окно **Guides**

В верхней части этого диалогового окна находится селектор цвета, задающий цвет направляющих. По умолчанию направляющие имеют зеленый цвет.

Флажок **Show guides** позволяет скрыть направляющие. По своей функции он аналогичен пункту-выключателю **Show Guides** в подменю **Guides** меню **View**.

Флажок **Snap to guides** включает или выключает "приклеивание" графики к направляющим. Он аналогичен пункту-выключателю **Snap to Guides** в подменю **Snapping** меню **View**.

Флажок **Lock guides** включает или выключает блокировку направляющих, чтобы их нельзя было ни переместить, ни удалить. Его "обязанности" такие же, как и у пункта-выключателя **Lock Guides** в подменю **Guides** меню **View**.

Раскрывающийся список **Snap accuracy** позволяет указать, как близко рисуемый примитив должен быть помещен к направляющей, чтобы быть к ней

"приклеенным". В этом списке доступны три пункта: **Must be close** (должен быть близко), **Normal** (значение по умолчанию) и **Can be distant** (может быть достаточно далеко). Скорее всего, придется поэкспериментировать с различными пунктами этого списка, чтобы подобрать подходящий.

Кнопка **Clear All** позволяет удалить все направляющие разом. Аналогичную задачу, кстати, выполняет пункт **Clear Guides** подменю **Guides** меню **View**.

Задав нужные параметры, следует нажать кнопку **OK**, чтобы применить их. Если мы передумали, нажмем кнопку **Cancel**. Если же мы хотим, чтобы заданные нами параметры использовались при создании других направляющих, т. е. стали значениями по умолчанию, нажмем кнопку **Save Default**.

Еще одно весьма полезное средство позиционирования, предлагаемое Flash, — это *координатная сетка*. Можно сказать, что это набор направляющих, выводимых самим Flash через равные промежутки. Использовать координатную сетку можно как вместе с координатными линейками, так и отдельно от них.

Чтобы вывести на экран координатную сетку, включим пункт-выключатель **Show Grid** в подменю **Grid** меню **View** или нажмем комбинацию клавиш <Ctrl>+<'>. Сама координатная сетка показана на рис. 2.22: видно, что она имеет вид множества тонких линий, наложенных на рабочую область.

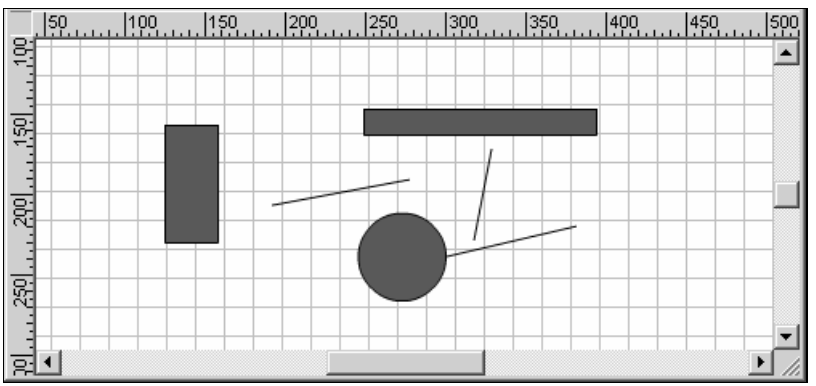


Рис. 2.22. Координатная сетка

При рисовании или правке графики примитивы "приклеиваются" к линиям координатной сетки так же, как и к направляющим. Чтобы временно убрать "приклеивание", нужно отключить пункт-выключатель **Snap to Grid** в подменю **Snapping** меню **View** или нажать комбинацию клавиш <Ctrl>+ +<Shift>+<'>.

Выбор пункта **Edit Grid** подменю **Grid** меню **View** или комбинация клавиш <Ctrl>+<Alt>+<G> выводят на экран диалоговое окно **Grid** (рис. 2.23). С помощью этого диалогового окна мы можем задать некоторые параметры координатной сетки.

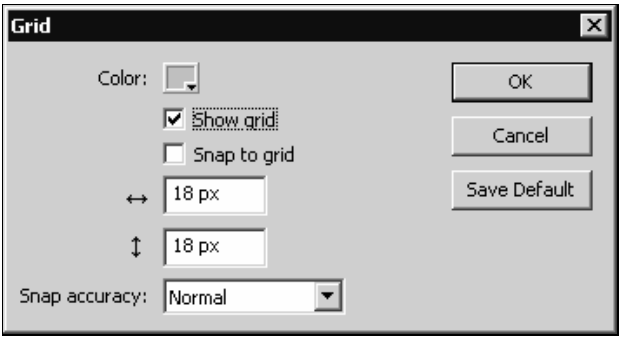


Рис. 2.23. Диалоговое окно **Grid**

В верхней части окна находится селектор цвета **Color**, с помощью которого задается цвет линий координатной сетки. По умолчанию он серый.

Флажок **Show grid** включает или выключает вывод на экран координатной сетки. По своей функции он аналогичен пункту-выключателю **Show Grid** в подменю **Grid** меню **View**.

Флажок **Snap to grid** включает или выключает "приклеивание" графики к линиям координатной сетки. Он аналогичен пункту-выключателю **Snap to Grid** в подменю **Grid** меню **View**.

Следующие два поля ввода служат для установки горизонтального и вертикального шага линий координатной сетки. Эти значения задаются в тех единицах измерения, что были выбраны при задании параметров документа Flash в диалоговом окне **Document Properties** (см. рис. 2.15). По умолчанию эта единица измерения — пиксел.

Раскрывающийся список **Snap accuracy** позволяет указать, как близко рисуемый примитив должен быть помещен к линии координатной сетки, чтобы быть к ней "приклеенным". В этом списке доступны четыре пункта: **Must be close** (должен быть близко), **Normal** (значение по умолчанию), **Can be distant** (может быть достаточно далеко) и **Always Snap** ("приклеивается" всегда, независимо от расстояния). С этим списком, опять же, придется поэкспериментировать.

Выбрав нужные параметры, нажмем кнопку **OK**, чтобы применить их. Если мы передумали, нажмем кнопку **Cancel**. Если же мы хотим, чтобы заданные нами параметры использовались во всех вновь создаваемых документах, т. е. стали значениями по умолчанию, нажмем кнопку **Save Default**.

Работа с файлами

Предположим, что мы создали какой-то документ Flash и теперь хотим сохранить его в файле. Что нужно делать в этом случае?

Чтобы сохранить открытый в активном окне документ в файле документа Flash, достаточно выбрать пункт **Save** в меню **File** или нажать комбинацию клавиш <Ctrl>+<S>. Если сохранение выполняется первый раз, т. е. документ еще ни разу не был сохранен, на экране появится стандартное диалоговое окно сохранения файла Windows, где мы должны будем задать имя файла. Впоследствии Flash будет сохранять документ, не спрашивая имени файла (поскольку мы его уже задали).

Если в работе находятся несколько документов, можно сохранить сразу их все, выбрав в меню **File** пункт **Save All**. Это быстрее, чем выполнять сохранение документов по одному.

Во время работы с документом в нем может накопиться всякий "мусор": сведения о недавно выполненных действиях, импортированные и впоследствии удаленные растровые изображения и фильмы и пр. При сохранении этот "мусор" остается в документе. Чтобы удалить его, нужно выбрать пункт **Save and Compact** в меню **File**. Flash выполнит сохранение документа в файле и при этом удалит из него весь "мусор".

Часто бывает нужно пересохранить документ, т. е. записать его в другой файл с другим именем. Для этого достаточно выбрать пункт **Save As** меню **File** или нажать комбинацию клавиш <Ctrl>+<Shift>+<S>. На экране появится стандартное диалоговое окно сохранения файла Windows, где мы сможем задать имя нового файла.

Открыть сохраненный в файле документа можно, выбрав пункт **Open** в меню **File** или нажав комбинацию клавиш <Ctrl>+<O>. После этого на экране появится стандартное диалоговое окно открытия файла Windows, в котором мы сможем выбрать нужный файл.

Еще один способ открыть файл — это воспользоваться подменю **Open Recent** меню **File**. В этом подменю перечислены в виде пунктов имена нескольких последних открывавшихся нами документов.

Если мы после многочисленных изменений пожелаем загрузить последнюю сохраненную версию документа, то мы выберем пункт **Revert** меню **File**.

Flash тотчас загрузит документ, сохраненный в файле; все прошедшие после последнего сохранения изменения будут потеряны. Пункт **Revert** меню **File** недоступен, если текущий документ ни разу не был сохранен в файле или ни разу не был изменен после его открытия.

Чтобы закрыть документ, нужно выбрать пункт **Close** меню **File** или нажать комбинацию клавиш <Ctrl>+<W>. Хотя проще всего будет закрыть это окно (имеется в виду окно документа, а не окно программы). А если выбрать пункт **Close All** меню **File**, будут закрыты все открытые во Flash документы.

А закрыть саму программу Flash можно четырьмя способами:

- выбрать пункт **Exit** меню **File**;
- нажать комбинацию клавиш <Ctrl>+<Q>;
- щелкнуть кнопку закрытия главного окна;
- нажать системную комбинацию клавиш <Alt>+<F4>.

Если при этом во Flash был открыт, исправлен и не сохранен какой-то документ, Flash предложит нам его сохранить. На экране появится окно-предупреждение с тремя кнопками; кнопка **Да** выполняет сохранение документа, кнопка **Нет** отменяет сохранение, а кнопка **Отмена** запрещает Flash завершить свою работу.

Интерактивная справка Flash

В любом серьезном программном продукте имеется *интерактивная справка*, содержащая полное описание всех его возможностей. Flash 8 — серьезный программный продукт, поэтому он тоже содержит интерактивную справку. Давайте выясним, как ею пользоваться — это очень пригодится нам в дальнейшем.

Содержимое интерактивной справки отображается в панели **Help** (рис. 2.24). Чтобы вывести ее на экран, достаточно включить пункт-выключатель **Flash Help** меню **Help** или нажать клавишу <F1>.

Панель **Help** разделена на две части. Они отделяются друг от друга серой полосой, которую мы можем перемещать мышью, меняя тем самым размеры этих частей.

В левой части панели **Help** расположен древовидный список *статей справки*, составляющих собственно справку. Можно сказать, что справка состоит из статей, как книга — из глав и параграфов. Мы можем свертывать и развертывать "ветви" этого списка, делая двойной щелчок мышью на значках,

находящихся левее названий этих статей. Чтобы просмотреть содержимое нужной статьи, нужно щелкнуть на ее названии.

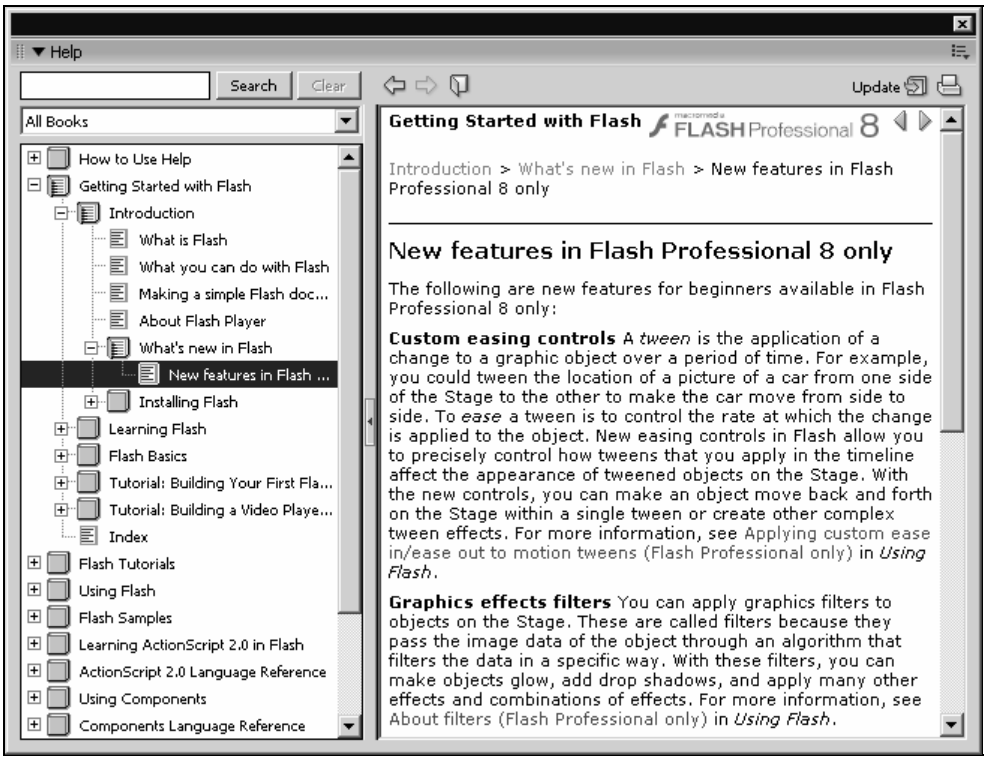


Рис. 2.24. Панель **Help**

Давайте перечислим некоторые "ветви" самого верхнего уровня этого списка, которые будут нам полезны.

- **How to Use Help** — в терминологии фирмы Microsoft это "справка по справке", инструкция по пользованию интерактивной справкой.
- **Getting Started with Flash** — начальные сведения о Flash, перечисление новых возможностей, появившихся во Flash 8, описание установки и запуска пакета и введение в его пользовательский интерфейс.
- **Flash Tutorials** — описания различных приемов работы во Flash, предназначенные для начинающих.
- **Using Flash** — описание всех возможностей Flash по созданию графики и анимации.

- **Flash Samples** — список примеров, которые можно найти в комплекте поставки Flash. Как говорится, лучше один раз увидеть.
- **Learning ActionScript 2.0 in Flash** — руководство по языку программирования сценариев ActionScript.
- **ActionScript 2.0 Language Reference** — справочник по языку ActionScript.

Содержимое выбранной нами статьи выводится в правой части панели **Help**. Ее мы описывать не будем, потому что описывать тут, собственно, нечего.

Если нам понадобится выполнить поиск всех статей, содержащих какой-либо текст, мы введем искомый текст в поле ввода, находящееся в левой части небольшого инструментария панели **Help**, и нажмем кнопку **Search**. После этого в иерархическом списке слева появятся заголовки всех найденных статей, и нам останется только выбрать нужную щелчком мыши.

Чтобы вернуться к обычному иерархическому списку, отображающему все статьи справки, достаточно нажать кнопку , находящуюся в инструментарии панели **Help**.

Также можно выполнить поиск текста в открытой в данный момент статье справки. Для этого нужно выбрать пункт **Find** дополнительного меню панели **Help** или нажать комбинацию клавиш <Ctrl>+<F>. После этого на экране появится диалоговое окно **Find** (рис. 2.25).

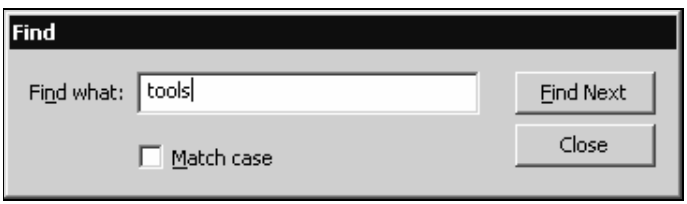


Рис. 2.25. Диалоговое окно **Find**

Искомый текст вводится в поле ввода **Find what**. Если нужно, чтобы Flash выполнял поиск с учетом регистра букв, нужно включить флажок **Match case**. Сам поиск запускается нажатием кнопки **Find Next**.

Если текст, введенный в поле ввода **Find what** окна **Find**, встретится в тексте открытой статьи, он будет выделен. После этого можно снова нажать кнопку **Find Next**, чтобы выполнить поиск следующего вхождения искомого текста в статье. С тем же успехом можно выбрать в дополнительном меню панели пункт **Find Again** или нажать клавишу <F3>, если окно **Find** уже закрыто (для этого достаточно нажать кнопку **Close**).

Кнопка инструментария панели **Help** и пункт **Print** дополнительного меню позволят нам распечатать выбранную статью справки на принтере.

Все просмотренные нами статьи справки Flash сохраняет в особом списке, называемом *историей*. Мы можем "путешествовать" по этому списку, пользуясь кнопками , которые находятся в том же инструментарии. Левая кнопка выполняет перемещение на предыдущую статью в истории, а правая — на следующую.

Есть и еще одна, к сожалению, весьма востребованная, возможность, предоставляемая справкой Flash, — это *обновление* ее статей через Интернет. Разработчики фирмы Macromedia зачастую так спешат с выпуском своих творений на рынок, что не успевают их доделать. Например, предыдущая версия Flash — MX 2004 — содержала неполную справку, и автору пришлось три раза выполнять ее обновление по Сети. (Правда, все недостающие статьи справки были потом включены во второй пакет обновления.)

Для обновления справки нужно подключиться к Интернету, нажать кнопку

и подождать, пока Flash проверит, есть ли на сайте фирмы Macromedia соответствующие обновления. Если таковые обновления есть, Flash выведет на экран небольшое окно-предупреждение. Чтобы запустить сам процесс обновления, следует нажать кнопку **Yes** и подождать, пока все обновления будут загружены, распакованы и установлены. Ждать, скорее всего, придется долго. Когда процесс закончится, Flash выведет окно-предупреждение с соответствующим текстом.

Если же никаких обновлений на сайте Macromedia нет, Flash выведет окно-предупреждение, сообщающее об этом. Его можно тут же закрыть.

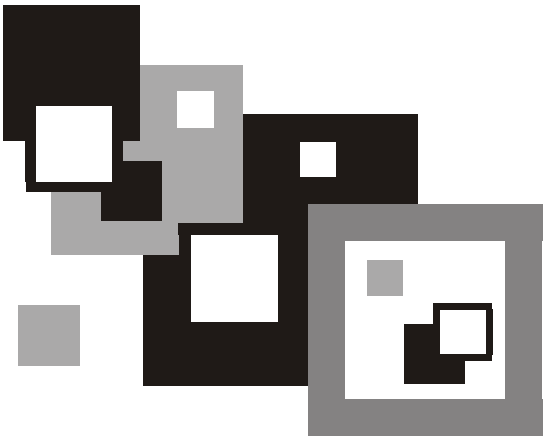
Как уже говорилось, вызвать панель **Help** мы можем также и выбором пункта **Help**, который присутствует в дополнительном меню всех панелей. При этом панель **Help** сразу же покажет нам соответствующую случаю статью, описывающую либо данную панель, либо предоставляемый ею набор инструментов. Собственно, поэтому справка Flash называется интерактивной — она "знает", в чем мы нуждаемся.

Ну что ж! Кажется, вводный курс, посвященный интерфейсу Flash, можно заканчивать. Пора переходить к более серьезным занятиям.

Что дальше?

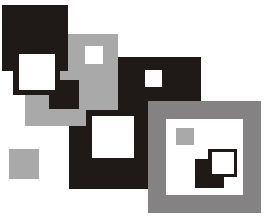
Мы изучили пользовательский интерфейс пакета Macromedia Flash 8. Конечно, мы коснулись не всех возможностей этого пакета, и многие из них будут рассмотрены в последующих главах. Здесь же были описаны только основные моменты, без знания которых в работе с Flash — никуда.

Теперь мы готовы к тому, чтобы нарисовать наше первое изображение. Этим мы и займемся в следующей главе.



ЧАСТЬ II

РАБОТА СО СТАТИЧНОЙ ГРАФИКОЙ



Глава 3

Простейшие операции рисования

Вот и состоялось наше первоначальное знакомство с Flash 8. Можно сказать, что теперь мы официально представлены друг другу. Пора переходить собственно к рисованию.

Но сначала давайте вспомним все, что мы знаем о Flash-графике. И заодно узнаем о ней кое-что новое.

Во-первых, графика Flash является, строго говоря, гибридной, т. е. состоит из набора примитивов (наипростейших геометрических фигур), описываемых математическими формулами, а также (возможно, но не обязательно) — импортированных растровых изображений и фильмов. (Также могут использоваться импортированные звуковые клипы.) Во-вторых, для рисования примитивов используются инструменты рисования Flash, которые можно выбрать с помощью главного инструментария или нажав определенную клавишу на клавиатуре. Сложные изображения могут состоять из множества примитивов, и чем сложнее изображение, т. е. чем больше примитивов оно содержит, тем больше по размерам получится файл документа Flash (а также результирующий файл Shockwave/Flash).

А теперь — внимание! Фигуры, нарисованные во Flash, как правило, состоят из контура и заливки. *Контур* — это набор линий, ограничивающих фигуру, а *заливка* — это ее, так сказать, наполнение. Существуют также фигуры без заливки; к ним относятся, в частности, линии. Также возможно с помощью некоторых инструментов создать одну заливку без контура.

Ну что ж, можно приступать... Запустим Flash, если мы еще этого не сделали. И начнем.

Базовые средства управления цветом

Но сначала давайте поговорим о средствах управления цветом, с помощью которых мы сможем задать цвет рисуемых нами контуров и заливок. Без этого никуда — ведь изобразительное искусство суть искусство цвета, даже если этот цвет всего один — черный.

Для задания цвета контуров и заливок служит уже знакомая нам по *главе 2* область задания цвета, находящаяся в середине главного инструментария и имеющая заголовок **Colors**. Она приведена на рис. 3.1. Рассмотрим ее подробнее.

Рис. 3.1. Область задания цвета в главном инструментарии

Прежде всего, заметим, что в нижней части этой области находятся три небольшие кнопки. Эти кнопки включают особые цветовые режимы, которые мы рассмотрим в *главе 6*. В данный же момент нас больше интересуют два селектора цвета, занимающие бóльшую часть области **Colors**. С их помощью и задаются цвета рисуемой графики.

> ***НА ЗАМЕТКУ***
>
> Селекторами цвета главного инструментария отнюдь не ограничивается арсенал Flash по управлению цветом. Существуют еще несколько панелей, позволяющих задать цвет, но мы подробно рассмотрим их в *главе 6*, полностью посвященной работе с цветом.

Мы видим, что селекторов цветов в области **Colors** два. Верхний из них задает цвет контуров — об этом говорит значок карандаша слева от селектора. Нижний задает цвет заливок, что подтверждает значок опрокинутого ведра.

Нужно иметь в виду, что селекторы цветов, расположенные в области **Colors** главного инструментария, задают цвета только вновь создаваемой графики. То есть если мы сменим цвет линии с черного на синий, следующие начер-

ченные нами линии будут синими, но вычерченные ранее останутся черными. О том, как изменить цвет уже нарисованных примитивов, мы поговорим позже.

Инструменты рисования

Flash 8 предоставляет нам довольно много инструментов рисования. Мы рассмотрим их по очереди, в порядке возрастания сложности.

Инструмент *Line* (*"Линия"*)

Этот инструмент служит для вычерчивания прямых линий. Чтобы выбрать его, следует щелкнуть в главном инструментарии по кнопке или нажать на клавиатуре клавишу <N>. После этого курсор мыши примет вид небольшого крестика — это значит, что теперь мы можем проводить мышью прямые линии.

Делается это так. Помещаем курсор мыши в то место, где должно находиться начало линии, и нажимаем левую кнопку мыши. После этого, не отпуская кнопки, протаскиваем мышь в то место, где у нас будет конец линии. Пока мы буксируем мышь, от начала линии за ее курсором будет тянуться "резиновая" линия, показывающая, как пройдет прямая, если мы прямо сейчас отпустим кнопку мыши. В нужный момент отпускаем левую кнопку мыши — и прямая линия готова.

Мы можем проводить таким образом линии любой длины и под любым углом. Если нам нужно провести линию под углом, кратным 45°, то во время вычерчивания линии мы будем удерживать нажатой клавишу <Shift>. В этом случае рисуемая линия будет автоматически "подгоняться" под подходящий угол.

При рисовании прямых линий мы можем включить модификатор **Snap to Objects** (Приклеивание), для чего достаточно нажать кнопку , находящуюся в области модификаторов главного инструментария. После этого при буксировке мыши, если какой-либо из концов рисуемой линии окажется достаточно близко к уже нарисованному контуру, направляющей или линии координатной сетки, курсор-крестик мыши будет заменен курсором-кружком. Если теперь отпустить кнопку мыши, завершив рисование линии, ее конец будет автоматически "приклеен" к ближайшей линии контура. Flash по мере сил помогает нам.

Чтобы отключить модификатор "приклеивание" (ну не нужна нам такая помощь, не нужна — мы и сами с усами), нужно еще раз щелкнуть по уже на-

жатой кнопке , и она "отожмется". Подобные кнопки, которые могут быть включенными и отключенными, называются *кнопками-выключателями*.

Мы можем изменять цвет линии с помощью верхнего селектора цвета контуров, что находится в области задания цветов главного инструментария. Селектор цвета заливок (нижний) никакого влияния на рисуемые прямые линии не оказывает, так как у линий нет заливок.

Рассмотрим теперь два следующих инструмента, позволяющих рисовать более сложные графические фрагменты, состоящие из нескольких примитивов: "прямоугольник" и "эллипс".

Инструменты *Rectangle* (*"Прямоугольник"*) и *Oval* (*"Эллипс"*)

Эти инструменты очень похожи. Разница проявляется лишь в результате их применения. Поэтому мы и описываем их здесь вместе.

Чтобы выбрать инструмент "прямоугольник", нужно щелкнуть в главном инструментарии по кнопке или нажать клавишу <R>. Если вы почему-то не находите в главном инструментарии эту кнопку, найдите кнопку , щелкните по ней и, не отпуская левой кнопки мыши, подержите ее немного. На экране появится небольшое меню, показанное на рис. 3.2; выберите в нем пункт **Rectangle Tool (R)**. После этого курсор мыши примет вид небольшого крестика — сигнал, что инструмент "прямоугольник" выбран.

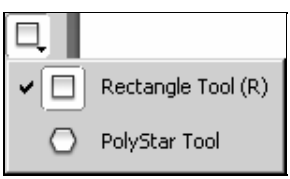


Рис. 3.2. Кнопка с меню

Такие кнопки "с секретом" называются *кнопками с меню*. Обычно для того чтобы открыть это меню, нужно просто щелкнуть по кнопке. Кнопка — единственное исключение из этого правила.

В случае инструмента "эллипс" все намного проще. Щелкнем по кнопке или нажмем клавишу <O>. Курсор мыши также примет вид небольшого крестика.

Прямоугольник рисуется следующим образом. Поместим курсор мыши в то место, где у нас будет находиться его левый верхний угол, и нажмем левую кнопку мыши. После этого, не отпуская кнопку, протащим мышь в то место, где у нас будет находиться правый нижний угол прямоугольника. Пока мы буксируем мышь, Flash отображает "резиновый" прямоугольник, так что мы всегда видим возможный результат наших трудов. Переместив курсор в нужную точку, отпустим левую кнопку мыши — и прямоугольник будет нарисован.

Эллипс рисуется аналогичным образом. Выбираем точку, соответствующую левому верхнему углу воображаемого прямоугольника, в который будет вписан эллипс, нажимаем левую кнопку мыши и буксируем мышь в точку правого нижнего угла этого прямоугольника. В процессе перетаскивания мыши Flash будет показывать нам "резиновый" эллипс. Когда он примет нужную форму, отпускаем левую кнопку мыши и наслаждаемся полученным результатом.

Мы можем рисовать таким образом прямоугольники и эллипсы любого вида. Если же нам нужно нарисовать квадрат или круг, то во время рисования будем удерживать нажатой клавишу <Shift>.

Есть и другой способ рисования прямоугольников и эллипсов. Выбираем соответствующий инструмент, нажимаем клавишу <Alt> и удерживаем ее нажатой во время рисования фигуры. Далее, ставим курсор мыши в точку, где должен находиться центр (именно центр, а не левый верхний угол) прямоугольника или эллипса, нажимаем левую кнопку мыши и растягиваем прямоугольник (эллипс). Когда "резиновый" прямоугольник или эллипс примет нужные размеры, отпускаем кнопку мыши, и фигура готова.

Каждый нарисованный нами прямоугольник и эллипс будет иметь заливку. Цвет заливки задается нижним селектором цвета в области **Colors** главного инструментария. А цвет контура задается, как мы помним, с помощью верхнего селектора цвета.

Пользуясь модификатором **Set Corner Radius** (Скругленные углы), мы можем рисовать прямоугольники со скругленными углами. Нажмем кнопку , находящуюся в области **Options** главного инструментария (инструмент "прямоугольник" должен быть выбран), и на экране появится небольшое диалоговое окно **Rectangle Settings**, показанное на рис. 3.3. В единственном поле ввода **Corner radius** задаем радиус кривизны углов прямоугольника, после чего нажимаем кнопку **OK** для подтверждения ввода данных или **Cancel** для отмены.

Если был введен радиус кривизны углов, отличный от нуля, и нажата кнопка **OK**, все нарисованные за этим прямоугольники будут иметь скругленные углы. Чтобы вернуть прямоугольникам обычные, прямые углы, нужно снова

вызвать на экран диалоговое окно **Rectangle Settings**, ввести в поле ввода **Corner radius** значение 0 и не забыть нажать **OK**.

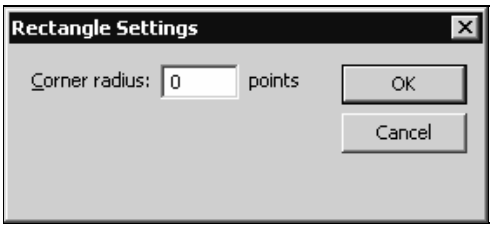


Рис. 3.3. Диалоговое окно **Rectangle Settings**

Кстати, вызвать окно **Rectangle Settings** можно также двойным щелчком на кнопке главного инструментария.

Есть еще один способ создания скругленных углов у прямоугольников. Он состоит в том, что одновременно с буксировкой мыши при рисовании очередного прямоугольника нажимаются клавиши <стрелка вверх> и <стрелка вниз>. Клавиша <стрелка вверх> уменьшает радиус кривизны углов, а клавиша <стрелка вниз> его увеличивает. Изменение радиуса отображается на "резиновом" прямоугольнике, так что мы всегда сможем вовремя остановиться.

Если выбраны инструменты "прямоугольник" или "эллипс", также становится доступным модификатор "приклеивание", описанный в предыдущем разделе.

С простейшими геометрическими фигурами мы покончили. Перейдем к более сложным.

Инструмент *PolyStar* (*"Многоугольник"*)

Этот инструмент позволяет быстро рисовать довольно сложные многоугольники. Он появился в предыдущей версии Flash — MX 2004.

Чтобы выбрать этот инструмент, нажмем кнопку главного инструментария. Если вы почему-то не находите в главном инструментарии эту кнопку, найдите кнопку , щелкните по ней и, не отпуская левой кнопки мыши, подержите ее немного. На экране появится небольшое меню, показанное на рис. 3.2; выберите пункт **PolyStar Tool**. После этого курсор мыши примет вид небольшого крестика — сигнал, что инструмент "многоугольник" выбран.

Чтобы нарисовать многоугольник, поместим курсор мыши в то место, где у нас будет располагаться его центр, и нажмем левую кнопку мыши. После этого, не отпуская кнопку, буксируем мышь, пока не получим многоугольник нужного размера. Как и всегда, Flash поможет нам, отображая "резиновый" многоугольник в процессе перемещения мыши. Чтобы закончить рисование, отпустим левую кнопку мыши.

Мы можем рисовать таким образом многоугольники любой формы. Если же нам нужно нарисовать многоугольник, чьи оси симметрии располагаются строго по горизонтали или вертикали, то во время рисования будем удерживать нажатой клавишу <Shift>.

Каждый нарисованный нами многоугольник будет иметь заливку. Цвет заливки задается нижним селектором цвета в области **Colors** главного инструментария. А цвет контура задается, как мы прекрасно помним, с помощью верхнего селектора цвета.

Мы можем задать различные параметры рисуемых нами многоугольников, воспользовавшись диалоговым окном **Tool Settings** (рис. 3.4). Вызвать это окно можно нажатием кнопки **Options**, находящейся в панели **Properties**; эта кнопка доступна только при выбранном инструменте "многоугольник". Если панель **Properties** на экране отсутствует, следует включить пункт-переключатель **Properties** подменю **Properties** меню **Windows** или нажать комбинацию клавиш <Ctrl>+<F3>.

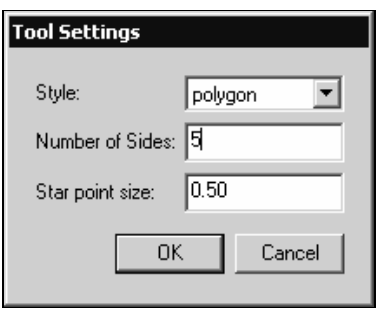


Рис. 3.4. Диалоговое окно **Tool Settings**

Раскрывающийся список **Style** позволит нам задать стиль рисуемого многоугольника. Здесь доступны два пункта: **polygon** (рисование собственно многоугольника) и **star** (рисование звезды).

В поле ввода **Number of Sides** вводится количество углов многоугольника или лучей звезды.

Поле ввода **Star point size** служит для задания степени заостренности лучей звезды. Сюда можно ввести любое значение от 0 до 1; значения, близкие к нулю, вызывают рисование звезды с заостренными лучами, а значения, близкие к единице, — обычного многоугольника. (Собственно, многоугольник от звезды отличается только тем, что значение поля **Star point size** для него близко к единице.)

Введя нужные значения, не забудем нажать кнопку **OK** для их сохранения. После этого все вновь рисуемые нами многоугольники будут иметь новые параметры. Если мы передумали изменять параметры инструмента "многоугольник", нажмем кнопку **Cancel**.

При выбранном инструменте "многоугольник" мы можем пользоваться модификатором "приклеивание".

На рис. 3.5 показаны примеры некоторых фигур, нарисованных с помощью инструмента "многоугольник".

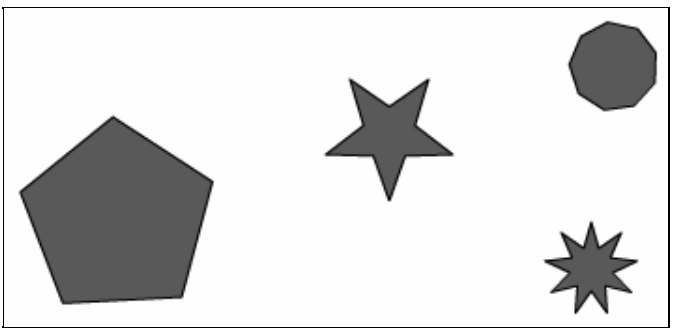


Рис. 3.5. Примеры фигур, нарисованных с помощью инструмента "многоугольник"

Инструмент *Pencil* (*"Карандаш"*)

Этот инструмент позволяет рисовать линии любой (как говорят профессиональные компьютерные художники — *свободной*) формы. Работает он так же, как обычный карандаш, отчего и получил такое название.

Чтобы выбрать инструмент "карандаш", щелкнем в главном инструментарии по кнопке или нажмем клавишу <Y>. Курсор примет вид карандаша.

Чтобы нарисовать линию, поместим курсор в точку, где она должна начинаться, и нажмем левую кнопку мыши. После этого, не отпуская левой кнопки, двигаем мышь по нужной траектории и, когда курсор окажется в

точке, где должен быть конец линии, отпустим кнопку мыши. Flash будет рисовать проводимую нами линию прямо во время движения мыши. Пример такой линии показан на рис. 3.6. Если нам нужно провести строго прямую горизонтальную или вертикальную линию, то во время рисования мы будем удерживать нажатой клавишу <Shift>.

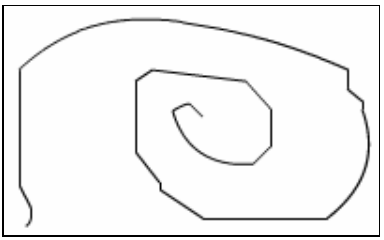


Рис. 3.6. Линия, проведенная с помощью инструмента "карандаш"

Мы можем задавать цвет линии, пользуясь верхним селектором цвета, расположенным в области **Colors** главного инструментария. Нижний селектор цвета никакого влияния на проводимые "карандашом" линии не оказывает, поскольку они не имеют заливки.

Можно заметить, что Flash автоматически сглаживает проведенную нами линию. Человеческая рука — не очень точный инструмент, и Flash это учитывает, устраняя мелкие неровности проводимых вами линий. Однако иногда такая "услужливость" может навредить: например, если нам нужно нарисовать что-то очень точно, со всеми неровностями и шероховатостями. Для регулирования степени сглаживания служит модификатор **Straighten** (Сглаживание линий), связанный с кнопкой , расположенной в области **Options** главного инструментария. Небольшая черная стрелка в правом нижнем углу этой кнопки говорит о том, что при нажатии кнопки на экране должно появиться меню, показанное на рис. 3.7. (То есть это кнопка с меню.)

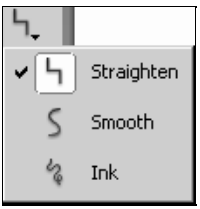


Рис. 3.7. Раскрытое меню кнопки модификатора "сглаживание линий"

Меню, показанное на рис. 3.7, имеет три пункта, задающих степень сглаживания:

- **Straighten** — самое сильное сглаживание, при котором готовые линии получаются прямыми или изломанными, а приближенные подобия геометрических фигур преобразуются в точные фигуры;
- **Smooth** — менее сильное сглаживание; готовые линии почти всегда получаются кривыми;
- **Ink** — минимальное сглаживание.

Если нам нужно нарисовать достаточно точную фигуру из прямых линий, мы выберем сглаживание **Straighten**. Если наша геометрическая фигура должна содержать кривые линии, смело выберем **Smooth**. Для точного рисования сложных контуров следует выбрать степень сглаживания **Ink**.

Инструмент *Pen* (*"Перо"*)

Этот мощнейший инструмент позволит нам рисовать как прямые, так и кривые линии, точно указывая их начальную и конечную точки и радиус кривизны. Кривые линии, рисуемые с помощью этого инструмента, называют также *кривыми Безье* по имени математика, создавшего формулу для описания этих кривых. Кроме того, "перо" предоставляет возможность создания ломаных линий, состоящих из множества прямых и кривых отрезков.

Чтобы выбрать инструмент "перо", щелкнем в главном инструментарии по кнопке или нажмем клавишу <P>. Курсор мыши при этом примет вид чертежного рейсфедера.

Рисование прямых с помощью "пера" выполняется следующим образом. Поместим курсор мыши в точку, где должно находиться начало прямой, и щелкнем левой кнопкой мыши. (Да-да — нужно именно щелкнуть!) В этом месте на листе появится небольшая полая точка. Далее поместим курсор в точку, где должен быть конец прямой, и снова щелкнем мышью. Прямая тотчас будет проведена.

Существует возможность проводить прямые линии под углом, кратным 45°. Для этого нам нужно при втором щелчке удерживать нажатой клавишу <Shift>.

Если мы продолжим щелкать мышью в других точках рабочего листа, то построим ломаную линию, аналогичную показанной на рис. 3.8. Чтобы завершить рисование ломаной, при создании последней ее точки мы сделаем двойной щелчок мышью вместо одинарного. Также для этого можно сразу после последнего щелчка выбрать другой инструмент или щелкнуть где-либо в пустом пространстве рабочего листа, удерживая нажатой клавишу <Ctrl>.

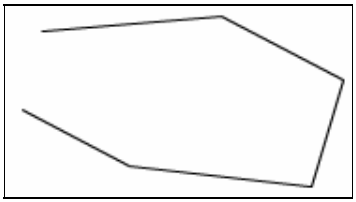


Рис. 3.8. Ломаная линия, нарисованная с помощью инструмента "перо"

Пользуясь инструментом "перо", мы можем рисовать многоугольники любой формы с заливкой. Для этого мы нарисуем ломаную, которая станет контуром этого многоугольника, и замкнем его. Замкнуть его проще всего, подведя курсор к начальной точке многоугольника точек так, чтобы ниже и правее ее появился небольшой кружок, и щелкнуть левой кнопкой мыши. После этого Flash завершит создание контура и поместит в него заливку (рис. 3.9).

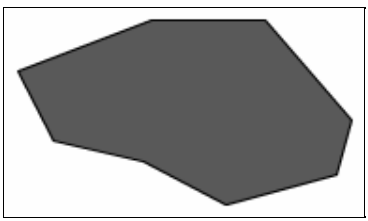


Рис. 3.9. Созданный с помощью инструмента "перо" многоугольник с заливкой

Как с помощью "пера" создаются прямые линии, мы выяснили. Пора приниматься за кривые.

Начальная точка кривой линии создается точно так же, как начальная точка прямой. После этого выберем место на рабочем листе, где будет находиться конечная точка кривой, и щелкнем левой кнопкой мыши, но отпускать ее не будем. Вместо этого переместим курсор в каком-либо направлении, лучше — перпендикулярном направлению получающейся прямой линии. Мы увидим, как при этом изменится форма создаваемой кривой (рис. 3.10).

Мы можем действовать немного по-другому. Выберем на листе точку, где должно находиться начало кривой, поставим в нее курсор, нажмем левую кнопку и, не отпуская ее, переместим мышь. После этого отпустим кнопку мыши, переместим курсор на место, где должен быть конец кривой, и

щелкнем левой кнопкой. Мы получим ту же самую кривую — конечно, не в точности такую же, но аналогичную.

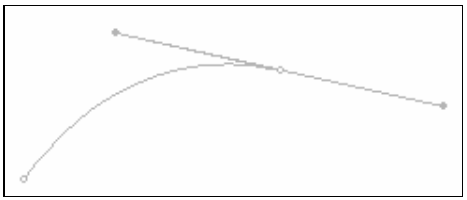


Рис. 3.10. Рисование кривой с помощью инструмента "перо"

И, наконец, можно совместить оба эти подхода. Поэкспериментируйте и посмотрите на получившиеся результаты — это лучший способ понять, как работает инструмент "перо".

Главное правило таково: мы должны перемещать мышь в том направлении, в каком создаваемая кривая должна быть выгнута. Если мы будем при этом удерживать нажатой клавишу <Shift>, курсор мыши будет перемещаться под ближайшим углом, кратным 45°. *Касательная* — прямая линия, заканчивающаяся темными точками, рисуемая Flash при создании кривой — поможет нам выбрать точную форму кривой.

Основное условие при создании кривых Безье — не жалеть времени на экспериментирование с инструментом "перо". В конце концов, практика в некоторых случаях может дать несравнимо больше, тем теория. Перефразируя известную пословицу, можно сказать, что лучше один раз что-то сделать самостоятельно, чем сто раз услышать, как это делается.

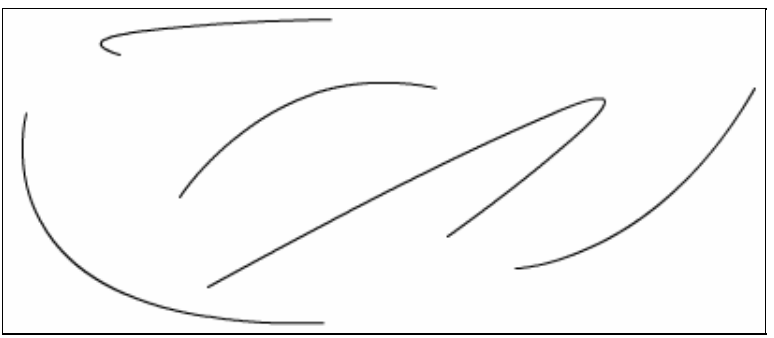


Рис. 3.11. Примеры геометрических фигур, созданных с помощью инструмента "перо"

Поэтому автор призывает вас: экспериментируйте! Посмотрите на рис. 3.11 — там показаны некоторые примеры сложных геометрических фигур, созданных из кривых Безье с помощью "пера". Попробуйте создать аналогичные фигуры.

Остается добавить, что мы можем задавать цвет линии, пользуясь верхним селектором цвета, расположенным в области **Colors** главного инструментария. Нижний селектор цвета, задающий цвет заливки, в этом случае влияния не оказывает.

Инструмент *Paint Bucket* (*"Ведро с краской"*)

Все рассмотренные нами ранее инструменты были предназначены, в основном, для рисования контуров. Конечно, при использовании некоторых из них также создавались и заливки, но это была "личная инициатива" Flash, искренне желающего нам помочь. Теперь же настала пора заняться заливками вплотную. Так вот, инструмент "ведро с краской" служит именно для создания заливок.

Этот инструмент ведет себя аналогично реальному ведру с краской: если "опрокинуть" его над замкнутым контуром, краска "разольется" и заполнит его внутренность. Если же контур имеет "прореху" (например, мы забыли его замкнуть), заливка создана не будет. Как видите, Flash не допустит, чтобы "краска" "вытекла" наружу и испортила изображение.

Чтобы выбрать инструмент "ведро с краской", щелкнем в главном инструментарии по кнопке или нажмем клавишу <K>. Курсор мыши при этом примет вид опрокинутого ведра с краской.

Использовать этот инструмент очень просто. Выберем нужную замкнутую фигуру, проследим за тем, чтобы она была действительно замкнута (иначе заливка не будет создана), и щелкнем на ней левой кнопкой мыши. Flash тотчас создаст заливку, заполняющую эту фигуру. Пример такой заливки показан на рис. 3.12.

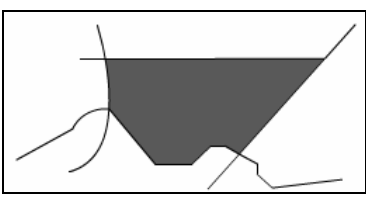


Рис. 3.12. Пример заливки, созданной при помощи инструмента "ведро с краской"

Выбор цвета заливки производится с помощью нижнего селектора цвета в области **Colors** главного инструментария. Цвет, заданный с помощью верхнего селектора, в этом случае роли не играет.

Если контур нарисованной нами фигуры в некоторых местах не замкнут, не стоит огорчаться. Flash автоматически устранит эти огрехи, если, конечно, они действительно невелики — в противном случае Flash вообще не создаст заливку. Размер разрыва, которого "не замечает" Flash, можно задать с помощью модификатора **Gap Size** (Размер разрыва). Этот модификатор имеет вид кнопки с меню (рис. 3.13) и расположен в области **Options** главного инструментария.

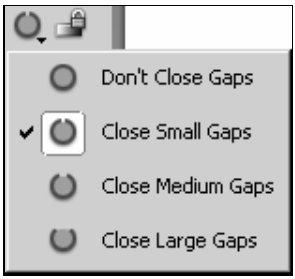


Рис. 3.13. Кнопка-модификатор "размер разрыва" с раскрытым меню

Меню этого модификатора предлагает для выбора четыре пункта:

- **Don't Close Gaps** — весь контур должен быть замкнут, иначе заливка создана не будет;
- **Close Small Gaps** — Flash закрывает только маленькие разрывы;
- **Close Medium Gaps** — Flash закрывает промежутки средних размеров;
- **Close Large Gaps** — Flash закрывает большие разрывы.

И еще: если мы попытаемся закрасить сам рабочий лист, выбрав "ведро с краской" и щелкнув на пустом пространстве листа, заливка создана не будет. Flash может создать заливку только внутри замкнутого контура — нужно это помнить. Чтобы изменить цвет фона изображения, следует воспользоваться диалоговым окном **Document Properties**. Работа с ним была подробно описана в *главе 2*.

Инструмент *Brush* (*"Кисть"*)

Этот инструмент ведет себя аналогично реальной кисти. С его помощью мы можем наносить мазки различной формы и толщины. Причем эти мазки

представляют собой не что иное, как заливки фигурной формы без видимых контуров. Несколько таких мазков показано на рис. 3.14.

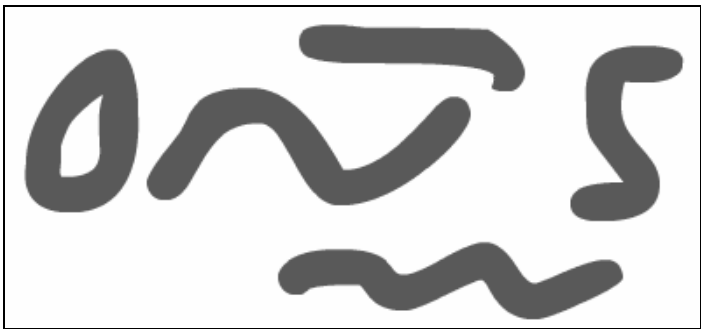


Рис. 3.14. Мазки, проведенные инструментом "кисть"

Чтобы выбрать инструмент "кисть", щелкнем в главном инструментарии по кнопке или нажмем клавишу <B>. Курсор при этом может принимать разные формы, в зависимости от настроек, заданных с помощью модификаторов (см. ниже).

Инструмент "кисть" используется аналогично "карандашу". Мы ставим курсор в точку, где должен начинаться мазок, нажимаем левую кнопку, двигаем мышь, рисуя сам мазок, и, в конце концов, заканчиваем рисование, отпустив левую кнопку мыши.

Если нам нужно нарисовать строго прямой горизонтальный или вертикальный мазок, то во время рисования мы будем удерживать нажатой клавишу <Shift>. Это правило справедливо для многих инструментов Flash.

Цвет мазка выбирается с помощью нижнего селектора цвета, расположенного в области **Colors** главного инструментария. Это вполне логично, так как "кисть" часто используется для создания заливок. Цвет контуров, задаваемый верхним селектором цвета, при этом роли не играет.

Кроме цвета мазка, Flash предоставляет нам возможность выбора формы и размера кисти. Для этого служат модификаторы **Brush Shape** (Форма кисти) и **Brush Size** (Размер кисти), расположенные, как и все остальные модификаторы, в области **Options** главного инструментария. Эти модификаторы представляют собой обычные раскрывающиеся списки, в которых в виде графических изображений представлены все доступные во Flash формы и размеры "кисти". Модификатор "размер кисти" в раскрытом виде показан на рис. 3.15, а модификатор "форма кисти" — на рис. 3.16. Видно, что возможности выбора достаточно велики.

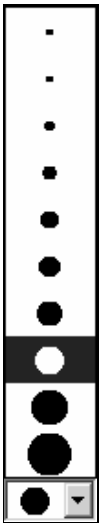


Рис. 3.15. Список модификатора "размер кисти" в открытом виде

Рис. 3.16. Список модификатора "форма кисти" в открытом виде

Как только мы выберем с помощью вышеупомянутых модификаторов форму и (или) размер "кисти", форма и (или) размер курсора мыши изменится. Таким образом, мы всегда будем знать, какой "кистью" мы работаем.

Еще один модификатор — **Paint Mode** (Режим закраски) — позволяет задать режим работы "кисти", т. е. определяет, будет ли "кисть" закрашивать линии или будет ограничиваться лишь заливками. Этот модификатор имеет вид кнопки с меню (рис. 3.17).

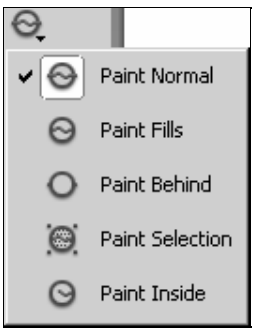


Рис. 3.17. Кнопка модификатора "режим закраски" с раскрытым меню

В меню модификатора "режим закраски" доступно пять пунктов:

- **Paint Normal** — обычное рисование, когда закрашиваются любые линии и любые заливки (рис. 3.18);

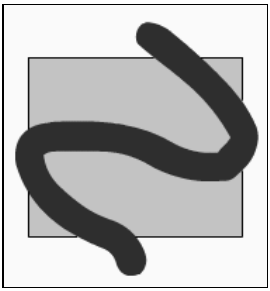


Рис. 3.18. Результат применения режима закраски **Paint Normal**

- **Paint Fills** — закрашиваются только заливки и пустые области, линии не закрашиваются (рис. 3.19);

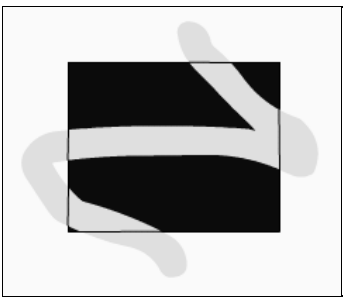


Рис. 3.19. Результат применения режима закраски **Paint Fills** (обратите внимание — линии контура остались незакрашенными)

- **Paint Behind** — закрашиваются только пустые области, линии и заливки не закрашиваются (рис. 3.20);
- **Paint Selection** — закрашивается только выделенная с помощью инструмента "стрелка" заливка (об инструменте "стрелка" см. далее); невыделенные заливки, любые линии и пустые области остаются незакрашенными (рис. 3.21);
- **Paint Inside** — закрашивается только заливка, находящаяся либо в пределах, либо за пределами замкнутого контура. То, что находится по другую

сторону этого контура, не закрашивается. Линии и пустые области также не закрашиваются (рис. 3.22).

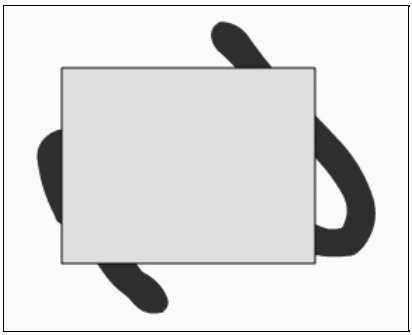


Рис. 3.20. Результат применения режима закраски **Paint Behind**

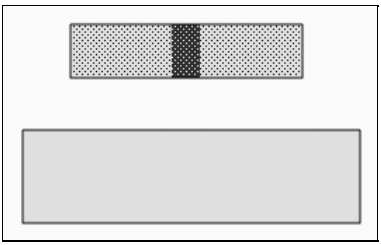


Рис. 3.21. Результат применения режима закраски **Paint Selection** (заливка верхнего прямоугольника была выделена с помощью инструмента "стрелка", а заливка нижнего — нет)

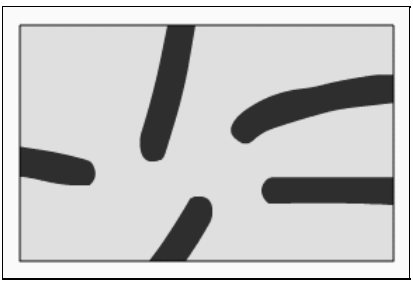


Рис. 3.22. Результат применения режима закраски **Paint Inside** (начало каждого мазка находилось внутри прямоугольника, а конец — вне его)

Правка графики

Итак, создавать графику мы научились. Теперь давайте выясним, как ее править.

Под правкой графики мы будем понимать изменение ее формы, а также ее полное или частичное удаление. Как добавить новые линии или фигуры, мы уже прекрасно знаем — достаточно воспользоваться одним из рассмотренных выше инструментов рисования. Кроме того, редактирование графики также подразумевает изменение ее цвета, но об этом мы подробнее поговорим в *главе 6*.

Для изменения графики Flash предоставляет четыре мощных инструмента: "стрелка", "белая стрелка", "лассо" и "ластик". Первые два инструмента предназначены для выделения и правки элементов изображения, третий — для "вырезания" из фигур "лоскутков" произвольной формы, а четвертый — для полного или частичного удаления фигур или их частей. Пользуясь этими четырьмя инструментами, мы сможем проделывать со своей графикой все что угодно.

Но прежде чем править графику, ее нужно выделить, указав тем самым, какой именно фрагмент нашего изображения мы хотим исправить. Давайте выясним, как это делается.

Выделение графики

Для *выделения* фрагментов графики служит очень мощный инструмент **Selection Tool** (Стрелка[1]). Чтобы его выбрать, нужно щелкнуть по кнопке в главном инструментарии или нажать клавишу <V> на клавиатуре. Курсор мыши после этого примет вид небольшой черной стрелки.

Как и в других аналогичных Windows-приложениях, для того чтобы выделить какой-либо фрагмент изображения, нужно щелкнуть на нем мышью. После этого он станет выделенным, о чем недвусмысленно скажет штриховка из черных точек (рис. 3.23), которой заполняются выделенные фигуры. Однако Flash вносит в этот несложный процесс свои коррективы.

Геометрическая фигура, нарисованная с помощью инструментов Flash, фактически состоит из двух частей. Первая часть — это контур, который образуют внешние линии фигуры. Контур может быть замкнутым или незамкнутым.

[1] Термин автора. Более распространенным является перевод "Инструмент выделения". — *Прим. ред.*

Если контур замкнут, в составе фигуры может присутствовать и вторая часть — заливка. (Выше уже говорилось, что контур и заливка, — совершенно разные виды примитивов.) Например, из контура и заливки состоят нарисованные нами прямоугольники и эллипсы. Конечно, бывают фигуры, состоящие из одного контура, например, линии, но разговор сейчас не о них.

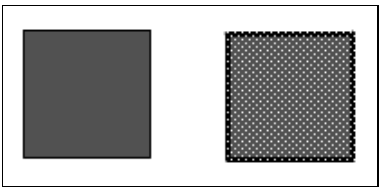


Рис. 3.23. Слева — невыделенный прямоугольник, справа — выделенный

Вышесказанное означает, что можно выделить отдельно контур какой-либо фигуры и произвести с ним какие-то действия, не затрагивая заливку. И наоборот, выделив заливку этой или другой фигуры и что-либо с ней сделав, мы никак не затронем контур. Более того, мы даже можем отделить заливку от контура, фактически превратив их в две абсолютно независимые фигуры.

Чтобы выделить контур фигуры, следует щелкнуть на нем мышью (предполагается, что мы уже выбрали инструмент "стрелка"). Конечно, это довольно трудно, особенно если он выполнен тонкими линиями, но мы можем изменить масштаб отображения (как это сделать, было рассказано в *главе 2*). Круг с выделенным контуром показан на рис. 3.24.

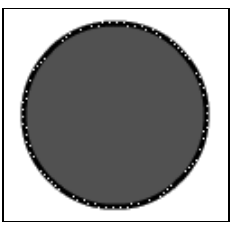


Рис. 3.24. Круг с выделенным контуром

С контурами, состоящими из нескольких отрезков, дело обстоит несколько сложнее. Если мы щелкнем мышью по одному из таких отрезков, будет выделен только он, а остальные отрезки останутся невыделенными.

Выделить заливку совсем просто — как правило, она значительно "объемистее" любых линий, и промахнуться мимо нее невозможно. Просто щелкнем на ней — результат этого действия показан на рис. 3.25.

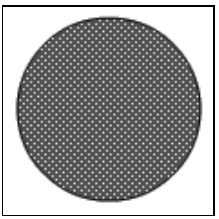


Рис. 3.25. Круг с выделенной заливкой

Но как поступить, если нужно выделить сразу все отрезки, из которых состоит контур, или сразу и контур, и заливку? Есть целых три способа сделать это.

Во-первых, практически любое Windows-приложение, работающее с графикой, предоставляет возможность *выделения перетаскиванием* мыши. Поместим курсор мыши в какую-либо точку на листе Flash, нажмем левую кнопку мыши и, не отпуская ее, протащим мышь так, чтобы захватить в получающийся воображаемый прямоугольник (*прямоугольник выделения*) нужный нам фрагмент изображения.

Во-вторых, мы можем воспользоваться одной уникальной возможностью Flash. Поместим курсор мыши на контур или заливку (лучше на заливку) и выполним двойной щелчок левой кнопкой мыши (*выделение двойным щелчком*). Этот способ применяется для выделения определенного фрагмента изображения и всех примыкающих к нему фрагментов, будь то линии или заливки.

В-третьих, мы можем выделить один примитив, щелкнув на нем мышью, а потом нажать клавишу <Shift> и, не отпуская ее, щелкнуть на остальных примитивах, которые хотим выделить. Этот способ выделения тоже поддерживает большинство Windows-приложений, работающих с графикой. Называется он *множественное выделение*.

Если мы в процессе множественного выделения (клавиша <Shift> еще нажата) щелкнем уже по выделенному фрагменту, он перестанет быть выделенным. Мы можем воспользоваться этим свойством для ускорения процесса, выделив большую группу фрагментов перетаскиванием мыши, нажав <Shift> и щелчками мыши сняв выделение с ненужных фрагментов. Опытные пользователи Flash очень часто так поступают.

Как бы то ни было, лучше всего попрактиковаться с выделением различных фрагментов графики разными способами. Нарисуйте несколько фигур посложнее и поэкспериментируйте с ними. Приобретенные навыки очень помогут вам в дальнейшем.

Напоследок осталось сказать совсем немного. Если мы хотим выделить все, что находится у нас в рабочей области, мы выберем пункт **Select All** меню **Edit** или контекстного меню рабочей области или нажмем комбинацию клавиш <Ctrl>+<A>. Снять же выделение со всех элементов позволяет выбор пункта **Deselect All** меню **Edit** или контекстного меню рабочей области или комбинация клавиш <Ctrl>+<Shift>+<A>. Снять выделение можно также, щелкнув в пустом пространстве рабочей области, не занятом графикой.

Иногда бывает нужно скрыть выделение фрагментов, т. е. сделать так, чтобы штриховка, которой помечаются выделенные фрагменты, не показывалась на экране. Для этого достаточно отключить пункт-выключатель **Hide Edges** меню **View** или нажать комбинацию клавиш <Ctrl>+<H>. Впоследствии мы можем снова включить этот пункт, чтобы вновь вывести на экран штриховку.

Фрагментация и слияние графики

Прежде чем мы приступим к рассмотрению способов правки уже созданной графики, поговорим еще об одной особенности Flash. Это так называемая *фрагментация* графики. Суть ее в том, что монолитное, казалось бы, изображение в определенных случаях "разваливается" на множество независимых фрагментов. Иногда это бывает полезно, иногда — не очень.

Рассмотрим классический пример фрагментации. Нарисуем эллипс и рассечем его прямой линией. У нас получится фигура, показанная на рис. 3.26.

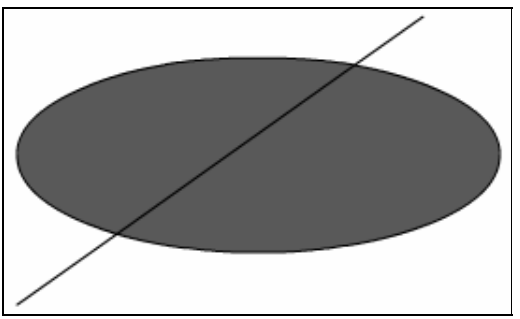


Рис. 3.26. Изначальная фигура, нарисованная для демонстрации эффекта фрагментации графики во Flash

После этого выберем инструмент "стрелка" и щелкнем в пустом пространстве рабочей области, чтобы снять любое выделение. И щелкнем на какомлибо из концов линии, выступающих за пределы перечеркнутого эллипса. В результате получим картину, показанную на рис. 3.27. Хорошо видно, что вместо того чтобы выделить всю линию, Flash выделил только ее фрагмент, "отрезанный" контуром эллипса. А это значит, что выделенный нами конец линии теперь представляет собой совершенно независимый фрагмент.

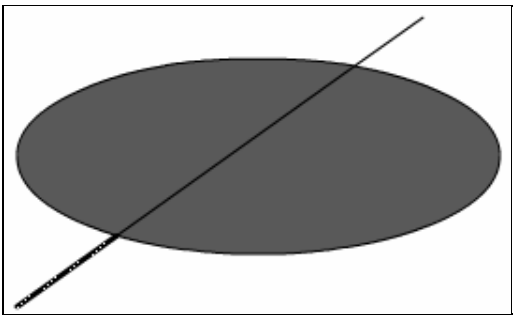


Рис. 3.27. При щелчке мышью будет выделен только фрагмент линии, а не вся линия

Более того: линия, рассеченная эллипсом натрое (проверьте — это так!), и сама рассекла эллипс на четыре части (две половинки контура и две половинки заливки). На рис. 3.28 это очень хорошо видно.

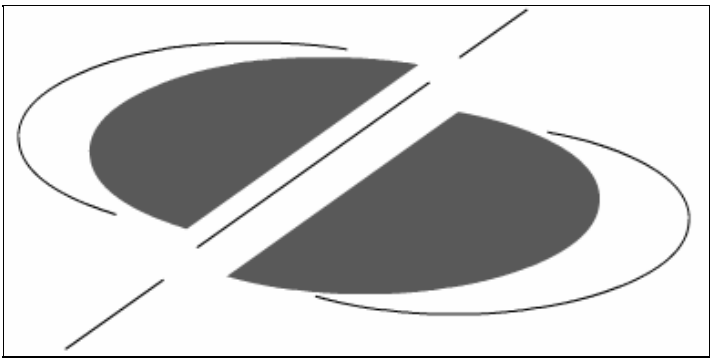


Рис. 3.28. Фрагменты изначальной фигуры, изображенной на рис. 3.26

Это и есть фрагментация. Монолитная, казалось бы, фигура "развалилась" сама собой на семь частей, каждая из которых стала абсолютно независимой.

С фрагментацией тесно связана и другая особенность Flash-графики — эффект *слияния* заливок. Заключается он в том, что если две или более заливки, нарисованных одним цветом, соприкасаются, они сливаются, превращаясь в единое целое.

Давайте нарисуем изображение, представляющее собой два перекрывающихся мазка "кистью", выполненных одним цветом (рис. 3.29). После этого щелкнем мышью на одном из этих двух мазков. Мы увидим, что будет выделена вся фигура, образованная перекрещивающимися мазками.

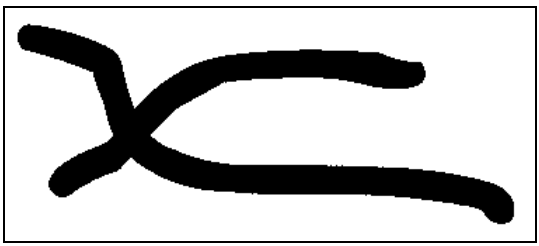


Рис. 3.29. Фигура, нарисованная для демонстрации эффекта слияния графики

Исходя из вышеописанного, можно вывести два правила поведения любой Flash-графики, состоящей из нескольких соприкасающихся примитивов. Причем вид примитива в этом случае роли не играет. Итак, фрагментируются:

- пересекающиеся примитивы разных цветов;
- пересекающиеся и изломанные линии;
- пересекающиеся линии и заливки, даже одноцветные (хотя пересекающиеся одноцветные заливки сливаются).

Сливаются же только заливки, нарисованные одним цветом. Как видим, эти правила достаточно просты.

Спору нет, и фрагментация, и слияние часто бывают полезными. Часто, но не всегда. Иногда бывает нужно избежать фрагментации и слияния сложных фигур. Сделать это можно четырьмя способами:

- выполнить группировку фрагмента, который не должен ни фрагментироваться, ни сливаться с другими фрагментами;
- нарисовать фрагмент, который не должен фрагментироваться, используя режим объектного рисования;
- разнести фрагменты, которые не должны фрагментироваться другими фрагментами или сливаться с ними, по отдельным слоям (о слоях *см. главу 5*);

- сохранить созданный фрагмент изображения как образец в библиотеке (о библиотеках и образцах *см. главу 8*).

Первый способ используется, в основном, как временное решение проблемы фрагментации и слияния. Чтобы решить эту проблему раз и навсегда, следует использовать второй, третий или четвертый способы.

Группировка фрагментов

Выше мы уже упоминали о группировке. Выясним теперь, что это такое.

Группировка позволяет временно объединить несколько разрозненных фрагментов изображения в единое целое — группу фрагментов. После выполнения группировки мы можем работать с получившейся группой как с обычным, "монолитным" фрагментом. Группа фрагментов не подвергается ни фрагментации, ни слиянию.

Чтобы сгруппировать графические фрагменты, выделим их и выберем пункт **Group** меню **Modify** или нажмем комбинацию клавиш <Ctrl>+<G>. Результат показан на рис. 3.30.

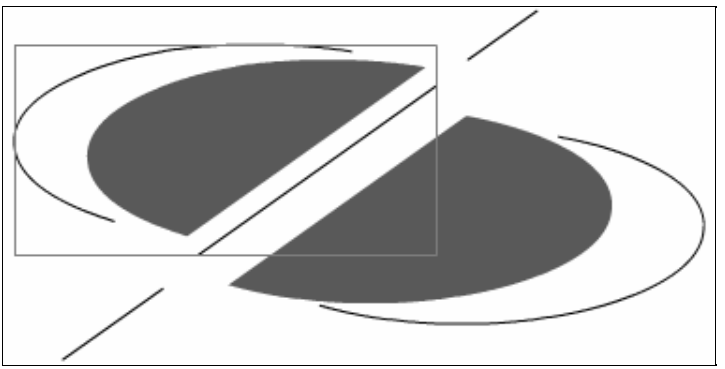


Рис. 3.30. Выделенная группа фрагментов изображения, показанного на рис. 3.26

Созданная нами таким образом группа ведет себя как единое целое. При щелчке мышью на ней она будет выделена целиком. Выделенная группа фрагментов, в отличие от обычной, не сгруппированной графики, не заполняется штриховкой, а помещается в тонкую синюю рамку.

Закончив работу с группой фрагментов, мы, вероятно, захотим *разбить* ее на отдельные составляющие. Для этого выделим группу и выберем в меню **Modify** либо пункт **Ungroup**, либо пункт **Break Apart** (этот же пункт доступен

в контекстном меню выделенной группы). Также мы можем нажать одну из комбинаций клавиш — <Ctrl>+<Shift>+<G> или <Ctrl>+<B>. Группа будет тотчас разбита на отдельные составляющие ее фрагменты.

Если нам понадобится исправить один из фрагментов, составляющих группу, не разбивая ее, мы сделаем следующее. Выделим эту группу и выберем пункт **Edit Selected** в меню **Edit** (или одноименный пункт в контекстном меню выделенной группы) или просто дважды щелкнем эту по группе. После этого в рабочей области будут показаны только фрагменты, составляющие выделенную группу, а все остальное будет закрашено серым цветом. Изменив то, что нужно, либо выберем в меню **Edit** пункт **Edit All**, либо щелкнем по кнопке , расположенной в левой части инструментария документа, либо дважды щелкнем по пустому пространству на рабочей области, чтобы вернуться к обычному режиму работы.

Объектное рисование

Объектное рисование появилось только во Flash 8. Это особый режим рисования графики, при котором Flash превращает все рисуемые фигуры в группы. Разумеется, эти группы не подвергаются ни фрагментации, ни слиянию, что часто бывает полезно.

Режим объектного рисования доступен при выбранных инструментах "линия", "прямоугольник", "эллипс", "карандаш", "перо" и "кисть", т. е. практически во всех изученных нами инструментах Flash. Включается он с помощью кнопки-выключателя модификатора **Object Drawing** (Объектное рисование), находящейся в области **Options** главного инструментария. Будучи нажатой, эта кнопка активизирует режим объектного рисования, а будучи отключенной, — обычный режим *графического рисования*, в котором мы работали до этого момента.

Для переключения между режимами графического и объектного рисования можно использовать также клавишу <J>. Пожалуй, так будет быстрее и удобнее.

Нарисованные в режиме объектного рисования фрагменты мы можем править точно так же, как и группы. И, конечно же, мы можем в любой момент выполнить их разбиение.

Правка графики

Ну что ж, пора приниматься за правку нарисованного. Как говорили древние греки — чаще поворачивай стиль. Проверим, выбран ли у нас инстру-

мент "стрелка", так как подавляющее большинство операций по правке графики выполняются именно с его помощью. Если же для каких-то операций понадобятся другие инструменты, автор об этом предупредит.

Перемещение и удаление графики

Перемещение и удаление графики — самые простые и наглядные операции. В самом деле, что может быть проще, чем двинуть мышкой или нажать клавишу?..

Перемещение фрагментов изображения выполняется очень просто. Выделим нужный фрагмент (или фрагменты) и мышью перетащим его на новое место. Если нам нужно переместить графику в направлении под углом, кратным 45°, во время перетаскивания будем удерживать нажатой клавишу <Shift>.

Для точного перемещения графики можно воспользоваться клавишами-стрелками клавиатуры. Выделим нужный фрагмент и будем нажимать клавиши-стрелки, пока он не окажется на нужном месте. При однократном нажатии на клавишу-стрелку происходит смещение на один пиксел. Если же мы при этом будем удерживать клавишу <Shift>, при каждом нажатии выбранный фрагмент будет смещаться сразу на 8 пикселов.

На новое место можно переместить не сам выделенный фрагмент, а его копию (т. е. выполнить *дублирование* фрагмента). Для этого просто перетащим его, удерживая нажатой клавишу <Alt>. Мы также можем выбрать пункт **Duplicate** в меню **Edit** или нажать комбинацию клавиш <Ctrl>+<D>, чтобы сначала продублировать выделенный фрагмент, а потом перетащить созданную таким образом его копию на нужное место.

Удалить ненужный фрагмент графики совсем просто. Для этого выделим его и либо нажмем клавишу <Del> или <Backspace>, либо выберем пункт **Clear** в меню **Edit**.

"Приклеивание" графики

Еще в *главе 2* было сказано, что фрагмент изображения, перетаскиваемый по рабочему листу, может "приклеиваться" к другим фрагментам, линиям сетки и направляющим. Эта возможность сильно облегчает рисование сложных геометрических фигур, состоящих из нескольких примитивов.

Здесь нужно иметь в виду вот что. "Приклеивание" перемещаемого фрагмента выполняется той его точкой, за которую мы "ухватились" мышью. Эта

точка (назовем ее *точкой приклеивания*) отображается на фрагменте в виде черного кружка (рис. 3.31).

Рис. 3.31. Точка приклеивания (черный кружок на контуре)

Обычно "приклеивание" помогает рисовать, однако может принести и вред — например, если мы хотим нарисовать очень сложную и извилистую кривую с помощью инструмента "карандаш" или провести две линии очень близко друг к другу. Поэтому Flash предоставляет вам возможность временно отключить "приклеивание".

Сделать это очень просто. Если мы выбрали инструмент "стрелка", становится доступным уже знакомый нам модификатор **Snap to Objects** (Приклеивание). Он представляет собой кнопку-выключатель , расположенную в области **Options** главного инструментария. Если же главный инструментарий скрыт, мы можем воспользоваться пунктом-выключателем **Snap to Objects** подменю **Snapping** меню **View**. А самый быстрый способ включить или отключить "приклеивание" — нажать комбинацию клавиш <Ctrl>+ +<Shift>+</>.

Отключить "приклеивание" к линиям координатной сетки или направляющим также очень просто. В первом случае нужно отключить пункт-выключатель **Snap to Grid** подменю **Snapping** меню **View** или нажать комбинацию клавиш <Ctrl>+<Shift>+<'>. Во втором случае, по аналогии, следует отключить пункт-выключатель **Snap to Guides** подменю **Snapping** меню **View** или нажать комбинацию клавиш <Ctrl>+<Shift>+<;>.

Специально для любителей работать при крупном масштабе (400% и более) Flash предоставляет возможность *приклеивания к отдельным пикселам* рабочей области. Чтобы его активизировать, нужно включить пункт-выключатель **Snap to Pixels** подменю **Snapping** меню **View**. После этого на экране отобразится так называемая *пиксельная сетка* — родная сестричка координатной сетки, но отображающая отдельные пикселы изображения. Как только мы отключим приклеивание (давайте писать это слово без кавычек) к пикселам, пиксельная сетка также пропадет.

Чтобы временно отключить приклеивание к пикселам во время рисования или правки графики, нажмем и будем удерживать клавишу <C>; когда мы ее отпустим, приклеивание включится снова. Если же мы отключили приклеивание к пикселам, нажатие клавиши <C> временно его включит. А клавиша <X> позволит нам скрыть (или показать, если приклеивание к пикселам отключено) пиксельную сетку.

Если мы включим пункт **Snap Align** подменю **Snapping** меню **View**, Flash поможет выровнять графические фрагменты по их горизонтальной или вертикальной границе (*выравнивающее приклеивание*). При этом, когда мы "поднесем" мышью фрагмент достаточно близко к любой горизонтальной или вертикальной границе другого фрагмента, на экране появится штриховая линия — касательная к этой границе (так называемая *направляющая*). То же справедливо и для самого рабочего листа: как только мы приблизим фрагмент к его горизонтальной или вертикальной границе, Flash покажет нам штриховую линию. Теперь, если мы отпустим левую кнопку мыши, перетаскиваемый фрагмент "приклеится" к соответствующей границе.

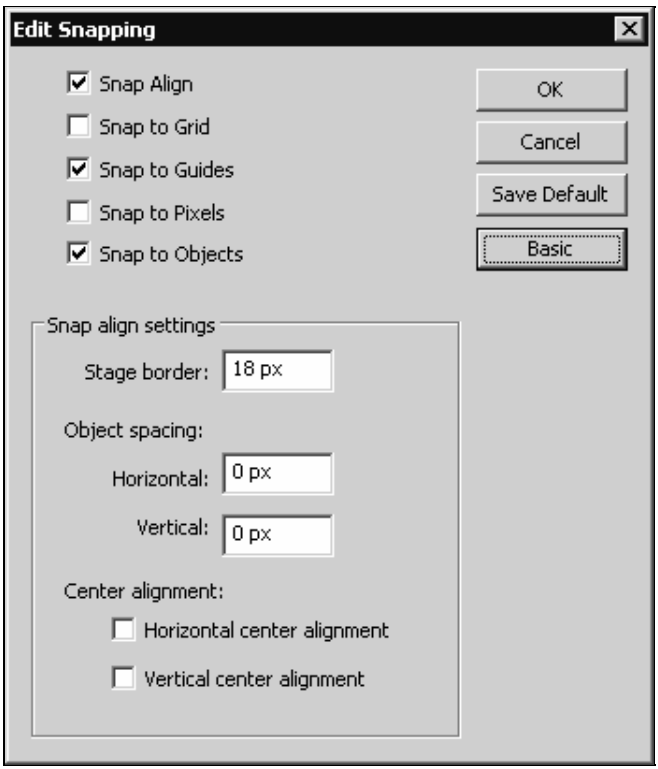


Рис. 3.32. Диалоговое окно **Edit Snapping**

Помимо всего, Flash предоставляет нам возможность настройки параметров выравнивающего приклеивания. Выберем пункт **Edit Snapping** подменю **Snapping** меню **View** или нажмем комбинацию клавиш <Ctrl>+</>, и на экране появится диалоговое окно **Edit Snapping**. Сразу же нажмем кнопку **Advanced** этого окна, чтобы получить доступ к дополнительным

элементам управления, предназначенным для настройки нужных нам параметров (рис. 3.32).

В верхней части диалогового окна **Edit Snapping** находятся флажки, соответствующие рассмотренным нами ранее пунктам-выключателям подменю **Snapping** меню **View**. Повторно рассматривать мы их не будем. Давайте лучше обратим внимание на элементы управления, расположенные ниже.

В поле ввода **Stage border** задается расстояние в пикселах, на которое перетаскиваемый фрагмент должен приблизиться к границе рабочего листа, чтобы сработало выравнивающее приклеивание. А в полях ввода **Horizontal** и **Vertical** вводится то же расстояние, но для горизонтальной и вертикальной границы других фрагментов соответственно, и тоже в пикселах.

Флажки **Horizontal center alignment** и **Vertical center alignment** позволят нам включить (или отключить) приклеивание к центру рабочего листа по горизонтали и вертикали соответственно.

Введя нужные значения, не забудем нажать кнопку **OK**. Если мы передумали менять параметры притягивания, нажмем кнопку **Cancel**.

Кстати, подменю **Snapping** со всеми рассмотренными нами пунктами также присутствуют в контекстном меню рабочей области. Так что совсем не обязательно лезть в главное меню, чтобы включить или отключить приклеивание.

Работа с буфером обмена

Все современные более-менее сложные Windows-приложения позволяют работать со стандартным *буфером обмена* Windows. Пользователь может помещать в буфер обмена фрагменты текстов и изображений и затем вставлять их в другое место этого же или другого документа, возможно, открытого в другой программе.

Разумеется, Flash также поддерживает работу с буфером обмена. И предоставляет стандартный набор операций, которые мы сейчас рассмотрим.

Мы можем *вырезать* выделенный фрагмент из листа и поместить его в буфер обмена; при этом сам выделенный фрагмент с листа пропадает. Чтобы вырезать выделенный фрагмент, выберем пункт **Cut** меню **Edit** или контекстного меню выделенного фрагмента или нажмем комбинацию клавиш <Ctrl>+<X>.

Мы также можем *скопировать* в буфер обмена выделенный фрагмент, оставив его на листе. Для этого выберем пункт **Copy** меню **Edit** или контекстного меню выделенного фрагмента или нажмем комбинацию клавиш <Ctrl>+<C>.

Чтобы поместить (*вставить*) содержимое буфера обмена на рабочий лист, выберем пункт **Paste in Center** меню **Edit** или контекстного меню рабочей

области или нажмем комбинацию клавиш <Ctrl>+<V>. Содержимое буфера обмена будет помещено в центр рабочего листа.

Если нужно поместить содержимое буфера обмена точно в ту же позицию, где находится скопированный или находился выделенный фрагмент, следует выбрать пункт **Paste in Place** меню **Edit** или контекстного меню или нажать комбинацию клавиш <Ctrl>+<Shift>+<V>. Эта операция называется *вставкой в исходную позицию.*

Несколько слов о том, как Flash при вставке обрабатывает графику, созданную в других программах. При вставке текста из другой программы на рабочем листе создается новый текстовый блок (о текстовых блоках и работе с ними см. *главу 7*). При вставке векторной графики на рабочем листе создается группа, которая может быть разбита на отдельные фрагменты и отредактирована. При вставке растрового изображения оно также помещается на рабочий лист (об импорте растровой графики *см. главу 10*).

Есть еще одна возможность вставки графики или текста в другом формате, которая также является стандартной для многих Windows-приложений — *вставка с преобразованием формата.* Пользуясь ей, можно поместить содержимое буфера обмена на рабочий лист в другом формате, например, преобразовать векторное изображение в растровое. Иногда это бывает полезно.

Чтобы выполнить вставку с преобразованием формата, выберем пункт **Paste Special** меню **Edit**. На экране появится стандартное диалоговое окно специальной вставки Windows (рис. 3.33). В списке, занимающем бóльшую часть этого окна, выберем нужный формат вставки. После этого нажмем кнопку **OK** для выполнения вставки или кнопку **Отмена** (Cancel) для отказа от нее.

Рис. 3.33. Диалоговое окно специальной вставки Windows

Изменение цвета графики

Мы можем изменять цвет линий контура и заливки, просто выделив их и задав новый цвет с помощью селекторов цвета, расположенных в области **Colors** главного инструментария. Кроме того, мы можем воспользоваться аналогичными селекторами цвета, расположенными в центре панели **Properties** (рис. 3.34).

Рис. 3.34. Селекторы цвета, расположенные на панели **Properties**

Нужно только учитывать, что оба этих набора селекторов цвета также меняют цвета, которыми будет нарисована новая графика.

Правка контура фигуры

Контур любой фигуры, нарисованной средствами Flash, можно править, тем самым изменяя и деформируя фигуру. Это выполняется с помощью все того же инструмента "стрелка" аналогично тому, как мы только что перемещали графику с места на место.

Нужно сказать сразу, что для изменения формы фигуры нужно, прежде всего, снять с нее выделение. Если этого не сделать, фигура будет частично или полностью перемещена на другое место, а это совсем не то, что мы хотим сделать.

Итак, для того чтобы изменить форму какой-либо фигуры, будь это простейший прямоугольник или сложная кривая, нужно выбрать инструмент "стрелка", снять с фигуры выделение, найти на контуре фигуры нужную точку и переместить ее с помощью мыши. Перемещение выполняется очень просто: ставим на нужную точку курсор мыши, нажимаем левую кнопку, перемещаем мышь и отпускаем кнопку. В результате этого форма фигуры изменится.

Таким образом мы можем удлинять и укорачивать прямые линии, перетаскивая их за концы (так называемые *конечные точки*). Также мы можем превращать прямые линии в кривые, захватив мышью любую точку на их середине и "вытянув" ее в нужную сторону. При этом на линии будет создана *точка искривления*, обозначающая место, где эта линия искривляется. Посмотрим на рис. 3.35 — там показан пример того, что можно сделать с прямой линией.

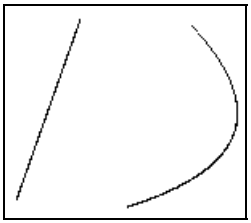


Рис. 3.35. Результат правки прямой линии
(слева — исходная линия, справа — результат правки)

В случае более сложных фигур мы можем перемещать углы (в терминологии Flash — *угловые точки*, т. е. точки соединения линий), тем самым вытягивая или "съеживая" их (фигур) контуры. Например, обычный прямоугольник можно "вытянуть" по диагонали за угол (рис. 3.36) или выгнуть любую из его сторон, превратив ее в кривую (рис. 3.37).

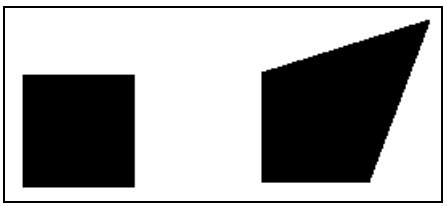


Рис. 3.36. Результат "вытягивания" угла прямоугольника
(слева — исходный прямоугольник, справа — результат изменения формы)

Рис. 3.37. Результат "вытягивания" стороны прямоугольника
(слева — исходный прямоугольник, справа — результат изменения формы)

Если фигура имеет заливку, то форма заливки тоже изменится так, чтобы заполнить получившийся в результате контур. Собственно, это заметно по рис. 3.36 и 3.37. Если же контур фигуры в результате перемещения перестанет быть замкнутым, заливка сохранит форму, которую имела перед этим.

Flash предоставляет также возможность создания на прямой новой угловой точки. Обычно они задаются при создании фигуры, но иногда впоследствии бывает нужно добавить дополнительную угловую точку. Для этого выполним

уже знакомую нам операцию по "вытягиванию" линии (прямой или кривой), но "вытягивать" ее будем, удерживая нажатой клавишу <Ctrl>. В том месте, где мы "захватили" линию мышью, появится новая угловая точка. Результат может выглядеть так, как показано на рис. 3.38 — заметим, что мы создали острый угол, которого раньше не было.

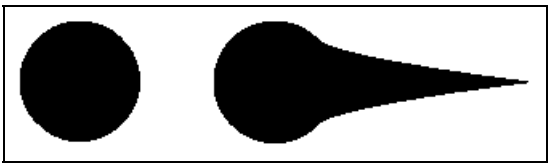


Рис. 3.38. Результат создания угловой точки на окружности (слева — исходная окружность, справа — фигура после изменения формы)

Теперь самое время немного попрактиковаться. Нарисуйте какую-нибудь достаточно сложную геометрическую фигуру и немного "поиздевайтесь" над ней. Попробуйте выполнить все описанные нами операции по нескольку раз и посмотрите сами, что из этого получится.

Точная правка контуров

Мы узнали, как можно изменять форму контуров с помощью инструмента "стрелка". Однако во многих случаях рассмотренный способ может оказаться слишком грубым. Некоторые особо точные операции нельзя выполнить при помощи "стрелки". Поэтому создатели Flash предусмотрели другой инструмент — **Subselection Tool** (Белая стрелка[2]). Он позволяет нам выполнять самые сложные действия над линиями контура, как прямыми, так и кривыми.

Чтобы выбрать инструмент "белая стрелка", следует щелкнуть по кнопке , находящейся в главном инструментарии, или нажать клавишу <A> на клавиатуре. После этого курсор мыши примет вид небольшой белой стрелки.

С помощью "белой стрелки" мы можем выделять фрагменты графики, как делали это ранее уже знакомым нам инструментом "стрелка". Результат такого выделения показан на рис. 3.39. Нужно только иметь в виду, что инструмент "белая стрелка" не позволяет выделять заливки — только контуры.

[2] Термин автора. Более распространенным является перевод "Инструмент частичного выделения". — *Прим. ред.*

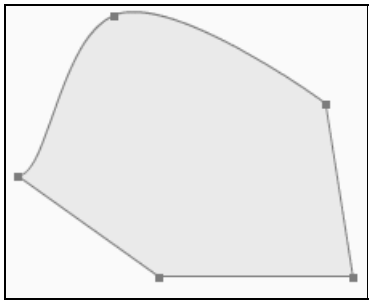


Рис. 3.39. Графика, выделенная с помощью инструмента "белая стрелка"

Посмотрим внимательно на рис. 3.39. На контуре нарисованной фигуры хорошо заметны точки, обозначенные небольшими белыми квадратами. Это угловые точки — точки смыкания линий, прямых и кривых. Перемещая эти точки с помощью мыши или пользуясь клавишами-стрелками клавиатуры, мы можем изменять форму контура.

Щелкнем в любой угловой точке. Мы увидим, что она изменит вид (вместо белого квадратика появится зеленый) — это означает, что угловая точка выделена. Теперь мы можем удалить эту точку, если она не нужна, для чего достаточно нажать клавишу <Del>. Результат удаления одной из угловых точек нашей фигуры показан на рис. 3.40.

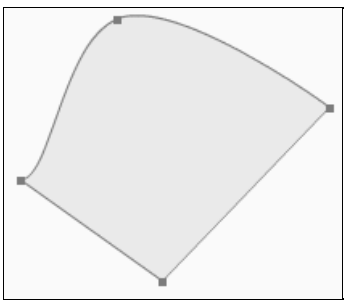


Рис. 3.40. Результат удаления одной из угловых точек фигуры, показанной на рис. 3.39

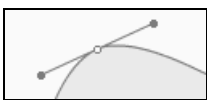


Рис. 3.41. Выделенная точка искривления

Теперь щелкнем в точке, расположенной в левом верхнем углу нашей фигуры. На рис. 3.41 показан результат этого действия. Видно, что теперь эта точка имеет вид не квадратика, а окружности. Это точка искривления, и она показывает, в каком месте линия изгибается. Мы также можем переместить эту точку с помощью мыши или клавиш-стрелок. Кроме того, можно перемещать точки, расположенные на концах синей касательной, чтобы изменить форму получившейся кривой.

Мы можем выполнить множественное выделение сразу нескольких угловых точек или точек искривления. Для этого при выбранном инструменте "белая стрелка" щелкнем в этих точках, удерживая нажатой клавишу <Shift>. А проще всего выделить их перетаскиванием мыши, опять же при выбранном инструменте "белая стрелка".

Мы можем преобразовать имеющуюся угловую точку в точку искривления, для чего достаточно выделить ее и начать перетаскивать при нажатой клавише <Alt>. Для обратного преобразования — точки искривления в угловую точку — придется воспользоваться инструментом "перо". Выберем его в главном инструментарии, щелкнем где-либо на контуре, чтобы его выделить, а потом щелкнем еще раз, уже в нужной точке (а второй щелчок вообще удалит эту точку).

С помощью инструмента "перо" можно также поместить на кривую (но не на прямую!) линию новую точку искривления. Выберем этот инструмент, щелкнем сначала по контуру, чтобы его выделить, а потом — в нужном месте нужной линии контура. Повторный щелчок преобразует точку искривления в угловую точку, а третий — вообще удалит ее.

Уф! Ох, и сложности!.. Не мешало бы поэкспериментировать со всем этим богатством для закрепления полученных знаний. Как вы считаете?

Сложное выделение. Инструмент *Lasso* (*"Ласссо"*)

Мы уже рассмотрели различные способы выделения графики. Добавим к ним еще парочку способов так называемого *сложного* выделения. При сложном выделении выделяются не целые примитивы, а их части, что позволяет художнику создавать еще более замысловатые геометрические фигуры и, стало быть, еще более оригинальную графику.

Прежде всего, рассмотрим уже знакомый нам инструмент "стрелка". Все ли мы о нем знаем? Оказывается, нет.

Давайте нарисуем на рабочем листе Flash небольшой эллипс и попытаемся выделить с помощью перетаскивания мыши его небольшую часть. Для этого сначала установим курсор мыши выше нарисованного нами эллипса и чуть левее его центра. После этого нажмем левую кнопку мыши и протащим

мышь, пока курсор не окажется ниже эллипса и чуть правее его центра. Отпустим кнопку мыши и посмотрим, что у нас получилось. А получиться у нас должно что-то похожее на фигуру, приведенную на рис. 3.42.

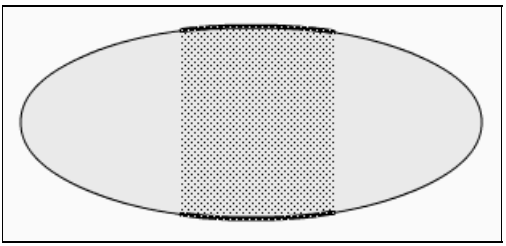


Рис. 3.42. Эллипс, фрагмент которого выделен с помощью перетаскивания мышью

Теперь проверим, действительно ли фрагмент эллипса выделен, — и мы сможем проделывать над ним все, что захотим. Для пробы перенесем выделенный фрагмент на другое место. Ура, получилось (см. рис. 3.43)!

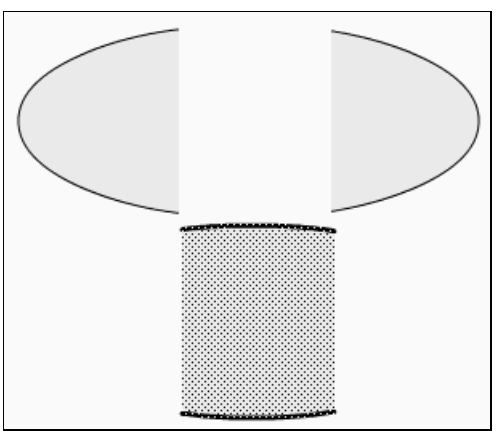


Рис. 3.43. Эллипс, выделенный фрагмент которого был перенесен на другое место

Так что с помощью уже знакомого нам инструмента "стрелка" можно выделять перетаскиванием не только целые фигуры, но и их фрагменты. Это может нам очень пригодиться в дальнейшем, когда мы начнем много работать с Flash. В самом деле, каких геометрических фигур можно наделать!

Но как быть, если нужно выделить не прямоугольный фрагмент, а "лоскут" неправильной формы? "Стрелка" здесь нам не поможет. Нужен другой инструмент, который называется **Lasso** ("Лассо").

Чтобы выбрать инструмент "лассо", щелкнем по кнопке , находящейся в главном инструментарии, или нажмем клавишу <L> на клавиатуре. Курсор мыши примет вид небольшого лассо.

Как пользоваться "лассо"? Ставим курсор мыши в точку, где собираемся начать выделение, нажимаем левую кнопку мыши и, не отпуская ее, рисуем *область выделения*, охватывающую нужный нам "лоскут". Не забудем замкнуть контур этой области (впрочем, Flash сделает это за нас, когда мы отпустим кнопку мыши). Отпускаем кнопку мыши — и "лоскут" выделен (рис. 3.44).

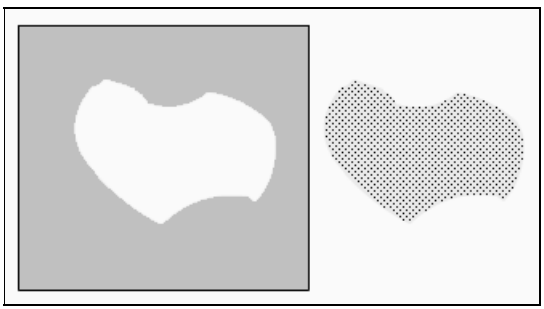


Рис. 3.44. Фрагмент прямоугольника, выделенный с помощью инструмента "лассо"

Инструмент "лассо" имеет несколько дополнительных возможностей, которые могут помочь в работе. Эти возможности включаются особыми модификаторами, доступными, как обычно, в области **Options** главного инструментария, если выбран этот инструмент. Давайте их рассмотрим.

Модификатор **Polygon Mode** (Полигон) переключает "лассо" в такой режим, когда "лоскут" выделения представляет собой не замкнутую фигуру произвольной формы, а многоугольник, состоящий из отрезков прямых линий. Чтобы включить его, нужно щелкнуть кнопку-выключатель .

При включенном модификаторе "полигон" работа с "лассо" протекает так. Мы выбираем место, где будет находиться начальная точка многоугольника выделения, ставим туда курсор мыши, щелкаем левой кнопкой и протаскиваем мышь, "вытягивая" за собой первый отрезок. Подведя курсор к месту, где должна находиться вторая точка многоугольника, опять щелкаем левой кнопкой. Далее точно таким же образом мы щелчками мыши

отмечаем третью, четвертую и все остальные точки этого многоугольника. Чтобы замкнуть контур, на последней точке мы выполним не одинарный, а двойной щелчок. Пример такой многоугольной области выделения приведен на рис. 3.45.

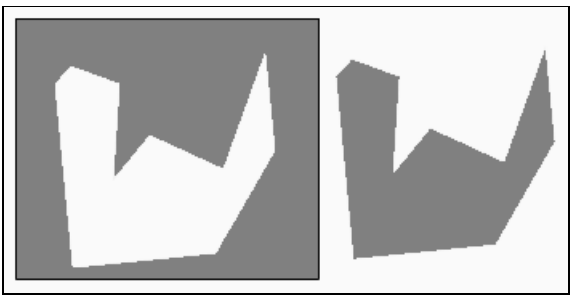


Рис. 3.45. Фрагмент прямоугольника, выделенный с помощью инструмента "лассо" при включенном модификаторе "полигон"

Создание многоугольной области выделения похоже на рисование многоугольников с помощью инструмента "перо". Научиться пользоваться модификатором "полигон" легче, если держать в памяти эту аналогию.

Находясь в обычном режиме работы "лассо", мы можем временно переключиться в режим "полигона". Для этого при рисовании области выделения нужно удерживать нажатой клавишу <Alt>.

Частичное стирание графики. Инструмент *Eraser* (*"Ластик"*)

Компьютерный (да и обычный, "бумажный") художник не только рисует. Увы, иногда приходится и удалять только что нарисованное. В конце концов, художник — человек, а не программа, и он может ошибиться. А иногда стирание необходимо, если нужно создать сложную геометрическую фигуру путем удаления части другой фигуры, более простой.

Конечно, мы всегда можем выделить часть графики, которую нужно удалить, и нажать клавишу <Del>. Выделить фрагмент можно как с помощью уже знакомой нам "стрелки", так и "заарканив" ее инструментом "лассо". Но Flash предлагает еще один инструмент для *частичного удаления* нарисованных контуров и заливок — **Eraser** ("Ластик"). Рассмотрим его.

Инструмент "ластик" ведет себя как обычный резиновый ластик, которым мы стираем на бумаге карандашные пометки. Если брать аналогии из Flash,

то "ластик" похож на "кисть": он работает по тому же принципу, только не закрашивает лист мазками, а стирает все, чего мы им коснемся.

Чтобы выбрать инструмент "ластик", щелкнем по кнопке , находящейся в главном инструментарии, или нажмем клавишу <E> на клавиатуре. Курсор мыши при этом может принимать разные формы, в зависимости от настроек, заданных с помощью модификаторов (см. ниже).

Используется инструмент "ластик" аналогично инструменту "кисть". Мы ставим курсор мыши в начальную точку, нажимаем левую кнопку, двигаем мышь, стирая графику, и, в конце концов, отпускаем кнопку мыши, закончив стирание. Пример того, что может получиться у нас в результате "стирания" "ластиком", показан на рис. 3.46.

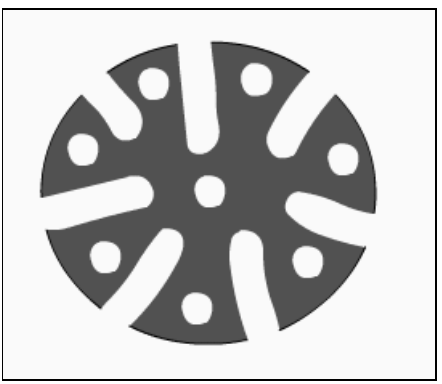


Рис. 3.46. Пример использования инструмента "ластик"

Если при работе нужно двигать "ластик" строго по горизонтали или вертикали, то во время перемещения его мы будем удерживать нажатой клавишу <Shift>. Кроме того, мы можем быстро стереть все нарисованное на рабочем листе, дважды щелкнув на кнопке выбора "ластика" в главном инструментарии. Это значительно быстрее, чем последовательно нажимать сначала комбинацию клавиш <Ctrl>+<A> (выделение всей графики), потом — клавишу <Del> (собственно удаление). Главное только — не удалить случайно все свои труды.

Так же, как и в случае с "кистью", Flash предоставляет возможность выбора формы и размера ластика. Для этого служит единственный модификатор **Eraser Shape** (Форма ластика), расположенный в области **Options** главного инструментария. Этот модификатор представляет собой обычный раскрывающийся список, в котором в виде графических изображений представлены все доступные во Flash формы и размеры "ластика" (рис. 3.47).

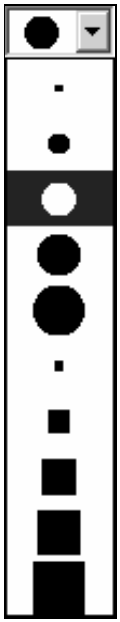


Рис. 3.47. Список модификатора "форма ластика" в раскрытом виде

Как только мы выберем с помощью этого модификатора форму и размер "ластика", форма и размер курсора мыши изменятся. Таким образом, мы всегда будем знать, каким "ластиком" пользуемся.

Еще один модификатор — "режим стирания" — позволяет задать режим работы "ластика", т. е. установить, будет ли "ластик" стирать линии контура или будет ограничиваться лишь заливками. Этот модификатор имеет вид кнопки с меню (рис. 3.48).

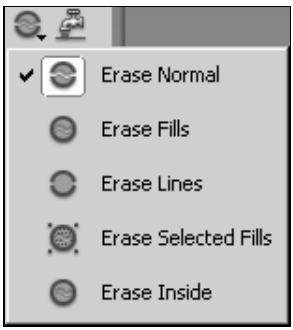


Рис. 3.48. Кнопка модификатора "режим стирания" с раскрытым меню

В меню модификатора "режим стирания" доступны пять пунктов:

- **Erase Normal** — обычный режим, когда стираются любые контуры и любые заливки (см. рис. 3.46);
- **Erase Fills** — стираются только заливки, контуры не стираются (рис. 3.49);
- **Erase Lines** — стираются только контуры, заливки не стираются (рис. 3.50);

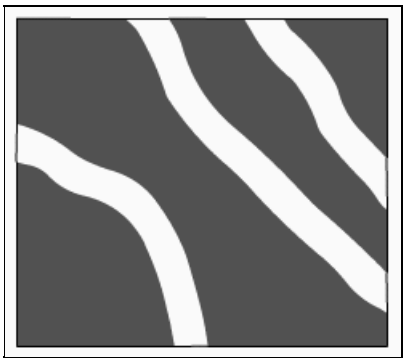


Рис. 3.49. Результат применения режима стирания **Erase Fills** (обратим внимание — линии контура остались нестертыми)

Рис. 3.50. Результат применения режима стирания **Erase Lines** (обратим внимание — стерлись только линии контура, а заливка осталась нестертой)

- **Erase Selected Fills** — стирается только выделенная с помощью инструмента "стрелка" заливка, невыделенные заливки и контуры остаются нестертыми (рис. 3.51);

- **Erase Inside** — стирается только заливка, находящаяся в пределах контура. То, что находится по другую сторону контура, не стирается. Сам контур также не стирается (рис. 3.52).

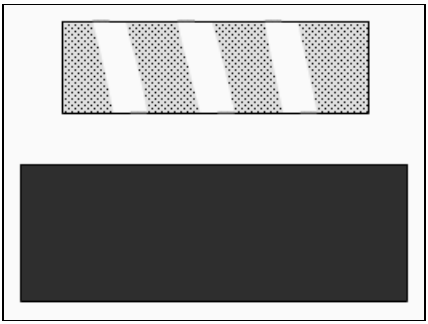


Рис. 3.51. Результат применения режима стирания **Erase Selected Fills** (заливка верхнего прямоугольника была выделена с помощью инструмента "стрелка", а заливка нижнего — нет)

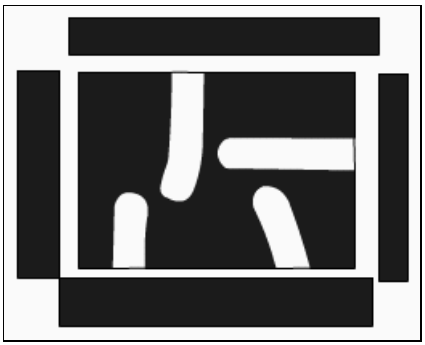


Рис. 3.52. Результат применения режима стирания **Erase Inside** (была стерта заливка центрального прямоугольника, причем мазки были направлены наружу, а внешние прямоугольники затронуты не были)

Но что делать, если нужно быстро стереть фрагмент графики, например, заливку или линию? Можно, конечно, переключиться на инструмент "стрелка", выделить нужный фрагмент и нажать клавишу <Del>. А можно воспользоваться модификатором **Faucet** (Кран). С помощью такого "крана" мы просто "смоем" ненужный фрагмент с рабочего листа.

Модификатор "кран" включается кнопкой-выключателем , которая доступна только при выбранном инструменте "ластик". После включения этого

модификатора мы можем просто щелкать на фрагментах графики, подлежащих удалению. Для отключения "крана" еще раз щелкнем по вышеупомянутой кнопке.

Дополнительные возможности работы с контурами

А теперь перечислим некоторые дополнительные возможности по работе с контурами, предоставляемые Flash.

Ранее было сказано, что при рисовании некоторыми инструментами (в частности, "карандашом" и "кистью") Flash автоматически сглаживает получившуюся графику. Однако Flash предлагает средства и для сглаживания уже нарисованной графики. Это особые модификаторы, доступные в области **Options** главного инструментария, если выбран инструмент "стрелка".

Первый из этих модификаторов так и называется — **Smooth** (Сглаживание). Он сглаживает кривые, устраняет рывки, сдвиги, нарушения кривизны и прочие дефекты, а также по мере возможности упрощает линии, иначе говоря, *оптимизирует* их. Благодаря такой оптимизации уменьшается количество описывающих их точек, а значит, уменьшается размер результирующего файла Shockwave/Flash и время, необходимое для вывода кривых на экран. Умелая и тщательная оптимизация всегда стоит затраченного на нее времени, так как выигрыш потом окупится многократно.

Чтобы выполнить сглаживание какой-либо кривой с помощью этого модификатора, выделим эту кривую и нажмем кнопку . (Если на рабочем листе ничего не выделено, эта кнопка недоступна.) Также можно выбрать пункт **Smooth** подменю **Shape** меню **Modify** или контекстного меню выделенной линии. Чтобы выполнить более сильное сглаживание, мы нажмем эту кнопку или выберем этот пункт меню несколько раз подряд. На рис. 3.53 показан результат применения модификатора "сглаживание" к кривой.

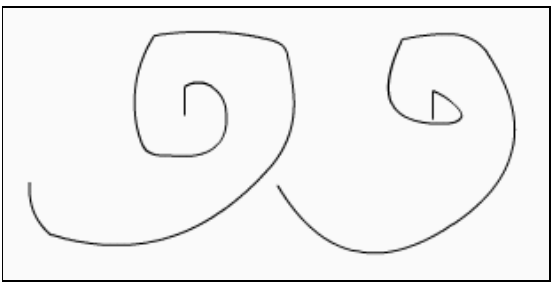


Рис. 3.53. Результат применения модификатора "сглаживание" (слева — исходная кривая, справа — ее вид после применения модификатора)

Здесь нужно сразу сказать, что применение модификатора "сглаживание" к прямым линиям и геометрическим фигурам, чей контур состоит из прямых, не дает никакого эффекта. Также не дает эффекта применение этого модификатора к простым кривым линиям с постоянным радиусом кривизны.

Модификатор **Straighten** (Спрямление) выполняет совершенно противоположную задачу. Он спрямляет кривые, заостряет углы и пытается привести нарисованные нами геометрические фигуры к простейшим: прямая, прямоугольник, треугольник, эллипс. При этом он тоже оптимизирует графику.

Чтобы испытать модификатор "спрямление" в действии, выделим кривую, которую нужно спрямить, и нажмем кнопку . (Если на рабочем листе ничего не выделено, эта кнопка также недоступна.) Также можно выбрать пункт **Straighten** подменю **Shape** меню **Modify** или контекстного меню выделенной линии. Опять же, чтобы выполнить более сильное спрямление, следует нажать эту кнопку или выбрать этот пункт меню несколько раз подряд. Пример применения модификатора "спрямление" к группе линий показан на рис. 3.54.

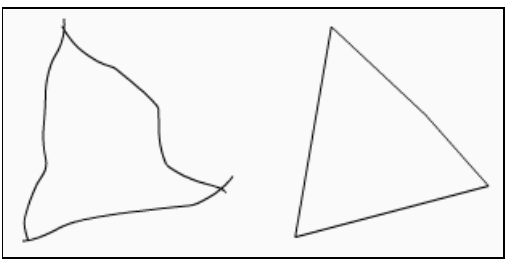


Рис. 3.54. Результат применения модификатора "спрямление" (слева — исходная группа линий, справа — ее вид после применения модификатора)

Применение модификатора "спрямление" к прямым линиям и фигурам, чей контур состоит из прямых линий, не дает никакого эффекта. Также, по-видимому, он никак не действует на кривые линии с постоянным радиусом кривизны.

Есть еще один способ сглаживания (и оптимизации) кривых. Выделим на рабочем листе контур, который нужно оптимизировать, и выберем пункт **Optimize** подменю **Shape** меню **Modify** или нажмем комбинацию клавиш <Ctrl>+<Alt>+<Shift>+<C>. На экране появится диалоговое окно **Optimize Curves**, показанное на рис. 3.55.

С помощью движкового регулятора **Smoothing** выбирается степень сглаживания кривых. Крайнее левое значение вообще отключает всякое сглажива-

ние, а крайнее правое — задает высшую степень оптимизации. По умолчанию движок установлен посередине шкалы — вероятно, это лучший выбор.

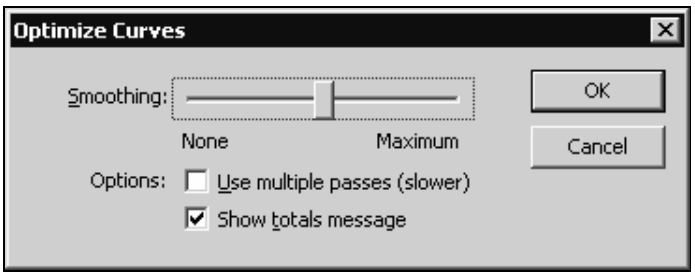


Рис. 3.55. Диалоговое окно **Optimize Curves**

Будучи включенным, флажок **Use multiple passes (slower)** заставляет Flash оптимизировать выбранный фрагмент несколько раз подряд, пока не станет ясно, что дальнейшая оптимизация бесполезна. Такая *многопроходная оптимизация* выполняется значительно медленнее обычной, *однопроходной*, но позволяет достичь лучших результатов. По умолчанию этот флажок отключен.

Флажок **Show totals message** включает или выключает вывод в конце процесса оптимизации окна, содержащего сведения о результатах (рис. 3.56). По умолчанию этот флажок включен.

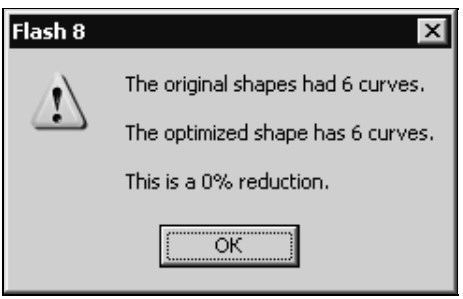


Рис. 3.56. Диалоговое окно сведений о результатах оптимизации кривых

Установив нужные параметры оптимизации, нажмем кнопку **OK**. (Если мы передумали сглаживать выделенный фрагмент, следует нажать кнопку **Cancel**.) Ознакомимся со сведениями о результатах сглаживания и оптимизации и закроем диалоговое окно, содержащее эти сведения, нажатием кнопки **OK**.

Осталось рассмотреть совсем немногое, а именно — три интересные возможности Flash по работе с заливками и фигурами, содержащими заливки.

Первая из них — это возможность преобразования линий в заливки. Зачем это может пригодиться, непонятно, так как Flash 8, в отличие от предыдущих версий Flash, позволяет применить к линиям градиентный цвет (о градиентных цветах будет рассказано в *главе 6*). Правда, разработчики Flash утверждают, что преобразование линий в заливки в некоторых случаях ускоряет отображение графики, хотя размер результирующего файла Shockwave/Flash может при этом увеличиться.

Чтобы преобразовать линию в заливку, выделим нужную линию (это может быть как простая прямая линия, так и сложный контур, содержащий множество кривых Безье) и воспользуемся пунктом **Convert Lines to Fills**, находящимся в подменю **Shape** меню **Modify**.

Здесь нужно дать небольшие пояснения. После преобразования наша (бывшая) линия будет выглядеть как очень тонкая заливка, заключенная между очень тонкими линиями. Если мы попробуем изменить ее форму, воспользовавшись инструментом "стрелка", то сразу это увидим. На рис. 3.57 видно, что на самом деле мы переместили одну из тонких внешних линий, и заливка сразу же "распухла", чтобы заполнить образовавшееся свободное пространство.

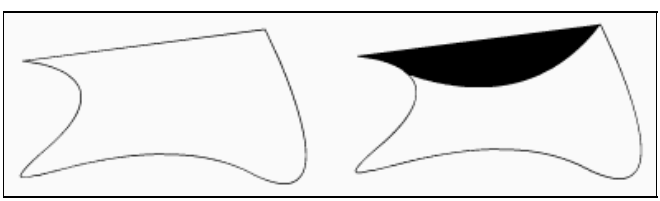


Рис. 3.57. Результат изменения формы контура, преобразованного в заливку (слева — до изменения, справа — после)

Вторая возможность — увеличение или уменьшение размера заливки на определенное число пикселов — позволяет, в частности, получить очень интересный эффект при управлении размером заливки, созданной из линии — линия станет визуально толще или тоньше. Однако при этом может исказиться форма заливки и пропасть контур (если он есть). Поэтому используйте данную функцию осмотрительно.

Для изменения размера какой-либо заливки выделим ее (она может быть заключена или не заключена в контур). Далее выберем пункт **Expand Fill** подменю **Shape** меню **Modify**. На экране появится диалоговое окно **Expand Fill** (рис. 3.58).

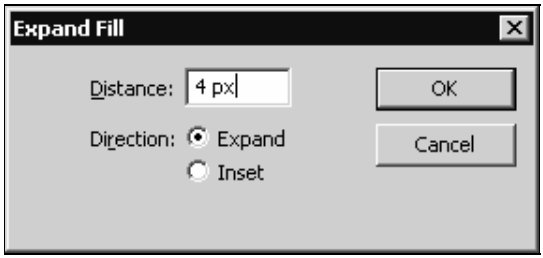


Рис. 3.58. Диалоговое окно **Expand Fill**

В поле ввода **Distance** укажем расстояние, на которое увеличивается или уменьшается заливка, в пикселах. С помощью переключателей группы **Direction** зададим, будет ли заливка увеличиваться (переключатель **Expand**) или уменьшаться (**Inset**). После этого остается только нажать кнопку **OK** для выполнения действия или **Cancel** для отказа от него.

Результат последовательного выполнения сначала уменьшения, а потом — увеличения заливки показан на рис. 3.59. Видно, как после этих операций у заливки пропал контур, а сама заливка исказилась.

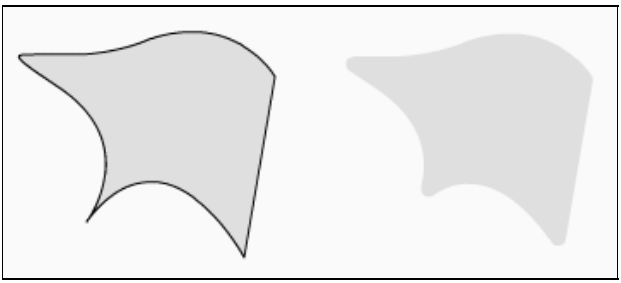


Рис. 3.59. Результат последовательного увеличения и уменьшения заливки (слева — до изменения, справа — после)

И, наконец, рассмотрим третью возможность по работе с заливками — возможность сглаживания контура, в который она помещена. При этом контур не только сглаживается, но еще и делается как можно менее заметным: его толщина уменьшается до минимума, а цвет задается таким же, как и у заливки. Это также может быть полезно во многих случаях, например, для того, чтобы удалить из контура лишние линии.

Выделим на рабочем листе фигуру, контур которой нужно сгладить. Далее выберем пункт **Soften Fill Edges** подменю **Shape** меню **Modify**. На экране появится диалоговое окно **Soften Fill Edges** (рис. 3.60).

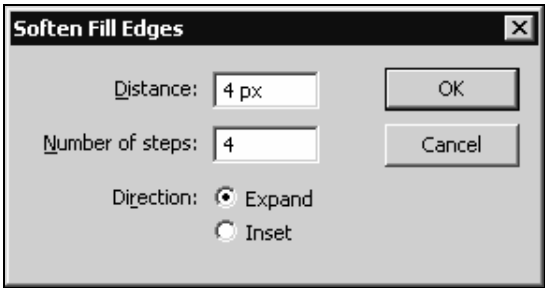


Рис. 3.60. Диалоговое окно **Soften Fill Edges**

В поле ввода **Distance** укажем толщину контура, который должен получиться в результате сглаживания, в пикселах.

В поле ввода **Number of steps** зададим максимальное количество кривых, которое будет использовано для создания нового, сглаженного контура. Чем больше кривых будет при этом использовано, тем более сложным получится контур, но тем бóльшим окажется результирующий файл Shockwave/Flash и тем дольше будет выводиться графика. Поэтому нужно выбрать разумный компромисс — возможно, для этого понадобится немного поэкспериментировать.

С помощью переключателей группы **Direction** задается, будет ли фигура увеличена (переключатель **Expand**) или уменьшена (**Inset**) для достижения эффекта сглаживания контура.

Введя нужные данные, нажмем кнопку **OK** для выполнения действия или **Cancel** для отказа от него.

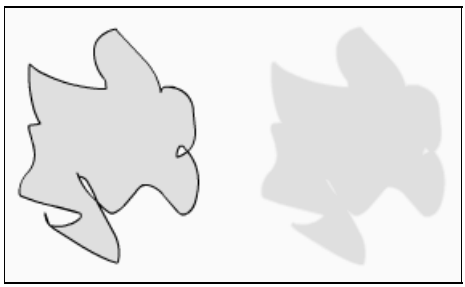


Рис. 3.61. Результат сглаживания контура фигуры (слева — фигура до изменения, справа — после)

Результат сглаживания контура, примененного к сложной фигуре, нарисованной с помощью инструмента "карандаш", показан на рис. 3.61. Обратим

внимание, что после этой операции контур стал невидимым, а его форма — несколько менее сложной. Также были устранены некоторые дефекты контура, в частности, ненужные линии.

Отмена и повтор выполненных действий

Flash, как и многие другие Windows-приложения, поможет нам исправить случайные ошибки, обязательно возникающие при работе. Мы всегда сможем отменить результат нескольких последних выполненных нами действий, восстановив то, что было до них. Эта возможность, иногда называемая *откатом*, позволит нам экспериментировать смелее, пробуя различные инструменты и преобразования. Если нам не понравится результат этих экспериментов, мы всегда сможем "откатиться" назад.

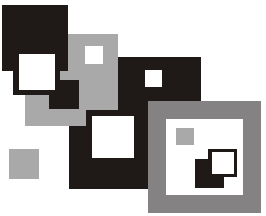
Чтобы выполнить откат, мы выберем пункт **Undo <*название операции*>** меню **Edit** или нажмем комбинацию клавиш <Ctrl>+<Z>. Последовательно выбирая этот пункт (или нажимая клавишную комбинацию), мы можем отменять сразу несколько последних выполненных действий. Нужно только помнить, что по умолчанию Flash в состоянии отменить не более 100 последних действий; информация о более ранних действиях удаляется в целях экономии оперативной памяти. Однако в настройках программы Flash (*см. главу 25*) число действий, информация о которых хранится Flash, можно изменить.

Иногда бывает нужно, наоборот, отменить результат отмены последнего вашего действия ("откатить" сам откат). Такая операция называется *возвратом*. Для выполнения возврата нужно сразу же после отката выбрать пункт **Redo <*название операции*>** меню **Edit** или нажать комбинацию клавиш <Ctrl>++<Y>.

Что дальше?

Вот и все о простейших операциях рисования и инструментах, реализующих их. В следующей главе мы выясним, какие операции Flash может проделывать с уже нарисованными изображениями, а именно — изучим его возможности по трансформации и искажению графики.

Глава 4



Трансформация графики

Что ж, рисовать графику во Flash мы научились. Править тоже. Теперь выясним, что мы еще можем делать с уже нарисованной графикой.

В этой главе мы узнаем, как выполняются операции так называемой *трансформации* графики: сдвиг, вращение, изменение размеров и более сложные искажения. Еще мы поговорим о возможностях, предлагаемых объектным режимом рисования Flash 8; они касаются объединения двух или нескольких примитивов в один. Ну, и рассмотрим некоторые полезные инструменты, которые помогут нам точно разместить фрагменты изображения на рабочем листе. Именно с этих инструментов мы и начнем.

Простейшие трансформации

В этом разделе описываются простейшие манипуляции над фрагментами нашего изображения. Можно сказать, продолжим тему правки графики, начатую в *главе 3*.

Выравнивание и распределение

Выравнивание фрагментов изображения — это размещение их на рабочем листе особым образом: вдоль какой-либо из сторон листа, по его центру или, скажем, по правой стороне самого правого выделенного фрагмента. *Распределение* же — это размещение фрагментов графики таким образом, чтобы их центры или границы находились на одинаковом расстоянии друг от друга по горизонтали или вертикали либо чтобы их ширина или высота была одинаковой.

Выравнивание графики выполняется с помощью панели **Align** (рис. 4.1). Чтобы вызвать ее на экран, нужно включить пункт-выключатель **Align** меню **Window** или нажать комбинацию клавиш <Ctrl>+<K>.

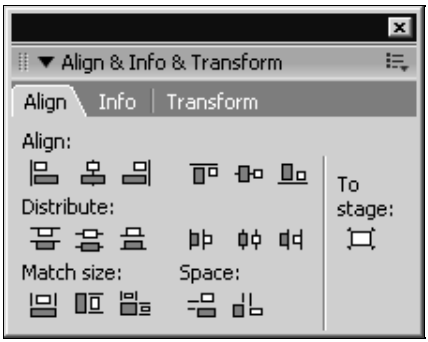


Рис. 4.1. Панель **Align**

Эта панель содержит пять групп кнопок, с помощью которых выполняются различные операции по выравниванию и распределению графики. Им соответствуют различные пункты подменю **Align**, находящегося в меню **Modify**. Рассмотрим их по очереди.

С помощью кнопок группы **Align** выполняется собственно выравнивание выделенных фрагментов изображения. (Обратим внимание: для единственного выделенного фрагмента все эти операции не имеют смысла за одним лишь исключением — если выравнивание выполняется относительно рабочего листа.) Ниже перечислены все операции, связанные с кнопками этой группы, в порядке слева направо, а в скобках указаны соответствующие им пункты подменю **Align** меню **Modify** и комбинации клавиш:

- выравнивание по левой границе самого левого из выделенных фрагментов (пункт **Left** или комбинация клавиш <Ctrl>+<Alt>+<1>);
- выравнивание выделенных фрагментов по их центральной вертикальной оси (пункт **Horizontal Center**, комбинация клавиш <Ctrl>+<Alt>+<2>);
- выравнивание по правой границе самого правого из выделенных фрагментов (пункт **Right**, комбинация клавиш <Ctrl>+<Alt>+<3>);
- выравнивание по верхней границе самого верхнего из выделенных фрагментов (пункт **Top**, комбинация клавиш <Ctrl>+<Alt>+<4>);
- выравнивание выделенных фрагментов по их центральной горизонтальной оси (пункт **Vertical Center**, комбинация клавиш <Ctrl>+<Alt>+<5>);
- выравнивание по нижней границе самого нижнего из выделенных фрагментов (пункт **Bottom**, комбинация клавиш <Ctrl>+<Alt>+<6>).

Группа кнопок **Distribute** отвечает за распределение выделенных фрагментов, т. е. за перемещение их таким образом, чтобы их границы или центры находились на равном расстоянии друг от друга. (Опять же, эти операции не имеют смысла для единственного выделенного фрагмента.) Ниже перечислены операции, соответствующие кнопкам этой группы, в порядке слева направо, вместе с соответствующими им пунктами подменю **Align** меню **Modify** и комбинациями клавиш (если они есть):

- распределение верхних границ — перемещение фрагментов по горизонтали таким образом, чтобы их верхние границы находились на одинаковом расстоянии друг от друга;
- распределение центров выделенных фрагментов по вертикали, как показано на рис. 4.2 (пункт **Distribute Heights**, комбинация клавиш <Ctrl>+ +<Alt>+<9>);
- распределение нижних границ;
- распределение левых границ — перемещение фрагментов по вертикали таким образом, чтобы их левые границы находились на одинаковом расстоянии друг от друга;
- распределение центров по горизонтали (пункт **Distribute Widths**, комбинация клавиш <Ctrl>+<Alt>+<7>);
- распределение правых границ.

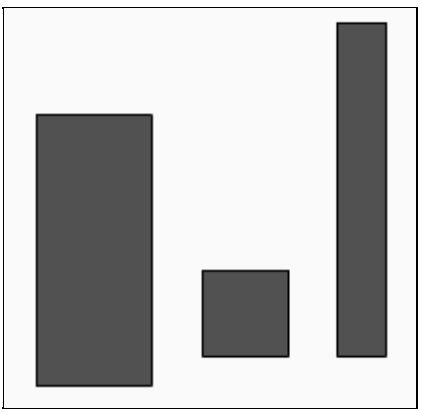


Рис. 4.2. Распределение центров по вертикали. Видно, что центры всех прямоугольников находятся на равном расстоянии друг от друга

С помощью кнопок группы **Match Size** задаются критерии выравнивания горизонтальных и (или) вертикальных размеров выделенных фрагментов.

Они перечислены ниже в порядке слева направо, вместе с соответствующими им пунктами подменю **Align** меню **Modify** и комбинациями клавиш (если они есть):

- все фрагменты должны быть одинаковой ширины (пункт **Make Same Widths**, комбинация клавиш <Ctrl>+<Alt>+<Shift>+<7>);
- все фрагменты должны быть одинаковой высоты (пункт **Make Same Heights**, комбинация клавиш <Ctrl>+<Alt>+<Shift>+<9>);
- все фрагменты должны быть одинаковой ширины и высоты.

С группой кнопок **Space** связаны две операции выравнивания расстояний между выделенными фрагментами по горизонтали и (или) вертикали. Вот они в порядке слева направо:

- вертикальные расстояния между фрагментами должны быть одинаковыми;
- горизонтальные расстояния между фрагментами должны быть одинаковыми.

Последняя группа панели **Align** содержит одну-единственную кнопку-выключатель **To Stage**. Эта кнопка, будучи включенной, применяет операции выравнивания или распределения выделенных фрагментов относительно рабочего листа. То есть если мы применим выравнивание по верхней границе к выделенным фрагментам при включенной кнопке **To Stage**, они будут выровнены не по верхней границе самого верхнего из них, а по верхней границе рабочего листа (см. рис. 4.3 и 4.4). Кнопке **To Stage** соответствуют пункт-выключатель **To Stage** подменю **Align** меню **Modify** и комбинация клавиш <Ctrl>+<Alt>+<8>.

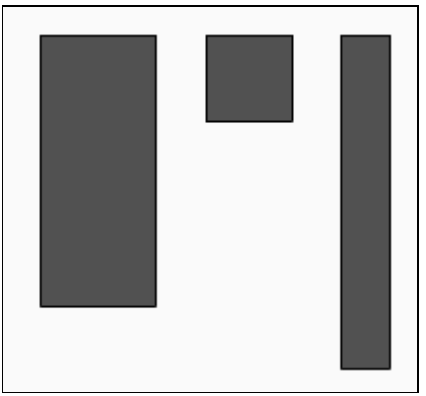


Рис. 4.3. Выравнивание по верхней границе при выключенной кнопке **To Stage**

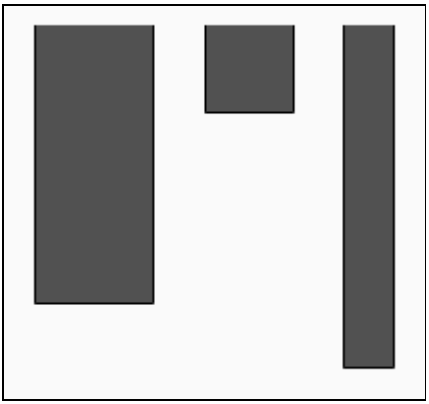


Рис. 4.4. Выравнивание по верхней границе при включенной кнопке **To Stage**

Кстати, при включенной кнопке **To Stage** операцию выравнивания можно применить и в том случае, если выделен только один фрагмент изображения.

Перемещение и изменение размеров

Мы уже знаем, как перемещать выделенные фрагменты по рабочему листу с помощью мыши и клавиш-стрелок. Однако наши знания далеко не полны. Здесь мы рассмотрим еще один способ перемещения выделенной графики — путем задания числовых значений в полях ввода, расположенных на панелях **Properties** и **Info**. Точно так же можно менять размеры выделенного фрагмента графики.

Выделим на рабочем листе какой-либо фрагмент нарисованного изображения и посмотрим на панель **Properties**. В ее нижнем левом углу находится набор из четырех полей ввода (рис. 4.5). С их помощью задается местонахождение выделенного фрагмента и его размеры.

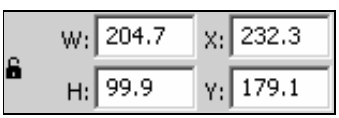


Рис. 4.5. Набор полей ввода **W**, **H**, **X** и **Y** панели **Properties**

В полях ввода **W** и **H** задаются соответственно ширина и высота выделенного фрагмента. В полях ввода **X** и **Y** указываются координаты фрагмента,

горизонтальная и вертикальная. Сразу же после ввода каких-либо значений в эти поля нужно нажать клавишу <Enter>, чтобы Flash выполнил соответствующие изменения на рабочем листе.

Нужно иметь в виду, что величины в эти поля вводятся в тех единицах измерения, которые мы выбрали в диалоговом окне **Document Properties**. Это окно было описано в *главе 2*.

Левее этих полей ввода находится небольшая кнопка-выключатель с изображением замка. Если "замок" включен, Flash при изменении одного из размеров выделенного фрагмента — горизонтального или вертикального — будет изменять другой размер так, чтобы сохранить пропорции данного фрагмента. Такие штуки приходится вытворять очень часто, так что эту кнопку следует принять на вооружение.

Теперь перейдем к панели **Info** (рис. 4.6). Чтобы вызвать ее на экран, следует включить пункт-выключатель **Info** меню **Window** или нажать комбинацию клавиш <Ctrl>+<I>.

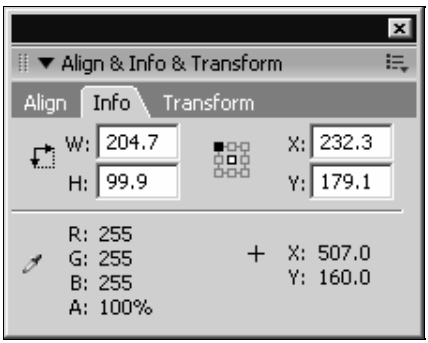


Рис. 4.6. Панель **Info**

Эта панель содержит те же четыре поля ввода, рассмотренные нами выше. Казалось бы, зачем она нужна?

Обратим внимание на значок, находящийся левее полей ввода **X** и **Y**. Это переключатель точки отсчета, позволяющий выбрать точку выделенного фрагмента, относительно которой будут отсчитываться координаты (*точку отсчета*).

Этот переключатель имеет девять точек, которые могут быть выбраны в качестве текущей точки отсчета. В реальности же некоторые из них всегда запрещены — об этом говорит их серая закраска. Доступные точки закрашены белым цветом (на рис. 4.6 есть только одна такая точка — центральная). Та же точка, которая выбрана в данный момент в качестве текущей точки от-

счета, закрашена черным (на рис. 4.6 — верхняя левая). Это значит, что изначально точкой отсчета всех фрагментов является их верхний левый угол.

Чтобы изменить точку отсчета с помощью переключателя, достаточно щелкнуть мышью по одной из доступных точек. Она тотчас станет черной, т. е. выделенной. Правда, для этого придется хорошо "прицелиться", так как точки на переключателе очень малы.

В нашем случае (см. рис. 4.6) в качестве точки отсчета мы можем выбрать либо верхнюю левую, либо центральную точку фрагмента. Какую именно из них выбрать — зависит от каждого конкретного случая.

В нижней половине панели **Info** находятся информационные текстовые поля. Поля **R**, **G**, **B** и **A** показывают параметры цвета точки, находящейся в данный момент под курсором мыши, — долю в этом цвете соответственно красной, зеленой, синей составляющих и уровень прозрачности (подробнее о цвете и работе с ним мы поговорим в *главе 6*). Если под курсором мыши находится свободное пространство рабочего листа, в этих полях отображаются прочерки. А расположенные в нижнем правом углу текстовые поля **X** и **Y** показывают текущие координаты курсора мыши.

Зеркальное отражение

Часто бывает нужно выполнить зеркальное отражение какого-либо фрагмента изображения относительно горизонтальной или вертикальной оси. Flash предоставляет и такую возможность.

Выделим нужный фрагмент и выберем пункт **Flip Horizontal** или **Flip Vertical** подменю **Transform** меню **Modify**. Первый пункт выполняет отражение фрагмента по горизонтальной оси, второй — по вертикальной.

Управление порядком перекрытия

В *главе 3* мы выяснили, что рисуемая во Flash графика подвергается фрагментации и слиянию. Это значит, что, если мы нарисуем эллипс и пересечем его прямой линией, и эллипс, и линия будут рассечены на части, а пересекающиеся одноцветные мазки, сделанные инструментом "кисть", сольются в монолитную фигуру. Это прекрасно видно на рис. 3.28 и 3.29.

Но если при рисовании графики включить режим объектного рисования, задействовав соответствующий модификатор, рисуемые примитивы будут представлять собой независимые фрагменты графики (фактически — группы). Такие фрагменты не будут подвергаться ни фрагментации, ни слиянию. Об этом мы также узнали в *главе 3*.

А что все это значит? А значит это то, что созданные в режиме объектного рисования примитивы могут перекрывать друг друга. Мы можем сами в этом убедиться, нарисовав в этом режиме, скажем, два эллипса и поместив их друг на друга.

Обычно примитивы, нарисованные позже, перекрывают примитивы, нарисованные раньше. Но часто возникает необходимость перенести нарисованные ранее примитивы "наверх" и, наоборот, "упрятать" нарисованные позднее "вниз", т. е. поменять *порядок перекрытия* нарисованных в режиме объектного рисования примитивов.

За порядок перекрытия "отвечают" пункты подменю **Arrange** меню **Modify** (это же подменю, кстати, находится в контекстном меню выделенного примитива). Вот эти пункты:

- **Bring to Front** (комбинация клавиш <Ctrl>+<Shift>+<стрелка вверх>) перемещает выделенный примитив на самый верх;
- **Bring Forward** (комбинация клавиш <Ctrl>+<стрелка вверх>) перемещает выделенный примитив "этажом" выше в порядке перекрытия;
- **Send Backward** (комбинация клавиш <Ctrl>+<стрелка вниз>) перемещает выделенный примитив "этажом" ниже в порядке перекрытия;
- **Send to Back** (комбинация клавиш <Ctrl>+<Shift>+<стрелка вниз>) перемещает выделенный примитив на самый низ.

Повторим еще раз: все эти операции имеют смысл только для фрагментов изображения, созданных в режиме объектного рисования. Обычная графика, нарисованная в режиме графического рисования, не может перекрывать друг друга, так как подвергается фрагментации и слиянию.

Объединение фрагментов изображения

Продолжим разговор о режиме объектного рисования, точнее, о нарисованных в этом режиме примитивах. Что мы с ними еще можем делать?

Мы можем *объединять* такие фрагменты и получать после этого совершенно новые фигуры. Для таких случаев Flash 8 припас для нас подменю **Combine Objects** меню **Arrange**, с которыми мы сейчас познакомимся.

Для пробы давайте создадим в режиме объектного рисования два эллипса с заливками разных цветов и разместим их на рабочем листе так, чтобы они немного накладывались друг на друга (рис. 4.7). После этого выделим оба этих эллипса и поочередно применим к ним пункты вышеупомянутого подменю.

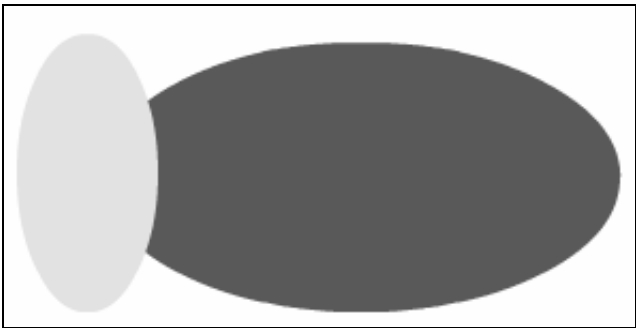


Рис. 4.7. Изначальное изображение,
созданное для иллюстрации различных режимов объединения фрагментов

При выборе пункта **Union** Flash создаст новую фигуру, вмещающую все содержимое изначальных выделенных фрагментов. Эта фигура также станет "объектным" фрагментом.

Кстати, эту операцию мы можем применить и к любому выделенному фрагменту, созданному в режиме графического рисования. В этом случае Flash превратит его в "объектный" фрагмент.

Если выбрать пункт **Intersect**, Flash возьмет пересекающуюся часть выделенных нами фрагментов и на ее основе создаст новую фигуру. При этом пересекающаяся часть будет принадлежать тому из выделенных фрагментов, что располагался выше всех в порядке перекрытия.

Давайте посмотрим на рис. 4.8 — там показана фигура, которую мы получим после применения к нашим эллипсам пункта **Intersect**. Мы видим там фрагмент меньшего эллипса (это видно по более светлой заливке), который был общим с бóльшим эллипсом. Поскольку меньший эллипс был выше большего, Flash взял именно его фрагмент для создания новой фигуры.

Рис. 4.8. Фигура, полученная из изображенной на рис. 4.7
после применения пункта **Intersect**
подменю **Combine Objects** меню **Arrange**

Заметим, что после применения пункта **Intersect** изначальные фрагменты будут удалены. В дальнейшем об этом не следует забывать.

Пункт **Punch**, вероятно, следует применять только к двум выделенным фрагментам — по крайней мере, так можно будет получить более предсказуемый результат. После выбора этого пункта Flash удалит из нижнего в порядке перекрытия фрагмента часть, общую с верхним фрагментом. В результате получится фигура, содержащая "урезанный" нижний фрагмент; верхний фрагмент также будет удален (рис. 4.9).

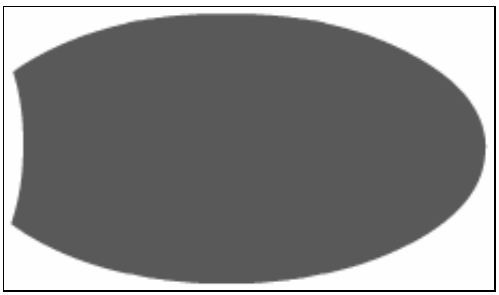


Рис. 4.9. Фигура, полученная из изображенной на рис. 4.7 после применения пункта **Punch** подменю **Combine Objects** меню **Arrange**

А пункт **Crop** оставляет в результирующей фигуре только те части выделенных фрагментов, кроме верхнего, что являются общими с самым верхним фрагментом. Верхний фрагмент при этом удаляется (рис. 4.10).

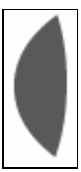


Рис. 4.10. Фигура, полученная из изображенной на рис. 4.7 после применения пункта **Crop** подменю **Combine Objects** меню **Arrange**

Нелишне напомнить, что описанные выше операции имеют смысл только для фрагментов изображения, нарисованных в режиме объектного рисования. Если же нужно применить их к обычному, "графическому" фрагменту, то следует преобразовать его в "объектный" фрагмент, для чего достаточно выделить его и выбрать пункт **Union** подменю **Combine Objects** меню **Arrange**.

Более сложные трансформации

От простого — к сложному. Это вечный путь познания. Последуем им и мы.

Сложные трансформации графики (вращение, сдвиг, изменение размеров и пр.) выполняются во Flash 8 с помощью особого инструмента, называемого **Free Transform** (Трансформатор). Чтобы выбрать его, достаточно щелкнуть по кнопке главного инструментария или нажать клавишу <Q>. Курсор мыши примет вид черной стрелки.

Теперь нам останется только щелкнуть на нужном фрагменте изображения, чтобы его выделить. И читать дальше.

Изменение размеров

Мы уже знаем, как можно изменить размеры выделенного фрагмента. Для этого достаточно указать новые значения в полях ввода **W** и **H**, находящихся в панели **Properties** (см. рис. 4.5) и **Info** (см. рис. 4.6), и нажать клавишу <Enter>. В общем, ничего сложного.

Однако изменить размеры фрагмента графики можно намного проще и удобнее — с помощью мыши. Для этого используется модификатор **Scale** (Изменение размера), доступный при выбранном инструменте "трансформатор". Чтобы включить его, нужно щелкнуть по кнопке-выключателю .

Если главного инструментария на экране нет, можно воспользоваться главным меню. Для этого нам даже не придется выбирать инструмент "трансформатор". Просто включаем пункт-выключатель **Scale** подменю **Transform** меню **Modify** — и можем управлять размерами выделенного фрагмента.

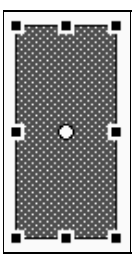


Рис. 4.11. Выделенный фрагмент изображения при включенном модификаторе "изменение размера"

Посмотрим теперь на выделенный нами фрагмент графики. Он должен выглядеть так, как показано на рис. 4.11, — он словно вписан в уже знакомый

нам по *главе 3* прямоугольник выделения, на сторонах и углах которого на этот раз расположены черные квадратики — *маркеры изменения размеров*.

Чтобы изменить какой-либо из геометрических размеров выделенного фрагмента, достаточно перетащить мышью нужный маркер. Маркеры на вертикальных сторонах прямоугольника выделения меняют ширину фрагмента, а маркеры на горизонтальных сторонах — его высоту. Если нам нужно изменить одновременно и ширину, и высоту фрагмента, мы воспользуемся одним из маркеров, расположенных в углах (на вершинах) прямоугольника выделения. А если нам нужно сделать то же самое, но без искажения пропорций фрагмента, мы начнем перетаскивать угловой маркер, нажмем клавишу <Shift> и будем удерживать ее, пока не закончим перетаскивание.

Есть еще один способ изменения размеров выделенного фрагмента — воспользоваться панелью **Transform** (рис. 4.12). Чтобы вывести ее на экран, мы включим пункт-выключатель **Transform** меню **Window** или нажмем комбинацию клавиш <Ctrl>+<T>.

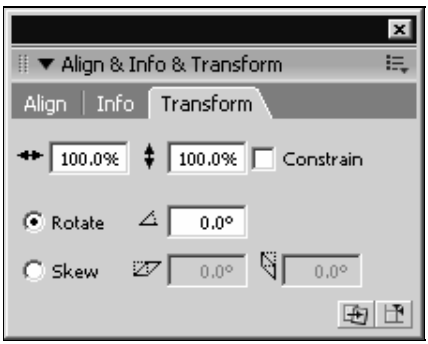


Рис. 4.12. Панель **Transform**

Использование этой панели чем-то напоминает использование панели **Info**. Нужные размеры фрагмента вводятся в два верхних поля ввода: горизонтальный размер — в левое, вертикальный — в правое. Причем размеры эти указываются не в обычных единицах измерения, заданных нами при установке параметров документа Flash (*см. главу 2*), а в процентах относительно текущего размера. Мы можем вводить значения от 1 до 1000%. После ввода не забудем нажать клавишу <Enter>.

Если мы включим флажок **Constrain**, расположенный правее этих полей ввода, Flash будет сохранять пропорции фрагмента. При этом если мы введем в одно поле ввода новое значение, содержимое второго поля ввода изменится — Flash сам вычислит новое значение, чтобы соблюсти эти пропорции.

Вращение и сдвиг

Вращение и сдвиг выделенного фрагмента также осуществляются с помощью инструмента "трансформатор". Однако модификатор для этого используется другой — **Rotate and Skew** (Вращение и сдвиг). Он включается с помощью кнопки-выключателя . Также можно включить пункт-выключатель **Rotate and Skew** подменю **Transform** меню **Modify**.

Выделенный фрагмент графики имеет при этом вид, показанный на рис. 4.11, т. е. такой же, как при выбранном модификаторе "изменение размеров". Он также вписан в прямоугольник выделения, но имеет другие маркеры, выполняющие другие функции. Так, на углах прямоугольника выделения расположены квадратные *маркеры вращения*, а на сторонах — *маркеры сдвига*.

Чтобы выполнить вращение выделенного фрагмента, мы перетащим мышью один из угловых маркеров — неважно, какой. Чтобы повернуть фрагмент на угол, кратный 45°, при перетаскивании маркера будем удерживать нажатой клавишу <Shift>. Чтобы выполнить Сдвиг, мы перетащим мышью маркер, находящийся на нужной стороне прямоугольника: горизонтальной — для сдвига по горизонтали или вертикальной — для сдвига по вертикали.

В центре прямоугольника выделения находится белая точка. Это центр вращения фрагмента, вокруг него осуществляется поворот. Его также называют *точкой фиксации*. Мы можем перетащить точку фиксации в другое место и даже вынести за пределы прямоугольника. Чтобы вернуть ее обратно в центр прямоугольника выделения, следует дважды щелкнуть на ней мышью.

Если нужно быстро повернуть выделенный фрагмент на 90°, проще всего воспользоваться главным меню. Пункт **Rotate 90° CW** подменю **Transform** меню **Modify** поворачивает фрагмент на 90° по часовой стрелке, а пункт **Rotate 90° CWW** — против часовой стрелки. Вместо выбора этих пунктов можно нажать комбинацию клавиш <Ctrl>+<Shift>+<9> и <Ctrl>+<Shift>+ +<7> соответственно.

И, конечно же, мы можем повернуть или сдвинуть фрагмент, воспользовавшись панелью **Transform** (см. рис. 4.12). Выведем ее на экран. После этого включим переключатель **Rotate** и введем в расположенное справа от него поле ввода нужный угол поворота. Чтобы выполнить сдвиг, нам нужно будет включить переключатель **Skew** и ввести в расположенные справа от него поля ввода нужные углы сдвига: горизонтального — в левое поле ввода, вертикального — в правое. Теперь остается нажать клавишу <Enter> — и дело сделано.

Если мы хотим одновременно повернуть графический фрагмент и изменить его размеры, мы можем использовать особую функцию Flash. Выберем пункт **Scale and Rotate** подменю **Transform** меню **Modify** или нажмем комбинацию клавиш <Ctrl>+<Alt>+<S>. На экране появится диалоговое окно

Scale and Rotate (рис. 4.13). Зададим в поле ввода **Scale** новый масштаб фрагмента в процентах, а в поле ввода **Rotate** — угол поворота в градусах. После этого останется только нажать кнопку **OK**; если же мы хотим отказаться от манипуляции над фрагментом, нажмем кнопку **Cancel**.

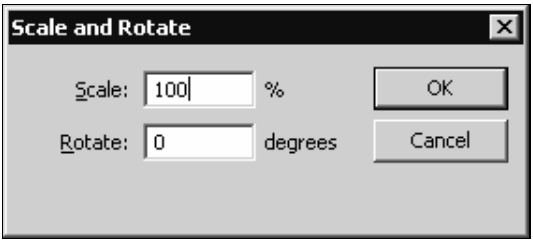


Рис. 4.13. Диалоговое окно **Scale and Rotate**

Искажение формы

На взгляд автора, возможность искажения формы во Flash включена напрасно, и модификатор **Distort** (Искажение формы) на самом деле не так уж нужен. Все операции, которые он предоставляет в распоряжение пользователя, можно проделать, используя хорошо знакомый нам по *главе 3* инструмент "стрелка". Хотя, может быть, этот модификатор позволяет начинающим пользователям изменить форму контура более простым и наглядным способом. Поэтому мы его все же рассмотрим.

Чтобы включить модификатор "искажение формы", нужно нажать кнопку-выключатель . Также можно включить пункт-выключатель **Distort** подменю **Transform** меню **Modify** или одноименный пункт контекстного меню выделенного фрагмента. Выделенный нами фрагмент станет таким, как показано на рис. 4.14. В углах и на сторонах прямоугольника выделения появятся *маркеры искажения формы*, имеющие вид черных квадратиков.

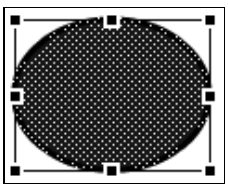


Рис. 4.14. Фрагмент изображения, выделенный при включенном модификаторе "искажение формы"

Мы можем перемещать маркеры, расположенные на сторонах прямоугольника выделения. При этом будут изменяться размеры графического фрагмента. Если же мы переместим угловой маркер, форма фрагмента исказится — "вытянется" или "втянется" соответственно. Например, из обычного квадрата можно, "вытянув" один его угол и "втянув" противоположный, получить фигуру, показанную на рис. 4.15.

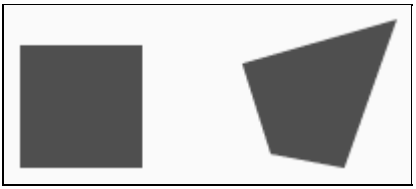


Рис. 4.15. Результат искажения формы квадрата (слева — изначальный квадрат, справа — искаженный)

Деформация

Также во Flash имеются средства, позволяющие произвольно деформировать графические фрагменты. Это модификатор **Envelope** (Огибающая) инструмента "трансформатор". Сейчас мы выясним, как им пользоваться.

Чтобы включить модификатор "огибающая", нужно нажать кнопку-выключатель . Также можно включить пункт-выключатель **Envelope** в подменю **Transform** меню **Modify** или одноименный пункт контекстного меню выделенного фрагмента. После этого фрагмент примет вид, показанный на рис. 4.16.

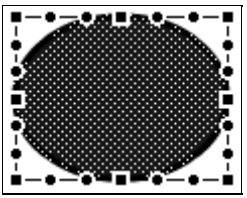


Рис. 4.16. Фрагмент изображения, выделенный при включенном модификаторе "огибающая"

В этом случае прямоугольник выделения имеет много *маркеров огибающей*. Принцип работы с ними можно выразить одной фразой: выбираем нужный маркер и тащим его мышью. В результате этого выделенный фрагмент будет

искажаться. Например, показанный на рис. 4.16 эллипс можно превратить в нечто, совсем не напоминающее эллипс — см. рис. 4.17.

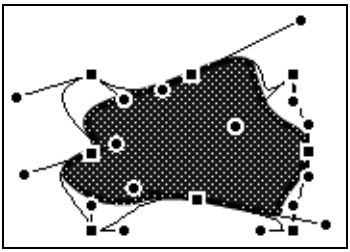


Рис. 4.17. Эллипс, показанный на рис. 4.16, после деформации

Вообще, стоит поэкспериментировать с модификатором "огибающая", чтобы понять, как он работает. Нарисуйте несколько фигур и испытайте этот модификатор в действии.

Свободная трансформация

Если при выбранном инструменте "трансформатор" не включен ни один модификатор, то Flash по широте душевной включает все эти модификаторы одновременно. В этом случае говорят о так называемой *свободной трансформации*. При свободной трансформации мы можем как угодно изменять выделенный фрагмент, не заботясь о том, какой модификатор нужно включить. Однако работать в этом случае становится довольно сложно.

Чтобы выбрать свободную трансформацию, проще всего отключить все кнопки-выключатели, "ответственные" за модификаторы инструмента "трансформатор". Все эти кнопки находятся в области **Options** главного инструментария. Также мы можем включить пункт-выключатель **Free Transform** подменю **Transform** меню **Modify** или контекстного меню выделенного фрагмента. Выделенный фрагмент примет вид, показанный на рис. 4.11.

Чтобы выполнить то или иное преобразование, поместим курсор мыши над нужным маркером прямоугольника выделения. Курсор мыши примет при этом "говорящую" форму. Скорее всего, нам придется некоторое время подвигать курсор возле разных маркеров, прежде чем он эту форму примет. После этого "захватим" маркер и переместим его на новое место, как и в случае других модификаторов.

По мнению автора, свободная трансформация более пригодна для достаточно опытных художников. Начинающим же поначалу будет удобнее пользо-

ваться различными модификаторами инструмента "трансформатор". (Вероятно, так же считают и разработчики пакета Flash, если они предусмотрели в своем детище все эти модификаторы.)

Дополнительные возможности

В этом разделе описаны некоторые дополнительные возможности по трансформации графики, которые могут нам пригодиться.

Трансформация копии графического фрагмента

Иногда требуется не просто выполнить какую-либо трансформацию над выделенным фрагментом, а создать его трансформированную копию в буфере обмена Windows, чтобы потом вставить ее в этот же или другой документ. Flash предоставляет такую возможность с одним ограничением: поддерживаются только трансформации, доступные в панели **Transform** (см. рис. 4.12): изменение размеров, вращение и сдвиг. Но, как говорится, и на том спасибо...

Воспользовавшись панелью **Transform**, зададим нужные трансформации. После этого нажмем кнопку — она расположена в правом нижнем углу этой панели. В буфере обмена будет создана трансформированная копия выделенного фрагмента, в то время как сам выделенный фрагмент не изменится.

Сброс всех трансформаций, примененных к фрагменту

Когда мы с помощью панели **Transform** применяем какую-то трансформацию к выделенному фрагменту — вращаем его, сдвигаем или меняем его размеры, — Flash сохраняет в памяти его изначальное состояние. Поэтому, если мы захотим вернуться к исходному фрагменту — такому, каким он был до этой трансформации (выполнить *сброс трансформаций*), нам достаточно будет выбрать пункт **Remove Transform** подменю **Transform** меню **Modify** или нажать комбинацию клавиш <Ctrl>+<Shift>+<Z>. Также мы можем нажать кнопку , расположенную в правом нижнем углу панели **Transform**. После этого мы сможем сказать своему многострадальному фрагменту словами старой песни: "каким ты был, таким остался".

ВНИМАНИЕ!

Имейте в виду, что таким образом можно отменить только самую последнюю примененную к фрагменту трансформацию. Все предыдущие трансформации отмене не подлежат. Кроме того, повторим, таким способом возможно сбросить только те трансформации, которые были выполнены с помощью панели **Transform**.

Блокировка фрагмента изображения

Очень часто бывает нужно выполнить какие-либо манипуляции с фрагментом, являющимся частью сложного изображения. При этом другие части этого изображения, расположенные рядом, мешают выполнить эти манипуляции; если мы попытаемся, например, сдвинуть фрагмент, то существует риск сдвинуть и соседние. Бывает и наоборот: нужно выполнить манипуляции над "соседями" какого-либо фрагмента, не затронув сам этот фрагмент. А поскольку изображение очень сложное, сделать это часто весьма затруднительно.

Специально для таких случаев Flash предоставляет возможность *заблокировать* какой-либо фрагмент, а именно, сделать его не выделяемым инструментом "стрелка" и немодифицируемым. Заблокированный фрагмент ведет себя как фон рабочего листа: мы можем щелкать на нем мышью без всякого риска для себя и для него.

Чтобы заблокировать какой-либо фрагмент, прежде всего, следует преобразовать его в группу, т. е. сгруппировать. Для этого достаточно выделить нужный фрагмент и выбрать пункт **Group** меню **Modify** или нажать комбинацию клавиш <Ctrl>+<G>. Разумеется, фрагменты, созданные при включенном режиме объектного рисования, группировать совершенно не обязательно.

Теперь выберем пункт **Lock** подменю **Arrange** меню **Modify** или нажмем комбинацию клавиш <Ctrl>+<Alt>+<L>. После этого фрагмент (точнее, созданная на его основе группа) будет заблокирован. Попробуйте щелкнуть на нем мышью — он никак не прореагирует.

Закончив нужные действия, этот фрагмент следует разблокировать. Для этого достаточно выбрать пункт **Unlock All** подменю **Arrange** меню **Modify** или нажать комбинацию клавиш <Ctrl>+<Alt>+<Shift>+<L>. Ну, а для разгруппировки его служат пункт **Ungroup** меню **Modify** и комбинация клавиш <Ctrl>+<Shift>+<G>.

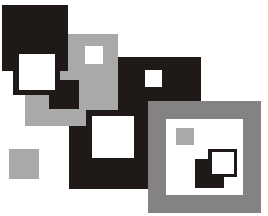
Заметим, что пункт меню, предназначенный для разблокирования фрагмента, носит название **Unlock All**, а не **Unlock** (т. е. "разблокировать все", а не "разблокировать"). Так как заблокированный фрагмент нельзя выделить, мы

не можем разблокировать отдельно взятый фрагмент, оставив другие блокированными. Поэтому Flash разблокирует сразу все заблокированные ранее фрагменты.

Что дальше?

Вот мы и закончили знакомство с возможностями Flash 8 по трансформации графики. Их немало, что и говорить.

Познакомимся теперь еще с одной возможностью Flash — слоями, служащими для разделения фрагментов изображения с целью избежать их фрагментации и слияния. Без него не обойтись при создании анимации, о которой пойдет речь в *части III* этой книги. Слоям полностью посвящена следующая глава.



Глава 5

Работа со слоями

В этой главе речь пойдет о слоях Flash и их использовании.

Слой (в терминологии Flash — layer) можно рассматривать как лист прозрачной пленки, лежащий на рабочем листе Flash. Мы можем рисовать на любом слое, используя изученные в *главе 3* инструменты рисования. При этом нарисованная на слое графика будет скрывать все, что находится на нижележащих слоях точно под ней.

Слои помогут нам разнести различные фрагменты изображения так, что они не будут ни фрагментироваться, ни сливаться, поскольку они не будут соприкасаться друг с другом. (О фрагментации и слиянии *см. главу 3.*) Пожалуй, это лучшее решение проблемы нежелательной фрагментации и ненужного слияния. Кроме того, слои очень активно применяются при создании трансформационной анимации, речь о которой пойдет в *главах 12* и *14.*

Слои сами по себе не увеличивают размер файла документа Flash и результирующего файла Shockwave/Flash. Увеличивает его только графика, расположенная в этих слоях.

Создание слоев и управление ими

Когда в *главе 2* мы рассматривали окно документа Flash, то ограничились только собственно рабочей областью. Теперь самое время приняться за панель **Timeline**, о которой мы ранее только упомянули. Если эта панель почему-то отсутствует, следует включить пункт-выключатель **Timeline** меню **Window** или нажать комбинацию клавиш <Ctrl>+<Alt>+<T>.

В данный момент нам больше всего пригодится левая часть панели **Timeline**, в которой находится нечто, напоминающее собой таблицу из четырех колонок (рис. 5.1). Это список всех слоев, присутствующих в документе.

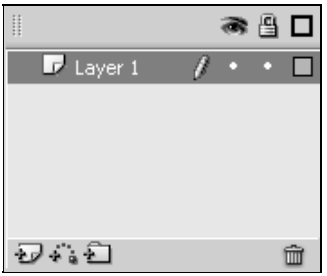


Рис. 5.1. Список слоев документа, находящийся в левой части панели **Timeline**

Если список слоев слишком узок, мы можем его расширить. Для этого достаточно переместить вправо толстую серую линию, отделяющую его от правой части панели **Timeline**.

Список слоев организован в виде таблицы. В самой левой его колонке перечислены уникальные имена слоев — по этим именам слои однозначно идентифицируются Flash.

В данный момент (см. рис. 5.1) документ содержит всего один слой под именем `Layer 1`. Этот слой автоматически формируется самим Flash при создании нового документа. Разумеется, если нам понадобятся другие слои, мы всегда сможем их добавить.

Слои в списке располагаются в том порядке, в каком они "сложены" на рабочем листе. Графика, нарисованная на вышележащих слоях, будет при этом перекрывать графику, нарисованную точно под ней на нижележащих слоях. Сами же слои прозрачные, они ничего собой не закрывают.

Мы можем выделить нужный слой в списке, для чего достаточно на нем щелкнуть. После выделения слоя вся помещенная на нем графика будет выделена. И наоборот, если мы выделим какой-либо фрагмент графики, в списке слоев панели **Timeline** будет автоматически выделен слой, на котором находится этот фрагмент.

Мы также можем использовать множественное выделение. Если нужно выделить непрерывную группу слоев, мы щелкнем на первом слое выделяемой группы, нажмем клавишу <Shift> и, удерживая ее, щелкнем на последнем слое. Если же нам понадобится выделить несколько несвязанных слоев, то мы сначала щелкнем на первом, а потом продолжим щелкать на остальных слоях, удерживая нажатой клавишу <Ctrl>. (Кстати, такая же методика применяется для выделения файлов в Проводнике Windows.)

Чтобы создать новый слой, проще всего нажать кнопку **Insert Layer** ( ), расположенную под списком слоев. Также можно выбрать пункт **Layer** под-

меню **Timeline** меню **Insert** или пункт **Insert Layer** контекстного меню выделенного слоя. Новый слой будет помещен в список над ранее выделенным слоем и сам автоматически станет выделенным.

Всем вновь создаваемым слоям Flash дает имена по умолчанию вида `Layer <Порядковый номер>`. Чтобы переименовать слой, достаточно дважды щелкнуть на его имени, после чего там появится небольшое поле ввода, в котором будет видно старое имя слоя. Вводим туда новое имя и нажимаем клавишу <Enter> для его сохранения или клавишу <Esc> для отказа от переименования слоя. Имена слоев могут включать практически любые символы: русские и латинские буквы, цифры, пробелы и знаки подчеркивания.

Мы можем перемещать слои в списке, меняя порядок перекрытия одного другим. Для этого достаточно просто перетащить мышью нужный слой или выделенную группу слоев на новое место.

Удалить ненужный слой можно разными способами. Проще всего выделить слой, который требуется удалить, и нажать кнопку **Delete Layer** ( ), расположенную в нижней части списка слоев. Также можно перетащить ненужный слой прямо на эту кнопку. Ну и, наконец, можно выбрать пункт **Delete Layer** в контекстном меню выделенного слоя.

> ***ВНИМАНИЕ!***
>
> Слой удаляется вместе со всей расположенной в нем графикой.

Flash позволяет удалить любой слой, созданный пользователем. Но самый первый слой, образованный самим Flash при создании нового документа, удалить невозможно.

А теперь давайте посмотрим еще раз на рис. 5.1. Как мы уже знаем, он представляет собой таблицу из четырех колонок. Все они перечислены ниже в порядке слева направо:

- имя слоя;
- видимость слоя (помечена изображением глаза);
- блокировка слоя (помечена изображением замка);
- отображение содержимого слоя разными цветами (помечена черным прямоугольником).

Колонка имени слоя нам уже знакома. А зачем нужны остальные?

Вторая колонка позволяет управлять *видимостью* слоя, т. е. будет ли содержимое слоя отображаться в рабочей области. Иногда бывает полезно скрыть некоторые слои, чтобы расположенная на них графика не мешала работе. Для этого достаточно щелкнуть мышью на точке, находящейся в этой колонке, — и вместо точки появится красный крестик, означающий, что данный слой скрыт. Чтобы вновь "открыть" скрытый слой, следует щелкнуть на

красном крестике; крестик исчезнет, и слой со всем его содержимым снова станет видимым.

Третья колонка предоставляет возможность *заблокировать* слой, после чего графика, находящаяся в таком слое, станет недоступной для выделения и правки. Это может понадобиться, например, если мы захотим исправить часть сложного изображения, расположенную в одном слое, не затрагивая содержимое остальных слоев. Чтобы заблокировать слой, опять же, нужно щелкнуть мышью на точке, находящейся в этой колонке. После щелчка вместо точки появится красный крестик, и слой будет заблокирован. Этот крестик вместе с блокировкой слоя пропадет при следующем щелчке мышью.

Последняя — четвертая — колонка позволяет отобразить содержимое слоя схематично, в виде одних только контуров, окрашенных в какой-либо один цвет (*схематичный режим* слоев). Это часто помогает правильно определить, в каком слое находится тот или иной фрагмент изображения. Включается и отключается схематичное отображение все так же — щелчком мышью.

Если щелкнуть, скажем, на точке во второй колонке, удерживая клавишу <Ctrl>, то будут скрыты сразу все слои. Такого же результата можно добиться, если щелкнуть на значке "глаз" в заголовке колонки. Если же щелкнуть на точке в колонке, удерживая клавишу <Alt>, то будут скрыты все слои, кроме того, которому "принадлежит" эта точка. Все эти приемы действуют и для других колонок списка слоев.

Если дважды щелкнуть на значке, находящемся левее имени слоя, на экране появится диалоговое окно **Layer Properties** (рис. 5.2), позволяющее задать различные параметры данного слоя. Это окно можно вызвать также, выбрав пункт **Layer Properties** подменю **Timeline** меню **Modify** или пункт **Properties** контекстного меню выделенного слоя.

В поле ввода **Name** задается имя слоя. Флажок **Show** включает или отключает отображение содержимого слоя на рабочем листе, т. е. управляет видимостью слоя. Флажок **Lock** позволяет заблокировать слой. А флажок **View layer as outlines** включает или отключает схематичный режим для данного слоя. Все это нам уже знакомо.

С помощью селектора цвета **Outline color** мы можем задать цвет, которым будут отображаться контуры графики, если для данного слоя включен схематичный режим. Впрочем, этим селектором цвета мы вряд ли будем часто пользоваться: Flash и сам довольно удачно распределяет цвета по соответствующим слоям.

С помощью раскрывающегося списка **Layer height** мы можем задать высоту строки списка соответствующей данному слою. Этот раскрывающийся список содержит три пункта: **100%** (значение по умолчанию), **200%** и **300%**. На рис. 5.3 показан слой `Layer 3`, представленный в списке строкой утроенной

высоты (пункт **300%** раскрывающегося списка **Layer height**). Такая возможность пригодится нам при работе со звуком (*см. главу 17*).

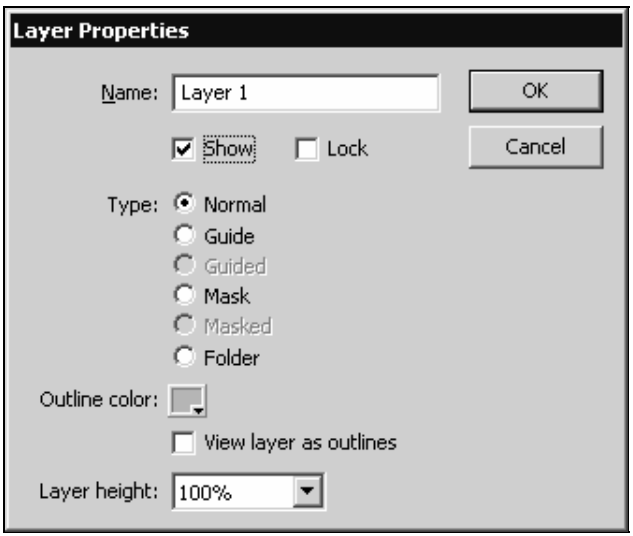


Рис. 5.2. Диалоговое окно **Layer Properties**

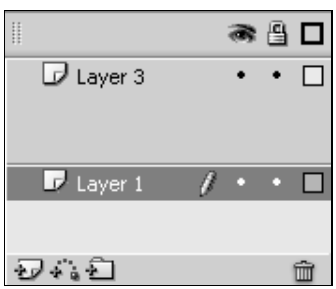


Рис. 5.3. Слой, представленный строкой утроенной высоты

С помощью набора переключателей **Type** задается *тип* слоя. Выбор переключателя **Normal** делает слой *обычным*, т. е. слоем, содержащим некую графику и не имеющим какого-то специального назначения. Остальные поддерживаемые Flash типы слоев, имеющие специальное назначение, мы рассмотрим в *главе 14*.

Задав нужные параметры слоя, следует нажать кнопку **OK**. Чтобы отменить их, достаточно нажать кнопку **Cancel**.

Использование папок для организации слоев

Для организации наборов файлов на диске компьютера применяются папки. Папки позволяют собирать вместе файлы, относящиеся к одной теме: отчеты к отчетам, личные письма к личным письмам, фотографии любимого бульдога к фотографиям любимой тещи. Правильная организация папок позволяет держать файлы в порядке и быстро их находить.

Для организации слоев Flash в иерархическую структуру также применяются *папки*. Используя папки, мы можем разложить по отдельным "полочкам" слои, содержащие похожие элементы изображения: надписи к надписям, кнопки к кнопкам, растровые изображения к растровым изображениям и т. д. Благодаря папкам мы сможем навести порядок в сложном документе Flash и не запутаться в нем, если понадобится что-то срочно найти.

Чтобы создать папку, нужно нажать кнопку **Insert Layer Folder** ( ), расположенную в нижней части списка слоев. Также можно выбрать пункт **Layer Folder** подменю **Timeline** меню **Insert** или пункт **Insert Folder** контекстного меню выделенного слоя. Вновь созданная папка появится выше выделенного слоя (рис. 5.4) и получит имя по умолчанию вида `Folder <Порядковый номер>`.

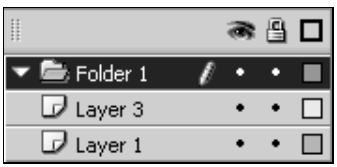


Рис. 5.4. Папка списка слоев (самый верхний пункт списка)

Чтобы переместить слой или несколько слоев в папку, просто выделим их и перетащим мышью на строку списка, обозначающую данную папку, или на один из слоев, уже находящихся в папке. Папка с помещенными в нее слоями показана на рис. 5.5.

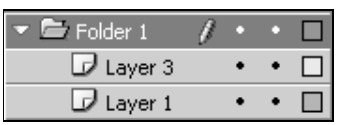


Рис. 5.5. Папка с помещенными с нее слоями `Layer 1` и `Layer 3`

Чтобы извлечь слой из папки, мы просто "вытащим" его мышью наружу. Точно таким же образом мы можем перемещать слои из одной папки в другую. И, конечно же, мы можем перемещать слои внутри папки.

Выделяются папки точно так же, как слои, причем мы можем использовать как обычное, так и множественное выделение. При выделении папки также выделяются все находящиеся в ней слои.

Flash также поддерживает создание папок, вложенных в другие папки. Чтобы создать вложенную папку, просто выделим любой слой, находящийся в папке, и нажмем кнопку **Insert Layer Folder** или выберем один из описанных выше пунктов меню.

Также мы можем переместить уже существующую папку в другую, перетащив ее мышью. При этом будут также перемещены все слои, находящиеся в папке.

Мы можем переименовывать папки, блокировать, управлять видимостью и переключать в схематичный режим все находящиеся в них слои. Эти операции для папок выполняются точно так же, как и для слоев.

Левее имени папки находится значок папки, а еще левее — небольшая треугольная стрелка. Если эта стрелка указывает вниз, то папка раскрыта, и мы можем наблюдать все содержащиеся в ней слои. Если же эта стрелка указывает вправо, то папка закрыта. Щелкая стрелку мышью, мы можем раскрывать и закрывать папку. Также мы можем выбрать в контекстном меню папки или находящегося в ней слоя пункт **Expand Folder** (развернуть папку) или **Collapse Folder** (свернуть папку). А пункты **Expand All Folders** (развернуть все папки) и **Collapse All Folders** (свернуть все папки) позволят нам соответственно развернуть или свернуть все папки списка слоев.

Чтобы удалить ненужную папку, нужно ее выделить и нажать уже знакомую нам кнопку **Delete Layer** — она имеет власть и над папками. Также можно выбрать пункт **Delete Folder** в контекстном меню выделенной папки. После этого Flash выведет окно-предупреждение, сообщающее о том, что папка будет удалена вместе со всеми находящимися в ней слоями. Для удаления папки нужно нажать **Yes**, для отказа от удаления — кнопку **No**.

И, наконец, мы можем превратить слой в папку. Для этого нужно вызвать на экран диалоговое окно **Layer Properties** (см. рис. 5.2), выбрать в группе **Type** этого окна переключатель **Folder** и нажать кнопку **OK**. Нужно только помнить, что при этом содержимое слоя безвозвратно пропадет.

Также есть возможность превратить папку в слой. Для этого требуется выбрать переключатель **Normal** группы **Type** диалогового окна **Layer Properties**. При этом слои, вложенные в преобразуемую папку, останутся нетронутыми.

Распределение графики по слоям

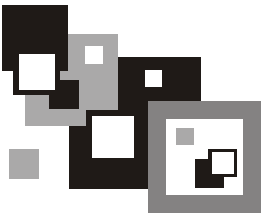
Иногда бывает нужно "разбросать" фрагменты изображения, находящиеся в одном слое, по разным слоям. В терминологии Flash эта операция называется *распределением графики по слоям*. При этом Flash сам создаст нужное количество слоев и переместит на них выделенные нами фрагменты: каждый фрагмент на "свой" слой.

Чтобы выполнить распределение фрагментов изображения по слоям, нужно сначала выделить их. Если эти фрагменты сгруппированы, разбивать их не нужно — Flash достаточно умен, чтобы выполнить это самостоятельно. После этого следует выбрать пункт **Distribute to Layers** подменю **Timeline** меню **Modify** или в контекстном меню выделенных фрагментов или нажать комбинацию клавиш <Ctrl>+<Shift>+<D>.

Что дальше?

Вот мы и закончили разговор о слоях. Точнее, прервали. Когда мы будем учиться создавать во Flash анимацию, мы вернемся к слоям.

Следующая глава будет посвящена средствам, предоставляемым Flash для управления цветами. Вы удивитесь, насколько они мощные! А иначе и быть не могло — ведь художник пользуется красками, и красок у него бывает очень много.



Глава 6

Работа с цветом и стилем линий

Что такое рисование? В основном — работа с цветом. Настроение любой картины создается в значительной степени соответствующим подбором цветов. Более того, настоящий художник должен иметь превосходное цветовое зрение, чтобы различить и перенести на бумагу (холст, дерево или иной носитель) все оттенки красок окружающего мира. И любой — и "бумажный", и компьютерный — художник сам готовит себе краски.

Именно поэтому разговор о цветах и управлении ими автор вынес в отдельную главу. Именно поэтому он не описал полностью работу с цветами в *главе 3*, рассказывающей о рисовании, а ограничился только базовыми средствами, без которых на первых порах не обойтись. Базовые средства хороши только для начинающих художников, рисующих на рабочем листе свои первые фигуры и не слишком озабоченных подбором цветов. Приготовить свои "краски" с их помощью вы не сможете.

В этой главе мы поведем разговор о средствах, предлагаемых Flash для работы с цветом. Мы подробнее рассмотрим уже знакомый нам селектор цвета, дополнительные инструменты и специальные панели.

Кроме того, мы рассмотрим некоторые инструменты для задания стиля линий. В терминологии Flash это значит, будет ли линия сплошной или штриховой, как будет оформлено пересечение линий и какой вид будут иметь их концы.

Инструменты выбора цвета

Но начнем мы отнюдь не с приготовления "красок". Первым делом мы поговорим о различных инструментах выбора цвета, доступных во Flash. По-

мимо области задания цвета главного инструментария и панели **Properties**, для этого используется особая панель **Color Mixer**, с помощью которой можно также создавать новые цвета, не существующие в селекторе. Да и сам селектор цвета таит в себе много сюрпризов.

Использование селектора цвета

Наличие селектора цвета — отличительная черта всех продуктов фирмы Macromedia. С ним мы уже знакомы. Мы умеем выбрать цвет с его помощью. Но всех его возможностей мы еще не знаем.

Давайте взглянем на рис. 6.1, на котором изображен открытый селектор цвета, находящийся в области задания цвета главного инструментария. Как его открыть, мы знаем еще из *главы 3*.

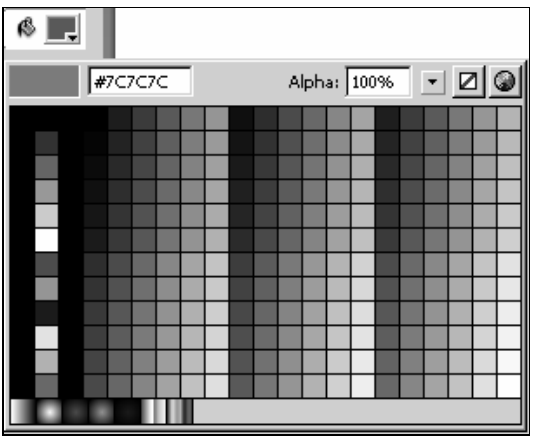


Рис. 6.1. Открытый селектор цвета

Мы видим, что бóльшая часть окна селектора цвета занята набором цветных квадратиков, представляющих все доступные для выбора цвета, — так называемой *палитрой*. Чтобы выбрать требуемый цвет, нужно поместить курсор над соответствующим квадратиком и щелкнуть левой кнопкой мыши. Окно селектора цвета после этого закроется. Закрыть его без выбора цвета можно нажатием клавиши <Esc>.

Однако селектор цвета предоставляет еще одну интересную возможность. Давайте откроем его и присмотримся к курсору. Нетрудно заметить, что он после этого примет вид небольшой пипетки. Щелкнем этой "пипеткой" в нужной точке экрана — и цвет этой точки будет выбран селектором. Таким

образом мы можем "позаимствовать" для работы цвет любого окна Windows, выбрать цвет из открытой в графическом редакторе фотографии и т. п.

В верхней части окна селектора находится небольшое поле ввода. В этом поле ввода отображается шестнадцатеричный *код цвета* той точки, которая в данный момент находится под курсором мыши. Все цвета, отображаемые на экране компьютера, однозначно идентифицируются их кодами. Попробуем перемещать курсор (имеющий вид пипетки) по экрану и посмотрим, как будет изменяться содержимое этого поля ввода. Разумеется, мы также можем ввести нужное значение кода цвета вручную, если, конечно, оно нам известно.

О кодах цветов мы еще поговорим в этой главе. А пока вернемся к селектору цвета.

В верхнем правом углу открытого окна селектора цвета находится небольшая кнопка . С ее помощью мы можем вызвать на экран стандартное диалоговое окно выбора цвета Windows (рис. 6.2). Это может понадобиться, если мы не смогли найти в палитре селектора нужного нам цвета. Как пользоваться этим окном, описано в интерактивной справке Windows.

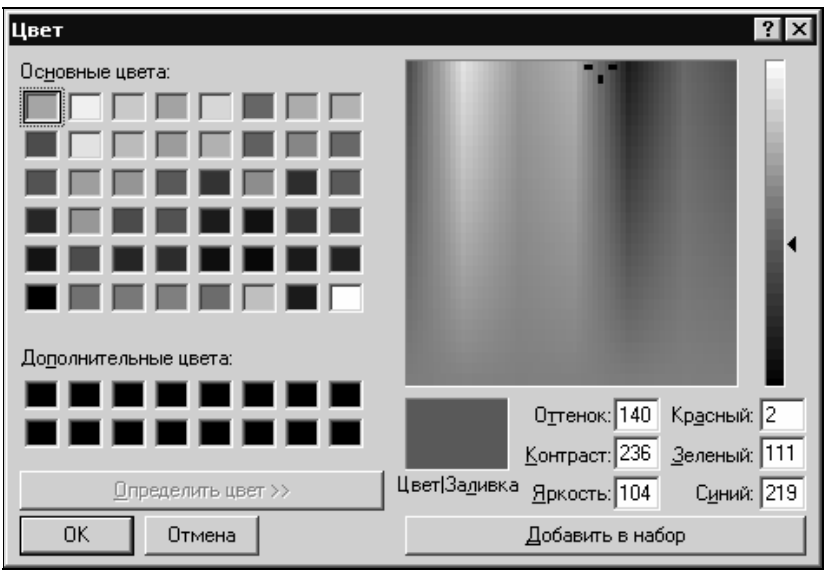


Рис. 6.2. Стандартное диалоговое окно выбора цвета Windows

Если в главном инструментарии выбран инструмент "прямоугольник", "эллипс" или "перо", то в верхнем правом углу окна селектора цвета появит-

ся кнопка . Это *кнопка отключения цвета*, позволяющая запретить рисование контура или заливки, в зависимости от того, в каком селекторе цвета она была нажата. То есть если мы откроем селектор цвета фона, нажмем эту кнопку и нарисуем прямоугольник, последний не будет содержать заливку.

Если кнопка отключения цвета была нажата, то соответствующий селектор цвета будет перечеркнут (рис. 6.3) — "отключен". Чтобы снова "включить" его, достаточно открыть его и выбрать любой цвет.

Рис. 6.3. "Отключенный" селектор цвета

Область *Colors* главного инструментария

Теперь посмотрим на область задания цветов **Colors** главного инструментария. Она тоже таит немало полезных сюрпризов.

В нижней части этой области находится еще одна кнопка отключения цвета. Она имеет такое же обозначение, как и аналогичная кнопка селектора цвета, только меньше по размерам. Чтобы "отключить" с ее помощью какой-либо селектор цвета, нужно щелкнуть на его обозначении (значке карандаша или ведра с краской), чтобы его выбрать (выбранный селектор цвета отображается в утопленном виде), и нажать эту кнопку. Чтобы "включить" селектор цвета, его нужно снова выбрать и еще раз щелкнуть по этой кнопке.

Кнопка , расположенная там же, задает для цветов линий и заливок значения по умолчанию. Во Flash это соответственно черный и белый цвета. А кнопка позволит нам мгновенно поменять местами цвета линий и заливок.

Создание новых цветов. Смеситель

Конечно, палитра селектора цвета весьма богата, и во многих случаях ее богатства может хватить. Однако для серьезной работы часто нужны цвета, которые в ней не присутствуют. Поэтому Flash предоставляет средства, позволяющие создавать новые цвета, применять их к выбранным на рабочем листе фрагментам изображения и добавлять при необходимости в палитру.

Этот инструмент — особая панель **Color Mixer**, называемая также *смесителем цветов*. Именно так мы и будем называть, для краткости, эту панель.

Чтобы вывести смеситель цветов на экран, нужно включить в меню **Window** пункт-выключатель **Color Mixer** или нажать комбинацию клавиш <Shift>+ +<F9>. Появившийся на экране смеситель цветов показан на рис. 6.4.

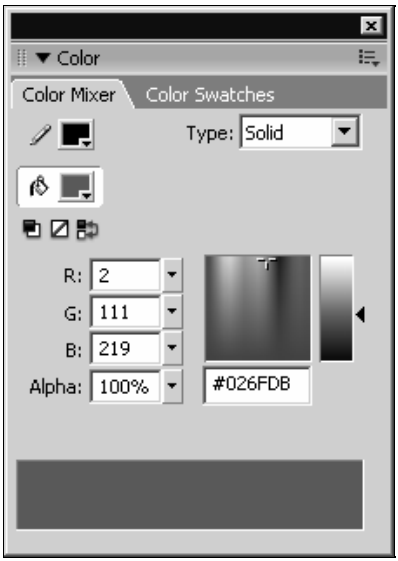


Рис. 6.4. Смеситель цветов (панель **Color Mixer**)

В левом верхнем углу смесителя цветов находятся два уже знакомых нам селектора цвета линии и заливки, которые можно выбирать щелчком мыши и использовать для выбора цветов. Также там присутствуют три дополнительные кнопки, аналогичные рассмотренным чуть выше кнопкам области **Colors** главного инструментария. Они пригодятся нам, если мы убрали с экрана главный инструментарий.

В нижней части смесителя цветов расположены две большие панели задания цвета. Пользоваться ими очень просто. Для этого сначала выбираем нужный селектор цвета — линии или заливки. После этого щелкаем мышью по нужной точке центральной (квадратной) панели, чтобы выбрать сам цвет, затем щелкаем по нужной точке правой (прямоугольной) панели, чтобы задать его интенсивность. Заданный таким образом цвет появится в выбранном селекторе и, одновременно, в расположенной в нижней части панели большой области предварительного просмотра цвета.

Под центральной (квадратной) панелью задания цвета находится уже знакомое нам поле ввода кода цвета. Так что мы можем быстро выбрать нужный цвет, введя туда его код, или, наоборот, узнать код выбранного с помощью других инструментов цвета.

А что же набор из четырех полей ввода, находящийся левее центральной панели задания цвета? Сейчас мы о нем поговорим.

Но сначала нужно выяснить все о *режимах задания цвета*. Как мы уже знаем, каждый цвет в компьютере однозначно описывается определенным набором чисел — кодом цвета. Так вот: режим задания цвета как раз и определяет, что собой представляет этот набор чисел и как он интерпретируется компьютером (точнее, программным обеспечением). Flash поддерживает два режима задания цвета, которые мы сейчас рассмотрим.

- Режим *RGB* (от английского Red, Green, Blue — красный, зеленый, синий). Числовой набор, описывающий цвет, представляет собой три числа, показывающие содержание в нем красной, зеленой и синей составляющей соответственно. Каждое число может принимать значения от 0 (данная составляющая отсутствует) до 255 (максимальное значение составляющей). Каждое такое число занимает один байт; таким образом, цвет кодируется тремя байтами. Этот режим выбран по умолчанию.
- Режим *HSB* (от английского Hue, Saturation, Brightness — цвет, насыщенность, яркость). Числовой набор, описывающий цвет, представляет собой три числа, задающие его оттенок, насыщенность и яркость соответственно. Каждое число также занимает один байт, и цвет, опять же, кодируется тремя байтами.

Задание нужного цветового режима осуществляется путем выбора соответствующего пункта-переключателя дополнительного меню смесителя. Пункт **RGB** этого меню соответствует режиму задания цвета RGB, пункт **HSB** — режиму HSB.

А вот теперь обратимся к набору полей ввода, который уже привлек наше внимание. Собственно задание чисел, описывающих цвет, производится с помощью трех полей ввода этого набора. Для режима RGB это поля ввода **R**, **G** и **B**, где вводятся соответственно доля красной, зеленой и синей составляющей цвета. Для режима HSB это поля ввода **H**, **S** и **B** — оттенок, насыщенность и яркость.

Ниже этих трех полей ввода находится четвертое — **Alpha**, задающее уровень прозрачности цвета. Этот уровень задается в процентах: 100% соответствуют полной непрозрачности цвета (фрагменты изображения, окрашенные этим цветом, полностью непрозрачны), а 0% — его полной прозрачности (окрашенные таким цветом фрагменты полностью прозрачны, т. е. невидимы).

Если внимательно посмотреть на любое из этих полей ввода, то справа от него можно заметить небольшую кнопку со стрелкой, направленной вниз, — *регулятор*. При нажатии на эту кнопку на экране появится шкала регулятора с движком; перемещая мышью этот движок, можно задать нужное значение для поля ввода (рис. 6.5). Чтобы убрать шкалу регулятора, достаточно щелкнуть мышью в любом месте окна или на любом другом элементе управления. Такие *поля ввода с регулятором* часто используются во Flash.

Рис. 6.5. Шкала регулятора с движком

В правом верхнем углу селектора цветов находится небольшой раскрывающийся список. С его помощью задается *вид цвета*. В данное время нас интересуют только два пункта этого списка. Пункт **Solid** задает обычный сплошной цвет, а пункт **None** — отсутствие цвета (т. е., соответственно, линия или заливка рисуемой фигуры будет невидима). Остальные пункты списка мы рассмотрим далее в этой главе.

Если в процессе работы мы создадим какой-либо очень удачный цвет и захотим сохранить его на будущее, мы выберем в дополнительном меню смесителя пункт **Add Swatch**. Созданный нами цвет будет добавлен в нижнюю строку палитры селектора цвета (см. рис. 6.1), где мы сможем в любой момент его выбрать для использования в рисуемом изображении.

Задание цвета линий и заливок

Ну зачем еще раз рассказывать о задании цвета линий и заливок? Что мы еще об этом не знаем? Неужели Flash — настолько мощная программа, что, сколько ни говори об одном и том же, все будет мало?

Увы, мало... Ведь, по сути, мы знакомы только с базовыми средствами задания цвета, позволяющими только задать сплошной цвет или отсутствие всякого цвета. А ведь Flash 8 может сделать цвет линии или заливки градиентным или вообще заполнить ее растровым графическим изображением. Вот как раз об этом мы сейчас и узнаем.

Задание сплошного цвета или отсутствия цвета

Окрасить уже нарисованную линию или заливку в другой цвет проще простого. Выделим нужную линию или заливку и выберем новый цвет в соответствующем селекторе цвета, расположенном в области **Colors** главного инструментария или в панели **Properties** (см. рис. 3.34). Сразу же после этого выделенная линия (заливка) будет закрашена выбранным нами цветом. Кроме того, именно этим цветом будут закрашиваться все линии (заливки), которые мы нарисуем потом.

Описанным выше образом можно выбрать для линии или заливки только сплошной цвет, т. е. равномерное ее заполнение однотонным цветом. Именно таким цветом мы рисовали во Flash наши первые фигуры.

Для задания нового сплошного цвета линии и заливки мы также можем воспользоваться смесителем цветов (см. рис. 6.4). Выделим линию (заливку), проверим, выбран ли в раскрывающемся списке вида цвета пункт **Solid**, и зададим нужный цвет с помощью уже знакомых нам инструментов.

Пункт **None** раскрывающегося списка вида цвета позволит нам задать отсутствие цвета для выделенной линии (заливки). Это значит, что, если мы выделим линию (заливку) и выберем этот пункт, она (выделенная линия или заливка) безвозвратно пропадет. (Хотя, конечно, проще удалить ее нажатием клавиши <Del>.)

Задание сложных цветов[1]

С *простыми* — сплошным и "отсутствующим" — цветами мы разобрались. В самом деле, что там такого сложного — мы уже без малого четыре главы ими пользуемся.

А вот со *сложными* цветами — градиентным и графическим — дело обстоит, извините за каламбур, сложнее. Для их задания нам придется выполнить значительно больше манипуляций с разными элементами управления смесителя цветов. Но и результат будет не в пример эффектнее.

Итак, просьба любить и жаловать — сложные цвета Flash!

---

[1] Автор в настоящей книге использует термины "сложный цвет", "градиентный цвет", "графический цвет" для обозначения способа окрашивания объектов, отличного от закраски однородным (сплошным) цветом. Таким образом, в указанных выше выражениях слово "цвет" имеет другое, отличное от "классического" значение и больше соответствует понятию "закраска". — *Прим. ред.*

Градиентные цвета

В *градиентных цветах* цвет плавно меняется от одного значения к другому, например, от белого к черному. Вы можете увидеть пример градиентного цвета, если посмотрите на любое окно любой программы, работающей в операционной системе Microsoft Windows 98 или более новой ее версии — цвет его заголовка "плывет" от синего к светло-голубому.

Прежде чем мы продолжим разговор о градиентных цветах (иногда их для краткости называют *градиентами*), давайте введем один термин. Назовем цвета, на основании которых создан градиентный цвет, *ключевыми*. Например, для заголовка окна Windows 98 ключевыми цветами будут синий и светло-голубой.

> ***НА ЗАМЕТКУ***
>
> Предыдущие версии Flash позволяли окрашивать в градиентные цвета только заливки, но не линии. Поэтому тогда шла речь о градиентных заливках, а не о цветах.

Flash позволяет создавать два вида градиентных цветов: линейный и радиальный. В *линейном градиентном цвете* цвет меняется линейно при движении от одного конца фигуры к другому (рис. 6.6). В *радиальном градиентном цвете* цвет меняется при движении от центра воображаемой окружности, вписанной в фигуру, к ее периферии (рис. 6.7). Можно сказать, что названия этих цветов говорят сами за себя.

Чтобы окрасить выделенную линию или заливку линейным или радиальным градиентным цветом, придется воспользоваться смесителем. Выберем в раскрывающемся списке видов цвета соответственно пункт **Linear** (линейный градиентный цвет) или **Radial** (радиальный градиентный цвет) — и линейный, и радиальный градиентные цвета создаются одинаковым образом. После этого смеситель изменит свой вид (рис. 6.8).

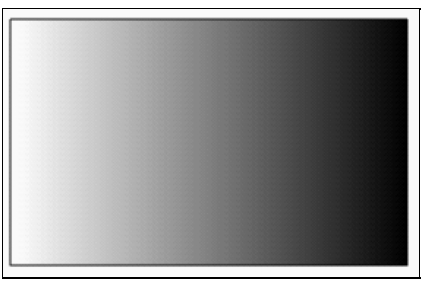


Рис. 6.6. Заливка, окрашенная линейным градиентным цветом

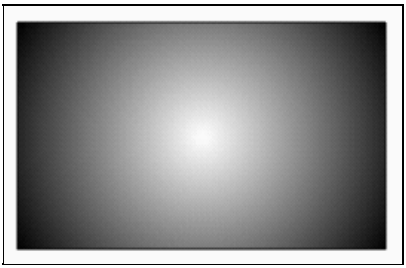


Рис. 6.7. Заливка, окрашенная радиальным градиентным цветом

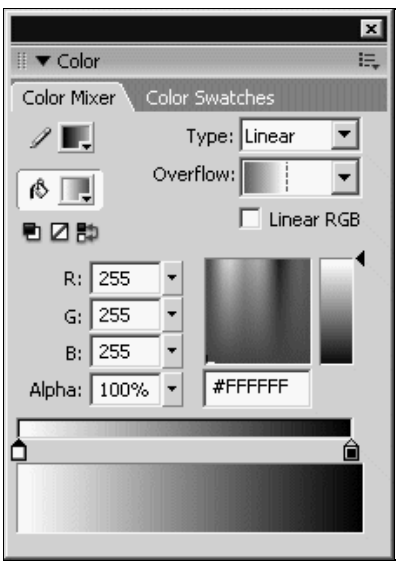


Рис. 6.8. Вид смесителя цветов при выборе градиентного цвета

Прежде всего, в нижней его части появится *линейка* для задания ключевых цветов заливки, имеющая вид шкалы, окрашенной в плавно перетекающие друг в друга цвета. Эта линейка схематично представляет наш будущий градиентный цвет и показывает на нем точки, окрашенные ключевыми цветами. Сами же ключевые цвета задают *указатели цвета*, имеющие вид небольших серых движков. Изначально этих движков всего два — это значит, что наш будущий градиентный цвет содержит только два ключевых цвета, — но может быть и больше.

Но как задать ключевой цвет, связанный с тем или иным указателем цвета? Очень просто. Щелкнем мышью по первому указателю цвета, после чего он станет черным — выбранным. Воспользовавшись селектором цвета, полями

ввода кода цвета или панелями задания цвета, мы задаем первый ключевой цвет. После этого щелчком мыши выбираем второй указатель цвета и задаем ключевой цвет для него. Получившийся градиентный цвет будет показан в расположенной в нижней части смесителя цветов области предварительного просмотра.

Еще мы можем перемещать указатели цветов мышью по линейке, чтобы подобрать нужное расстояние между ключевыми цветами в результирующем градиентном цвете. Так можно создать интересные цветовые эффекты.

Вообще, с градиентными цветами, как и со многими другими возможностями Flash, лучше всего поэкспериментировать. Это позволит вам, что называется, набить руку.

Только что мы создали наш первый градиентный цвет. Он содержит всего два ключевых цвета; такие градиентные цвета используются очень часто. Но наиболее эффектно, конечно, выглядят градиентные цвета, состоящие из нескольких ключевых цветов (*многоцветные*). Пример заливки, закрашенной трехцветным градиентом, показан на рис. 6.9 — белый цвет плавно перетекает в черный, а потом — снова в белый. Сейчас мы выясним, как создаются такие цвета.

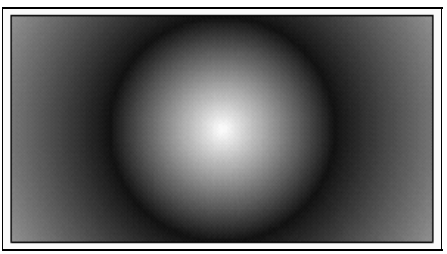


Рис. 6.9. Заливка, окрашенная многоцветным градиентным цветом

Прежде всего, нам будет нужно добавить на линейку новые указатели цвета. Для этого надо выбрать нужное место на линейке и щелкнуть чуть ниже ее, на уровне остальных указателей. После этого мы сможем задать для вновь созданного указателя цвета ключевой цвет и, если нужно, переместить его.

Для заливки, показанной на рис. 6.9, вид линейки задания цвета будет такой, как показано на рис. 6.10.

Внимание!

Flash поддерживает градиентные цвета, имеющие до восьми ключевых цветов включительно.

Рис. 6.10. Вид линейки задания цветов для заливки, показанной на рис. 6.9

Чтобы удалить ненужный указатель цвета, достаточно "захватить" его мышью и "сдернуть" вниз.

> ***ВНИМАНИЕ!***
>
> Удалить два первоначальных указателя цветов, созданных самим Flash, невозможно.

Раскрывающийся список **Overflow** позволяет задать, как ключевые цвета располагаются внутри градиента. В качестве пунктов этого списка используются "говорящие" рисунки, так что проблем с ним быть не должно.

Созданный градиентный цвет мы можем сохранить в палитре селекторов цвета Flash. Для этого нужно будет выбрать в дополнительном меню смесителя пункт **Add Swatch**. Впрочем, мы уже говорили об этом.

Графические цвета

Графический цвет используют для заполнения линии или заливки растрового графического изображения (так называемое *изображение-заливка*). Часто такие заливки выглядят очень эффектно, но не следует забывать, что растровое изображение может значительно увеличить размер результирующего файла Shockwave/Flash.

Чтобы создать графический цвет, сначала нужно, опять же, вызвать на экран смеситель цветов. Далее выбираем в раскрывающемся списке видов цвета пункт **Bitmap**. Если мы еще ни разу не создавали графический цвет, Flash предупредительно выведет на экран стандартное диалоговое окно открытия файла Windows. В этом окне мы должны будем выбрать файл растрового изображения, которое будет использовано для создания графического цвета.

После того как мы выберем файл нужного нам изображения и нажмем кнопку открытия, смеситель цветов примет вид, показанный на рис. 6.11.

В нижней части смесителя цветов в этом случае появится список всех выбранных нами растровых изображений (на рис. 6.11 там присутствует только одно). Чтобы применить какое-либо из них в качестве графического цвета, достаточно щелкнуть по нему мышью. Пример заливки, закрашенной графическим цветом, показан на рис. 6.12.

Чтобы добавить в список новое растровое изображение, нужно щелкнуть по кнопке **Import**. Flash выведет на экран стандартное диалоговое окно откры-

тия файла Windows, в котором мы сможем выбрать нужный файл, после чего будем должны нажать кнопку открытия.

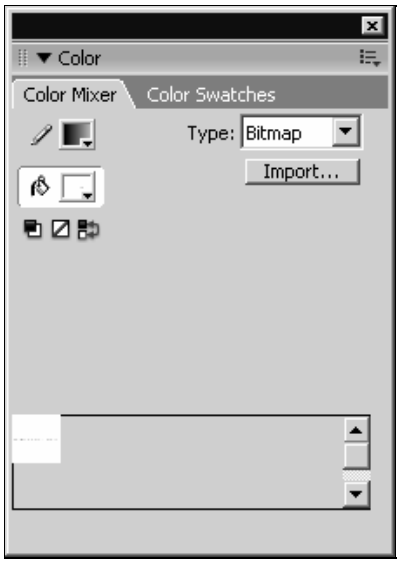


Рис. 6.11. Вид смесителя цветов при выборе графического цвета

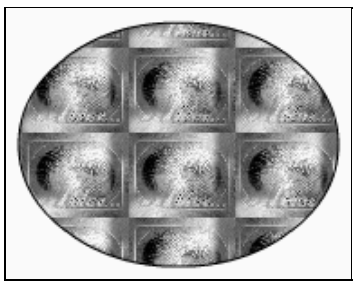


Рис. 6.12. Графическая заливка

Работа со сложными цветами

Так, создавать сложные цвета мы научились. Теперь настала пора познакомиться со средствами для работы с ними. Мы научимся применять к цветам трансформации, аналогичные трансформациям фрагментов изображения, рассмотренным в *главе 4*, и создавать фиксированные цвета.

Трансформация сложных цветов

Если при создании градиентной или графической заливки мы случайно щелкнули мышью не в той точке внутри замкнутого контура, и заливка получилась не такой, какая нам нужна, мы всегда можем исправить свою ошибку. В частности, мы можем изменить направление линейного градиента и радиус воображаемой окружности, на основе которой строится радиальный градиент. Подобные операции называются *трансформациями цвета.*

Для трансформаций цвета служит особый инструмент, называемый **Gradient Transform** ("Трансформатор градиента"). Хоть он и называется трансформатором именно градиента, но с его помощью можно манипулировать и графическим цветом. В этом разделе, кстати, работу с этим инструментом мы начнем рассматривать на примере именно графического цвета — так будет нагляднее.

Чтобы выбрать инструмент "трансформатор градиента", достаточно щелкнуть по кнопке , находящейся в главном инструментарии, или нажать клавишу <F> клавиатуры. Теперь, чтобы настроить цвет какой-либо фигуры, щелкнем ее мышью. Эта фигура будет выделена особым образом (рис. 6.13). Как мы видим, она помещается в уже знакомый нам прямоугольник выделения; на разных его сторонах и в центре находятся маркеры, с помощью которых и выполняется трансформация цвета.

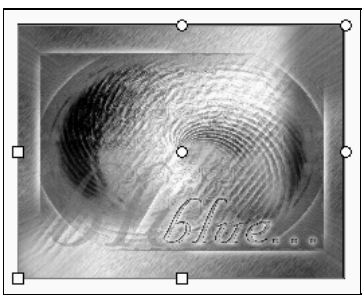


Рис. 6.13. Заливка, окрашенная графическим цветом и выделенная с помощью инструмента "трансформатор градиента"

Внимание!

Выделить фигуру, окрашенную сплошным цветом, при выбранном инструменте "трансформатор градиента" невозможно. Трансформации цветов доступны только для сложных цветов.

Что же можно делать с выделенной таким образом фигурой?

Во-первых, мы можем *смещать* ее цвет. Для этого нужно перетащить мышью круглый белый маркер, находящийся в центре прямоугольника выделения, — *маркер смещения*. Результат такой операции показан на рис. 6.14 — графический цвет был смещен вправо и вверх.

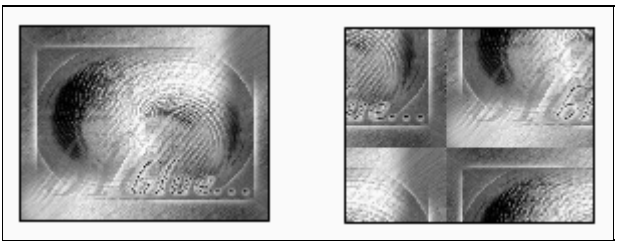


Рис. 6.14. Смещение графического цвета вправо и вверх (слева — исходная заливка, справа — результат смещения)

Во-вторых, можно изменять геометрические размеры образца цвета. Чтобы добиться этого, нужно перетащить мышью один из трех больших квадратных маркеров изменения размера. Маркер на левой стороне прямоугольника выделения меняет ширину образца цвета, а маркер на нижней стороне — его высоту. Если же нам нужно менять одновременно и ширину, и высоту прямоугольника с сохранением его пропорций, воспользуемся маркером, расположенным в его левом нижнем углу. Пример изменения размеров образца графического цвета показан на рис. 6.15 — он был сжат по горизонтали.

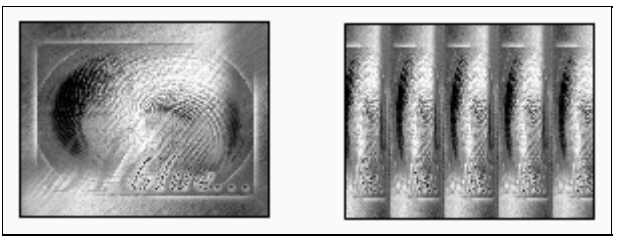


Рис. 6.15. Сжатие образца графического цвета по горизонтали (слева — исходная заливка, справа — результат сжатия)

На рис. 6.15 хорошо видно, что, если размеры образца графического цвета оказываются меньше размеров заполненной им фигуры, для заполнения ее будет использовано несколько изображений-заливок, составляющих графи-

ческий цвет. Опытные художники говорят в этом случае, что программа выполняет закраску *мозаикой* (по-английски: *tile*).

В-третьих, можно вращать образец графического цвета вокруг центра, обозначенного маркером смещения. Для этого нужно перетаскивать мышью круглый маркер, расположенный в правом верхнем углу прямоугольника выделения, — маркер вращения. Результат, достигаемый при этом, показан на рис. 6.16. В этом случае также может быть использована мозаика, так как по высоте изображение-заливка оказалось у́же, чем нужно для того, чтобы заполнить фигуру целиком.

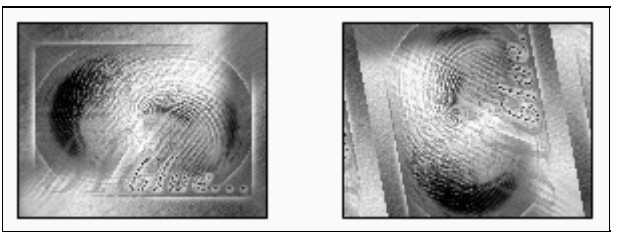


Рис. 6.16. Вращение образца графического цвета (слева — исходная заливка, справа — результат вращения)

В-четвертых, образец графического цвета можно сдвигать относительно заполняемой им фигуры. Для этого достаточно перетащить мышью один из двух круглых маркеров сдвига, расположенных на правой и верхней стороне прямоугольника выделения. Верхний маркер "отвечает" за сдвиг по горизонтали, правый — соответственно за сдвиг по вертикали. Результат сдвига образца графического цвета показан на рис. 6.17.

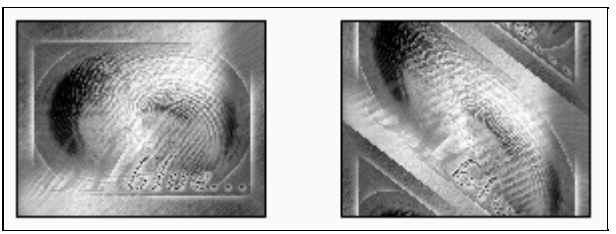


Рис. 6.17. Сдвиг образца графического цвета по вертикали (слева — исходная заливка, справа — результат сдвига)

Многие из этих действий мы можем выполнять и над линейными градиентными цветами. А именно — смещение, изменение одного из размеров и вращение. На рис. 6.18 показаны все доступные маркеры.

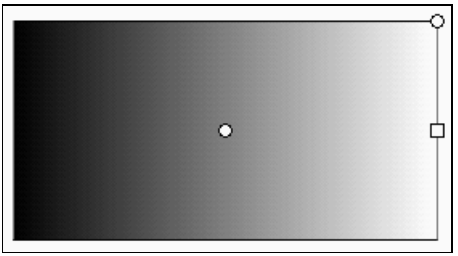


Рис. 6.18. Заливка, окрашенная линейным градиентным цветом и выделенная с помощью инструмента "трансформатор градиента"

В случае же радиального градиентного цвета все немного по-другому. Посмотрим на рис. 6.19 — и сразу станет понятно, почему. Прежде всего, здесь нужно говорить не о прямоугольнике, а об *окружности выделения*. Далее, сразу не понятно, какой маркер за что "отвечает". Хорошо заметны только маркер смещения (он всегда находится в центре) и маркер изменения размера, а именно, ширины (он находится справа и имеет вид квадрата — это единственный квадратный маркер из всех изображенных на рисунке). Остальные же маркеры мы сейчас рассмотрим.

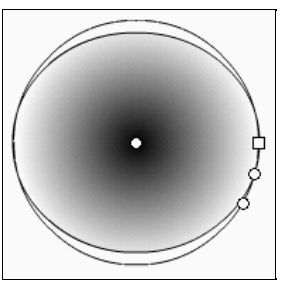


Рис. 6.19. Заливка, окрашенная радиальным градиентным цветом и выделенная с помощью инструмента "трансформатор градиента"

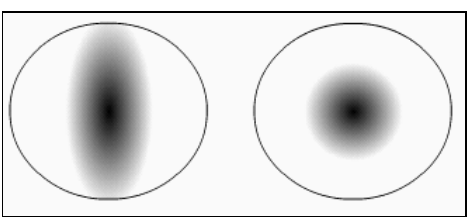


Рис. 6.20. Сжатие по ширине (слева) и уменьшение радиуса окружности (справа) образца радиального градиентного цвета

Если, начиная от маркера изменения ширины образца, следовать вдоль окружности выделения по часовой стрелке, первым встретившимся нам маркером будет *маркер изменения радиуса* окружности. (Не путайте его с маркером изменения ширины!) Попробуем переместить оба этих маркера и посмотрим, что получится (рис. 6.20).

Если продолжать наше воображаемое путешествие по окружности, то следующим маркером будет маркер вращения. Как им пользоваться, мы уже знаем.

Фиксированные цвета

Обычно, если мы окрасим заливки в градиентный или графический цвет, Flash делает эти цвета *независимыми* друг от друга. При этом считается, что каждая заливка на рабочем листе существует сама по себе, вне зависимости от того, какие заливки ее окружают. Это хорошо заметно на рис. 6.21.

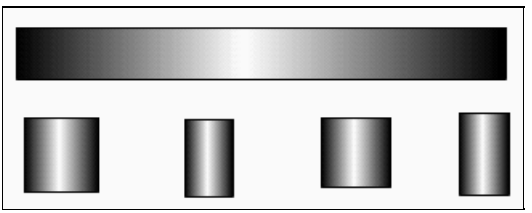


Рис. 6.21. Набор заливок, окрашенных в независимые цвета

Однако иногда бывает нужно создать набор заливок, окрашенных в *зависимые* или *фиксированные* цвета. Например, такой, какой показан на рис. 6.22 — там хорошо видно, что один цвет как бы "растягивается" на несколько заливок, в то время как его начало зафиксировано в одной из них.

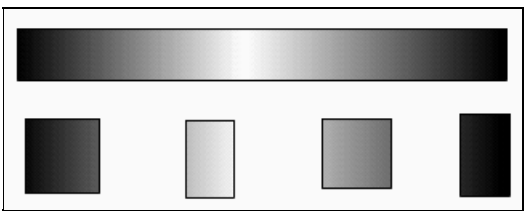


Рис. 6.22. Набор заливок, окрашенных в фиксированные цвета

Чтобы создать фиксированный цвет, следует воспользоваться модификатором **Lock Fill** (Фиксация заливки), доступным при выборе инструментов "кисть" или "ведро с краской". Он включается кнопкой-выключателем ; кнопка эта, как и все другие кнопки модификаторов, находится в области **Options** главного инструментария.

Процесс создания заливки, окрашенной в фиксированный цвет, состоит из нескольких этапов. Давайте их рассмотрим.

1. Выбираем инструмент "кисть" или "ведро с краской".
2. Включаем или отключаем, при желании, модификаторы этого инструмента, кроме модификатора "фиксация заливки".
3. Щелкаем мышью внутри того контура, в котором хотим создать начало заливки, окрашенной фиксированным цветом.
4. Включаем модификатор "фиксация заливки".
5. Продолжаем щелкать внутри других контуров, на которые мы хотим распространить созданную на предыдущем шаге заливку.
6. Закончив, обязательно отключаем модификатор "фиксация заливки", если не хотим, чтобы созданная нами заливка распространялась и на другие контуры нашего изображения.

Как видим, создать заливку, окрашенную фиксированным цветом, очень просто. А чтобы добавить ей эффектности, мы можем воспользоваться модификатором "трансформатор заливки" и изменить ее.

Работа с линиями

Ну все, с цветами и их заданием мы разобрались. И даже успели немного поэкспериментировать с ними.

Теперь самое время рассмотреть средства Flash для работы с линиями контуров. Это, прежде всего, средства для задания *стиля линии* — это в терминологии Flash ее начертание: сплошная, точечная, штриховая, штрихпунктирная и т. п. Плюс еще один вспомогательный инструмент, который может нам потом пригодиться.

Задание стиля линий

Для задания стиля линий используется панель **Properties**. Она предоставит соответствующие элементы управления, если мы выделим на рабочем листе какую-либо линию или контур. Эти элементы управления показаны на рис. 6.23.

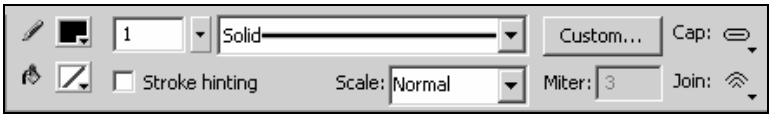


Рис. 6.23. Панель **Properties** при выделенной линии

С помощью большого раскрывающегося списка мы можем выбрать любой из набора предопределенных стилей линии: тонкая, толстая, штриховая, неровная, точечная и др. Левее раскрывающегося списка расположено уже знакомое нам поле ввода с регулятором, задающее толщину линии в типографских пунктах (в этих единицах измеряются размеры шрифтов) от 0 до 200. Ну, и с помощью верхнего селектора цвета мы можем задать цвет линии.

Если нам не хватит предопределенных стилей линии, щелкнем по кнопке **Custom**, расположенной правее раскрывающегося списка стилей линии. На экране появится диалоговое окно **Stroke Style**, показанное на рис. 6.24.

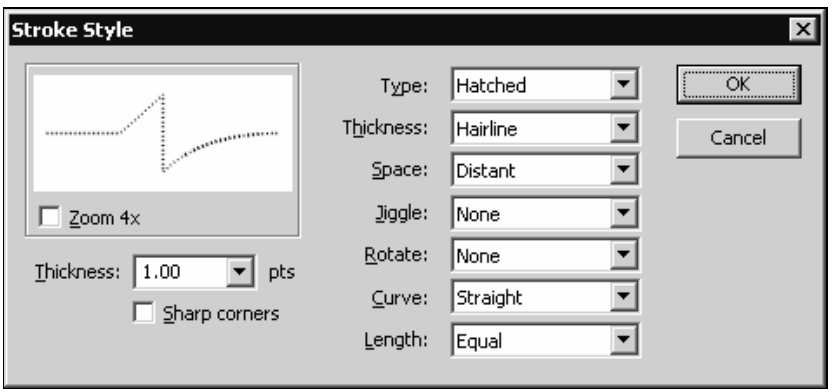


Рис. 6.24. Диалоговое окно **Stroke Style**

В верхнем левом углу этого диалогового окна находится область предварительного просмотра, в которой показывается результат наших экспериментов над линией. Флажок **Zoom 4x** позволяет включить или отключить четырехкратное увеличение содержимого области просмотра, если мы пожелаем посмотреть, из чего состоит наша линия, в деталях.

В поле ввода со списком **Thickness** задается толщина линии в типографских пунктах.

Флажок **Sharp corners** включает или отключает "заострение" концов линий.

В раскрывающемся списке **Type** выбирается тип линии:

- **Solid** — сплошная линия;
- **Dashed** — пунктирная линия;
- **Dotted** — точечная линия;
- **Ragged** — имитация линии, проведенной вручную;
- **Stipple** — линия, состоящая из чернильных пятен;
- **Hatched** — линия, состоящая из поперечных черточек.

Если выбран пункт **Dashed**, то ниже раскрывающего списка **Type** появятся два дополнительных поля ввода **Dash**. В левом вводится длина штрихов, а в правом — расстояние между штрихами. Обе эти величины задаются в типографских пунктах.

Если выбран пункт **Dotted**, в диалоговом окне появится одно дополнительное поле ввода **Dot spacing**. В нем вводится расстояние между отдельными точками, измеряемое также в пунктах.

Если выбран пункт **Ragged**, в диалоговом окне появятся три дополнительных раскрывающихся списка: **Pattern**, **Wave height** и **Wave length**.

В раскрывающемся списке **Pattern** задается дополнительный тип линии:

- **Solid** — непрерывная линия;
- **Simple** — линия, состоящая из отрезков одинаковой длины;
- **Random** — линия, состоящая из отрезков разной длины;
- **Dotted** — линия, состоящая из отрезков одинаковой длины, разделенных точками;
- **Random Dotted** — линия, состоящая из отрезков разной длины, разделенных точками;
- **Triple Dotted** — линия, состоящая из отрезков одинаковой длины, разделенных двойными точками;
- **Random Triple Dotted** — линия, состоящая из отрезков разной длины, разделенных двойными точками.

В раскрывающемся списке **Wave height** задается "волнистость" линии. Он содержит четыре пункта: **Flat**, **Wavy**, **Very Wavy** и **Wild**, задающих соответственно полное отсутствие "волнистости", малую, среднюю и сильную "волнистость".

В раскрывающемся списке **Wave length** задается длина получившихся "волн"; пункты **Very Short** (очень короткие), **Short** (короткие), **Medium** (средней длины) и **Long** (длинные) меняют длину "волн" от минимальной до максимальной.

Если в раскрывающемся списке **Type** выбран пункт **Stipple**, то в диалоговом окне появятся три дополнительных раскрывающихся списка: **Dot size**, **Dot variation** и **Density**.

В раскрывающемся списке **Dot size** задается размер пятен — от минимального до максимального — с помощью пунктов **Tiny** (крошечный), **Small** (малый), **Medium** (средний) и **Large** (большой).

С помощью раскрывающегося списка **Dot variation** задается разница в размерах точек, составляющих линию. Этот список имеет четыре пункта:

- **One Size** — пятна имеют одинаковый размер;
- **Small Variation** — размер пятен немного меняется;
- **Varied Sizes** — размер пятен меняется сильно;
- **Random Sizes** — размер каждого пятна выбирается случайным образом. Практически выбор этого пункта дает тот же эффект, что и выбор пункта **Varied Sizes**.

В раскрывающемся списке **Density** устанавливается, насколько близко друг от друга находятся отдельные пятна. Здесь доступны четыре пункта:

- **Very Dense** — пятна расположены настолько близко друг к другу, что сливаются в одну линию;
- **Dense** — пятна расположены близко друг к другу, сливаясь во множество непрерывных отрезков;
- **Sparse** — пятна расположены далеко друг от друга и не сливаются;
- **Very Sparse** — пятна расположены очень далеко друг от друга.

В случае же выбора в раскрывающемся списке **Type** пункта **Hatched** в диалоговом окне **Line Style** появятся целых шесть дополнительных раскрывающихся списков. Рассмотрим их.

Раскрывающийся список **Thickness** задает толщину штрихов, составляющих линию. Он имеет четыре пункта: **Hairline** (очень тонкие, "волосяные" штрихи), **Thin** (тонкие), **Medium** (средней толщины) и **Thick** (толстые).

Раскрывающийся список **Space** позволяет задать расстояние между отдельными штрихами. Он также имеет четыре пункта: **Very Close** (очень близко друг к другу), **Close** (близко), **Distant** (далеко) и **Very Distant** (очень далеко).

Раскрывающийся список **Jiggle** задает величину сдвига отдельных штрихов друг относительно друга, иначе говоря, их "разболтанность". Доступны четыре пункта: **None** (без сдвига), **Bounce** (небольшой сдвиг), **Loose** (средний сдвиг) и **Wild** (полнейшая "разболтанность").

Раскрывающийся список **Rotate** задает величину поворота отдельных штрихов друг относительно друга. Доступны четыре пункта: **None** (без поворота),

Slight (небольшие углы поворота), **Medium** (средний поворот) и **Free** (угол поворота может достигать 90° — настоящая "свистопляска" штрихов).

Раскрывающийся список **Curve** позволяет задать степень искривленности штрихов. Он имеет четыре пункта: **Straight** (никакой искривленности, все штрихи прямые), **Slight Curve** (небольшая искривленность), **Medium Curve** (средняя искривленность) и **Very Curved** (большая искривленность).

Последний раскрывающийся список — **Length** — позволяет задать, насколько меняется размер соседних штрихов. Здесь имеются четыре пункта: **Equal** (все штрихи одинаковы), **Slight Variation** (размеры изменяются в небольших пределах), **Medium Variation** (размеры меняются в средних пределах) и **Random** (размер каждого штриха выбирается случайным образом).

Задав нужный стиль линии, нужно нажать кнопку **OK** для его применения или **Cancel** для его отмены. Заданный нами стиль будет тотчас применен к выделенной линии. Более того: все линии, которые мы потом нарисуем, будут иметь этот стиль.

Дополнительные параметры стиля линий

Теперь настало время обратиться к инструментам, появившимся во Flash 8. Эти инструменты также доступны из панели **Properties** и позволяют задать дополнительные параметры стиля линий.

Начнем мы с флажка **Stroke hinting**. Если его включить, то все выделенные и вновь рисуемые линии будут выровнены по пикселам рабочего листа. В результате Flash для вывода этих линий не будет использовать сглаживание, что может положительно сказаться на скорости их вывода и качестве изображения. Но если требуется нарисовать что-то очень точно, флажок **Stroke hinting** лучше отключить.

Кнопка с меню **Cap** (рис. 6.25) позволяет задать вид концов линий. Меню этой кнопки содержит три пункта: **None** (нет особого оформления концов линий), **Round** (скругленные) и **Square** (Квадратные).

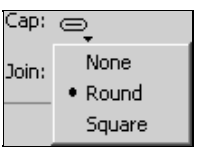


Рис. 6.25. Кнопка с меню **Cap** панели **Properties**

Другая кнопка с меню — **Join** (рис. 6.26) — позволяет задать вид углов, образуемых смыкающимися линиями. Ее меню также содержит три пунк-

та: **Miter** (Заостренные углы), **Round** (Скругленные углы) и **Bevel** (Затупленные углы).

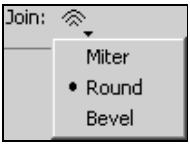


Рис. 6.26. Кнопка с меню **Join** панели **Properties**

Если в меню кнопки **Join** выбран пункт **Miter**, становится доступно поле ввода **Miter**. В этом поле вводится величина заостренности углов в типографских пунктах. Если заостренный угол, образуемый смыкающимися линиями, выступает более чем на заданную величину, Flash его обрезает.

Инструмент *Ink Bottle* (*"Чернильница"*)

Воспользовавшись инструментом "чернильница", мы можем изменить цвет и стиль контура любой фигуры, щелкнув в любом ее месте, даже в области заливки. Чтобы выбрать этот инструмент, нужно щелкнуть по кнопке , находящейся в главном инструментарии, или нажать клавишу <S> клавиатуры. Курсор мыши примет вид небольшой опрокинутой чернильницы.

После этого задаем нужные цвет и стиль линии с помощью панели **Properties** и щелкаем в любом месте фигуры, контур которой мы хотим изменить. Проще всего, разумеется, щелкнуть на ее заливке — она все-таки объемнее, чем линии контура, и промахнуться мимо нее невозможно.

Инструмент *Eyedropper* (*"Пипетка"*)

Иногда бывает необходимо скопировать параметры (цвет и стиль) какой-либо линии или заливки и применить их к другой линии или заливке. Конечно, можно просто выставить их параметры вручную, но, согласитесь, это долго и неудобно. Проще воспользоваться инструментом "пипетка", предусмотренным как раз для такого случая.

Чтобы выбрать инструмент "пипетка", щелкнем по кнопке , находящейся в главном инструментарии, или нажмем клавишу <I> клавиатуры. Курсор мыши примет вид пипетки.

Выбрав этот инструмент, щелкнем на линии или заливке, параметры которых мы хотим скопировать. Если мы щелкнем на линии, Flash автоматически выберет инструмент "чернильница", если же на заливке — "ведро с краской". В последнем случае также автоматически включится модификатор "фиксация заливки"; если он не нужен, мы можем его отключить.

Теперь, чтобы применить выбранные параметры к другим линиям или заливкам, нам остается только щелкнуть на них. Просто, не правда ли?

Работа с палитрами

Когда мы знакомились с селектором цветов, то назвали весь доступный в его окне набор цветов палитрой. Также мы узнали, что в эту палитру можно добавлять новые цвета, созданные с помощью смесителя цветов. После этого мы в любой момент сможем выбрать и применить к выделенному фрагменту изображения созданные нами цвета, воспользовавшись селектором цвета.

Так вот, палитра селектора цвета содержит набор всех цветов, что мы использовали в данном изображении Flash. Каждый документ Flash содержит свою собственную палитру, которая сохраняется в файле документа. Когда мы экспортируем готовое изображение, палитра перекочевывает в результирующий файл Shockwave/Flash. Отсюда следует, что если палитра велика, то результирующий файл, соответственно, увеличивается в размерах. Умело подобрав палитру и, возможно, ограничив количество используемых в изображении цветов, мы сможем уменьшить размер этого файла.

Давайте выясним, каким образом можно управлять палитрой документа Flash.

Вся работа над содержимым палитры осуществляется с помощью панели **Color Swatches** (рис. 6.27). Чтобы вызвать ее на экран, нужно включить пункт-выключатель **Color Swatches** меню **Window** или нажать комбинацию клавиш <Ctrl>+<F9>.

Эта панель состоит из двух частей. В верхней ее части находится уже знакомая нам палитра сплошных цветов, в нижней — образцы различных видов градиентных и графических цветов, созданных как самим Flash при создании данного документа, так и нами в процессе работы. Мы можем перемещать мышью серую полосу, разделяющую эти части, меняя их размеры.

Выбрать любой цвет в палитре можно, как обычно, щелчком мышью. Выбранный цвет выделяется белым квадратом. Если на рабочем листе у нас была выделена какая-либо линия или заливка, то выбранный в панели **Color Swatches** цвет будет тотчас к ней применен.

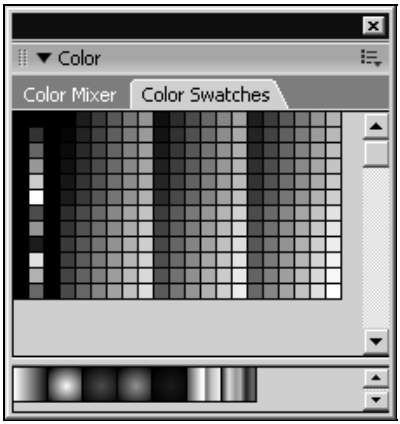


Рис. 6.27. Панель **Color Swatches**

Мы уже знаем, как добавлять в палитру новые цвета. Давайте выясним, что еще можно с ними делать.

Если Flash позволяет нам добавлять в палитру новые цвета, то уж конечно он должен позволить нам и удалять их. Для этого нужно всего лишь выделить ненужный цвет и выбрать пункт **Delete Swatch** дополнительного меню этой панели.

Кроме того, есть возможность создать копию (продублировать) выбранного цвета. Это пригодится, если мы собираемся создать новый цвет, слегка отличающийся от уже существующего. Выделяем цвет, который мы хотим продублировать, и выбираем пункт **Duplicate Swatch** дополнительного меню.

Если мы создали, с нашей точки зрения, исключительно удачную палитру и хотим приберечь ее для дальнейшего использования, мы можем сохранить ее в виде файла на диске. Файл этот может иметь расширение clr (так называемые файлы *цветовых наборов* Flash) или act (*цветовые таблицы*). Цветовые наборы Flash могут быть использованы только в самом Flash, а цветовые таблицы поддерживаются также другими графическими программами, в частности, Macromedia Fireworks и Adobe Photoshop.

Чтобы сохранить цветовую палитру в файле, мы выберем пункт **Save Colors** дополнительного меню. На экране появится стандартное диалоговое окно сохранения файла Windows. Задаем имя файла, выбираем его формат и нажимаем кнопку сохранения.

Также мы можем сохранить созданную нами палитру в качестве *палитры по умолчанию*, которая будет использоваться во всех вновь создаваемых документах Flash. Для этого выберем пункт **Save as Default** дополнительного меню.

Чтобы загрузить сохраненную ранее палитру из файла, затерев существующую, мы выберем в дополнительном меню пункт **Replace Colors**. Если же мы хотим добавить в существующую палитру цвета из сохраненной, выберем пункт **Add Colors**. В обоих этих случаях на экране появится стандартное диалоговое окно открытия файла Windows. Выбираем формат файла, сам файл и нажимаем кнопку открытия.

Мы также можем загружать цвета из графических изображений формата GIF. При этом также возможно как добавить цвета к существующей палитре, так и полностью заменить ее.

Чтобы загрузить палитру по умолчанию, заменив ею существующую, выберем в дополнительном меню пункт **Load Default Colors**. А чтобы очистить палитру, удалив из нее все цвета, оставив только черный, белый и простейший черно-белый линейный градиент, воспользуемся пунктом **Clear Colors**.

Пункт **Web 216** дополнительного меню позволит нам загрузить так называемую *безопасную палитру* WWW. Эта палитра включает в себя 216 цветов, которые гарантированно будут отображаться на любом компьютере. Если мы создаем изображения для публикации в Интернете, то будет лучше всего использовать именно эту палитру.

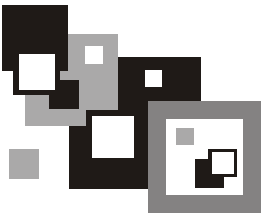
Последний пункт дополнительного меню **Sort by Color** позволит нам отсортировать цвета в палитре по оттенку. К сожалению, вернуть обычный порядок сортировки цветов после этого невозможно.

Что дальше?

Вот мы и научились работать с цветами. Разговор об этом вышел долгим и обстоятельным — сами посудите, ведь любой художник, "бумажный" или компьютерный, фактически всегда работает с цветом. Цвет — основа живописи, из мазков различных цветов состоят бессмертные шедевры...

Ну да ладно, бог с ними, с шедеврами! Нас-то что дальше ждет?

А ждет нас работа с текстом. Мы научимся создавать текстовые блоки и управлять ими. А также выясним все о том, как Flash поддерживает различные шрифты.



Глава 7

Работа с текстом

Текст? Но зачем графическому пакету поддерживать работу с текстом?

Во-первых, часто бывает так, что лучше сказать пару слов, чем рисовать мегабайты хитроумной графики. Во-вторых, текст зачастую является составной частью изображения, как например заголовок сайта или название художественного фильма. В-третьих, без работы с текстом не обойтись при создании Web-сайтов, полностью реализованных на Flash, ведь главное на сайте — это текст, не так ли?

Вот поэтому Flash 8 и поддерживает работу с текстом. Причем текст этот может быть набран разными шрифтами, с различным типом выравнивания; можно также задать абзацные отступы и красную строку. Более того, Flash позволяет применить к тексту различные трансформации, изменить его цвет и даже анимировать. (Тут мы немного забежали вперед — анимация будет описана в *части III* этой книги.) Вообще, во Flash с текстом (как и с графикой) можно делать все, что угодно, и даже чуть больше.

В этой главе мы поговорим о возможностях, предоставляемых Flash для работы с текстом.

Текстовые блоки и работа с ними

Любой текст, который мы поместим на рабочий лист, будет заключен в так называемый *текстовый блок*. Это особый примитив, мало чем отличающийся от уже знакомых нам "графических" примитивов — линий и заливок. О работе с текстовыми блоками мы сейчас и поговорим.

Работа с текстовыми блоками

Создание текстовых блоков выполняется с помощью особого инструмента **Text** (Текст). Чтобы выбрать его, нужно щелкнуть по кнопке A главного инструментария или нажать на клавиатуре клавишу <T>. После этого курсор мыши примет вид небольшого крестика с буквой A.

Как только мы выберем инструмент "текст", мы можем приступить к созданию текстовых блоков. Сделать это можно двумя способами; при этом мы получим текстовые блоки двух видов с весьма сильно различающимися свойствами.

Первый способ создания текстового блока — самый простой. Мы просто щелкаем в той точке рабочего листа, где должен находиться левый верхний угол создаваемого текстового блока. После этого там появится небольшой синий прямоугольник с мигающим текстовым курсором. Этот прямоугольник — и есть только что созданный нами текстовый блок, пока еще не содержащий никакого текста.

Текст в текстовый блок вводится так же, как в обычном текстовом редакторе. Для переноса текста на новую строку при наборе текста используется клавиша <Enter>. Также мы можем редактировать текст, пользуясь клавишами-стрелками, <Backspace>, <Del> и др.

Пользуясь мышью или стандартными клавиатурными комбинациями, мы можем выделять фрагменты текста и работать с буфером обмена Windows. В частности, мы можем выделять слова двойным щелчком мыши и выделять все содержимое текстового блока нажатием клавиатурной комбинации <Ctrl>+<A>, перемещать и копировать его в буфер обмена Windows.

Также мы можем перенести текст из другого Windows-приложения, например, Microsoft Word. В случае достаточно большого текста это может оказаться более удобным, чем набор его прямо во Flash, так как Word содержит, кроме всего прочего, систему проверки орфографии, которая, несмотря на все ее недостатки, часто бывает полезна.

Введем текст в текстовый блок и посмотрим на него. То, что мы увидим, показано на рис. 7.1.

Рис. 7.1. Открытый текстовый блок с фиксированной высотой

Прежде всего, мы увидим сам введенный нами текст, ставший содержимым этого текстового блока. Также хорошо заметен черный прямоугольник выделения с маленькими черными маркерами изменения размера. И — еще один, незнакомый нам маркер, имеющий вид окружности и находящийся в правом верхнем углу прямоугольника выделения. Об этом маркере мы поговорим потом.

Еще мы можем заметить текстовый курсор — это значит, что прямо сейчас можно добавить в текстовый блок новый текст или исправить уже введенный. Говорят, что текстовый блок в этом случае *открыт*.

Давайте все-таки попробуем добавить в текстовый блок еще несколько символов. При этом можно заметить, что текстовый блок растягивается в ширину, чтобы вместить все вводимые символы. Высота же текстового блока остается постоянной, а именно, такой, чтобы вместить ровно одну строку текста. Такой текстовый блок называется *блоком с фиксированной высотой* и идеально подходит для создания коротких надписей.

Теперь давайте либо выберем в главном инструментарии другой инструмент (например, "стрелку"), либо нажмем клавишу <Esc>. Текстовый блок после этого *закроется*, и мы уже не сможем работать с помещенным в него текстом. Но при этом он останется выделенным, и мы можем, например, переместить его на другое место мышью.

Закрытый текстовый блок, выделенный с помощью "стрелки", отображается на рабочем листе так, как показано на рис. 7.2. В этом случае он будет окружен синим прямоугольником выделения с маленькими синими маркерами изменения размеров. Невыделенный же текстовый блок отображается Flash просто как текст — его содержимое без всяких прямоугольников выделения.

Рис. 7.2. Закрытый текстовый блок, выделенный с помощью инструмента "стрелка"

Подробно о работе с текстовыми блоками мы поговорим чуть позже. А пока что давайте выясним, что же это за второй способ создания текстовых блоков.

Второй способ создания текстового блока аналогичен рисованию прямоугольника или эллипса. Помещаем курсор мыши в том месте, где будет располагаться левый верхний угол текстового блока, и нажимаем левую кнопку

мыши. Затем, не отпуская эту кнопку, протащим мышь в то место, где будет находиться правый нижний угол текстового блока. Пока мы буксируем мышь, Flash будет отображать синий "резиновый" прямоугольник, так что всегда можно следить за тем, что же у нас получается. Переместив курсор мыши в нужную точку, отпустим ее левую кнопку. Текстовый блок будет тотчас нарисован, и внутри него появится текстовый курсор.

Созданный таким образом текстовый блок (рис. 7.3) имеет неизменную ширину, но будет растягиваться по высоте, чтобы вместить в себя весь набранный нами текст (текстовый *блок с фиксированной шириной*). Разумеется, Flash при этом будет выполнять автоматический перенос строк в случае, если текст не умещается в текстовом блоке по ширине.

Рис. 7.3. Открытый текстовый блок с фиксированной шириной

На рис. 7.3 видно, что текстовый блок с фиксированной шириной отличается от его собрата с фиксированной высотой формой маркера, находящегося в правом верхнем углу прямоугольника выделения. Этот маркер имеет форму полого квадратика. В закрытом же виде такой текстовый блок отображается так же, как показано на рис. 7.2.

Текстовые блоки с фиксированной шириной хорошо подходят для помещения в изображение Flash больших фрагментов текста.

Итак, что мы можем делать с текстовыми блоками? Мы можем выделять их щелчком мыши; правда, при этом придется щелкать на одном из символов текста. Если текст набран очень мелким шрифтом, лучше всего применять выделение перетаскиванием. Выделенный текстовый блок можно переместить и удалить; делается это так же, как в случае уже знакомых нам линий и заливок.

Мы можем править текст, помещенный в текстовый блок, если откроем его. Чтобы открыть текстовый блок, нужно:

- если в данный момент выбран инструмент "стрелка", дважды щелкнуть на нужном текстовом блоке, либо щелкнуть один раз и выбрать пункт **Edit Selected** меню **Edit**;
- если выбран инструмент "текст", щелкнуть на текстовом блоке один раз.

Как только мы выберем любой другой инструмент или нажмем клавишу <Esc>, все сделанные нами правки будут применены, а текстовый блок — закрыт. Закрытый текстовый блок после этого останется выделенным.

А еще мы можем преобразовать текстовый блок с фиксированной высотой в текстовый блок с фиксированной шириной. Для этого достаточно изменить его ширину с помощью любого из соответствующих маркеров. Высоту текстового блока мы изменить не сможем — Flash сам будет управлять ей.

Также есть возможность превратить текстовый блок с фиксированной шириной в текстовый блок с фиксированной высотой. Для этого достаточно открыть его и дважды щелкнуть маркер, находящийся в правом верхнем углу. После этого Flash сам установит размеры текстового блока такими, чтобы "охватить" весь содержащийся в нем текст.

Форматирование текста

Но возможности Flash отнюдь не исчерпываются набором простого текста, разбитого на абзацы с помощью клавиши <Enter>. Текст может содержать фрагменты, набранные разными шрифтами, с разным выравниванием и отступами. Давайте же выясним, как средствами Flash можно оформить (*отформатировать*) набранный нами текст.

Форматирование отдельных символов

Форматирование отдельных символов текста во Flash, а именно, задание шрифта и цвета, выполняется с помощью панели **Properties**. Ее вид при выделенном фрагменте текста в текстовом блоке показан на рис. 7.4.

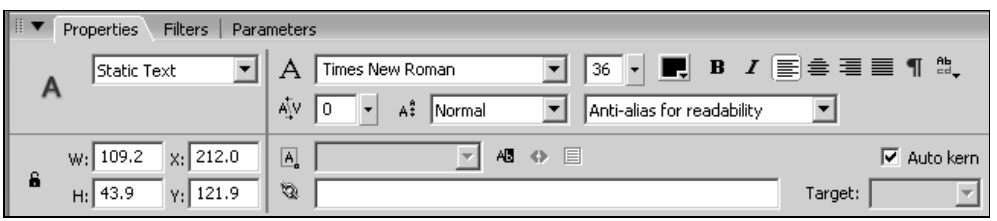


Рис. 7.4. Панель **Properties** после выделения фрагмента текста в текстовом блоке

Элементы управления, расположенные в нижнем левом углу этой панели, нам уже знакомы по *главе 4*. Они позволяют задать размеры и местоположение выбранного фрагмента изображения. Остальные же элементы управления "отвечают" исключительно за текст.

Здесь нужно сказать еще вот что. Если выделенный текстовый блок закрыт, все заданные в панели **Properties** параметры форматирования применяются ко всему тексту, помещенному в текстовый блок. Если же текстовый блок открыт, то эти параметры будут применены к вновь набираемому тексту. Если же мы выделим какой-либо фрагмент текста в открытом текстовом блоке, то заданные нами параметры будут применены к выделенному тексту.

Но вернемся к панели **Properties**.

Раскрывающийся список выбора шрифта, позволяющий задать шрифт текста, заметен сразу (слева от него стоит большая буква A). В этом списке приведены все шрифты, установленные в системе. Во время просмотра раскрытого списка в небольшом окошке для нашего удобства выводится пример текста, набранного этим шрифтом.

Правее списка выбора шрифта находится поле ввода с регулятором, с помощью которого задается размер (или, как говорят полиграфисты, *кегль*) шрифта в типографских пунктах. Еще правее имеются: кнопка селектора цвета, задающего цвет шрифта, и две кнопки, задающие его начертание, — **B** (полужирное) и ***I*** (курсивное).

Для выбора шрифта мы можем также воспользоваться подменю **Font** меню **Text** — его многочисленные пункты представляют все установленные в системе шрифты. Аналогично выбрать размер шрифта можно, воспользовавшись подменю **Size** меню **Text** — там перечислены наиболее употребительные размеры шрифтов. Такой способ выручает, если панели **Properties** нет на экране.

В том же меню **Text** находится подменю **Style**, позволяющее задать:

- полужирное начертание шрифта (пункт-переключатель **Bold** и комбинация клавиш <Ctrl>+<Shift>+<B>);
- курсивное начертание шрифта (пункт-переключатель **Italic** и комбинация клавиш <Ctrl>+<Shift>+<I>);
- обычный вид шрифта (пункт-переключатель **Plain** и комбинация клавиш <Ctrl>+<Shift>+<P>).

Прямо под списком выбора шрифта находится поле ввода с регулятором, с помощью которого задается так называемый *трекинг* — параметр, регулирующий пространство между символами. Это иногда бывает очень полезно — например, для выделения заголовков "разрядкой" текста.

Значение трекинга вводится в пунктах, причем можно вводить как положительные, так и отрицательные значения. В первом случае символы текста раздвигаются, а во втором — сближаются. При этом главное — не переборщить с трекингом, иначе символы или "разбредутся" слишком далеко друг

от друга, перестав быть единым текстом, или сольются друг с другом, и текст станет совершенно нечитаемым.

Задавать значения трекинга также можно, использовав подменю **Letter Spacing** меню **Text**. Пункт **Increase** (и комбинация клавиш <Ctrl>+<Shift>+ +<стрелка вправо>) увеличивает трекинг, а пункт **Decrease** (и комбинация клавиш <Ctrl>+<Shift>+<стрелка влево>) — уменьшает. Пункт **Reset** (и комбинация клавиш <Ctrl>+<Shift>+<стрелка вверх>) позволяет вернуть значение трекинга по умолчанию.

Правее поля задания трекинга расположен раскрывающийся список, задающий положение символов относительно *базовой линии* текста — воображаемой линейки, по которой "пишется" текст. В списке доступны три пункта:

- **Normal** — обычное положение текста, когда он "лежит" на базовой линии;
- **Superscript** — верхний индекс;
- **Subscript** — нижний индекс.

Мы также можем воспользоваться пунктами-выключателями **Superscript** и **Subscript** подменю **Style** меню **Text**. Первый превращает текст в верхний индекс, второй — в нижний. Если оба этих пункта отключены, текст имеет обычное положение на базовой линии.

В правом нижнем углу панели **Properties** расположен флажок **Auto Kern**. Он включает или отключает *кернинг* — специальное управление пространством между символами. Дело в том, что некоторые пары символов требуют задания различных значений промежутка между ними — например, промежуток между A и W должен быть меньше, чем между A и H. Именно этим и занимается кернинг; благодаря ему текст выглядит более ровным.

Форматирование абзацев

Форматирование целых текстовых абзацев, а именно, задание выравнивания и отступов, выполняется с помощью все той же панели **Properties** (см. рис. 7.5). Для этого используются другие, не рассмотренные еще нами элементы управления

Заданные параметры форматирования применяются к тому абзацу, на котором стоит текстовый курсор (естественно, текстовый блок должен быть открыт). Если текстовый блок закрыт, эти параметры будут применены сразу ко всем абзацам содержащегося в нем текста.

Выравнивание текста в абзаце задается с помощью находящегося в верхнем правом углу панели **Properties** набора из четырех кнопок-переключателей.

Для этого также можно воспользоваться пунктами-переключателями подменю **Align** меню **Text**.

Перечислим все кнопки этого набора в порядке слева направо:

- первая кнопка задает выравнивание по левому краю (пункт **Align Left** подменю **Align** меню **Text** и комбинация клавиш <Ctrl>+<Shift>+<L>);
- вторая — выравнивание по центру (пункт **Align Center** и комбинация клавиш <Ctrl>+<Shift>+<C>);
- третья — выравнивание по правому краю (пункт **Align Right** и комбинация клавиш <Ctrl>+<Shift>+<R>);
- четвертая — полное выравнивание (пункт **Justify** и комбинация клавиш <Ctrl>+<Shift>+<J>).

Правее этого набора кнопок находится кнопка **Edit format options** ( ). Если ее нажать, на экране появится небольшое диалоговое окно **Format Options** (рис. 7.5), с помощью которого мы можем задать другие параметры абзаца.

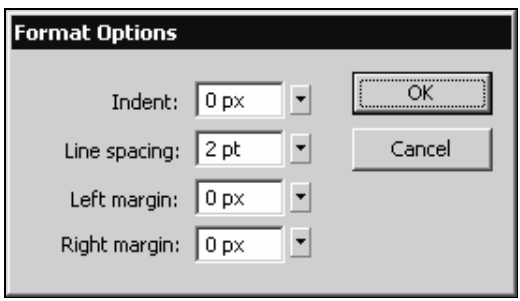


Рис. 7.5. Диалоговое окно **Format Options**

Поле ввода **Indent** задает значение отступа *красной строки*, т. е. расстояние между левой границей текстового блока и первым символом первой строки абзаца. Это значение задается в пикселах.

Поле ввода **Line spacing** задает расстояние между строками по вертикали в типографских пунктах.

Поля ввода **Left margin** и **Right margin** задают отступ, соответственно слева и справа, текста от границ текстового блока. Эти значения задаются в пикселах.

Чтобы применить заданные параметры, нужно нажать кнопку **OK**. Кнопка **Cancel** позволит нам от них отказаться.

Поддержка шрифтов во Flash

А теперь самое время поговорить о различных шрифтах и их поддержке во Flash.

Из *главы 1* мы помним, что все графические изображения можно разделить на векторные и растровые. Аналогично этому, на векторные и растровые можно разделить и все компьютерные шрифты. В растровых шрифтах символы хранятся в виде массива точек, а в векторных — в виде наборов параметров кривых. В этом смысле каждый шрифт можно считать набором изображений, соответственно векторных или растровых, представляющих его символы.

В операционных системах семейства Windows для вывода текста применяются и векторные, и растровые шрифты. В растровом виде хранятся почти все системные шрифты, которыми набраны заголовки окон, пункты меню и надписи к элементам управления. В частности, растровыми являются шрифты MS Sans Serif и Fixedsys. Векторными шрифтами обычно набираются текстовые документы; к ним относятся популярные шрифты Arial, Courier, Times New Roman, Tahoma и Verdana.

Векторные шрифты, используемые в системах Windows, хранятся в формате *TrueType*, разработанном фирмой Apple. Из других форматов хранения векторных шрифтов можно выделить разработанный фирмой Adobe формат *Type 1*, который применяют, в основном, в профессиональных программах верстки.

Здесь нужно сказать, что для набора текстов Flash позволяет использовать только векторные шрифты, т. е. шрифты форматов TrueType и Adobe Type 1. Любые растровые шрифты игнорируются и не выводятся в списке выбора шрифта панели **Properties**. Это первое, что мы должны запомнить.

А теперь представим себе такую довольно часто встречающуюся ситуацию. Мы создали изображение Flash и использовали в нем хитроумный шрифт, который не поставляется стандартно с Windows (где мы его взяли — другой вопрос). Далее мы экспортировали это изображение в формат Shockwave/Flash и каким-либо образом распространили. Это изображение получил пользователь и пытается его просмотреть. Так сможет ли он это сделать, если на его компьютере использованный нами шрифт не установлен?

Сможет. Дело в том, что когда мы экспортируем готовое изображение в формате Shockwave/Flash, Flash сохраняет в результирующем файле описания всех использованных в изображении шрифтов. Этот процесс называется *внедрением шрифтов*. При просмотре нашего изображения проигрыватель Flash извлекает эти описания и с их помощью отображает текст в том виде, как он должен выглядеть.

Самое интересное, что при внедрении шрифтов Flash старается по мере сил навести экономию. Он внедряет в результирующий файл Shockwave/Flash не все символы шрифтов, а только те, что реально были использованы в тексте. Так что если мы добавим в свое изображение единственный текстовый блок, содержащий коротенькую надпись, полученный после экспорта файл Shockwave/Flash увеличится в размерах ненамного.

> ***Внимание!***
>
> Сказанное выше относится только к файлам Shockwave/Flash. Если же вы передадите кому-то другому исходный документ Flash и у этого "другого" не окажется нужного шрифта, текст будет выведен в искаженном виде, так как Flash будет вынужден использовать один из тех шрифтов, чт. е. на компьютере получателя. Поэтому позаботьтесь о том, чтобы у вашего коллеги оказались в наличии все нужные для работы шрифты. Есть, правда, еще один способ решить проблему недостающих шрифтов — подстановка, но о ней мы поговорим в конце этой главы.

Всем хорош такой подход, кроме одного — если мы использовали в своем изображении много разных и, вдобавок, сложных шрифтов, результирующий файл Shockwave/Flash может оказаться очень большим. Как следствие, те, кто загружает наши творения из Интернета, могут, потеряв терпение, просто прервать загрузку. Что же делать?

Из этой ситуации есть четыре выхода.

Выход первый, спартанский. Нужно умерить аппетиты. Иначе говоря, использовать в создаваемых изображениях небольшой набор относительно простых шрифтов. В результате файл Shockwave/Flash хоть и "раздуется", но ненамного, и пользователи останутся довольны.

Выход второй, компромиссный. Мы можем в некоторых случаях использовать так называемые *шрифты-псевдонимы* Flash. (В терминологии Flash они названы device fonts, т. е. шрифты устройства.) Это шрифты _sans, _serif и _typewriter. Каждый из них представляет целую группу родственных шрифтов, а именно:

- _sans обозначает все шрифты без засечек (*серифов*), например, Arial или Helvetica;
- _serif обозначает шрифты с засечками, например, Times New Roman;
- _typewriter обозначает моноширинные шрифты, наподобие Courier или Lucida Console.

Так вот, шрифты-псевдонимы не внедряются в результирующий файл Shockwave/Flash. Вместо этого проигрыватель Flash будет искать подходящий шрифт из набора, установленного на компьютере клиента. Возможно, текст в этом случае отобразится не совсем так, как мы задумывали, зато ре-

зультирующий файл Shockwave/Flash получится меньшего размера, так как ничего лишнего в него не добавится.

Разработчики Flash настоятельно рекомендуют для набора обычного (информативного, а не декоративного) текста использовать именно шрифты-псевдонимы. Более того, утверждается, что на малых кеглях (10 пунктов и менее) текст, набранный шрифтами-псевдонимами, выглядит лучше, чем набранный обычными шрифтами. К тому же, шрифты-псевдонимы не внедряются в результирующий файл Shockwave/Flash, что само по себе является большим достоинством.

Чтобы задать для текста какой-либо шрифт-псевдоним, нужно выбрать соответствующий пункт (**_sans**, **_serif** или **_typewriter**) в раскрывающемся списке выбора шрифта панели **Properties**. Эти пункты расположены в самом верху данного списка.

Выход третий, для уверенных в себе и в потребителях своей графики. Можно указать, чтобы Flash не внедрял в результирующий файл Shockwave/Flash шрифт, которым набран текст в каком-либо текстовом блоке. Разумеется, это следует применять для тех шрифтов, которые есть у всех (это шрифты Arial, Courier и Times New Roman). В противном случае проигрыватель Flash будет вынужден подобрать похожий шрифт из установленных у пользователя, и текст, скорее всего, исказится.

Выход четвертый, что называется, для "экстремалов". Мы можем указать Flash преобразовать при экспорте заданный текстовый блок в растровое изображение. Насколько при этом уменьшится в размерах результирующий файл Shockwave/Flash, неизвестно, но в некоторых случаях стоит попробовать.

Параметры текстового блока

Задавать параметры можно не только для отдельных символов или отдельного абзаца текста, но и для всего текстового блока. Этих параметров немного, но они весьма важны.

Начнем с кнопки-выключателя , расположенной в панели **Properties** (см. рис. 7.4). Если она включена, пользователь может выделять и копировать в буфер обмена текст, помещенный в выделенный текстовый блок. По умолчанию же эта кнопка отключена, и текст в текстовом блоке не может быть выделен. (Это, кстати, полезно, если мы желаем защитить какие-либо тексты от несанкционированного копирования.)

А раскрывающийся список, задающий метод отображения шрифта, которым набран текст в текстовом блоке, требует особого разговора. Этот список находится в центре панели **Properties**, правее раскрывающегося списка, за-

дающего положение текста относительно базовой линии. В нем доступно пять пунктов, которые мы сейчас рассмотрим.

Пункт **Use device fonts** позволяет указать Flash не внедрять шрифт, которым набран текст в выделенном текстовом блоке, в результирующий файл Shockwave/Flash. Мы уже говорили об этом в предыдущем разделе, и не будем сейчас повторяться.

Пункт **Bitmap text (no anti-alias)** при выборе предписывает преобразовать при экспорте изображения в формат Shockwave/Flash преобразовать выделенный текстовый блок в растровое изображение. Об этом также говорилось в предыдущем разделе.

Пункт **Anti-alias for animation** задает особый метод сглаживания шрифта для улучшения качества анимации. При этом проигрыватель Flash может проигнорировать выравнивание абзацев и кернинг шрифта. Также текст, набранный мелким шрифтом, в текстовом блоке, для которого был выбран этот пункт, может плохо читаться.

Пункт **Anti-alias for readability**, выбранный по умолчанию, задает метод сглаживания шрифта для улучшения читаемости текста. При этом проигрыватель Flash будет выводить такой текст с наилучшим качеством, даже если текст этот набран мелким шрифтом.

Пункт **Custom anti-alias** предназначен для тех, кто желает сам управлять сглаживанием шрифта. После его выбора на экране появится диалоговое окно **Custom Anti-Aliasing** (рис. 7.6), в котором задаются параметры сглаживания шрифта.

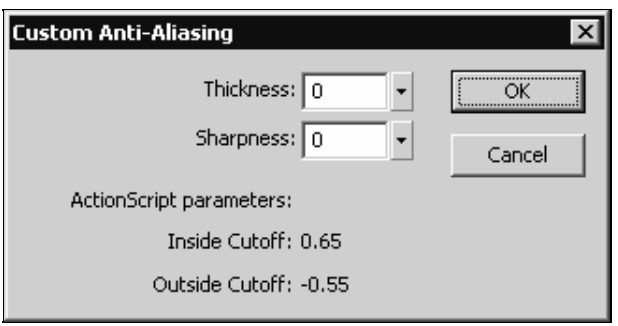


Рис. 7.6. Диалоговое окно **Custom Anti-Aliasing**

Поле ввода с регулятором **Thickness** служит для задания толщины сглаженных участков шрифтов. Большее значение этого параметра заставляет символы выглядеть более "толстыми".

Поле ввода с регулятором **Sharpness** задает величину плавности перехода между границами символов и фоном.

Задав нужные параметры сглаживания шрифта, надо нажать кнопку **OK**. Кнопка **Cancel** позволяет отказаться от задания этих параметров.

> ***НА ЗАМЕТКУ***
>
> Если мы выберем для текста в данном текстовом блоке шрифт-псевдоним, раскрывающийся список задания метода отображения шрифта будет недоступен. В этом случае в нем будет выбран пункт **Use device fonts**.

Трансформация текстовых блоков

Мы можем применять к текстовым блокам те же трансформации, что и к другим фрагментам изображения. Делается это точно так же — выделяем нужный текстовый блок, выбираем в главном инструментарии инструмент "трансформатор" и перемещаем соответствующий маркер. Таким образом, в частности, мы можем повернуть текстовый блок на 90°, создав таким образом вертикальный текстовый блок.

> ***ВНИМАНИЕ!***
>
> Для создания вертикальных текстовых блоков не следует пользоваться кнопкой с меню (самая правая в верхнем ряду на рис. 7.4). Эта кнопка предназначена для создания текстовых блоков, содержащих тексты на азиатских иероглифических языках, и результат их использования может быть совсем не тем, какой вы ожидаете.

А еще Flash предоставляет возможность работы с символами текста как с обычной графикой. Символы текста после операции разбиения, аналогичной разбиению группы, преобразуются в набор обычных контуров и заливок. Теперь мы можем менять форму отдельных символов текста, как если бы это были фрагменты обычной графики, нарисованные с помощью обычных инструментов рисования. В общем, творить с текстом все, что пожелаем.

Чтобы выполнить разбиение, нужно выделить требуемый текстовый блок и дважды выбрать пункт **Break Apart** меню **Modify**, в контекстном меню текстового блока или просто нажать комбинацию клавиш <Ctrl>+<B>. Почему дважды? Да потому, что после первого разбиения текстовый блок "распадется" на множество маленьких текстовых блоков, содержащих по одному символу текста, и только после второго каждый символ превратится в набор примитивов. Все получившиеся примитивы останутся выделенными.

Мы можем применять для преобразованного в графику текста любые знакомые нам приемы изменения формы и цвета. В частности, можно распо-

ложить символы такого текста по окружности или кривой (похоже, это единственный доступный во Flash прием сделать фигурный заголовок). Также мы можем создавать новые, причудливые шрифты или просто слегка изменять форму отдельных символов. На рис. 7.7 как раз изображен результат изменения формы обычной латинской буквы U с помощью инструмента "стрелка".

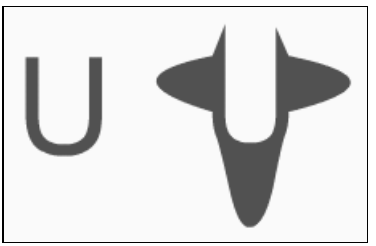


Рис. 7.7. Изменение формы литеры U
(слева — изначальный символ, справа — после изменения)

К несчастью, мы не сможем преобразовать набор полученных после разбиения текстового блока фрагментов обратно в текстовый блок. А это значит, что подобный текст не подлежит редактированию. Поэтому перед разбиением текста следует убедиться, что он не содержит ошибок.

Кстати, разбиение — это еще один, уже пятый по счету способ уменьшить размер файла Shockwave/Flash, содержащий изображение с текстовыми блоками. Ведь в этом случае внедрение шрифтов выполняться не будет. В самом деле, с точки зрения Flash, такое изображение вообще не содержит текста — только обычную графику.

Создание гиперссылок

Flash позволяет превратить фрагмент текста, находящегося в текстовом блоке, в *гиперссылку*. Делается это очень просто.

Прежде всего, выделим нужный фрагмент текста. Далее введем в большое поле ввода, находящееся вдоль нижнего края панели **Properties** (см. рис. 7.4), нужный нам интернет-адрес. Пример гиперссылки, созданной таким образом, показан на рис. 7.8.

Правее поля ввода интернет-адреса находится раскрывающийся список **Target**. С его помощью задается так называемая *цель гиперссылки*, т. е. то, будет ли Web-страница, на которую указывает гиперссылка, загружаться в

то же самое или в другое окно Web-обозревателя. Список содержит четыре пункта:

- **_blank** — Web-страница загрузится в отдельное окно Web-обозревателя;
- **_parent** — Web-страница загрузится в родительский набор фреймов (*фрейм* — своего рода "окошко" в окне Web-обозревателя, в которое может быть загружено независимое содержимое; отдельные фреймы объединяются в *наборы фреймов*);
- **_self** — Web-страница загрузится в текущий фрейм (значение по умолчанию);
- **_top** — Web-страница загрузится в текущее окно Web-обозревателя, заменив собой весь набор фреймов, если он есть.

Рис. 7.8. Гиперссылка, созданная средствами Flash (подчеркнутое слово "Macromedia")

Кроме того, мы сами можем ввести в этот список другое значение. Таким значением может быть имя фрейма, в который должна загрузиться Web-страница.

Специальные текстовые блоки

А теперь давайте поговорим о так называемых *специальных текстовых блоках*, т. е. о текстовых блоках, выполняющих особые функции. В данный момент они нам, правда, не пригодятся, зато понадобятся тогда, когда мы начнем писать программы в среде Flash (программированию посвящена *часть IV* данной книги).

Поля ввода

Поле ввода — это своего рода небольшой текстовый редактор, помещенный в окно Windows-программы и предназначенный для ввода одно- или многострочного текста. При этом Windows сама управляет текстовым вводом: мы просто набираем символы, пользуемся клавишами-стрелками, клавишами <Backspace>, <Del>, выделяем текст, используем буфер обмена, как в любом "большом" текстовом редакторе. Программисты фирмы Microsoft много

поработали над тем, чтобы сделать стандартные поля ввода Windows по-настоящему удобными.

Чтобы создать во Flash поле ввода, сначала нужно поместить на рабочий лист обычный текстовый блок и выделить его инструментом "стрелка" (при этом в текстовый блок необязательно вводить какой-либо текст). Далее находим в верхнем левом углу панели **Properties** небольшой раскрывающийся список (см. рис. 7.4) и выбираем в нем пункт **Input Text**. Если после этого мы откроем полученный текстовый блок, увидим то, что показано на рис. 7.9.

Рис. 7.9. Открытое поле ввода Flash

Итак, что мы видим? А видим мы полный набор маркеров изменения размера (они сохранятся и в том случае, если мы закроем поле ввода). Это значит, что мы можем менять также и вертикальный размер полученного поля ввода. А прямоугольный маркер из правого верхнего угла переместится в правый нижний — это знак того, что перед нами именно поле ввода.

Вид панели **Properties** после выделения поля ввода представлен на рис. 7.10. Как видим, там появились дополнительные элементы управления, которые нужно рассмотреть.

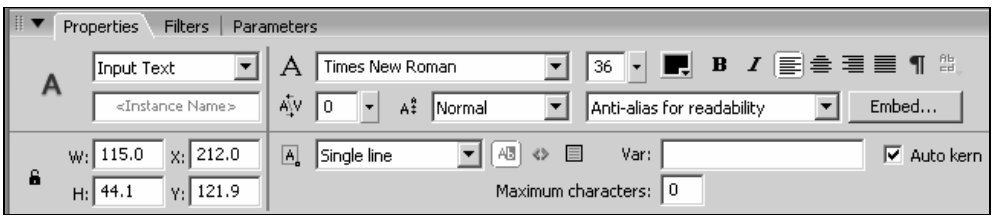


Рис. 7.10. Панель **Properties** после выделения поля ввода

Прежде всего, прямо под списком, которым мы только что пользовались, появилось поле, в котором нам нужно будет указать уникальное *имя*, идентифицирующее созданное нами поле ввода Flash. Каждое поле ввода, созданное на рабочем листе, должно иметь уникальное имя. Это имя может содержать буквы русского и латинского алфавита, цифры и знаки подчерки-

вания, причем оно обязательно должно начинаться либо с буквы, либо со знака подчеркивания.

Ниже поля ввода задания трекинга находится другой раскрывающийся список. С его помощью выбирается вид поля ввода. В этом списке доступны четыре пункта:

- **Single Line** — обычное *однострочное поле ввода*;
- **Multiline** — *многострочное поле ввода* (иначе говоря — область редактирования);
- **Multiline no wrap** — многострочное поле ввода, в котором не будет выполняться автоматический перевод строки (хотя пользователь все же может перенести строку вручную, нажав клавишу <Enter>);
- **Password** — *поле ввода пароля*. Аналогично обычному однострочному полю ввода, но скрывает введенный в него текст, отображая вместо него строку из "звездочек".

Правее этого списка находится набор их трех кнопок-выключателей. Первая из них ( ) нам уже знакома — она позволяет пользователю выделять находящийся в поле ввода текст. Как видно на рис. 7.10, эта кнопка нажата и недоступна, т. е. не может быть отключена.

Вторая кнопка-выключатель ( ), будучи включенной, предписывает Flash обрабатывать все встретившиеся в тексте теги HTML. Так, если мы заключим фрагмент текста в тег `<B>...</B>`, Flash выделит его полужирным шрифтом. Если же эта кнопка выключена, Flash выводит текст "как есть", вместе с любыми тегами HTML, которые там заданы.

Вообще говоря, эта возможность полезна, только если мы задаем содержимое поля ввода из сценария ActionScript, так что описана она здесь "с опережением". Подробную информацию о поддержке тегов HTML в полях ввода и динамических текстовых блоках можно найти в интерактивной справке Flash.

Третья кнопка-выключатель ( ) во включенном состоянии заставляет Flash показывать рамку вокруг поля ввода, отчего последнее напоминает стандартное поле ввода Windows. Если же она отключена, поле ввода имеет "плоский" вид.

Если созданное нами поле ввода должно являться частью оригинального интерфейса, лучше всего оставить эту кнопку отключенной. Если же мы желаем сделать поле ввода более привычным для пользователя или более заметным, можно включить эту кнопку.

Правее этих трех кнопок расположено поле ввода **Var**. В этом поле задается имя переменной ActionScript, в которую будет помещен введенный в созданное нами поле ввода текст. Переменная — это своего рода "ячейка камеры хранения" данных, созданная в памяти компьютера и имеющая уникаль-

ное имя, по которому ее можно однозначно определить. Мы можем обратиться к этой переменной по имени, извлечь из нее данные или поместить в нее новое значение. Во всех подробностях переменные будут рассмотрены в *главе 19*.

В поле ввода **Maximum characters** задается максимальное количество символов, которое сможет вводить в него пользователь. Если мы не хотим ограничивать это количество, то введем туда значение 0 (таково, кстати, значение поля по умолчанию).

А теперь давайте вернемся к теме внедрения шрифтов, но уже применительно к полям ввода (точнее, ко всем специальным текстовым блокам). Дело в том, что в этом случае никакое внедрение не выполняется — Flash сохраняет только имена использованных шрифтов. При воспроизведении готового изображения проигрыватель Flash сначала ищет эти шрифты на компьютере клиента и, если они есть, использует их для вывода текста. Если же подходящий шрифт не будет найден, то проигрыватель использует ближайший к искомому подобный шрифт, отчего изображение может исказиться. А это не всегда приемлемо.

Но мы все-таки можем заставить Flash внедрить в результирующий файл Shockwave/Flash описания избранных символов заданного для данного поля ввода шрифта. Сейчас мы увидим, как это делается.

Выбор символов шрифта для внедрения выполняется в диалоговом окне **Character Embedding** (рис. 7.11). Это окно появится на экране после нажатия кнопки **Embed** панели **Properties** (см. рис. 7.10).

Большой список этого окна позволит нам выбрать целые группы символов, которые должны быть внедрены в результирующий файл Shockwave/Flash. Он содержит довольно много пунктов; те же из них, что гарантированно нам пригодятся, приведены ниже:

- **All (39477 glyphs)** — все символы данного шрифта в количестве 39 477. Этот пункт приведен только для ознакомления; в реальности вряд ли кто-то будет им пользоваться;
- **Uppercase [A-Z] (27 glyphs)** — большие латинские буквы (27 символов);
- **Lowercase [a-z] (27 glyphs)** — малые латинские буквы (также 27 символов);
- **Numerals [0-9] (11 glyphs)** — арабские цифры (11 символов);
- **Punctuation [!@#$%...] (52 glyphs)** — знаки препинания и специальные обозначения (52 символа);
- **Basic Latin (95 glyphs)** — буквы большинства европейских языков (95 символов);
- **Cyrillic (499 glyphs)** — буквы кириллицы (499 символов). Пожалуй, самый важный пункт в нашем случае.

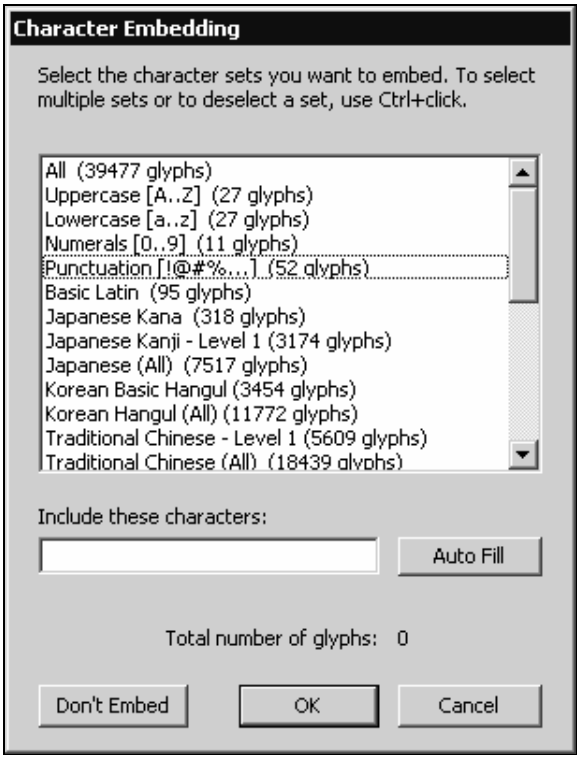


Рис. 7.11. Диалоговое окно **Character Embedding**

Как правило, в этом списке выделяются сразу несколько пунктов (правильно — кому нужны буквы без цифр!). Для этого сначала щелкнем мышью на первом пункте из набора тех, которые хотим выделить, потом нажмем клавишу <Ctrl> и, удерживая ее, щелкнем на остальных пунктах, пока не выделим все. После этого можно отпустить клавишу <Ctrl>.

Под списком шрифтов находится поле ввода **Include these characters**. В нем мы можем вручную ввести символы, которые должны быть внедрены. Такой способ полезен, если мы хотим внедрить только определенные символы шрифта. А если мы нажмем кнопку **Auto Fill**, Flash сам подставит в данное поле ввода все символы, что мы использовали в тексте выделенного поля ввода.

В самом низу окна **Character Options** находится небольшое текстовое поле **Total number of glyphs**, показывающее, сколько символов мы уже выбрали для внедрения. Так что мы сможем вовремя остановиться.

Кнопка **Don't Embed** позволит быстро удалить все символы, что мы выбрали для внедрения. При нажатии на нее будет очищено поле ввода **Include these**

characters, а в большом списке с выбранных нами пунктов пропадет выделение.

Задав параметры внедрения шрифтов в диалоговом окне **Character Embedding**, следует нажать кнопку **OK**, чтобы применить их. Если же нужно отменить эти параметры, нажмем кнопку **Cancel**.

Динамические текстовые блоки

Часто вместо того чтобы показывать обычный, неменяющийся текст, требуется выводить в текстовом виде результаты работы какого-либо сценария ActionScript. Например, такой сценарий может извлекать новости с сайта "Компьюленты", прогноз погоды — с сайта "Апорта", а курсы валют — с сайта "Рэмблера", добавлять к извлеченным данным текущее время и все это в виде текста выводить на экран. (Кстати, интересная идея...)

Вот как раз для вывода текста, являющегося результатом работы сценария, и применяются *динамические текстовые блоки*. (Обычные же текстовые блоки часто называют *статическими*.) Пожалуй, нужда в таких динамических текстовых блоках возникает чаще, чем в полях ввода.

Динамический текстовый блок создается практически так же, как поле ввода. Создаем обычный текстовый блок, выделяем его инструментом "стрелка" и находим в верхнем левом углу панели **Properties** небольшой раскрывающийся список (см. рис. 7.4). Вот только выбрать в этом списке нужно пункт **Dynamic Text**.

Выделенный динамический текстовый блок похож на поле ввода (см. рис. 7.9), да и панель **Properties** будет выглядеть примерно так же, как показано на рис. 7.10. Поэтому мы не будем давать описание находящихся в нем элементов управления, а ограничимся указанием на существующие отличия.

Для вывода текста в динамический текстовый блок используются все те же переменные. Сценарий ActionScript помещает в такую переменную выводимый текст, а сам вывод выполняет проигрыватель Flash. Имя этой переменной вводится во все то же поле ввода, что находится под списком, с помощью которого мы только что превратили обычный текстовый блок в динамический.

Теперь об отличиях. Отличие в плане интерфейса фактически имеется только одно и, вдобавок, оно нам уже знакомо. Кнопка-выключатель , включающая возможность выделения и копирования текста, помещенного в динамический текстовый блок, теперь доступна, и мы можем при желании ее отключить.

Отличие в плане возможностей состоит в том, что динамический текстовый блок может быть сделан *прокручиваемым*. Обычный динамический блок

должен непременно иметь такие размеры, чтобы в нем помещалось все его содержимое. Для прокручиваемого же блока это не обязательно: подобно областям редактирования Windows, он может "прокручивать" содержащийся в нем текст. Для управления прокруткой же нам придется писать соответствующие сценарии ActionScript — прокручиваемый текстовый блок не содержит полос прокрутки.

Чтобы сделать динамический текстовый блок прокручиваемым, нужно сначала его выделить. После этого следует включить пункт-выключатель **Scrollable** меню **Text** или одноименный пункт контекстного меню текстового блока. Если текстовый блок в данный момент открыт, можно дважды щелкнуть на квадратном маркере, расположенном в правом нижнем углу прямоугольника выделения, удерживая нажатой клавишу <Shift>. Готовый прокручиваемый текстовый блок показан на рис. 7.12; обратим внимание, что маркер, который мы щелкали, принял форму черного квадратика.

Рис. 7.12. Прокручиваемый динамический текстовый блок

Подстановка шрифтов

Напоследок мы рассмотрим один вопрос, который не касается непосредственно темы поддержки шрифтов Flash. Точнее, касается, но, как говорится, другим боком...

Представим себе такую довольно часто встречающуюся ситуацию. Мы создали во Flash какой-либо документ, использовав в нем текст, набранный очень редким хитроумным шрифтом. После этого мы передали этот документ (обратим внимание — сам документ, а не готовый файл Shockwave/Flash) на доработку своему коллеге, на компьютере которого этот шрифт не установлен. Что случится, когда он попытается открыть этот документ на своем компьютере? Ведь в этом случае Flash не найдет нужного шрифта и будет вынужден как-то выходить из положения.

И с честью выйдет из него! Как только Flash понадобится для отображения текста этот самый отсутствующий шрифт, он выведет на экран диалоговое окно **Missing Font Warning** (рис. 7.13).

Рис. 7.13. Диалоговое окно **Missing Font Warning**

Сообщение в этом окне гласит: "Flash для вывода текста нужен шрифт, который в данный момент не установлен в системе, и он будет вынужден выполнить *подстановку шрифта*", т. е. заменить отсутствующий шрифт другим, установленным в системе. Конечно, при этом текст исказится, но зато ваш коллега сможет просмотреть и отредактировать этот документ.

Для выполнения подстановки шрифта Flash предлагает нашему коллеге следующую альтернативу действий.

Первое и самое простое: отдать все на откуп Flash. В настройках Flash можно задать шрифт, который будет автоматически подставляться вместо отсутствующих (о настройках Flash см. *главу 23*). Чтобы выбрать этот вариант, следует нажать кнопку **Use Default** диалогового окна **Missing Font Warning**.

Второй и более "продвинутый" вариант: взять бразды правления в свои руки, для чего достаточно нажать кнопку **Choose Substitute** этого диалогового окна. После этого на экране появится диалоговое окно **Font Mapping**, показанное на рис. 7.14.

Бóльшую часть этого окна занимает список отсутствующих шрифтов. Он оформлен в виде таблицы из двух колонок: **Missing Fonts** (отсутствующие шрифты) и **Mapped To** (шрифты-замены). Изначально в колонке **Mapped To** из всех отсутствующих шрифтов присутствует только шрифт, заданный в настройках Flash; он обозначается пунктом **System Default Font**. Мы можем выбирать строки этого списка и задавать для соответствующих отсутствующих шрифтов заменители, выбирая нужные шрифты в раскрывающемся списке **Substitute font**. Если мы для какого-то отсутствующего шрифта хотим вернуть шрифт, заданный в настройках Flash, нажмем кнопку **System Default**.

Закончив подстановку шрифтов, нажмем кнопку **OK**, чтобы сохранить заданные нами параметры. Если мы хотим отказаться от них, нажмем кнопку **Cancel**; тогда вместо всех отсутствующих шрифтов будет подставлен шрифт, заданный в настройках Flash в качестве шрифта по умолчанию.

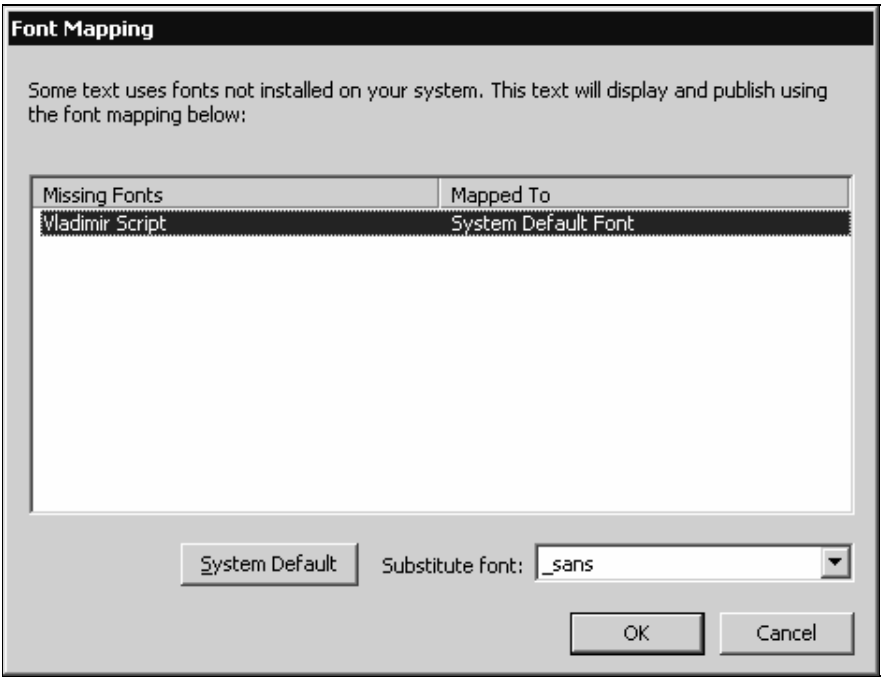


Рис. 7.14. Диалоговое окно **Font Mapping**

Впоследствии, если мы выделим текстовый блок, содержащий текст, набранный отсутствующим шрифтом, то увидим, что в раскрывающемся списке шрифтов панели **Properties** название этого (отсутствующего) шрифта помещено в скобки (рис. 7.15).

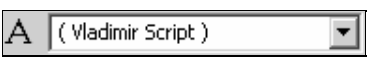


Рис. 7.15. Отсутствующий и замененный шрифт показывается в скобках

Интересная особенность Flash: после подстановки шрифтов он не сохраняет в документе сведений о шрифте-замене. То есть если наш коллега все-таки найдет недостающий шрифт и установит его на свой компьютер, то при последующем открытии вашего документа текст будет набран именно этим, изначальным шрифтом. Удобно, не правда ли?

Если мы еще раз посмотрим на рис. 7.13, то увидим, что диалоговое окно **Missing Font Warning** содержит флажок **Don't warn me again** (Больше не предупреждать меня). Если наш коллега включит его перед тем, как сделать

свой выбор, Flash будет всегда применять этот выбор, не задавая лишних вопросов.

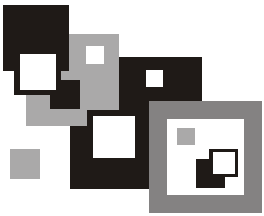
Наш коллега может также вызвать диалоговое окно **Font Mapping**, выбрав пункт **Font Mapping** в меню **Edit**. Это позволит ему управлять подстановкой шрифтов уже после того, как документ будет открыт.

Flash сохраняет для дальнейшего использования все установки, заданные в окне **Font Mapping**. И после открытия другого документа, содержащего отсутствующие шрифты, эти установки уже будут содержаться в диалоговом окне **Font Mapping**. Чтобы изменить их, наш коллега должен будет закрыть все открытые документы и выбрать все тот же пункт **Font Mapping** в меню **Edit**. На экране появится диалоговое окно **Font Mapping**, в котором он сможет выполнить любые изменения этих установок. Если же он хочет удалить какую-то установку для определенного шрифта, то должен будет выделить нужный пункт списка и нажать кнопку **Delete**, доступную только в этом случае.

Что дальше?

Вот мы и выяснили все о текстовых блоках и их поддержке во Flash. Мы познакомились также со средствами, позволяющими придать надлежащий вид тексту в изображении Flash на любом из компьютеров, на которых оно будет воспроизводиться. И, наконец, мы постигли все тонкости, связанные с подстановкой шрифтов.

Следующая глава будет посвящена образцам и библиотекам Flash. Это особое средство, позволяющее... Нет, об этом следует рассказывать как можно подробнее!



Глава 8

Образцы, экземпляры и библиотеки

В предыдущих главах мы много рисовали, редактировали, стирали, трансформировали, преобразовывали — в общем, не теряли времени даром. В этой главе мы рисовать не будем. Хватит, научились и рисовать, и править нарисованное, и даже стирать лишнее. Поговорим на этот раз о другом, а именно, об образцах и библиотеках и их использовании.

Рассмотрим такой достаточно часто встречающийся случай. Нам нужно поместить в изображение Flash множество одинаковых повторяющихся элементов. Для этого есть три пути. Первый: нарисовать каждый из этих элементов по отдельности, что, согласитесь, долго, трудоемко и просто глупо. Второй: нарисовать один элемент и продублировать его нужное количество раз, использовав пункт **Duplicate** меню **Edit**. Это намного проще и быстрее, так мы и поступим...

Стоп! А действительно ли — проще и быстрее? Хорошо, пусть мы все-таки создали изображение. А если нам срочно понадобится изменить все эти повторяющиеся элементы разом? Нам что же — придется все их перерисовывать?!

Но это еще полбеды. А главная беда в том, что размер результирующего файла Shockwave/Flash сильно увеличится. В самом деле, придется хранить в нем данные о каждом нарисованным нами однотипном элементе, которых в этом изображении очень много.

Есть ли выход из этой ситуации? Есть! Достаточно выбрать третий путь решения проблемы повторяющихся элементов. Он состоит из трех этапов.

1. Рисуем на рабочем листе первый из повторяющихся элементов нашего изображения.
2. Преобразуем нарисованный элемент в *образец* (в терминологии Flash — symbol) — своего рода шаблон для создания остальных повторяющихся элементов.

3. Создаем на рабочем листе нужное количество копий (*экземпляров*; в терминологии Flash — instance) созданного нами образца. Все — изображение готово!

Первое преимущество такого подхода налицо — становится очень просто создавать изображения с множеством повторяющихся элементов. Второе преимущество: поскольку данные экземпляра образца занимают совсем мало места, то размер результирующего файла Shockwave/Flash значительно уменьшится. Третье преимущество: чтобы изменить все повторяющиеся элементы изображения разом, нам достаточно будет исправить только образец, на основе которого они были созданы. Наконец, экземпляр образца — это "монолитный" фрагмент изображения, не подверженный фрагментации и слиянию, что во многих случаях может принести пользу.

А теперь — немного технических деталей. Созданный нами образец помещается Flash в особое хранилище, называемое *библиотекой*. Такая библиотека хранится в составе документа Flash. Каждому образцу дается при создании уникальное имя, по которому его можно опознать. Также образцу приписывается определенный *тип*, задающий его свойства.

Достаточно для начала. О других достоинствах образцов и их экземпляров мы поговорим далее, в процессе более подробного знакомства с ними. Также мы выясним, какие полезные сюрпризы припасли для нас библиотеки Flash.

Типы образцов Flash

Прежде всего нам необходимо познакомиться с типами образцов, поддерживаемых Flash. Каждый из этих типов обладает специфическими свойствами и имеет особую область применения. Всего этих типов пять.

Обычно тип образца задается при его создании, но впоследствии может быть изменен. При создании на основе образца экземпляра последний наследует от образца его тип. Однако мы можем изменить тип экземпляра, не затронув тип образца, — иногда без этого не обойтись.

Итак, перечислим пять типов образцов, поддерживаемых Flash.

- *Графический образец* (в терминологии Flash — graphic) представляет собой обычное статичное или анимированное изображение. Его экземпляры обычно используются как части более сложных изображений. Это простейший из всех образцов.
- *Образец-кнопка* (button) — это особый образец, представляющий собой обычную командную кнопку, аналогичную стандартной кнопке Windows. Экземпляры образцов-кнопок (они так и называются — *кнопки*) исполь-

зуются для создания графического пользовательского интерфейса в программах, написанных в среде Flash. Они будут рассмотрены во всех подробностях в *главе 18*.

- *Образец-клип* (movie clip) — самый сложный и мощный из образцов Flash. Образец-клип представляет собой своего рода фильм, являющийся независимой частью другого изображения или фильма. Обычно экземпляры таких образцов (*клипы*) используются для создания сложной анимации или программ, поскольку позволяют управлять своим поведением из сценариев. Мы подробно изучим их использование в *главе 13* и *части IV* этой книги.
- *Образец-шрифт* используется достаточно редко и только в разделяемых библиотеках. Как мы помним из *главы 7*, при экспорте изображения Flash в формат Shockwave/Flash в результирующий файл помещаются описания всех использованных в изображении шрифтов. Вместо этого можно преобразовать используемые в изображении шрифты в образцы-шрифты, поместить их в разделяемые библиотеки и подключить к изображениям, их использующим. Подробно создание образцов-шрифтов, как и разделяемых библиотек, будет рассмотрено в конце этой главы. Кстати, образец-шрифт — единственный тип образца, на основе которого нельзя создать экземпляр.
- *Импортированный образец* представляет импортированное в документ Flash растровое изображение, звук или видеоклип. Соответственно, импортированный образец может быть одного из трех *подтипов*: *образец-растровое изображение* (bitmap), *образец-звук* (sound) и *образец-импортированный клип* (embedded video). Все эти образцы будут рассмотрены в *главах 10*, *16* и *17*.

Из рассмотренных нами типов образцов только первые три из них могут быть полностью созданы во Flash. Образцы-шрифты создаются на основе установленных на компьютере шрифтов, т. е. нам нужно будет сначала найти и установить нужный шрифт. Ну, а с импортированными образцами все ясно...

Создание образцов

Есть два пути создания образца. Во-первых, мы можем нарисовать на рабочем листе какую-либо фигуру, а потом превратить ее в образец и поместить в библиотеку. Во-вторых, мы можем создать "пустой" образец и поместить его в библиотеку, а уже потом "наполнить" его графическим содержимым.

Предположим, что у нас на рабочем листе уже что-то нарисовано (рис. 8.1). Давайте преобразуем это "что-то" в образец.

Рис. 8.1. Фигура, которая будет преобразована в образец

Прежде всего, преобразуемую в образец фигуру нужно выделить. После этого выберем пункт **Convert to Symbol** меню **Modify**, контекстного меню выделенного фрагмента изображения или нажмем клавишу <F8>. На экране появится небольшое диалоговое окно **Convert to Symbol**, показанное на рис. 8.2.

Рис. 8.2. Диалоговое окно **Convert to Symbol**

Как уже говорилось в самом начале этой главы, каждый образец должен иметь уникальное имя, по которому Flash сможет его однозначно идентифицировать. Имя образца может содержать практически любые символы: буквы русского и латинского алфавита, цифры, знаки препинания, символы подчеркивания и пробелы. Вводится оно в поле ввода **Name**.

С помощью группы переключателей **Behavior** задается тип образца: графический образец (переключатель **Graphic**), образец-кнопка (переключатель **Button**) или образец-клип (переключатель **Movie clip**).

Правее группы переключателей **Behavior** находится переключатель точки отсчета **Registration**. Точка отсчета, как мы помним из *главы 4*, — это точка, относительно которой будет вестись отсчет местоположения экземпляра

этого образца и выполняться вращение экземпляра. Переключатель позволяет выбрать девять мест, куда можно поместить точку отсчета простым щелчком мыши на белой точке; по умолчанию она находится в левом верхнем углу образца и помечена черной точкой.

Закончив задание параметров образца, нажмем кнопку **OK**, чтобы создать его. Если мы передумали создавать новый образец, нажмем кнопку **Cancel**.

После того как созданный нами образец будет добавлен в библиотеку, фрагмент изображения, на основе которого он был создан, станет его экземпляром. Так что наше изображение не будет испорчено.

Теперь, чтобы посмотреть на созданный образец, откроем панель **Library** (рис. 8.3). Для этого выберем пункт **Library** в меню **Window** или нажмем комбинацию клавиш <Ctrl>+<L>.

Рис. 8.3. Панель **Library**

Эта панель разделена на две части. В нижней части находится список уже созданных образцов; мы можем выбирать эти образцы, щелкая мышью на их именах. Левее имен в этом списке находятся небольшие пиктограммы, обозначающие тип образца. В верхней же части панели находится область предварительного просмотра, где отображается содержимое образца, вы-

бранного в нижней части панели. Мы можем менять размеры этих частей, передвигая мышью серую полосу, разделяющую их. И конечно, мы можем менять размеры самой панели **Library**.

Кстати, если на экране присутствует панель **Library**, мы можем просто перетащить в нее выделенный фрагмент изображения, чтобы превратить его в образец. В этом случае на экране также появится диалоговое окно **Convert to Symbol** (см. рис. 8.2), в котором мы сможем задать параметры создаваемого образца.

Подробнее панель **Library** и работу с ней рассмотрим далее в этой главе. Сейчас же давайте узнаем о втором способе создание образца — "с нуля".

Чтобы создать и поместить в библиотеку новый "пустой" образец, нужно выбрать пункт **New Symbol** меню **Insert** или нажать комбинацию клавиш <Ctrl>+<F8>. Если панель **Library** открыта, также можно выбрать пункт **New Symbol** в ее дополнительном меню или нажать кнопку , которая находится в нижнем левом углу этой панели. На экране появится диалоговое окно **Convert to Symbol** (см. рис. 8.2), в котором мы сможем задать параметры создаваемого образца.

После нажатия кнопки **OK** окна **Convert to Symbol** рабочий лист очистится, и мы сможем нарисовать нужный образец. В центре листа виден небольшой черный крестик — это точка отсчета образца. К сожалению, мы не сможем переместить ее на другое место — вместо этого придется сместить сам нарисованный образец.

Закончив рисование образца, выберем пункт **Edit Document** в меню **Edit** или нажмем комбинацию клавиш <Ctrl>+<E>. Также можно нажать кнопку , расположенную в левой части инструментария документа. Flash вернется в обычный режим работы — редактирование всего изображения. А в списке панели **Library** появится только что созданный нами образец.

Создание экземпляров

Что же до экземпляров, то они создаются совсем просто. Достаточно вывести на экран панель **Library**, выбрать в списке образцов этой панели нужный образец и перетащить его на рабочий лист. На листе будет создан новый экземпляр выбранного нами образца (рис. 8.4).

Обратим внимание, что созданный нами экземпляр выделен тонким синим прямоугольником выделения — как группа или примитив, нарисованный в режиме объектного рисования. Это значит, что мы не можем его править, как обычную графику; для этого придется исправить образец, на основе которого он был создан. Также обратим внимание, что в центре экземпляра небольшим кружком помечена его точка отсчета.

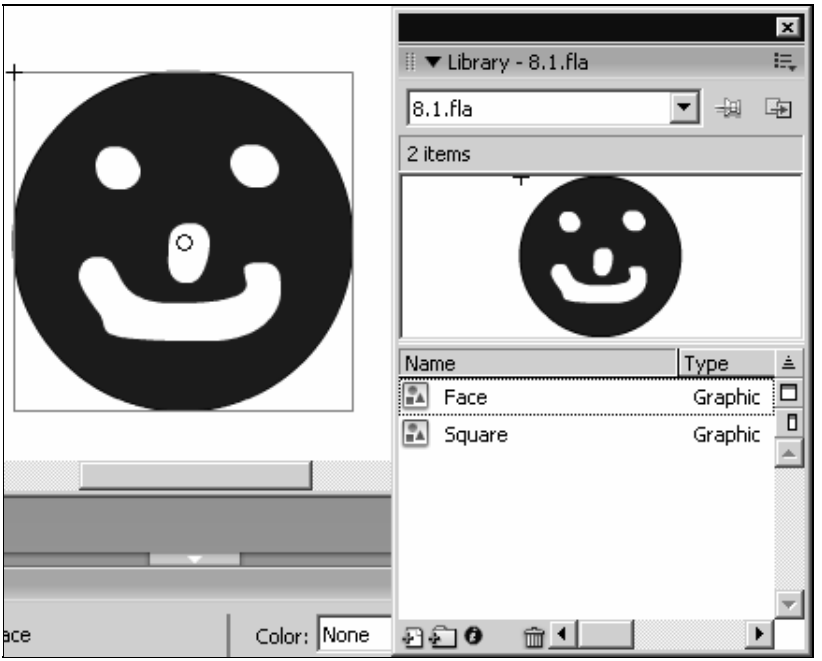


Рис. 8.4. Образец (в панели **Library**)
и созданный на его основе экземпляр (на рабочем листе)

Работа с экземплярами

Flash позволяет изменить некоторые параметры созданных экземпляров. В частности, можно изменять их цвета, задавать различные трансформации и пр. Давайте поговорим об этом.

Прежде всего, мы можем управлять порядком перекрытия экземплярами друг друга. Как это делается, мы узнали еще в *главе 4*, так что не будем сейчас повторяться.

Задание цветового эффекта

Еще можно задавать для выделенного на рабочем листе экземпляра так называемый *цветовой эффект*. Например, мы можем изменить оттенок экземпляра или сделать его полупрозрачным.

Задание цветового эффекта для экземпляра выполняется с помощью раскрывающегося списка **Color** (рис. 8.5), который находится в панели **Proper-**

ties. Этот список содержит пять пунктов, задающих различные цветовые эффекты. Рассмотрим их по порядку.

Рис. 8.5. Раскрывающийся список **Color** панели **Properties** (выбран пункт **Brightness**)

Выбор пункта **None** убирает все цветовые эффекты, заданные для экземпляра ранее. Фактически этот пункт возвращает экземпляр в первоначальное состояние.

Выбрав пункт **Brightness** (как на рис. 8.5), мы можем изменить *яркость* экземпляра, т. е. общую яркость всех цветов всех примитивов, составляющих этот экземпляр. При выборе этого пункта правее раскрывающегося списка **Color** появится поле ввода с регулятором, в которое и вводится нужное значение яркости. При этом доступны значения от −100% до 100%; отрицательные значения делают экземпляр более блеклым, положительные — более ярким.

Пункт **Alpha**, будучи выбранным, позволит изменить общую прозрачность экземпляра. В этом случае правее раскрывающегося списка **Color** также появится поле ввода с регулятором, где задается значение прозрачности. Доступен ввод значений от 0 до 100%. Чем меньше введенное значение, тем прозрачнее экземпляр; если же значение равно нулю, экземпляр становится полностью прозрачным, т. е. невидимым.

Если выбран пункт **Tint**, мы можем задать *оттенок* экземпляра, иначе говоря, цвет, который будет добавлен к цветам экземпляра. При выборе этого пункта в панели **Properties** появится несколько элементов управления, с помощью которых можно задать этот оттенок (рис. 8.6).

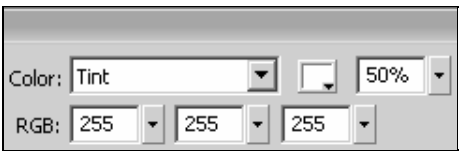


Рис. 8.6. Раскрывающийся список **Color** панели **Properties** (выбран пункт **Tint**)

Задать нужный оттенок можно с помощью расположенного правее списка **Color** селектора цвета. Также можно ввести соответственно его красную, зе-

леную и синюю составляющие в поля ввода с регуляторами **R**, **G** и **B**, расположенными под списком **Color**.

Правее селектора цвета (см. рис. 8.6) находится еще одно поле ввода с регулятором. С его помощью мы можем задать степень окрашенности экземпляра выбранным нами оттенком. Допустимый диапазон значений — от 0 до 100%. Чем больше это значение, тем сильнее преобладает заданный нами оттенок; нулевое значение задает полное его отсутствие, а выбор значения, равного 100%, приводит к закраске этим оттенком всего экземпляра, полностью заглушая его изначальные цвета.

Последний пункт раскрывающегося списка **Color — Advanced** — позволяет задать самые сложные цветовые эффекты. Если его выбрать, в панели **Properties** появится кнопка **Settings**, при нажатии которой на экране появится диалоговое окно **Advanced Effect**, показанное на рис. 8.7.

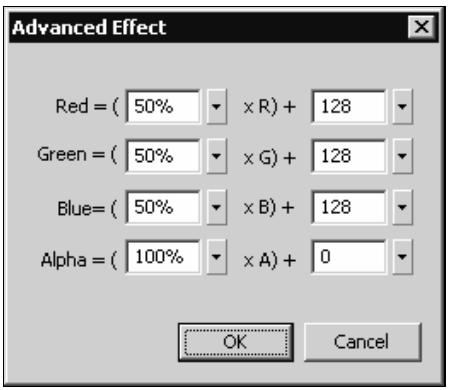


Рис. 8.7. Диалоговое окно **Advanced Effect**

Это диалоговое окно содержит четыре группы элементов управления, содержащие по два поля ввода с регулятором: **Red**, **Green**, **Blue** и **Alpha**. Они служат для изменения исходных цветов (красного, зеленого, синего) и прозрачности экземпляра соответственно. Это изменение задается следующим образом.

Возьмем для примера самую верхнюю группу полей ввода, задающую изменение красного цвета. Обозначим исходную долю красной составляющей R_0, значения, введенные в левое и правое поля ввода, — R и r, а окончательную долю красной составляющей — R_1. Тогда окончательная доля данной составляющей будет вычисляться по формуле

$$R_1 = R_0 \times R + r.$$

Предположим, например, что изначальная доля красной составляющей цвета экземпляра была 128, в левое поле ввода было введено значение 80%, а в правое — 24. Тогда окончательная доля красной составляющей будет равна

$$128 \times 0{,}8 + 24 = 126{,}4 \approx 126.$$

Как видно, цвет экземпляра можно задавать достаточно точно. Это может очень пригодиться при создании анимации.

Трансформация экземпляров

Мы можем применить к экземпляру почти все трансформации, доступные для обычного фрагмента изображения, не являющегося экземпляром: вращение, изменение размеров, сдвиг и пр. Также мы можем изменить точку отсчета экземпляра. Все эти операции были описаны в *главе 4*.

Невозможно использовать только трансформации, сильно меняющие форму экземпляра. К ним относятся деформация (модификатор "огибающая" инструмента "трансформатор") и искажение формы (одноименный модификатор того же инструмента).

Сброс всех трансформаций, примененных к экземпляру

Из *главы 4*, посвященной трансформации фрагментов изображения, мы узнали, что можно отменить (сбросить) последнюю примененную к фрагменту трансформацию, выполненную с помощью панели **Transform**. В случае экземпляров Flash предоставляет такую же возможность, за двумя исключениями.

- Сбрасывается не самая последняя, а все трансформации экземпляра. Это значит, что после выполнения операции сброса экземпляр принимает свой изначальный вид.
- Сбрасываются все трансформации, а не только примененные через панель **Transform**.

Напомним, что для сброса трансформаций нужно выбрать пункт **Remove Transform** подменю **Transform** меню **Modify** или нажать комбинацию клавиш <Ctrl>+<Shift>+<Z>. Также можно нажать кнопку , расположенную в правом нижнем углу панели **Transform**.

Смена типа экземпляра

Нам уже известно, что каждый экземпляр наследует свой тип от породившего его образца. И еще нам известно, что можно сменить тип экземпляра, не меняя типа самого образца. Сейчас мы узнаем, как это делается.

Операция смены типа выделенного экземпляра выполняется с помощью все той же панели **Properties**. В ее левом верхнем углу находится раскрывающийся список (рис. 8.8), в котором доступны три пункта: **Graphic** (Графический образец), **Button** (Образец-кнопка) и **Movie Clip** (Образец-клип). Назначение этих пунктов понятно без дополнительного разъяснения.

Рис. 8.8. Раскрывающийся список смены типа экземпляра, находящийся в панели **Properties**

Обратим внимание — возможно сменить тип экземпляра только на один из тех трех типов, образцы которых могут быть созданы в среде Flash. А вот, скажем, превратить графический образец в импортированный образец или образец-шрифт мы не сможем — у этих типов совсем другая природа.

Смена породившего экземпляр образца

Иногда бывает необходимо сменить образец, на основе которого был создан экземпляр, на другой, сохранив при этом все выполненные над экземпляром преобразования. Для этого достаточно выделить нужный экземпляр и нажать в панели **Properties** кнопку **Swap**, после чего на экране появится диалоговое окно **Swap Symbol** (рис. 8.9). Также можно выбрать пункт **Swap Symbol** подменю **Symbol** меню **Modify** или в контекстном меню выделенного экземпляра.

В списке, занимающем бóльшую часть этого диалогового окна, перечислены все имеющиеся в библиотеке текущего Flash-документа образцы. Образец, на основе которого был создан выделенный экземпляр, помечен хорошо заметной черной точкой.

Выберем в списке нужный образец. В небольшой области предварительного просмотра, расположенной левее и выше списка, будет показано его содержимое, так что мы всегда сможем выяснить, тот ли образец выбран. После этого нам останется только нажать кнопку **OK**, чтобы выполнить смену образца, или кнопку **Cancel** для отказа от этого.

Рис. 8.9. Диалоговое окно **Swap Symbol**

Разбиение экземпляра (преобразование в обычную графику)

Можно выполнить разбиение экземпляра, превратив его в обычную графику. После этого мы можем применять к полученному фрагменту изображения любые трансформации, даже использовать те инструменты, которые Flash не позволяет применять к экземплярам.

Операция разбиения экземпляра выполняется уже знакомым нам по *главе 3* способом. Выделяем нужный экземпляр и либо выбираем пункт **Break Apart** меню **Modify** или контекстного меню выделенного экземпляра, либо нажимаем комбинацию клавиш <Ctrl>+<B>. В результате Flash превратит этот экземпляр в обычную графику и выделит ее.

Работа с образцами

Мы научились работать с экземплярами, созданными на основе образцов. Давайте теперь выясним, как работать с самими образцами.

Правка образцов

Разумеется, время от времени мы будем править созданные нами образцы. В конце концов, нечасто удается сделать все так, как надо.

Flash предоставляет целых три способа исправить образец:

- в обычном режиме правки образца;
- "на месте";
- в отдельном окне документа Flash.

Мы можем выбрать любой из этих способов.

Проще всего переключиться в режим правки образца "на месте" (рис. 8.10). При этом мы сможем исправить образец прямо на рабочем листе, где находится наше изображение. Вся окружающая образец графика станет недоступной для выделения и правки; мы сможем править только сам образец.

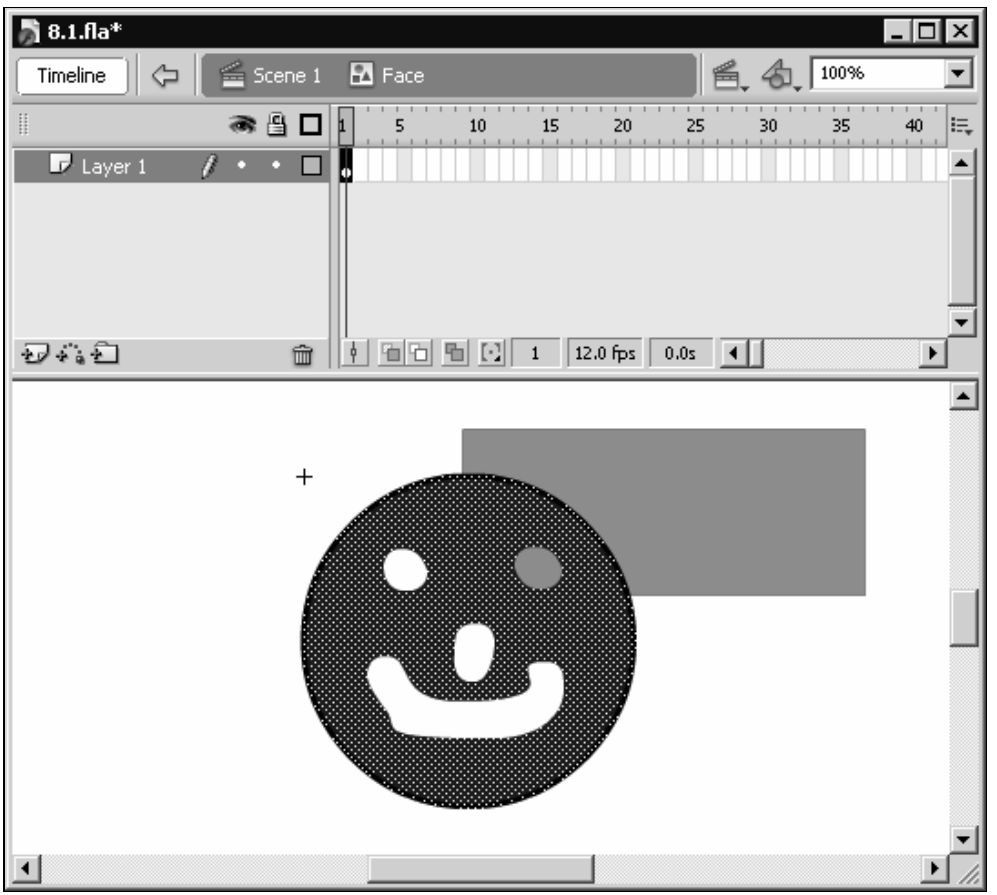


Рис. 8.10. Образец в режиме правки "на месте"

Чтобы переключиться в режим правки образца "на месте", нужно дважды щелкнуть по любому созданному на его основе экземпляру. Также можно выделить нужный экземпляр и выбрать пункт **Edit in Place** меню **Edit** или контекстного меню экземпляра.

Режим правки образца в отдельном окне документа Flash обладает тем преимуществом, что мы можем править образец, одновременно с ним имея перед глазами основное изображение. Конечно, для этого нам придется изменить размеры открытых окон Flash и переместить их, чтобы были видны и основное изображение, и исправляемый образец. (Как это сделать, было описано в *главе 2*.). Правда, не всем может понравиться, если в одной программе будет открыто слишком много окон...

Чтобы переключиться в режим правки образца в новом окне, достаточно выделить любой экземпляр, созданный на основе этого образца, и выбрать пункт **Edit in New Window** в его контекстном меню. При этом окно документа, в котором будет открыт образец, получит заголовок вида

<Имя файла документа>.fla:<Порядковый номер образца>,

так что мы сразу поймем, что открыто в этом окне.

Обычный режим правки образца — это такой режим, при котором Flash скрывает основное изображение, выводя на его место исправляемый образец. Так что мы будем видеть перед собой только выбранный для исправления образец — и больше ничего.

Переключиться в обычный режим правки образца можно одним из следующих способов:

- выделив любой экземпляр этого образца и выбрав пункт **Edit Symbols** меню **Edit** или нажав комбинацию клавиш <Ctrl>+<E>;
- выделив экземпляр и выбрав пункт **Edit Selected** того же меню;
- выбрав пункт **Edit** в контекстном меню выделенного экземпляра;
- нажав расположенную в правой части инструментария документа кнопку **Edit Symbols** и выбрав имя нужного образца в появившемся на экране меню (рис. 8.11);
- дважды щелкнув либо на пиктограмме слева от имени нужного образца в списке панели **Library** (см. рис. 8.3), либо на его изображении в области предварительного просмотра этой же панели;
- выделив нужный образец в списке панели **Library** и выбрав пункт **Edit** в контекстном меню этого списка или дополнительном меню панели.

Для правки образца можно применять все уже знакомые нам приемы работы с графикой. Единственное исключение: не удастся изменить точку отсчета образца штатными средствами. Для этого придется выделить весь образец

и переместить его так, чтобы точка фиксации пришлась на другое, нужное нам, место образца. (Однако штатными средствами мы можем изменить точку отсчета у любого экземпляра этого образца.)

Рис. 8.11. Кнопка **Edit Symbols** инструментария документа и ее раскрытое меню

Чтобы выйти из режима редактирования образца, проще всего выбрать пункт **Edit Document** меню **Edit**, нажать комбинацию клавиш <Ctrl>+<E> или щелкнуть по кнопке , расположенной в левой части инструментария документа. Если мы используем режим правки образца "по месту", то можем дважды щелкнуть мышью в области рабочего листа где-нибудь вне образца. Если же мы выбрали режим правки образца в отдельном окне документа, просто закроем это окно.

Дублирование образцов

Часто бывает нужно создать новый образец, лишь слегка отличающийся от уже имеющегося. В этом случае проще и быстрее всего выполнить дублирование образца, а потом уже исправить полученный дубликат.

Чтобы создать дубликат имеющегося в библиотеке образца, сначала нужно будет выделить любой созданный на его основе экземпляр. Далее следует выбрать пункт **Duplicate Symbol** подменю **Symbol** меню **Modify** или контекстного меню выделенного экземпляра. На экране появится диалоговое окно **Duplicate Symbol** (рис. 8.12).

Рис. 8.12. Диалоговое окно **Duplicate Symbol**

Ну, здесь и объяснять-то нечего. Вводим в единственное поле ввода **Symbol name** имя создаваемого образца-дубликата и нажимаем кнопку **OK**. Если мы передумали дублировать образец, нажмем кнопку **Cancel**.

После этого Flash выполнит следующие действия. Сначала он создаст новый образец, являющийся полной копией того, на основе которого был создан выделенный нами экземпляр. Далее он автоматически заменит образец, на основе которого был создан этот экземпляр, только что созданным образцом-дубликатом. Наконец, он выделит этот экземпляр, и нам останется только дважды щелкнуть на нем и внести во вновь созданный образец нужные исправления.

Если на экране присутствует панель **Library**, мы можем выделить нужный образец в ее списке и выбрать пункт **Duplicate** в контекстном или дополнительном меню этой панели. На экране появится диалоговое окно **Duplicate Symbol**, которое в этом случае будет похоже на уже знакомое нам диалоговое окно **Convert to Symbol** (см. рис. 8.2). Зададим в нем имя создаваемого образца и его тип и нажмем кнопку **OK** — образец-дубликат будет создан. Однако создавать новые экземпляры на основе этого образца нам уже придется самим.

Панель *Library*

Настало время познакомиться с этой панелью поближе. Хватит ходить вокруг да около!..

Начнем со списка образцов. Если внимательно к нему присмотреться, можно заметить, что он представляет собой таблицу, состоящую из множества колонок. Перечислим все имеющиеся в списке колонки:

- **Name** — имя образца;
- **Kind** — тип образца (`Graphic`, `Button`, `Movie Clip`, `Bitmap`, `Sound` или `Font`);
- **Use Counts** — количество экземпляров, созданных на основе этого образца; если же ни одного экземпляра создано не было, отображается прочерк;
- **Linkage** — состояние разделяемого или импортированного образца (о разделяемых и импортированных образцах будет рассказано далее в этой главе);
- **Date Modified** — дата и время последней правки образца.

Бывает, что значения, отображаемые в колонке **Use Counts** списка образцов, не всегда соответствуют действительности. Например, мы можем создать на рабочем листе несколько экземпляров какого-либо образца и обнаружить,

что в соответствующей колонке против его имени стоит прочерк. В этом случае достаточно выбрать пункт **Update Use Counts Now** дополнительного меню панели — и значения в колонке **Use Counts** отобразят реальное положение дел.

Для включения (выключения) автоматического обновления значений в колонке **Use Counts** при каждом создании и удалении экземпляров служит пункт-выключатель **Keep Use Counts Updated**. По умолчанию этот пункт выключен. Включать его тоже нужно осторожно, так как постоянное обновление значений в этой колонке отнимает системные ресурсы, которых всегда не хватает.

Но вернемся к списку образцов. Мы можем изменять ширину любой его колонки, перетаскивая мышью ее правую границу. Также можно сортировать пункты списка по значению какой-либо колонки, для чего достаточно щелкнуть на заголовке этой колонки. А кнопка , расположенная правее списка вровень с заголовками колонок, позволит нам изменить порядок сортировки на противоположный.

Ниже кнопки изменения порядка сортировки находятся еще две кнопки, показанные на рис. 8.13. Это кнопки быстрого изменения размера панели **Library**. Верхняя кнопка расширяет панель так, чтобы был виден весь список образцов со всеми его колонками (правда, в этом случае панель может занять весь экран, так что осторожнее с этой кнопкой). Нижняя же кнопка сужает панель так, чтобы была видна только колонка имен образцов. И конечно, мы можем сами менять размер панели **Library**.

Рис. 8.13. Кнопки быстрого изменения размера панели **Library**

Над областью предварительного просмотра панели **Library** отображается небольшая строка статуса. В ней показывается общее количество образцов, имеющихся в библиотеке (в это число входят также все папки, о которых рассказывается далее в данной главе).

Щелкнув в области предварительного просмотра правой кнопкой мыши, мы вызовем на экран небольшое контекстное меню. Включив в нем пункт-выключатель **Show Grid**, мы зададим показ в этой области координатной сетки. А пункты-переключатели **Movie's Background** и **White Background** задают фон области предварительного просмотра: заданный в параметрах документа и белый соответственно.

Управление образцами

Мы уже знаем, как добавлять в библиотеку новые образцы, как их править и изменять их тип. Давайте поговорим о других действиях, которые можно выполнить над образцами.

Можно выделять в списке сразу несколько образцов. Для этого следует выбрать щелчком первый образец из нужных нам, нажать клавишу <Ctrl> и, удерживая ее, продолжить щелкать на остальных образцах, которые требуется выделить. Можно также щелкнуть на первом образце, нажать клавишу <Shift> и, удерживая ее, щелкнуть на другом, — все образцы, находящиеся в списке между этими двумя, будут выделены.

Мы можем переименовать любой образец, независимо от того, были ли на его основе созданы экземпляры или нет. Для этого достаточно дважды щелкнуть на имени (не на пиктограмме) нужного образца в списке. (Можно также выделить нужный образец и либо воспользоваться пунктом **Rename** контекстного меню списка или дополнительного меню панели, либо просто нажать клавишу <F2>.) Вместо имени этого образца появится небольшое поле ввода, в котором будет подставлено старое имя. Введем туда новое имя и нажмем клавишу <Enter> для его сохранения или <Esc> для отказа от изменения имени.

После переименования образца, на основе которого уже были созданы экземпляры, эти экземпляры все равно останутся привязанными к нему. Так что мы можем сколько угодно переименовывать образцы, не опасаясь нарушить их связи с экземплярами.

Чтобы удалить ненужный образец, его следует выделить в списке панели **Library** и выбрать пункт **Delete** в контекстном или дополнительном меню или нажать кнопку **Delete** ( ), находящуюся на нижней кромке панели.

> ***ВНИМАНИЕ!***
>
> Образец удаляется без всякого предупреждения. Более того, все основанные на нем экземпляры также будут удалены с рабочего листа, и также без предупреждения. Если нужно вернуть удаленный образец и все основанные на нем экземпляры, следует выполнить описанную в *главе 3* операцию "отката".

Flash также позволяет быстро выделить все образцы, на основе которых не было создано ни одного экземпляра, — для этого достаточно выбрать пункт **Select Unused Items** в дополнительном меню панели. Впоследствии эти образцы можно будет удалить, если они нам действительно не нужны.

Мы можем изменить имя и тип любого образца, воспользовавшись диалоговым окном **Symbol Properties**, похожим на уже знакомое нам диалоговое окно **Convert to Symbol** (см. рис. 8.2). Чтобы вызвать его на экран, следует выделить нужный образец в списке панели и выбрать пункт **Properties**

контекстного меню или дополнительного меню. Также можно щелкнуть по кнопке **Properties** ( ), расположенной на нижней границе панели **Library**.

Новое имя образца вводится в поле ввода **Name**. Для задания типа образца используется группа переключателей **Behavior**. (Точку отсчета образца в этом случае мы изменить не сможем.) Задав новые данные, нужно нажать кнопку **OK** для их сохранения или **Cancel** для отказа от них.

Кстати, в диалоговом окне **Symbol Properties** доступна, кроме всего прочего, кнопка **Edit**. Нажав эту кнопку, мы сможем исправить образец в обычном режиме правки.

Тип образца можно сменить, воспользовавшись подменю **Type**, доступным в контекстном (но не дополнительном) меню. Пункт-переключатель **Graphic** превращает выбранный в списке образец в графический, пункт **Button** — в образец-кнопку, а пункт **Movie Clip** — в образец-клип.

Если в списке образцов выбран образец-клип или образец-звук, мы можем воспроизвести его прямо в области предварительного просмотра панели **Library**. Для этого выделим нужный образец в списке и выберем пункт **Play** контекстного или дополнительного меню. Чтобы остановить воспроизведение, воспользуемся пунктом **Stop**, также находящимся в контекстном или дополнительном меню. Но удобнее будет воспользоваться кнопками, находящимися в верхнем правом углу области предварительного просмотра ( ); правая кнопка запускает воспроизведение, а левая — останавливает его.

Использование папок

В *главе 5*, посвященной слоям, мы познакомились с папками. Эти папки, подобно папкам файловой системы нашего компьютера, позволяют объединить слои, хранящие сходную по назначению графику, в единое целое. Часто это бывает полезно, особенно если мы создаем очень сложное изображение.

Точно так же, как и слои, можно "складывать" в папки образцы, хранящиеся в библиотеке. Давайте же выясним, как это делается.

Самый простой путь создать новую папку и сразу же поместить в нее выделенный в списке панели **Library** образец — это выбрать пункт **Move to New Folder** контекстного или дополнительного меню. На экране в этом случае появится диалоговое окно **New Folder** (рис. 8.14). Введем в единственное поле ввода **Name** имя создаваемой папки и нажмем кнопку **OK**. (На имена папок накладываются те же ограничения, что и на имена образцов.) Чтобы отказаться от создания новой папки, нажмем кнопку **Cancel**.

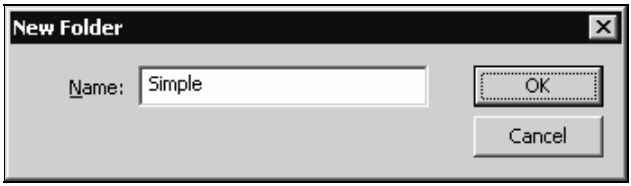


Рис. 8.14. Диалоговое окно **New Folder**

Созданная нами папка сразу же появится в списке образцов панели **Library** (рис. 8.15). В колонке **Kind** списка образцов против имени папки будет стоять слово `Folder` (папка).

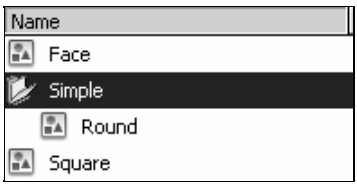


Рис. 8.15. Папка в списке образцов панели **Library** (выделена)

Щелкая на значке папки, мы можем раскрыть или закрыть ее. Для этого также можно пользоваться следующими пунктами контекстного и дополнительного меню панели **Library**:

- **Expand Folder** — раскрывает выделенную папку;
- **Collapse Folder** — закрывает выделенную папку;
- **Expand All Folders** — раскрывает все папки;
- **Collapse All Folders** — закрывает все папки.

К сожалению, поместить описанным выше способом другой образец в уже существующую папку мы не сможем. Если ввести в поле **Name** диалогового окна **New Folder** имя уже существующей папки, Flash откажется ее создавать. Поэтому для "разбрасывания" файлов по уже созданным папкам нужно пользоваться приемами, знакомыми нам по *главе 5*. Так, чтобы поместить образец в папку, его нужно перетащить на значок папки или на один из образцов, уже находящихся в ней. Чтобы "вынести" образец из папки, достаточно перетащить его куда-нибудь за пределы папки.

Чтобы просто создать новую "пустую" папку, проще всего нажать кнопку **New Folder** (), находящуюся на нижней кромке панели **Library**. Также можно выбрать пункт **New Folder** в дополнительном меню этой панели.

В списке образцов появится небольшое поле ввода; введем в него имя новой папки и нажмем клавишу <Enter>. Если же нажать клавишу <Esc>, новая папка получит имя вида `untitled folder <Порядковый номер>`.

Мы можем создать новый образец сразу в какой-либо папке. Для этого, перед тем как создать этот образец, выберем в нужной папке любой из уже находящихся в ней образцов.

Разумеется, Flash позволяет вкладывать одни папки в другие. Для этого достаточно переместить мышью одну папку на значок другой папки или на значок любой папки или образца, что вложены в эту папку.

Мы можем переименовать любую папку. Выполняется это так же, как переименование образца.

Точно так же — как в случае с образцом — выполняется удаление ненужной папки. Однако здесь нужно иметь в виду, что при удалении папки будут также удалены все находящиеся в ней образцы и папки.

Ранее было сказано, что каждый образец в библиотеке должен иметь уникальное имя. Это не совсем верно. На самом деле уникальные имена должны иметь все образцы, находящиеся в одной папке. Здесь опять прослеживается аналогия с файловой системой компьютера: на диске может находиться сколько угодно файлов с одинаковыми именами, если они при этом "разбросаны" по разным папкам. Такая структура может быть очень удобной: скажем, образцы одинакового назначения, находящиеся в разных папках, могут иметь одинаковые имена.

Но что случится, если мы переместим в какую-либо папку образец, имеющий то же имя, что и уже существующий в этой папке? В этом случае возникнет так называемый *конфликт имен* образцов, и Flash выведет диалоговое окно **Resolve Library Conflict**, показанное на рис. 8.16.

Рис. 8.16. Диалоговое окно **Resolve Library Conflict**

Если мы хотим сохранить оба этих образца, выберем переключатель **Don't replace existing items**. В этом случае перемещенный нами образец получит имя

вида <Старое имя> copy. Если же нужно, чтобы перемещенный образец заместил существовавший в папке, мы выберем переключатель **Replace existing items**. (Нужно только помнить, что ранее существовавший в папке образец будет удален безвозвратно.) После этого остается нажать кнопку **OK**, чтобы выполнить перемещение образца, или **Cancel**, чтобы отказаться от него.

Совместное использование образцов и библиотек

В свете последних тенденций глобализации и международного разделения труда будет нелишним разговор о совместном использовании образцов и целых библиотек. Речь в данном разделе пойдет о том, как сделать наши образцы доступными для других разработчиков.

Предположим, что мы работаем в большой организации, занимающейся созданием фильмов в среде Flash, в этакой Flash-киностудии. И каждый такой фильм создается несколькими людьми: один рисует персонажи, другой готовит фоновую графику, третий анимирует все, что должно двигаться, пятый занимается озвучиванием и т. д. Кино — искусство коллективное, и "сочинить" полнометражный фильм одному человеку крайне сложно. (Хотя, в принципе, возможно, что нам постоянно пытаются доказать мученики киноандерграунда.)

Допустим, наша работа — получить готовые персонажи и поместить их на готовую "сцену", где уже имеется фоновая графика. Рисовальщик персонажей передает нам документ Flash, в библиотеке которого содержатся эти самые персонажи в виде образцов. Затем специалист по фоновой графике приносит нам другой документ, куда мы должны поместить эти персонажи. И оба с чувством выполненного долга отправляются пить кофе (или что там пьют художники?). Теперь дело за нами.

К счастью, Flash предоставляет возможность открыть библиотеку другого документа и позаимствовать из нее нужные образцы. Это делается очень просто, а как — мы скоро узнаем.

Хорошо, мы это сделали! Но наши мучения не закончились. Едва мы успели скопировать персонажи в основной документ, приходит начальник и с сияющей улыбкой начинает излагать очередную свою (или чужую, выданную за свою) идею. Заключается она в следующем.

Пусть фильм, в создании которого мы принимаем участие, состоит из нескольких файлов Shockwave/Flash: Эпизод1.swf, Эпизод2.swf и т. д. Соответственно, все персонажи, участвующие более чем в одном эпизоде, встречаются сразу в нескольких файлах. А нельзя ли их как-то вынести из этих

файлов и поместить в отдельный файл? Налицо две выгоды от такого подхода. Во-первых, файлы Shockwave/Flash, составляющие фильм, получатся меньшими по размерам за счет того, что персонажи будут храниться не в каждом из этих файлов, а только в одном. Во-вторых, чтобы внести в облик какого-либо персонажа изменения — например, поменять ему прическу, достаточно будет изменить всего один файл и повторно его экспортировать.

Что мы ответим начальнику? Правильно — да! Запросто! Благо, Flash позволяет создавать разделяемые образцы и библиотеки. О них мы тоже поговорим в этой главе.

Копирование образцов из документа в документ

Самый удобный способ позаимствовать чужой образец — просто скопировать его из библиотеки нужного документа. Как мы уже знаем, Flash — многодокументное приложение, позволяющее открыть сразу несколько документов. Это значит, что мы можем открыть во Flash сразу два нужных нам документа и скопировать образцы из библиотеки одного документа в библиотеку другого.

Но как это сделать? Ведь панель **Library** всего одна! И отображает она содержимое библиотеки только того документа, что открыт в активном окне. А нам нужно иметь две панели — чтобы просматривать содержимое библиотек всех открытых документов!

Давайте взглянем на панель **Library** (см. рис. 8.3). Точнее, на небольшой инструментарий, находящийся в самой верхней ее части, выше строки статуса, где отображается количество образцов в библиотеке. Первое, что мы увидим, — это большой раскрывающийся список, показанный на рис. 8.17. В этом списке перечислены все открытые нами во Flash документы. Просто выбираем нужный — и панель **Library** выведет содержимое его библиотеки.

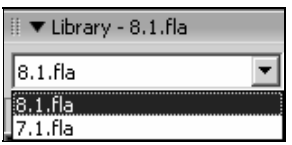


Рис. 8.17. Раскрывающийся список открытых документов, находящийся в панели **Library**

Это, конечно, замечательно, но как же вывести на экран содержимое сразу двух библиотек? А вот как. Нажмем кнопку **New library panel** (), находя-

щуюся в правой части небольшого инструментария панели **Library**. На экране появится вторая панель **Library**, изначально отображающая содержимое библиотеки того же документа, что и первая. Выбираем в раскрывающемся списке открытых документов этой панели нужный — и получаем то, что нам надо (рис. 8.18). Теперь мы можем перетаскивать образцы из одной библиотеки в другую.

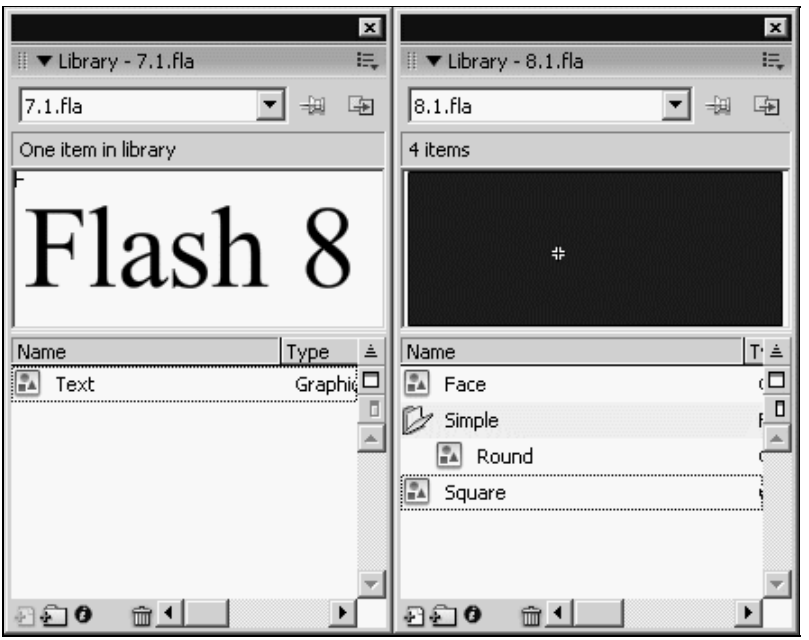


Рис. 8.18. Две панели **Library**, представляющие содержимое библиотек двух открытых во Flash документов

Собственно, нам совсем не обязательно открывать на экране несколько панелей **Library**. Мы можем просто перетащить экземпляры с рабочего листа одного документа на рабочий лист другого (конечно, в этом случае следует позаботиться о том, чтобы окна обоих документов были видны на экране) или скопировать их через буфер обмена. Flash при этом также выполнит копирование соответствующих образцов.

Если мы скопируем в библиотеку другого документа образец, имя которого совпадает с именем уже имеющегося в этой библиотеке образца, снова возникнет конфликт имен. На экране появится уже знакомое нам диалоговое окно **Resolve Library Conflict** (см. рис. 8.16), с помощью которого мы сможем указать Flash, как разрешить этот конфликт.

Иногда бывает нужно, чтобы при переключении между открытыми документами Flash панель **Library** всегда отображала содержимое библиотеки только одного документа. В этом случае мы переключимся в нужный документ и включим кнопку-выключатель **Pin current library** ( ), находящуюся в инструментарии этой панели.

Мы можем открыть нужный документ не целиком, а только его библиотеку. Для этого достаточно выбрать пункт **Open External Library** подменю **Import** меню **File** или нажать комбинацию клавиш <Ctrl>+<Shift>+<O>. На экране появится стандартное диалоговое окно открытия файла Windows; выберем в нем файл документа Flash, из которого хотим позаимствовать образцы, и нажмем кнопку открытия.

Обновляемые образцы

Присвоить чужой образец — оно, конечно, хорошо... Но вот что делать, если создатель этого образца изменил его? Скопировать повторно? Можно, конечно, поступить и так, но есть способ получше. А именно, можно заставить Flash автоматически проверять документ Flash, из которого мы скопировали образец, на предмет обновления, и самому выполнять копирование исправленных образцов.

Давайте введем пару терминов, которыми будем пользоваться в дальнейшем. Изначальный образец, изменения в котором мы хотим отслеживать, назовем *образцом-источником*. А скопированный нами образец, находящийся в нашем документе, назовем *обновляемым образцом*. Flash перед экспортом готового изображения в формат Shockwave/Flash самостоятельно или по нашей указке проверит, изменился ли образец-источник, и, если он изменился, снова скопирует его в библиотеку нашего документа, обновив соответствующий ему изменяемый образец.

Как сделать образец обновляемым? Очень просто. Сначала выведем на экран панель **Library**. Выделим в списке нужный образец и выберем пункт **Properties** контекстного или дополнительного меню, чтобы вызвать диалоговое окно **Symbol Properties**. Нажмем кнопку **Advanced**, расположенную в нижней части этого окна. Диалоговое окно **Symbol Properties** развернется в свой "полный" вид (рис. 8.19).

В левом нижнем углу этого диалогового окна видна кнопка **Browse**. Нажмем ее. На экране появится стандартное диалоговое окно открытия файла Windows. Выберем файл документа, в котором находится образец-источник, и нажмем кнопку открытия. После этого на экране появится диалоговое окно **Select Source Symbol**, внешне похожее на окно **Swap Symbol** (см. рис. 8.9). Выберем в списке этого окна нужный образец-источник и нажмем кнопку **OK**.

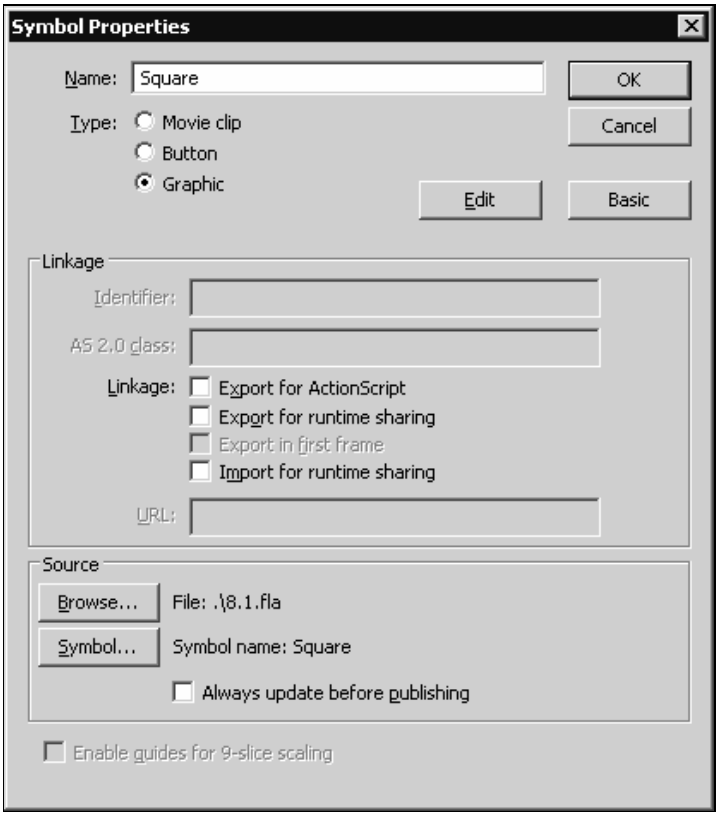


Рис. 8.19. Диалоговое окно **Symbol Properties** ("полный" вид)

Последнее, что нам останется, — это нажать кнопку **OK** окна **Symbol Properties**. Но перед этим настоятельно рекомендуется включить флажок **Always update before publishing** этого окна. Тогда Flash будет сам проверять образец-источник на предмет обновления перед экспортом нашего документа. Если мы не включим этот флажок, то сами должны будем запускать процесс проверки.

Чтобы выполнить обновление всех образцов вручную, нужно выбрать пункт **Update** контекстного меню или дополнительного меню панели **Library**. На экране появится диалоговое окно **Update Library Items** (рис. 8.20).

Бóльшую часть этого окна занимает список созданных нами обновляемых образцов. Каждый пункт этого списка имеет флажок, включив который, можно задать обновление для данного образца. (Изначально все эти флажки включены.) Процесс обновления запускается нажатием кнопки **Update**. Чтобы закрыть после этого окно **Update Library Items**, нужно нажать кнопку **Close**.

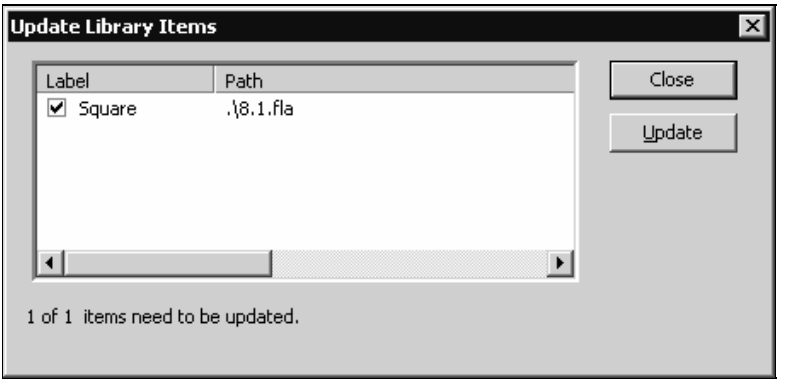


Рис. 8.20. Диалоговое окно **Update Library Items**

К несчастью, Flash никак не помечает обновляемые образцы в списке панели **Library**. Так что если мы захотим проверить, является ли какой-либо образец обновляемым, придется вызвать на экран диалоговое окно **Symbol Properties**. Если данный образец является обновляемым, в нижней части этого окна, над флажком **Always update before publishing** будут стоять имя файла документа Flash, содержащего образец-источник, и имя этого образца. Ну, и будет включен флажок **Always update before publishing**.

Вот теперь обновляемый образец создан. Мы можем проверить, действительно ли Flash отследит все обновления в этом образце.

Впоследствии мы можем изменить для любого обновляемого образца как образец-источник, так и файл документа Flash, в библиотеке которого он хранится. Для этого нужно открыть диалоговое окно **Symbol Properties**, нажать кнопку **Symbol**, выбрать нужный образец в диалоговом окне **Select Source Symbol** и нажать кнопку **OK**. Чтобы изменить файл документа, следует нажать кнопку **Browse** и выбрать нужный файл в стандартном диалоговом окне открытия файла Windows.

Разделяемые образцы

А еще Flash позволяет создавать *разделяемые образцы*. Такие образцы, будучи созданными в каком-либо изображении, могут быть использованы затем любым другим изображением Flash. При этом разделяемые образцы вообще не присутствуют в библиотеках использующих их изображений, а загружаются из библиотеки изображения, в котором они созданы.

Библиотека изображения Flash, содержащая разделяемые образцы, называется *разделяемой библиотекой*. Как правило, такое изображение не содержит

никакой графики — только саму разделяемую библиотеку, так что называть его изображением не совсем корректно.

Обычно в виде разделяемых образцов выполняют фрагменты графики, использующиеся сразу в нескольких изображениях Flash. Таким образом, во-первых, мы можем сильно сократить размеры результирующих файлов Shockwave/Flash, ведь все эти элементы будут присутствовать только в одном-единственном файле, содержащем разделяемую библиотеку. И, во-вторых, нам не придется создавать эти фрагменты во всех использующих их изображениях.

Разговор о разделяемых образцах будет долгим.

Создание разделяемых образцов

Начнем с создания разделяемых образцов. Сначала выделим в списке панели **Library** образец, который хотим сделать разделяемым, и вызовем на экран диалоговое окно **Symbol Properties**. Щелкнем по кнопке **Advanced**, чтобы развернуть его в "полный" вид (см. рис. 8.19). И рассмотрим элементы управления, расположенные в группе **Linkage**.

Первое, что нам понадобится, — это включить флажок **Export for runtime sharing**. Именно после этого Flash превратит выделенный образец в разделяемый. Нам останется только задать пару параметров, которые мы сейчас рассмотрим.

Прежде всего, каждый разделяемый образец должен иметь уникальный *идентификатор*. Использующие этот образец изображения применяют этот идентификатор, чтобы загрузить нужный им образец из разделяемой библиотеки.

На идентификатор разделяемого образца накладываются такие же ограничения, что на имена образцов, за тем исключением, что он не может содержать пробелов. Кроме того, идентификаторы разделяемых образцов должны быть уникальными в пределах всей библиотеки, в отличие от имен образцов, которые должны быть уникальны только в пределах папки.

Flash создаст идентификатор разделяемого образца сам, как только мы включим флажок **Export for runtime sharing**, и подставит его в поле ввода **Identifier**. Сформированный Flash идентификатор будет совпадать с именем образца, лишенным пробелов. Мы можем оставить его неизменным, а можем изменить.

Кроме идентификатора, Flash обязательно требует, чтобы мы ввели интернет-адрес, по которому будет находиться создаваемая нами разделяемая библиотека. Этот интернет-адрес вводится в поле ввода **URL**. Конечно, если мы точно знаем, где именно опубликуем разделяемую библиотеку, то обяза-

тельно его введем. В противном случае лучше ввести туда, скажем, интернет-адрес **http://localhost/** — он обозначает интернет-адрес компьютера, на котором мы сейчас работаем (*локальный хост*).

Задав нужные параметры, нажмем кнопку **OK**. И повторим все перечисленные выше операции для всех образцов, которые хотим сделать разделяемыми. (Обычно разделяемая библиотека включает в себя несколько разделяемых образцов.) Чтобы отказаться от создания разделяемого образца, мы нажмем кнопку **Cancel**.

После этого посмотрим на список образцов панели **Library**, а точнее, на колонку **Linkage** этого списка. У неразделяемого образца в этой колонке будет стоять прочерк, а у разделяемого — слово `Export: <Идентификатор разделяемого образца>`. Так что мы сразу увидим, какие образцы превратили в разделяемые.

Есть и еще один способ создания разделяемых образцов. Выделим в списке панели **Library** нужный образец и выберем пункт **Linkage** контекстного или дополнительного меню этой панели. На экране появится диалоговое окно **Linkage Properties** (рис. 8.21).

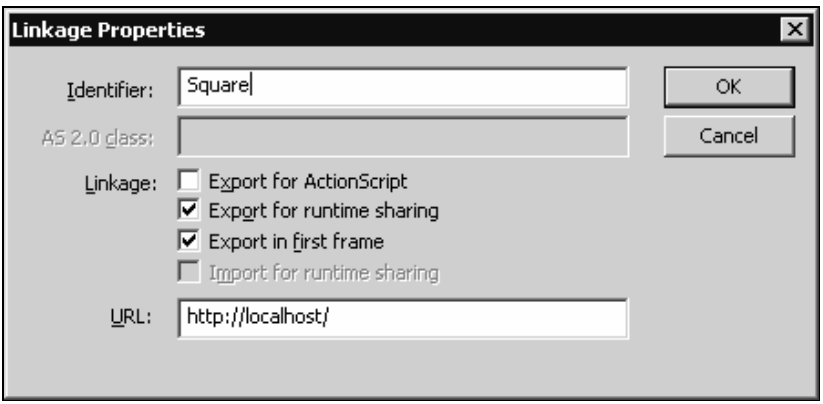


Рис. 8.21. Диалоговое окно **Linkage Properties**

Как видим, это окно содержит элементы управления, знакомые нам по группе **Linkage** окна **Symbol Properties**. Так что не будем повторяться.

Собственно, наша первая разделяемая библиотека почти готова. Осталось только экспортировать ее в формат Shockwave/Flash (о том, как это делается, *см. главу 21*).

Дальнейшие действия по созданию разделяемой библиотеки уже не связаны с Flash. Они включают в себя, прежде всего, помещение готовой разделяе-

мой библиотеки (файла в формате Shockwave/Flash) на какой-либо Web-сервер. Если же мы собираемся распространять свои Flash-творения другим способом, например, на компакт-дисках, то нужно будет записать файл разделяемой библиотеки на эти диски. Одним словом, мы должны сделать свою разделяемую библиотеку доступной.

Нужно только иметь в виду вот что. Во-первых, если проигрыватель Flash не сможет загрузить файл разделяемой библиотеки, он не сможет и правильно отобразить изображение, использующее разделяемые образцы. Во-вторых, проигрыватель Flash всегда загружает разделяемую библиотеку целиком, даже если ему нужен всего один образец, поэтому не следует делать такие библиотеки слишком большими.

Использование разделяемых образцов

Итак, наша разделяемая библиотека готова. Мы опубликовали ее на Web-сервере или распространили иным способом, например, на дискетах или по электронной почте. И теперь наш коллега хочет использовать один из содержащихся в ней разделяемых образцов. Что он для этого должен сделать?

Нужно сразу сказать, что использовать разделяемые образцы в своем документе он может двумя способами. Выбор способа зависит от того, имеет ли наш коллега доступ к исходному файлу документа нашей разделяемой библиотеки. Вообще, будет лучше, если мы предоставим ему этот файл — в этом случае он избежит некоторых проблем, о которых будет рассказано ниже.

Предположим, что мы предоставили своему коллеге этот файл. В таком случае он сможет воспользоваться первым способом.

Итак, наш коллега открывает свой документ Flash и собирается поместить в него разделяемый образец из нашей библиотеки. Для этого ему нужно сначала открыть исходный файл документа, в котором находится эта разделяемая библиотека. Он выбирает пункт **Open External Library** подменю **Import** меню **File** или нажимает комбинацию клавиш <Ctrl>+<Shift>+<O>, после чего выбирает наш файл в стандартном диалоговом окно открытия файла Windows и нажимает кнопку открытия. Напоследок ему останется вывести на экран две панели **Library**, отображающие содержимое обеих библиотек: принадлежащей его документу и разделяемой.

Теперь нашему коллеге останется перетащить нужный образец из панели **Library** разделяемой библиотеки на панель **Library** его документа, выполнив операцию *импорта образца*. Он также может перетащить этот образец прямо на рабочий лист; в этом случае Flash не только создаст экземпляр этого образца, но и скопирует образец в библиотеку его документа. Скопированный таким образом разделяемый образец будет помечен как *импортированный*: в

колонке **Linkage** списка образцов панели **Library** против его имени будет стоять слово `Import: <Имя разделяемого образца>`.

Самое интересное в том, что образец при этом копируется в библиотеку нового документа полностью, т. е. вместе со своим содержимым. Вероятно, это сделано для того, чтобы удобнее работать с такими образцами: при создании экземпляра разделяемого образца данные для этого будут взяты из библиотеки нового документа, чтобы лишний раз не тревожить разделяемую библиотеку (которая, к тому же, в данный момент может оказаться недоступной).

Но когда наш коллега выполнит экспорт готового документа в формат Shockwave/Flash, Flash не включит в результирующий файл разделяемые образцы, а поместит ссылки на них, указывающие на файл разделяемой библиотеки. Встретив эти ссылки, проигрыватель Flash загрузит файл разделяемой библиотеки и извлечет разделяемые образцы оттуда.

Кстати, наш коллега вполне сможет при желании исправить импортированный образец, воспользовавшись любым из описанных в начале этой главы режимов правки. При этом Flash выведет на экран окно-предупреждение, говорящее о том, что после правки этот образец перестанет быть импортированным, потеряет связь с разделяемой библиотекой, из которой был взят, и станет обычным образцом — "собственностью" нового документа. Нашему коллеге будет нужно нажать кнопку **OK** этого окна, чтобы все-таки выполнить правку образца, или **Cancel** — чтобы отказаться от этого.

Хорошо, наш коллега использовал какой-либо образец из разделяемой библиотеки в своем изображении. А мы тем временем исправили этот образец, снова экспортировали свою библиотеку и распространили ее. Но коллега об этом не узнает — в библиотеке его документа сохранятся данные старого образца! Как поступить в этом случае?

Проще всего нашему коллеге превратить использованный разделяемый образец в обновляемый — тогда он будет в курсе всех обновлений. Для этого он должен будет получить от нас новый исходный файл документа, содержащего разделяемую библиотеку, и привязать использованный им образец к нему. Как это сделать, было описано ранее в параграфе, посвященном обновляемым образцам.

Коллега может и проигнорировать все сделанные нами исправления. Все равно результирующий файл Shockwave/Flash будет использовать образцы из разделяемой библиотеки. А значит, все наши исправления будут актуальными, и творение нашего коллеги в любом случае отобразится нормально.

Наш коллега может изменить некоторые параметры использованного разделяемого образца. Для этого ему нужно выделить его в списке образцов панели **Library** и выбрать пункт **Properties** или **Linkage** контекстного или до-

полнительного меню панели. На экране появится одно из уже знакомых нам диалоговых окон — **Symbol Properties** или **Linkage Properties**.

Поле ввода **Identifier** служит, как мы помним, для задания идентификатора разделяемого образца. Если мы в своей разделяемой библиотеке изменили этот идентификатор, наш коллега сможет исправить его, воспользовавшись данным полем ввода.

В поле ввода **URL** задается интернет-адрес файла Shockwave/Flash разделяемой библиотеки. Изначально туда подставляется тот интернет-адрес, что мы ввели, когда превращали образец в разделяемый. Но не всегда можно точно знать, по какому именно интернет-адресу мы опубликуем нашу разделяемую библиотеку (мы, например, не знали, поэтому ввели в это поле адрес **http://localhost/**), так что возможность впоследствии исправить его очень пригодится нашему коллеге.

И, наверно, уже не стоит говорить о том, что после ввода всех нужных данных следует нажать кнопку **OK**.

А теперь предположим, что мы не дали своему коллеге исходный файл документа Flash с разделяемой библиотекой. Что он будет делать? Ведь он не сможет получить данные нужного ему разделяемого образца.

В этом случае нашему коллеге придется сначала создать новый образец, который заменит нужный ему разделяемый на время работы с документом (*образец-заглушка*). Образец может быть как "пустым", т. е. не содержащим никакой графики, так и "непустым" — это различие не принципиально. В любом случае образец-заглушка будет заменен после экспорта реальным разделяемым образцом.

Итак, образец-заглушка готов. Наш коллега выделяет его в списке образцов панели **Library** и вызывает на экран диалоговое окно **Symbol Properties**. Далее он устанавливает флажок **Import for runtime sharing**; если же этот флажок недоступен, то предварительно снимает все остальные флажки группы **Linkage**. Теперь ему останется только ввести идентификатор разделяемого образца в поле ввода **Identifier** и интернет-адрес файла Shockwave/Flash разделяемой библиотеки в поле ввода **URL** и нажать кнопку **OK**.

Разделяемые образцы-шрифты

В *главе 7*, говоря о работе с текстом в среде Flash, мы упомянули о проблеме, возникающей при экспорте готового изображения в формат Shockwave/Flash. В результирующий файл, как мы помним, помещаются описания всех шрифтов, использованных в документе, что может вызвать значительное увеличение размера этого файла. Для решения этой проблемы предлагалось несколько путей, которые не будем здесь перечислять — все они описаны в *главе 7*.

Но существует еще один путь решения данной проблемы, о котором мы еще не знаем. Это создание разделяемого образца-шрифта. Такой шрифт может быть затем использован в любом другом документе, как всякий разделяемый образец. Фактически при этом мы "выносим" описание этого шрифта из использующего его изображения и тем самым делаем содержащий его файл Shockwave/Flash меньше.

Создание разделяемого образца-шрифта происходит в два этапа. На первом этапе создается обычный, неразделяемый образец-шрифт. На втором этапе он делается разделяемым.

НА ЗАМЕТКУ

Создание обычных, неразделяемых, образцов-шрифтов лишено смысла. Описание шрифта все равно помещается в результирующий файл Shockwave/Flash при экспорте готового документа. Имеет смысл создавать только разделяемые образцы-шрифты, которые могут быть использованы в других документах.

Обычный образец-шрифт создается следующим образом. Выбираем в дополнительном меню панели **Library** пункт **New Font**. На экране появится диалоговое окно **Font Symbol Properties** (рис. 8.22).

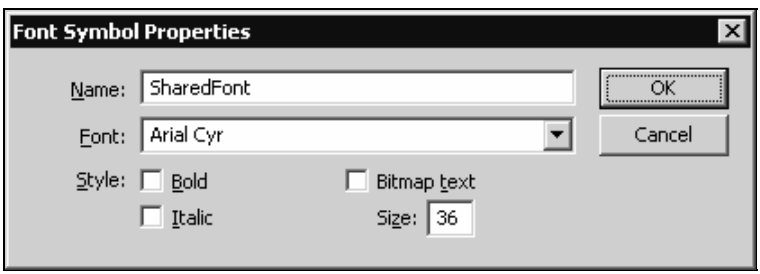


Рис. 8.22. Диалоговое окно **Font Symbol Properties**

В поле ввода **Name** задается имя создаваемого образца-шрифта. Это имя может совпадать с оригинальным именем шрифта, присутствующего в Windows. В раскрывающемся списке **Font** выбирается шрифт, на основе которого создается образец. Флажки **Bold** и **Italic** позволяют создать соответственно полужирный шрифт и курсив. Флажок **Bitmap text** заставляет Flash преобразовать символы получающегося образца-шрифта в растровые изображения; в некоторых случаях это может уменьшить размер файла Shockwave/Flash разделяемой библиотеки. А в поле ввода **Size** вводится размер шрифта.

Задав все нужные данные, нажмем кнопку **OK**. Если же мы передумали создавать образец-шрифт, нажмем кнопку **Cancel**.

Созданный таким образом образец-шрифт появится в списке образцов панели **Library**. К сожалению, Flash не предоставляет возможности посмотреть, как выглядят его символы. Для этого придется создать текстовый блок, ввести в нем какой-либо текст и отформатировать его этим шрифтом.

Если мы ошиблись в задании параметров образца-шрифта, то всегда можем исправить нашу ошибку. Выделим нужный образец-шрифт в списке панели **Library** и выберем пункт **Properties** контекстного или дополнительного меню. На экране появится диалоговое окно **Font Symbol Properties**, в котором мы сможем задать нужные параметры.

Так, образец-шрифт создан. Настала пора сделать его разделяемым.

Выделим образец-шрифт в списке образцов панели **Library** и выберем пункт **Linkage** контекстного или дополнительного меню этой панели. На экране появится диалоговое окно **Linkage Properties** (см. рис. 8.21). Зададим в нем нужные параметры (они описывались ранее в этой главе) и не забудем нажать кнопку **OK**.

Чтобы использовать готовый образец-шрифт в другом документе, мы поступим так же, как и в случае обычного образца. То есть откроем исходный файл документа, содержащий разделяемую библиотеку, воспользовавшись пунктом **Open External Library** подменю **Import** меню **File**, и перетащим образец-шрифт на панель **Library** нового документа.

После этого мы можем, как обычно, форматировать текст с использованием созданного шрифта, просто выбрав его в списке шрифтов панели **Properties** (см. рис. 7.4). Использованный образец-шрифт будет помечен в этом списке звездочкой (рис. 8.23).

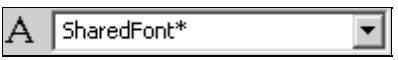


Рис. 8.23. Использованный образец-шрифт помечен в списке шрифтов звездочкой

Превращение разделяемого образца в обычный

Может получиться так, что мы захотим превратить использованный разделяемый образец в обычный, неразделяемый. Это делается следующим образом.

Вызовем на экран диалоговое окно **Symbol Properties** или **Linkage Properties** (какое — не имеет значения). Отключим переключатель **Import for runtime sharing**. И нажмем кнопку **OK**.

Библиотеки общего использования

В составе Flash поставляются три *библиотеки общего использования*, содержащие образцы, которые мы можем свободно использовать в своих изображениях. Чтобы открыть такую библиотеку, нужно выбрать соответствующий ей пункт в подменю **Common Libraries** меню **Window**. Все библиотеки, связанные с доступными пунктам этого подменю, перечислены в табл. 8.1.

Таблица 8.1. *Библиотеки общего использования, поставляемые в составе Flash*

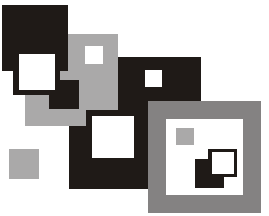
Название библиотеки	Описание
Buttons	Разнообразные кнопки (о кнопках *см. главу 18*)
Classes	Полезные объекты ActionScript 2.0 (об объектах *см. главу 19*)
Learning Interactions	Примеры различных Flash-приложений (написание приложений в среде Flash выходит за рамки данной книги)

Библиотеки общего использования — это обычные документы Flash (т. е. файлы с расширением fla). Найти их можно в папке en/Configuration/Libraries, расположенной в папке, в которой установлен сам Flash 8. Чтобы добавить во Flash свои библиотеки в качестве библиотек общего использования, достаточно просто поместить соответствующие файлы документов Flash в эту папку.

Что дальше?

На этом долгий рассказ об образцах, экземплярах и библиотеках можно считать в целом законченным. Конечно, в дальнейшем мы время от времени будем возвращаться к нему, чтобы узнать что-то новое. Образцы и библиотеки Flash — слишком обширная тема, чтобы описать ее в одной главе, даже такой большой, как эта.

А вот следующая глава будет совсем небольшой. В ней будет рассказано о новых возможностях, появившихся во Flash 8, — фильтрах и наложениях.



Глава 9

Использование фильтров и наложений

До сих пор мы изучали возможности, перекочевавшие во Flash 8 из предыдущих версий этой программы. Многие из них появились еще в самой первой версии Flash, вышедшей во второй половине 90-х годов прошлого века. Это, в основном, ключевые возможности Flash, касающиеся рисования и правки графики, простых трансформаций, управления цветом, слоями и пр.

А теперь мы познакомимся с двумя нововведениями, появившимися только во Flash 8. Это фильтры и наложения, позволяющие быстро создавать эффекты наподобие тени, размытия, сложения цветов и др. Пользователям замечательного пакета растровой Web-графики Macromedia Fireworks все это должно быть уже знакомо.

Начнем мы с фильтров, как более простых и эффектных нововведений Flash 8.

Фильтры

Фильтры Flash 8 позволяют применить к фрагменту изображения такие эффекты, как отбрасывание тени, "выдавливание", иллюзию тления и пр. Сейчас мы рассмотрим, как все это делается.

Сразу же нужно сказать вот что. Фильтры могут быть применены не ко всем фрагментам графики, а только кнопкам, клипам (экземплярам образцов-клипов) и текстовым блокам. Так что если мы хотим применить какой-то фильтр к обычному примитиву, придется преобразовать его в экземпляр образца-клипа. Как это делается, было описано в *главе 8*.

Создание фильтров

Для создания фильтров нам понадобится "универсальная" панель **Properties**. Только в этом случае нам понадобится переключиться на вкладку **Filters** этой панели (рис. 9.1). Также можно включить пункт-переключатель **Filters** подменю **Properties** меню **Window**.

Рис. 9.1. Вкладка **Filters** панели **Properties**

В левой части панели **Properties** после переключения на вкладку **Filters** появится большой список, в котором будут перечислены все созданные нами к данному моменту фильтры. (На рис. 9.1 этот список пуст.) Над этим списком будут находиться кнопки **Add filter** (со знаком "плюс") и **Remove filter** (со знаком "минус"), выполняющие создание нового и удаление выделенного в списке фильтра соответственно.

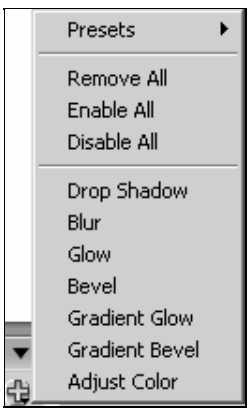


Рис. 9.2. Раскрытое меню кнопки **Add filter** панели **Properties**

Итак, чтобы создать новый фильтр, нам будет нужно выделить на рабочем листе нужный нам фрагмент изображения (кнопку, клип или текстовый

блок) и нажать кнопку **Add filter**. После этого на экране появится меню, показанное на рис. 9.2. В этом меню перечислены пункты, соответствующие доступным фильтрам, а также выполняющие различные вспомогательные операции. Пункты, создающие фильтры Flash, находятся в нижней части этого меню.

Фильтр будет создан сразу же после выбора соответствующего пункта меню кнопки **Add filter**. В списке фильтров появится и будет выбран пункт, носящий название этого фильтра. А справа от списка фильтров появятся элементы управления, с помощью которым мы сможем задать его параметры.

Мы можем создавать для выделенного фрагмента изображения несколько фильтров. При этом они будут применяться в том порядке, в котором перечислены в списке фильтров.

Фильтры, поставляемые в составе Flash

А теперь давайте подробно рассмотрим все фильтры, поставляемые в комплекте Flash, и все их параметры.

Отображение тени (*Drop Shadow*)

Фильтр, позволяющий создать тень для выделенного фрагмента изображения, создается выбором пункта **Drop Shadow** меню кнопки **Add filter**. Результат его применения показан на рис. 9.3.

Рис. 9.3. Результат применения к клипу фильтра **Drop Shadow**

Если выбрать в списке фильтров панели **Properties** пункт **Drop Shadow**, соответствующий этому фильтру, правее этого списка появятся элементы

управления для настройки его параметров. Эти элементы управления показаны на рис. 9.4.

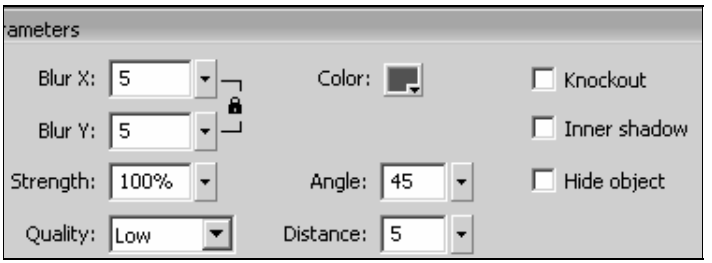


Рис. 9.4. Элементы управления панели **Properties** для настройки параметров фильтра **Drop Shadow**

Поля ввода с регуляторами **Blur X** и **Blur Y** позволяют задать размеры тени в пикселах. Если включить расположенную правее этих полей ввода кнопку-выключатель со знаком замка, Flash при изменении значения в одном из полей будет менять значение в другом поле, чтобы соблюсти пропорции тени.

Поле ввода с регулятором **Strength** позволяет задать степень насыщенности тени в процентах.

Раскрывающийся список **Quality** задает качество отрисовки тени. В нем доступны три пункта: **Low** (Низкое качество), **Medium** (Среднее) и **High** (Высокое). По умолчанию выбран пункт **Low**.

Здесь нужно помнить, что чем выше качество отрисовки тени, тем больше времени отнимает вывод изображения на экран. Разработчики вообще рекомендуют оставлять выбранным пункт **Low** (Низкое качество), если, конечно, не стоит задача получить изображение максимального качества.

Селектор цвета **Color** служит для задания цвета тени.

Поле ввода с регулятором **Angle** задает угол "падения" тени в градусах. Это поле ввода с регулятором несколько отличается от уже рассмотренных нами полей тем, что имеет круговой регулятор (*поле ввода с круговым регулятором*; рис. 9.5).

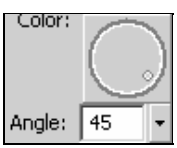


Рис. 9.5. Поле ввода с круговым регулятором

Видно, что это поле ввода имеет регулятор в виде круговой шкалы. Чтобы им воспользоваться, нужно сначала открыть его щелчком мышью по кнопке, расположенной правее поля ввода. Далее ставим курсор мыши на любую точку шкалы этого регулятора, нажимаем кнопку мыши и, не отпуская ее, буксируем мышь по вертикали, вверх или вниз. При буксировке мыши мы можем заметить, что небольшой, имеющий вид точки указатель движется по шкале регулятора. Задав нужное значение (его можно видеть в поле ввода), мы отпустим кнопку мыши, и регулятор пропадет с экрана.

Поле ввода с регулятором **Distance** позволяет задать расстояние между фрагментом изображения и тенью. Здесь можно вводить как положительные, так и отрицательные значения.

Флажок **Knockout** при включении "заливает" фрагмент изображения, к которому применяется фильтр, белым цветом. Это позволяет создать интересный эффект (рис. 9.6).

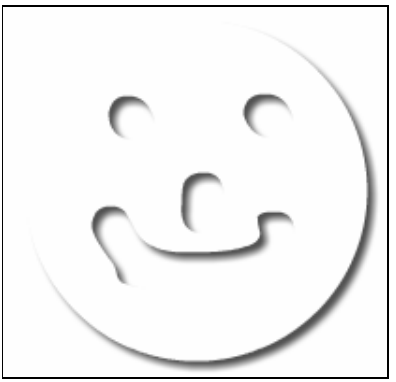


Рис. 9.6. Результат применения к клипу фильтра **Drop Shadow** с включенным флажком **Knockout**

Флажок **Inner shadow** позволяет отобразить тень "внутри" ее "владельца". А флажок **Hide object** скрывает "владельца" тени, оставляя саму тень видимой.

Размытие (*Blur*)

Фильтр, создающий эффект размытия выделенного фрагмента изображения, создается выбором пункта **Blur** меню кнопки **Add filter**. Результат его применения показан на рис. 9.7, а набор элементов управления для задания его параметров — на рис. 9.8.

Рис. 9.7. Результат применения к клипу фильтра **Blur**

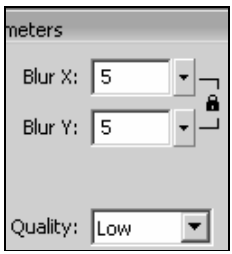


Рис. 9.8. Элементы управления панели **Properties** для настройки параметров фильтра **Blur**

Поля ввода с регуляторами **Blur X** и **Blur Y** позволяют задать размеры области размытости в пикселах. Если включить расположенную правее этих полей ввода кнопку-выключатель со знаком замка, Flash при изменении значения в одном из полей будет менять значение в другом поле, чтобы соблюсти пропорции области размытости.

Раскрывающийся список **Quality** задает качество отрисовки размытости. В нем доступны три пункта: **Low** (Низкое качество), **Medium** (Среднее) и **High** (Высокое). По умолчанию выбран пункт **Low**.

Эффект сияния (*Glow*)

Фильтр, создающий эффект сияния выделенного фрагмента изображения, создается выбором пункта **Glow** меню кнопки **Add filter**. Результат его применения показан на рис. 9.9, а набор элементов управления для задания его параметров — на рис. 9.10.

Рис. 9.9. Результат применения к клипу фильтра **Glow**

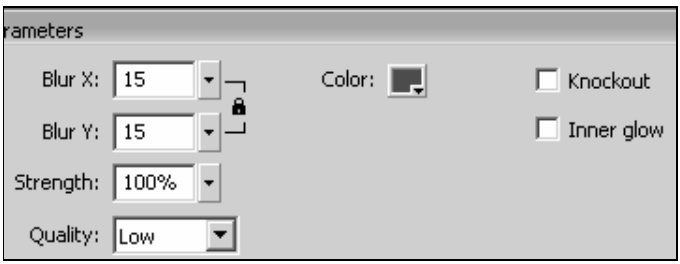


Рис. 9.10. Элементы управления панели **Properties** для настройки параметров фильтра **Glow**

Поля ввода с регуляторами **Blur X** и **Blur Y** позволяют задать размеры области "сияния" в пикселах. Если включить расположенную правее этих полей ввода кнопку-выключатель со знаком замка, Flash при изменении значения в одном из полей будет менять значение в другом поле, чтобы соблюсти пропорции области "сияния".

Поле ввода с регулятором **Strength** позволяет задать степень насыщенности "сияния" в процентах.

Раскрывающийся список **Quality** задает качество отрисовки эффекта тления. В нем доступны три пункта: **Low** (Низкое качество), **Medium** (Среднее) и **High** (Высокое). По умолчанию выбран пункт **Low**.

Флажок **Knockout** при включении "заливает" фрагмент изображения, к которому применяется фильтр, белым цветом. Флажок **Inner glow** позволяет отобразить имитацию "сияния" внутри ее "владельца".

Эффект вдавливания (*Bevel*)

Фильтр, создающий эффект вдавливания выделенного фрагмента изображения, создается выбором пункта **Bevel** меню кнопки **Add filter**. Результат его применения показан на рис. 9.11, а набор элементов управления для задания его параметров — на рис. 9.12.

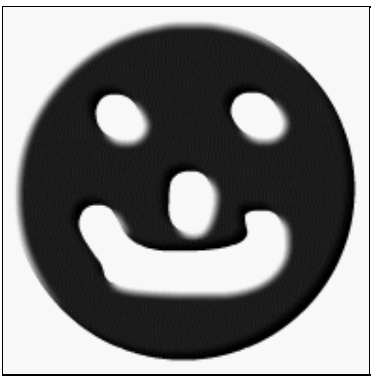


Рис. 9.11. Результат применения к клипу фильтра **Bevel**

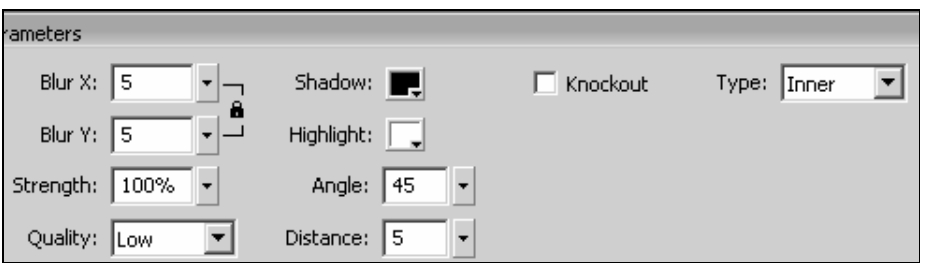


Рис. 9.12. Элементы управления панели **Properties** для настройки параметров фильтра **Bevel**

Поля ввода с регуляторами **Blur X** и **Blur Y** позволяют задать размеры области "вдавливания" в пикселах. Если включить расположенную правее этих полей ввода кнопку-выключатель со знаком замка, Flash при изменении значения в одном из полей будет менять значение в другом поле, чтобы соблюсти пропорции области "вдавливания".

Поле ввода с регулятором **Strength** позволяет задать степень "вдавленности" в процентах.

Раскрывающийся список **Quality** задает качество отрисовки эффекта вдавливания. В нем доступны три пункта: **Low** (Низкое качество), **Medium** (Среднее) и **High** (Высокое). По умолчанию выбран пункт **Low**.

Селекторы цвета **Shadow** и **Highlight** служит для задания цветов затененной и освещенной сторон "вдавленного" фрагмента изображения соответственно.

Поле ввода с регулятором **Angle** задает угол "вдавленности" в градусах, а поле ввода с регулятором **Distance** — ее "глубину". Причем "глубина" эта может принимать как положительные, так и отрицательные значения.

Флажок **Knockout** при включении "заливает" фрагмент изображения, к которому применяется фильтр, белым цветом.

Раскрывающийся список **Type** позволяет задать вид эффекта вдавливания. В нем доступны три пункта:

- **Inner** — эффект создается на самом фрагменте изображения (выбран по умолчанию);
- **Outer** — эффект создается вокруг фрагмента изображения, оставляя сам этот фрагмент нетронутым;
- **Full** — комбинация пунктов **Inner** и **Outer**.

Эффект градиентного сияния (*Gradient Glow*)

Фильтр, создающий эффект сияния выделенного фрагмента изображения и использующий для создания этого эффекта градиент, создается выбором пункта **Gradient Glow** меню кнопки **Add filter**. Результат его применения показан на рис. 9.13, а набор элементов управления для задания его параметров — на рис. 9.14.

Рис. 9.13. Результат применения к клипу фильтра **Gradient Glow**

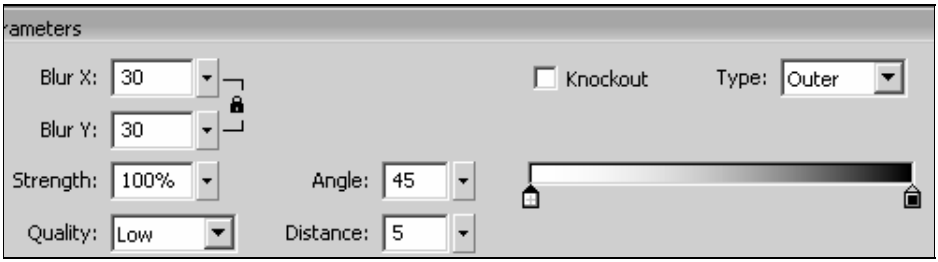


Рис. 9.14. Элементы управления панели **Properties** для настройки параметров фильтра **Gradient Glow**

Поля ввода с регуляторами **Blur X** и **Blur Y** позволяют задать размеры области "сияния" в пикселах. Если включить расположенную правее этих полей ввода кнопку-выключатель со знаком замка, Flash при изменении значения в одном из полей будет менять значение в другом поле, чтобы соблюсти пропорции области "сияния".

Поле ввода с регулятором **Strength** позволяет задать степень насыщенности "сияния" в процентах.

Раскрывающийся список **Quality** задает качество отрисовки эффекта сияния. В нем доступны три пункта: **Low** (Низкое качество), **Medium** (Среднее) и **High** (Высокое). По умолчанию выбран пункт **Low**.

Поле ввода с регулятором **Angle** задает угол направления эффекта в градусах, а поле ввода с регулятором **Distance** — ее величину отступа эффекта от фрагмента изображения, к которому применяется фильтр. Причем величина отступа может принимать как положительные, так и отрицательные значения.

Флажок **Knockout** при включении "заливает" фрагмент изображения, к которому применяется фильтр, белым цветом.

Раскрывающийся список **Type** позволяет задать вид эффекта градиентного тления. В нем доступны три пункта:

- **Inner** — эффект создается на самом фрагменте изображения;
- **Outer** — эффект создается вокруг фрагмента изображения, оставляя сам этот фрагмент нетронутым (выбран по умолчанию);
- **Full** — комбинация пунктов **Inner** и **Outer**.

Кроме того, в панели **Properties** имеется линейка для задания градиентного цвета, который используется при создании фильтра. На этой линейке находятся два указателя цвета, задающие ключевые цвета градиента.

Чтобы задать ключевой цвет для какого-либо указателя, достаточно щелкнуть по нему. После этого на экране появится окно селектора цвета, в котором мы сможем выбрать нужный цвет.

Чтобы добавить новый указатель цвета (и тем самым — новый ключевой цвет в градиент), нужно щелкнуть чуть ниже линейки, на уровне остальных указателей. Удалить ненужный указатель цвета можно, "сдернув" его мышью вниз.

Эффект градиентного вдавливания (*Gradient Bevel*)

Фильтр, создающий эффект вдавливания выделенного фрагмента изображения и использующий для создания этого эффекта градиентный цвет, создается выбором пункта **Gradient Bevel** меню кнопки **Add filter**. Результат его применения показан на рис. 9.15, а набор элементов управления для задания его параметров — на рис. 9.16.

Рис. 9.15. Результат применения к клипу фильтра **Gradient Bevel**

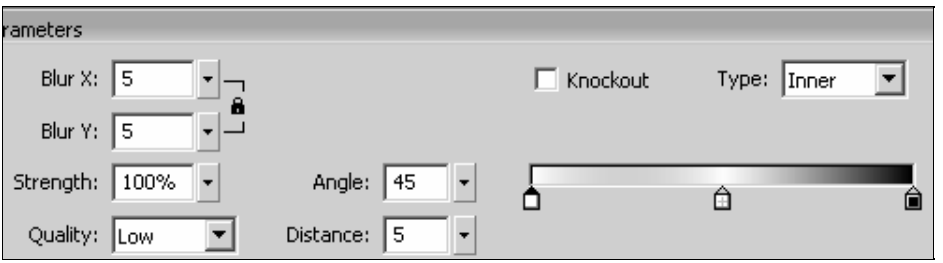


Рис. 9.16. Элементы управления панели **Properties** для настройки параметров фильтра **Gradient Bevel**

Поля ввода с регуляторами **Blur X** и **Blur Y** позволяют задать размеры области "вдавливания" в пикселах. Если включить расположенную правее этих

полей ввода кнопку-выключатель со знаком замка, Flash при изменении значения в одном из полей будет менять значение в другом поле, чтобы соблюсти пропорции области "вдавливания".

Поле ввода с регулятором **Strength** позволяет задать степень "вдавленности" в процентах.

Раскрывающийся список **Quality** задает качество отрисовки эффекта вдавливания. В нем доступны три пункта: **Low** (Низкое качество), **Medium** (Среднее) и **High** (Высокое). По умолчанию выбран пункт **Low**.

Поле ввода с регулятором **Angle** задает угол "вдавленности" в градусах, а поле ввода с регулятором **Distance** — ее "глубину". Причем "глубина" эта может принимать как положительные, так и отрицательные значения.

Флажок **Knockout** при включении "заливает" фрагмент изображения, к которому применяется фильтр, белым цветом.

Раскрывающийся список **Type** позволяет задать вид эффекта градиентного вдавливания. В нем доступны три пункта:

- **Inner** — эффект создается на самом фрагменте изображения (выбран по умолчанию);
- **Outer** — эффект создается вокруг фрагмента изображения, оставляя сам этот фрагмент нетронутым;
- **Full** — комбинация пунктов **Inner** и **Outer**.

Что касается самого градиентного цвета, использующегося при создании фильтра, то он задается с помощью уже знакомой нам линейки.

Настройка цвета (*Adjust Color*)

Фильтр, позволяющий настроить цвет выделенного фрагмента изображения, создается выбором пункта **Adjust Color** меню кнопки **Add filter**. Набор элементов управления для задания параметров этого фильтра показан на рис. 9.17.

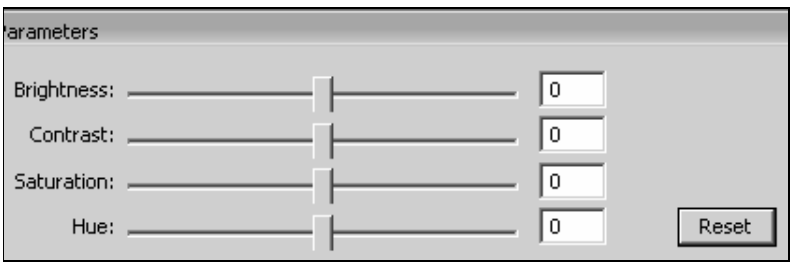


Рис. 9.17. Элементы управления панели **Properties** для настройки параметров фильтра **Adjust Color**

Как видно на рис. 9.17, панель **Properties** содержит набор из четырех регуляторов, перечисленных ниже:

- **Brightness** задает яркость выделенного фрагмента изображения;
- **Contrast** — контрастность;
- **Saturation** — насыщенность;
- **Hue** — оттенок.

Мы можем задавать как положительные, так и отрицательные значения всех этих параметров. Кроме того, мы можем ввести нужное значение любого из параметров вручную. Для этого служат поля ввода, находящиеся правее перечисленных выше регуляторов.

Кнопка **Reset** позволит нам быстро сбросить заданные параметры фильтра в значения по умолчанию (т. е. в нулевые значения).

Управление фильтрами

Так, создавать фильтры и задавать их параметры мы уже научились. Давайте выясним, что же мы можем еще делать с ними.

Выше говорилось, что мы можем применять к выделенному фрагменту изображения сразу несколько фильтров. При этом они будут применяться к этому фрагменту в том порядке, в котором перечислены в списке фильтров панели **Properties** (см. рис. 9.1). Так вот, мы можем перемещать фильтры в списке мышью, меняя их порядок.

Удалить ненужный фильтр также очень просто. Выделяем в списке фильтров соответствующий ему пункт и нажимаем кнопку **Remove filter**, находящуюся над списком и помеченную значком "минус".

Также можно удалить сразу все фильтры, созданные для выделенного фрагмента изображения. Для этого достаточно выбрать пункт **Remove All** меню кнопки **Add filter** (см. рис. 9.2).

Если нужно временно убрать действие какого-либо фильтра, мы можем щелкнуть мышью по значку зеленой галочки, находящейся левее названия соответствующего пункта списка фильтров. После щелчка значок галочки сменится красным символом X, и фильтр будет отключен. Чтобы снова включить фильтр, нужно щелкнуть мышью по символу X (после этого там снова появится зеленая галочка, говорящая, что фильтр включен).

Пункт **Disable All** меню кнопки **Add filter** позволяет отключить сразу все фильтры. А пункт **Enable All** того же меню, наоборот, все их включает.

Если мы создали, с нашей точки зрения, удачную комбинацию фильтров, то можем сохранить ее в виде *набора фильтров*. Для создания набора фильтров

достаточно выбрать пункт **Save As** подменю **Presets** меню кнопки **Add filter**. На экране появится диалоговое окно **Save Preset As** (рис. 9.18). Вводим в единственное поле ввода **Preset name** этого окна имя создаваемого набора фильтров и нажимаем кнопку **OK**. Если мы передумали создавать набор фильтров, нажмем кнопку **Cancel**.

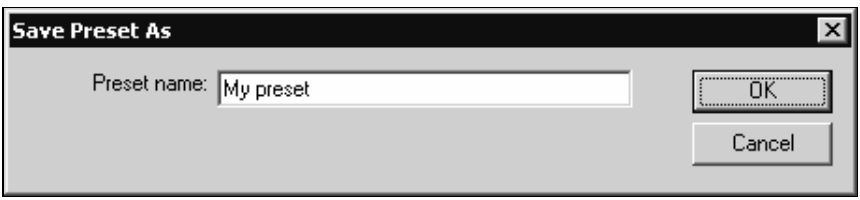


Рис. 9.18. Диалоговое окно **Save Preset As**

Все созданные таким образом наборы фильтров будут присутствовать в виде пунктов, находящихся в нижней части подменю **Presets** меню кнопки **Add filter**. И чтобы применить нужный набор фильтров к выделенному фрагменту изображения, нам будет достаточно выбрать соответствующий пункт этого подменю.

Рис. 9.19. Диалоговое окно **Delete Preset**

Чтобы удалить ненужный набор фильтров, достаточно выбрать пункт **Delete** подменю **Presets** меню кнопки **Add filter**. На экране появится диалоговое окно **Delete Preset** (рис. 9.19). Выбираем в большом списке набор фильтров, который хотим удалить, и нажимаем кнопку **Delete**. Чтобы закрыть это окно, не удаляя ни одного набора фильтров, достаточно нажать кнопку **Cancel**.

НА ЗАМЕТКУ

По идее, Flash 8 также позволяет переименовать выбранный набор фильтров. В подменю **Presets** меню кнопки **Add filter** доступен пункт **Rename**, после выбора которого на экране появляется диалоговое окно **Rename Preset**, похожее на окно **Delete Preset**. Но кнопка **Rename** этого окна, с помощью которой выполняется переименование набора фильтров, недоступна. Вероятно, это ошибка в программе.

Наложения

Наложения Flash 8 — это режимы смешивания цвета выделенного нами фрагмента изображения и цветов всех фрагментов, которые находятся под ним. При этом не имеет значения, лежат ли все эти фрагменты на одном слое или разнесены по разным слоям.

Наложения могут быть применены только к клипам (экземплярам образцов-клипов). Поэтому если нам нужно применить наложение к фрагменту изображения, не являющемуся клипом, придется сначала преобразовать его в образец-клип и применить наложение к его экземпляру.

Наложение задается с помощью панели **Properties**, а именно, пунктов раскрывающегося списка **Blend**, расположенного на вкладке **Properties** этой панели. Этот список показан на рис. 9.20.

Давайте перечислим все пункты этого списка и познакомимся с предлагаемыми ими режимами наложения.

- **Normal** — отключение всех наложений, т. е. фрагменты графики ведут себя, как обычно.
- **Layer** — то же самое, что при выборе пункта **Normal**. По крайней мере, автору книги не удалось обнаружить никаких отличий. Скорее всего, это ошибка в программе.
- **Darken** — более светлые пикселы перекрываемых фрагментов, имеющие цвет клипа, заменяются на пикселы клипа, остальные остаются неизменными.

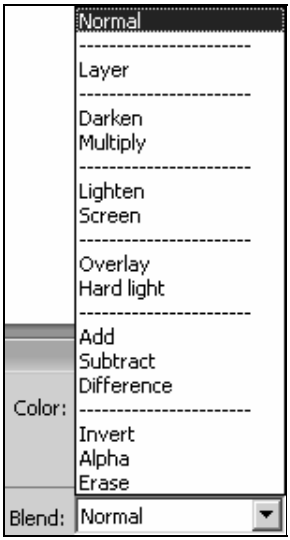


Рис. 9.20. Раскрывающийся список **Blend** панели **Properties**

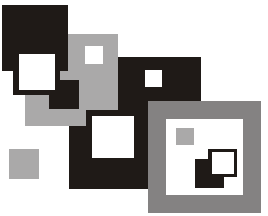
- **Lighten** — более темные пикселы перекрываемых фрагментов, имеющие цвет клипа, заменяются на пикселы клипа, остальные остаются неизменными.
- **Screen** — инвертированные пикселы клипа умножаются на пикселы перекрываемых фрагментов.
- **Overlay** — то же самое, что при выборе пункта **Screen**, только с учетом цвета пикселов перекрываемых фрагментов.
- **Hard light** — то же самое, что при выборе пункта **Screen**, только с учетом цвета пикселов клипа.
- **Difference** — цвет пикселов клипа вычитается из цвета пикселов перекрываемых фрагментов или наоборот, в зависимости от того, какие пикселы ярче.
- **Invert** — цвета пикселов клипа и перекрываемых фрагментов инвертируются.
- **Alpha** и **Erase** — клип просто исчезает. Насколько удалось выяснить автору этой книги, другой эффект выбор этих пунктов не дает. Вероятно, это также ошибка в программе.

Скорее всего, для создания нужного эффекта придется поэкспериментировать с различными наложениями. Как говорится, лучше один раз увидеть...

Что дальше?

Вот мы и познакомились с новыми возможностями, предлагаемыми Flash 8, — фильтрами и наложениями. Удобные, кстати, штуки — автору они пригодились в Macromedia Fireworks, значит, пригодятся и во Flash.

Flash 8, как мы помним из *главы 1*, — пакет гибридной графики. Это значит, что изображение Flash может содержать, наряду с векторными, и растровые фрагменты. О том, как импортировать такие фрагменты в документ Flash и что можно с ними делать потом, и расскажет следующая глава.



Глава 10

Импорт графики

Ну, до чего же хорош Flash 8! И рисовать в нем можно, и анимировать, и даже писать программы (эта тема, правда, выходит за рамки данной книги). И при этом размер получающихся файлов Shockwave/Flash настолько мал, что они могут распространяться через Интернет. Неудивительно, что именно он получил на просторах Всемирной сети такую популярность. Ура, да здравствует Flash 8, великий и могучий!

Но Flash не универсален. Например, он не содержит абсолютно никаких инструментов для работы с растровой графикой. То есть векторное изображение практически любой сложности мы в нем нарисуем, но создать изображение растровое — увы!.. Для работы с растровой графикой служат другие программы: Adobe Photoshop, Macromedia Fireworks, Microsoft Paint и др. Да и в области векторной графики Flash не так уж силен — он не позволяет делать многое, доступное более мощным пакетам, таким как CorelDRAW и Macromedia FreeHand.

Поэтому профессиональные художники для создания особо сложной графики, включающей как векторные, так и растровые фрагменты, используют несколько различных программ. Обычный комплект профессионала таков: Adobe Photoshop, CorelDRAW или Macromedia FreeHand, возможно, Macromedia Fireworks. Именно в этих пакетах зачастую подготавливаются изображения, которые потом станут составными частями документа Flash.

Подготовленные изображения помещаются в документ Flash с помощью особой операции *импорта* (импортируются, иначе говоря). И после этого с ними можно творить, что душе угодно: помещать на рабочий лист, трансформировать, раскрашивать в разные цвета и даже анимировать.

В этой главе речь как раз и пойдет о том, как выполнить импорт подготовленных в других пакетах изображений в документ Flash. (Flash также поддерживает импорт видео и звука, но об этом мы поговорим соответственно в *главах 16* и *17*.) Итак, запасемся изображениями в разных форматах и начнем...

Какие форматы графики поддерживает Flash

Сначала выясним, какие форматы графики поддерживает Flash 8. Их довольно много, и все они перечислены в табл. 10.1.

Таблица 10.1. *Форматы графики, которые можно импортировать во Flash*

Название формата	Расширение файлов
Adobe Acrobat	pdf
Adobe Illustrator	ai, eps
AutoDesk AutoCAD	dxf
FutureSplash	spl
GIF (обычный и анимированный)	gif
JPEG	jpg, jpe, jpeg
Macromedia FreeHand	fh7, ft7, fh8, ft8, fh9, ft9, fh10, fh11
Macromedia Shockwave/Flash	swf
PNG	png
Метафайлы Windows	wmf
Растровые файлы Windows	bmp
Расширенные метафайлы Windows	emf

А если на нашем компьютере установлен пакет Apple QuickTime версии не ниже 4, будут доступны для импорта еще несколько графических форматов, перечисленные в табл. 10.2.

Таблица 10.2. *Форматы графики, доступные для импорта во Flash при установленном пакете Apple QuickTime*

Название формата	Расширение файлов
Adobe PhotoShop	psd
Apple MacPaint	pntg
Apple QuickTime (статичные изображения)	qtif
TIFF	tif, tiff

Таблица 10.2 *(окончание)*

Название формата	Расширение файлов
Растровые файлы Macintosh	pct, pic
Растровые файлы Silicon Graphics	sgi
Растровые файлы Targa	tga

Да, список поддерживаемых Flash форматов статичной графики достаточно велик, но неполон. Так что если нам понадобится импортировать во Flash файл неподдерживаемого формата, придется найти программу, поддерживающую этот формат, и с ее помощью преобразовать данный файл в один из поддерживаемых форматов.

Импорт графики

Существуют два способа импортировать во Flash графику, созданную в другом графическом пакете. Рассмотрим их.

Первый способ очень прост. Выбираем пункт **Import to Stage** подменю **Import** меню **File** или нажимаем комбинацию клавиш <Ctrl>+<R>. На экране появится стандартное диалоговое окно открытия файла Windows. Находим нужный файл и нажимаем кнопку открытия файла этого диалогового окна. После этого выбранный нами графический файл будет импортирован и сразу же помещен на рабочий лист.

Второй способ еще проще. Запускаем нужный графический пакет, открываем в нем файл, содержащий нужное изображение, или создаем его заново, копируем это изображение в буфер обмена Windows и вставляем во Flash, выбрав один из трех пунктов: **Paste**, **Paste Special** или **Paste in Center** меню **Edit**. Первые два пункта нам уже знакомы; третий позволяет вставить находящееся в буфере обмена изображение прямо в центр рабочего листа.

Оба этих способа позволяют поместить импортированное изображение и в библиотеку в виде импортированного образца, и на рабочий лист. Если же мы хотим поместить импортируемое изображение только в библиотеку, то должны будем выбрать пункт **Import to Library** подменю **Import** меню **File**. На экране после этого появится стандартное диалоговое окно открытия файла Windows; находим в нем нужный файл и нажимаем кнопку открытия.

Особенности импорта разных графических форматов

Импорт файлов, сохраненных в некоторых графических форматах, имеет определенные особенности. Давайте поговорим о них подробнее.

Macromedia Fireworks

Пакет создания Web-графики Macromedia Fireworks — в чем-то "коллега" Flash. Он тоже создает интернет-графику, но только традиционную, растровую.

Fireworks хранит свои рабочие документы в формате PNG. Как и Flash, Fireworks может объединять в одном документе и растровую, и векторную графику (гибридная графика). Это становится возможным из-за того, что формат PNG, помимо собственно растрового изображения, может хранить дополнительные данные (*меги PNG*), в которых и записывается вся векторная графика. Экспорт готового изображения выполняется в форматах GIF, JPEG и PNG; разумеется, при этом вся векторная графика растрируется.

Поскольку и Flash, и Fireworks разработаны одной фирмой, было бы удивительно, если бы они не поддерживали форматов друг друга.

При импорте документа Fireworks на экране появится диалоговое окно **Fireworks PNG Import Settings** (рис. 10.1).

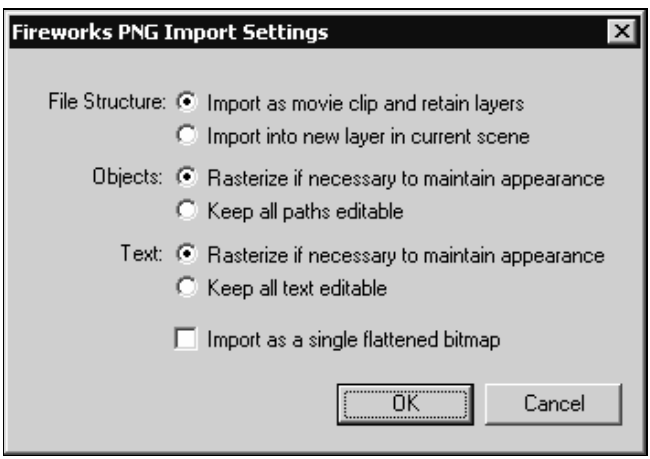


Рис. 10.1. Диалоговое окно **Fireworks PNG Import Settings**

Если мы сразу же включим флажок **Import as a single flattened bitmap**, Flash растрирует импортируемое изображение, переведя в растровый вид все его

векторные фрагменты. Если этот флажок включен, все остальные элементы управления окна **Fireworks PNG Import Settings** станут недоступными.

Если же флажок **Import as a single flattened bitmap** не включать, изображение Fireworks будет импортировано без изменений. С помощью остальных элементов управления окна **Fireworks PNG Import Settings** мы можем задать дополнительные параметры импорта.

С помощью переключателей группы **File Structure** задается вид, в котором будет импортирована графика Fireworks. Переключатель **Import as movie clip and retain layers**, включенный по умолчанию, позволяет импортировать изображение Fireworks как отдельный образец-клип с сохранением всех его слоев. Переключатель **Import into new layer in current scene** заставляет Flash поместить изображение Fireworks прямо на рабочий лист во вновь созданный слой.

Переключатели группы **Objects** задают, в каком виде будут импортированы векторные примитивы Fireworks, не поддерживаемые Flash. Если включен переключатель **Rasterize if necessary to maintain appearance**, такие примитивы будут превращены в растровые изображения и благодаря этому сохранят свой вид в точности. Если же включен переключатель **Keep all paths editable**, Flash преобразует их в максимально похожие "свои родные" примитивы, возможно с некоторым искажением.

Переключатели группы **Text** задают те же самые параметры, что и переключатели группы **Objects**, но только для текста. Так, если включен переключатель **Rasterize if necessary to maintain appearance**, текст, содержащий преобразования, не поддерживаемые Flash, будет растрирован. Если же включен переключатель **Keep all text editable**, никакой растеризации выполняться не будет, в результате чего некоторые текстовые блоки исходного изображения могут исказиться.

Задав нужные параметры, следует нажать кнопку **OK**. Чтобы отказаться от импорта файла Fireworks, нужно нажать кнопку **Cancel**.

Macromedia Freehand

Векторный графический редактор Macromedia Freehand версий 7—11 — одна из лучших программ создания сложной векторной графики. Файлы Freehand можно импортировать во Flash без особого труда и практически без потерь качества.

После импорта изображения Freehand на экране появится диалоговое окно **FreeHand Import** (рис. 10.2).

Группа элементов управления **Mapping** позволит нам задать действия, предпринимаемые Flash в отношении страниц и слоев изображения Freehand. Она в свою очередь содержит две группы переключателей, рассматриваемые ниже.

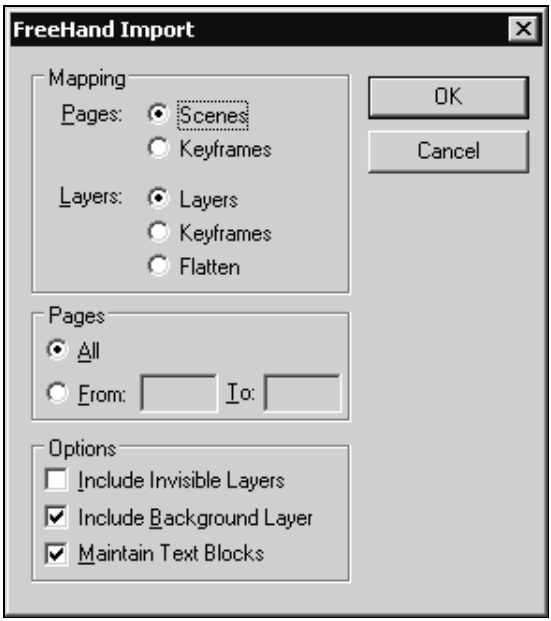


Рис. 10.2. Диалоговое окно **Freehand Import**

Группа переключателей **Pages** управляет импортом страниц. Переключатель **Scenes** заставляет Flash преобразовывать каждую страницу в сцену (Flash не поддерживает страниц), а переключатель **Keyframes** — в ключевой кадр. (О сценах и ключевых кадрах будет рассказано в *главе 11.*)

Группа переключателей **Layers** управляет импортом слоев. Переключатель **Layers** заставляет Flash преобразовать каждый слой FreeHand в собственный слой, переключатель **Keyframes** — в ключевой кадр, а переключатель **Flatten** указывает, что нужно поместить содержимое всех слоев FreeHand в один слой Flash.

Группа элементов управления **Pages** задает, какие страницы документа FreeHand должны быть импортированы. Если включить переключатель **All**, будут импортированы все страницы. Если выбрать второй переключатель, будут импортированы только страницы с номерами, указанными в полях ввода **From** (первая импортируемая страница) и **To** (последняя).

Флажки, расположенные в группе элементов управления **Options**, задают дополнительные параметры импорта графики FreeHand. Включение флажка **Include Invisible Layers** предписывает Flash импортировать все слои изображения FreeHand; если же этот флажок отключен, импортируются только видимые слои. Включение флажка **Include Background Layer** вызывает импорт также и фонового слоя FreeHand. А флажок **Maintain Text Blocks**, будучи включен, заставляет Flash преобразовать фрагменты текста из файла FreeHand в текстовые блоки.

Задав нужные параметры, нажмем кнопку **OK**. Если мы передумали импортировать файл FreeHand, нажмем кнопку **Cancel**.

А теперь рассмотрим некоторые особенности, присущие импорту графики FreeHand, сопроводив рекомендациями, которые могут оказаться полезными.

- Если графическое изображение, созданное во FreeHand, имеет градиентную заливку с более чем восемью цветами, Flash для ее имитации создаст несколько накладывающихся друг на друга заливок. Нужно учесть, что это может увеличить размер результирующего файла Shockwave/Flash, поэтому по возможности не следует использовать в исходном изображении FreeHand градиенты с более чем восемью ключевыми цветами.
- Как мы уже знаем, Flash выполняет слияние и фрагментацию накладывающихся друг на друга примитивов. FreeHand же этого не делает. Поэтому импортированная во Flash графика FreeHand может быть искажена. Чтобы предотвратить слияние и фрагментацию фрагментов импортированного изображения, при рисовании его во FreeHand нужно будет разместить перекрывающиеся фрагменты в разных слоях.
- При импорте изображений FreeHand углы линий будут скруглены.
- При импорте черно-белого изображения FreeHand оно будет преобразовано в цветной формат, так как Flash не поддерживает черно-белую графику. Это может увеличить размер результирующего файла Shockwave/Flash.
- Если при задании цветов изображения FreeHand был использован цветовой режим *CMYK* (Cyan, Magenta, Yellow, blacK — голубой, пурпурный, желтый, черный), то такие цвета будут автоматически преобразованы в RGB.
- Если изображение FreeHand содержит внедренные фрагменты в формате EPS, то перед импортом нам нужно будет установить флажок **Convert editable EPS when imported** на вкладке **Import** диалогового окна настройки FreeHand **Preferences**. Иначе Flash не сможет правильно отобразить такие изображения.

Adobe Illustrator и Adobe Acrobat

Flash позволяет импортировать файлы, созданные в Adobe Illustrator версии 6.0 или более ранних версий. Также мы можем импортировать электронные документы Adobe Acrobat, сохраненные в формате *PDF* (Portable Document Format, формат переносимых документов) версии 1.4 или более ранней.

Перед импортом графики Illustrator следует разгруппировать все сгруппированные фрагменты изображения на всех его слоях. Если этого не сделать, Flash не сможет правильно их обработать.

После импорта документа Illustrator или Acrobat на экране появится диалоговое окно **Import Options** (рис. 10.3).

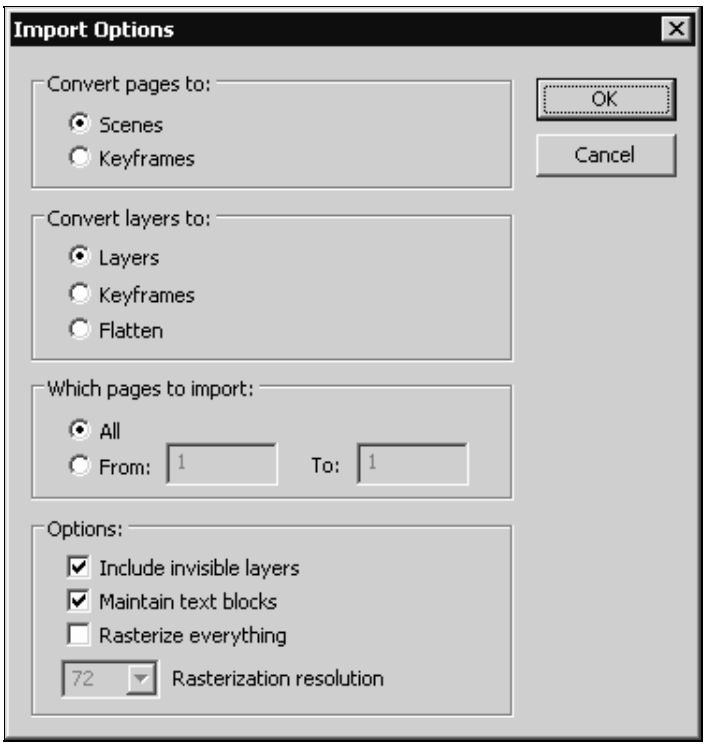


Рис. 10.3. Диалоговое окно **Import Options**

Если мы включим флажок **Rasterize everything** в группе **Options** и выберем разрешение результирующего изображения в поле со списком **Rasterization resolution**, Flash преобразует изображение Illustrator или Acrobat в растровое.

Чтобы импортировать изображение без изменений, нужно отключить этот флажок.

Группа переключателей **Convert pages to** управляет импортом страниц. Переключатель **Scenes** заставляет Flash преобразовать каждую страницу в сцену (Flash не поддерживает страницы), а переключатель **Keyframes** — в ключевой кадр. (О сценах и ключевых кадрах будет рассказано в *главе 11*.)

Группа переключателей **Convert layers to**, доступная при отключенном флажке **Rasterize everything**, управляет импортом слоев. Переключатель **Layers** заставляет Flash преобразовывать каждый слой исходного изображения в собственный слой, переключатель **Keyframes** — в ключевой кадр, а переключатель **Flatten** помещает содержимое всех слоев исходного изображения в один слой Flash.

С помощью группы элементов управления **Which pages to import** можно указать, какие страницы документа Illustrator или Acrobat должны быть импортированы. Если включить переключатель **All**, будут импортированы все страницы. Если включить второй переключатель этой группы, будут импортированы только страницы с номерами, указанными в полях ввода **From** (первая импортируемая страница) и **To** (последняя).

Установкой флажка **Include invisible layers** группы элементов **Options** включается импорт всех слоев изображения; если флажок снят, импортируются только видимые слои.

Флажок **Maintain text blocks** группы **Options**, будучи включенным, заставляет Flash преобразовать фрагменты текста в импортируемом изображении в текстовые блоки. Этот флажок доступен при отключенном флажке **Rasterize everything**.

Задав нужные параметры, нажмем кнопку **OK**. Если мы передумали импортировать документ Illustrator или Acrobat, нажмем кнопку **Cancel**.

AutoDesk AutoCAD

Flash позволяет импортировать чертежи и схемы AutoCAD версии 10.

При импорте графики в этом формате возможны некоторые проблемы. Давайте поговорим о них.

- AutoCAD не поддерживает стандартные системные шрифты. При импорте Flash будет выполнять подстановку шрифтов, отчего возможно искажение текста.
- AutoCAD не поддерживает создание заливок.
- Flash не позволяет импортировать трехмерные чертежи и схемы AutoCAD — только двухмерные.

- Все импортированные изображения AutoCAD, будучи помещены на рабочий стол, имеют размер 12×12 дюймов, независимо от заданного для них масштаба. Мы можем воспользоваться инструментом "трансформатор", чтобы увеличить или уменьшить их.
- Flash позволяет импортировать только файлы AutoCAD, сохраненные в текстовом формате. Если они сохранены в двоичном формате, следует открыть их в AutoCAD и пересохранить в текстовом формате.

Работа с импортированной растровой графикой

Хорошо, мы импортировали изображение в документ Flash. И что теперь с ним делать?

Да практически все, что мы уже делали с другими фрагментами графики, в том числе образцами и экземплярами: перемещать, трансформировать, задавать цветовые эффекты, переименовывать и удалять из библиотеки. Есть, правда, несколько исключений, но они не очень принципиальны.

Особенности работы с импортированными образцами

Из *главы 8* мы знаем, что импортированное растровое изображение преобразуется в образец-растровое изображение и помещается в библиотеку. Это значит, что мы можем поместить на рабочий лист сколько угодно экземпляров этого образца без опасения сильно увеличить размер результирующего файла Shockwave/Flash.

Кстати, при импорте растрового изображения Flash сохраняет в параметрах созданного образца-растрового изображения полное имя изначального файла, в котором это изображение хранится. Это нужно для того, чтобы предоставить нам возможность исправить этот файл прямо из среды Flash.

Что касается импортированного векторного изображения, то оно будет преобразовано в графический образец. Скорее всего, при этом в библиотеке также появится папка, содержащая множество других образцов-клипов, — там хранятся различные части этого изображения. Мы можем увидеть это, так сказать, воочию — достаточно импортировать в документ Flash любое достаточно сложное изображение FreeHand.

Бывает, что импортируемое изображение содержит и растровую часть, и векторные фрагменты (как, например, изображение Fireworks). В этом слу-

чае Flash поместит растровую часть в образец-растровое изображение, а векторные фрагменты — в графические образцы. Последние при этом будут помещены в папку.

Как работать с графическими образцами, мы уже знаем. Принципы работы с импортированными образцами примерно такие же, за тем исключением, что мы не сможем исправить их в среде Flash. Правка таких образцов выполняется другим способом, который мы сейчас рассмотрим.

Итак, чтобы исправить импортированный образец, выделим его в списке панели **Library** и выберем в контекстном или дополнительном меню пункт **Edit with**. После этого на экране появится стандартное диалоговое окно открытия файла Windows. Выберем в этом окне исполняемый файл нужной программы и нажмем кнопку открытия. Изначальный графический файл, в котором хранится выбранный нами образец, будет открыт в этой программе. Исправим его, сохраним и закроем программу, а Flash сам обновит в библиотеке соответствующий образец.

Если у нас на компьютере установлена программа Macromedia Fireworks, мы можем исправить растровый файл прямо в ней. Для этого нужно выбрать пункт **Edit with Fireworks** в контекстном или дополнительном меню панели **Library**.

В конце концов, мы можем исправить сам исходный файл изображения, использовав любую графическую программу, поддерживающую его формат. Недостатком такого подхода является то, что после правки необходимо будет обновить соответствующий образец в библиотеке вручную. Для этого нужно выделить его в списке панели **Library** и в контекстном или дополнительном меню выбрать пункт **Update**. На экране появится уже знакомое нам диалоговое окно **Update Library Items** (см. рис. 8.20). В этом окне включим флажки против нужных файлов и нажмем кнопку **Update** для запуска процесса обновления. После обновления образцов можно нажать кнопку **Close**, чтобы закрыть это окно.

Смена образца для выделенного экземпляра импортированного образца выполняется нажатием кнопки **Swap** панели **Properties**, выбором пункта **Swap Bitmap** подменю **Bitmap** меню **Modify** или пункта **Swap Bitmap** контекстного меню выделенного экземпляра. На экране появится диалоговое окно **Swap Bitmap**, похожее на уже знакомое нам окно **Swap Symbol** (см. рис. 8.9). Принципы работы с этими окнами аналогичны.

Импортированные растровые изображения мы можем использовать в качестве изображений-заливок для создания графических цветов. О графических цветах было подробно рассказано в *главе 6*.

Векторизация растровой графики

Иногда бывает нужно преобразовать импортированное растровое изображение в векторное (векторизовать его). Это может пригодиться, если мы захотим исправить импортированное растровое изображение прямо в среде Flash, не прибегая к другим программам. После векторизации растровое изображение будет разбито на отдельные графические примитивы, с которыми мы можем сделать все, что угодно.

Flash позволяет векторизовать только экземпляр образца-растрового изображения, но не сам этот образец. Выделим на рабочем листе нужный нам экземпляр и выберем пункт **Trace Bitmap** подменю **Bitmap** меню **Modify**. На экране появится диалоговое окно **Trace Bitmap** (рис. 10.4).

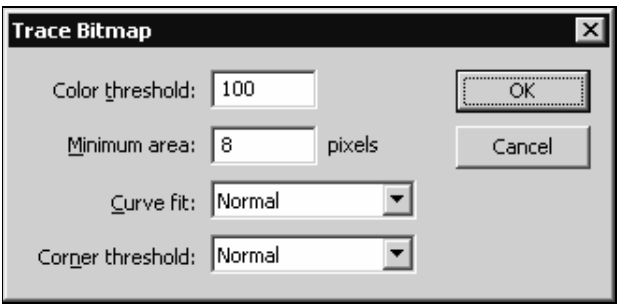


Рис. 10.4. Диалоговое окно **Trace Bitmap**

В поле ввода **Color threshold** задается числовое значение "разницы" между цветами. Если два соседних пиксела имеют цвета, различающиеся по значениям на число, меньшее, чем введенная "разница", они считаются одноцветными. Таким образом, чем больше значение "разницы" цветов, тем меньше будет цветов в результирующем векторном изображении. Допускается вводить значения от 1 до 500, значение по умолчанию — 100.

В поле ввода **Minimum area** указывается количество окружающих пикселов, которые будут взяты для определения цвета результирующего пиксела. Иными словами, это значение задает разрешающую способность системы векторизации Flash, и чем оно больше, тем меньше нюансов будет иметь результирующее изображение. Допускается вводить значения от 1 до 1000 пикселов, значение по умолчанию — 8 пикселов.

Раскрывающийся список **Curve fit** служит для задания точности передачи контуров результирующего векторного изображения. Доступны шесть пунктов, задающих точность от максимальной до минимальной: **Pixel** (Точность до пиксела), **Very Tight** (Очень точно), **Tight** (Точно), **Normal** (этот пункт

выбран по умолчанию), **Smooth** (Приближенно) и **Very Smooth** (очень приближенно, сильно сглаженные контуры).

Раскрывающийся список **Corner threshold** задает степень сглаживания углов контура, который получится после векторизации. Здесь доступны пункты **Many corners** (контуры результирующего изображения будут содержать много углов), **Normal** (значение по умолчанию) и **Few Corners** (немного углов, сильно сглаженные контуры).

Задав нужные параметры, нажмем кнопку **OK**. Если мы передумали векторизовать растровое изображение, нажмем кнопку **Cancel**.

Результат векторизации импортированного растрового изображения показан на рис. 10.5. Если внимательно присмотреться к нему, то можно заметить потерю качества. Так что лучше все-таки не векторизовать растровую графику без особой необходимости.

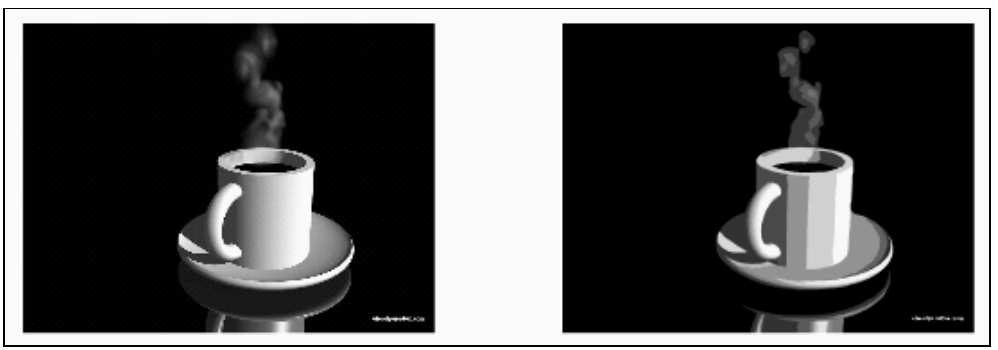


Рис. 10.5. Результат векторизации растрового изображения (слева — исходное изображение, справа — векторизованное)

Разбиение растровой графики. Модификатор *Magic Wand* (*"Волшебная палочка"*)

Если внимательно присмотреться к растровому изображению, находящемуся на рабочем листе, можно заметить, что оно состоит из множества частей, пикселы которых имеют одинаковый цвет. Так же думает и Flash. И, если его попросить, он может разделить это изображение на части — выполнить разбиение растрового изображения. После этого мы сможем очень просто перекрасить любую его часть в другой цвет, переместить его в другое место или вообще удалить с рабочего листа.

Чтобы выполнить разбиение растрового изображения (опять же, это должен быть экземпляр образца-растрового изображения, но не сам образец), сначала выделим его. После этого выберем пункт **Break Apart** меню **Modify** или контекстного меню растрового изображения или просто нажмем комбинацию клавиш <Ctrl>+<B>.

На первый взгляд, растровое изображение не изменится, но это только на первый взгляд. Давайте попробуем закрасить какую-либо его часть другим цветом. А для этого сначала нужно его выделить. И хорошо знакомая нам "стрелка" в этом случае не помощник.

Для выделения части растрового изображения нам понадобится инструмент "лассо". Выберем его и нажмем кнопку-выключатель **Magic Wand** ( ), включающую модификатор "волшебная палочка". После этого курсор мыши примет вид стилизованной волшебной палочки.

Для выделения части разбитого изображения достаточно просто щелкнуть на ней. Выделенная таким образом часть изображения показана на рис. 10.6. Если после этого щелкнуть на другой части изображения, она будет добавлена к выделению. А если щелкнуть на уже выделенной части, с нее будет снято выделение.

Рис. 10.6. Выделенная часть разбитого растрового изображения

Теперь мы можем воспользоваться инструментом "ведро с краской" или селекторами цвета в панели **Properties**, чтобы закрасить выделенные части изображения. Также можно выбрать инструмент "пипетка", чтобы создать на основе разбитого на части изображения графический цвет.

Мы можем настроить параметры модификатора "волшебная палочка". Эти настройки выполняются в диалоговом окне **Magic Wand Settings** (рис. 10.7),

которое появится на экране после нажатия кнопки главного инструментария.

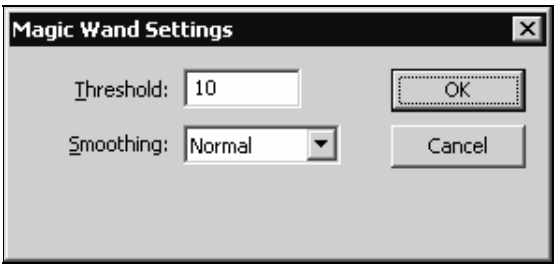


Рис. 10.7. Диалоговое окно **Magic Wand Settings**

В поле ввода **Threshold** задается числовое значение "разницы" между цветами соседних пикселов. Если два соседних пиксела имеют цвета, различающиеся на значение, меньшее, чем введенное в это поле число, они считаются одноцветными и попадут в одну выделенную с помощью "волшебной палочки" область. Таким образом, чем больше значение "разницы" цветов, тем больший диапазон цветов будет включать в себя выделенная часть изображения. Допускается вводить значения от 0 до 200, значение по умолчанию — 10.

Раскрывающийся список **Smoothing** служит для задания степени сглаживания контура, охватывающего выделенную область. Доступны четыре пункта, задающие точность от максимальной до минимальной: **Pixel** (Точность до пиксела), **Rough** (Грубые контуры), **Normal** (Значение по умолчанию) и **Smooth** (Плавные контуры).

Задав нужные параметры модификатора "волшебная палочка", нажмем кнопку **OK**, чтобы их сохранить. Нажатие кнопки **Cancel** вызовет отмену введенных параметров и возврат к старым значениям.

Немного об авторских правах

Все в этом мире кому-то принадлежит (ну, почти все...). Земля, вода, воздушное пространство, недра, заводы, дома, дороги, машины, приборы, вещи, мысли, открытия, изобретения и многое, многое другое является чьей-то собственностью. Да что там — даже эта книга является совместной собственностью автора и издательства "БХВ-Петербург". И, хотим мы или нет, этот закон приходится учитывать в повседневной жизни.

Он всесилен и в компьютерном мире. Тексты, изображения, программы, фильмы и музыка — все кому-то принадлежат. И этот "кто-то" требует соблюдения своих авторских прав, и зачастую весьма жестко. За нарушение его прав вполне реально получить на свою голову массу неприятностей.

Из этого следует один очень важный вывод. Прежде, чем позаимствовать что-либо чужое для своего блага (картинку, фильм, фрагмент текста или звуковой клип), нужно обязательно выяснить, как к этому отнесется владелец. Часто владелец относится к этому весьма демократично, но вполне вероятно, что нам придется хорошенько его попросить, пойти на некоторые уступки, а то и заплатить деньги за право использовать какое-то из его творений.

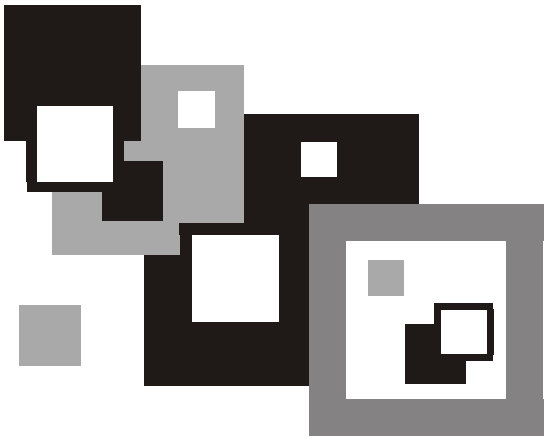
Обычно сведения об авторских правах на опубликованные на сайте тексты, изображения, программы или фильмы даются в нижней части главной страницы этого самого сайта или в специальном разделе. Если же эти тексты и изображения распространяются на компакт-дисках, авторские права описаны в особом файле, называющемся Copyright.txt. Если же текст или изображение является частью какой-либо программы, нам нужно будет обратиться к диалоговому окну сведений о программе или поискать в справке. Там описано, что разрешено делать с этим интеллектуальным богатством, а чего лучше не делать.

Всегда следует читать раздел авторских прав! Уж не стоит, наверно, говорить, что нарушение их чревато судебным преследованием — в России было немало таких прецедентов. В конце концов, воровать нехорошо — это вам скажет любой воспитанный человек. И будет прав.

Что дальше?

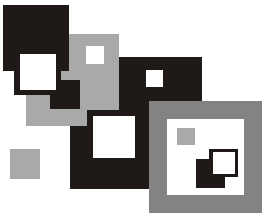
Вот мы и узнали, как импортировать во Flash-документ графику, созданную в другой программе. Теперь наши документы украсятся красивыми растровыми картинками... Нужно только следить, чтобы размер результирующего файла Shockwave/Flash сохранялся в разумных пределах.

Что ж, на этом разговор о создании статичной графики во Flash 8 закончен. Следующий шаг — освоение процесса создания анимации. Давайте добавим "жизни" нашим застывшим картинам, заставим их двигаться! Благо, во Flash это делается до того просто, что грех не воспользоваться этим удивительным пакетом.



ЧАСТЬ III

Анимация и звук



Глава 11

Покадровая анимация

Анимация — один из "коньков" Flash. Еще в самой первой версии этой программы фирма Macromedia сделала на нее ставку и не прогадала — анимация уже проникла в Интернет, и удивительные возможности Flash по созданию "мультиков" оказались востребованными. Можно сказать, что интернет-анимация — это Flash, а Flash — это интернет-анимация.

Что ж, и нам пора прикоснуться к таинству "оживления" неподвижных картинок. Благо, этих самых картинок у нас уже хватает. С чего начнем?

Давайте начнем с покадровой анимации. И вот почему.

Вспомним, что такое покадровая анимация (подробно о ней было рассказано в *главе 1*). Это последовательность отдельных кадров, изображающих различные фазы движения, которая очень быстро "прокручивается" перед зрителем, в результате чего у последнего возникает иллюзия непрерывного движения.

Создать покадровую анимацию очень просто — знай, рисуй себе кадр за кадром. Единственная оговорка: таких кадров может быть очень много, и работа над фильмом может сильно затянуться. Ну да ладно, на первых порах мы ограничимся простейшими фильмами. Главное в том, что при создании покадровой анимации мы будем пользоваться, в основном уже изученными в *части II* инструментами. Это позволит нам освоить работу с временно́й шкалой Flash и кадрами, не отвлекаясь на посторонние дела. Трансформационная же анимация, хоть и значительно менее трудоемка, но требует использования специальных инструментов; мы займемся ею в *главе 12*.

А еще мы рассмотрим одну очень интересную возможность Flash, связанную с так называемыми сценами. Пользуясь ими, можно разделить свой фильм на несколько отдельных частей. Но это будет в самом конце этой главы, когда мы уже достаточно хорошо натренируемся в рисовании и анимировании.

Итак, запасемся терпением — нам придется много рисовать. Освежим свои знания в области базовых инструментов рисования Flash. Создадим новый документ Flash. И начнем!

Создание покадровой анимации

Предположим, что нужно создать небольшой учебный фильм для средней школы. Этот фильм будет демонстрировать процесс деления клетки. Вы еще не забыли школьный курс биологии? Самое время его вспомнить.

Наш фильм будет содержать двенадцать кадров — на первый раз этого хватит. Зададим частоту кадров равной 4, для чего выберем в меню **Modify** пункт **Document**, введем в поле ввода **Frame rate** диалогового окна **Document Properties** (см. рис. 2.15) число 4 и нажмем кнопку **OK**. Четырех кадров в секунду школьникам вполне хватит, чтобы понять великое таинство жизни.

Осталось нарисовать первый кадр нашего фильма — единственную клетку, еще не помышляющую о делении. Она будет выглядеть примерно так, как показано на рис. 11.1. Выглядит она, конечно, не очень похоже на живую клетку, слишком упрощенно — всего два эллипса без заливок. Но для нашего первого фильма этого хватит, ведь он станет учебным не столько для гипотетических школьников, сколько для нас самих.

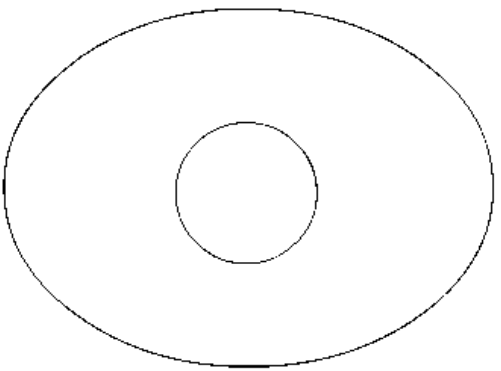


Рис. 11.1. Первый кадр нашего фильма — изначальное состояние клетки

Сохраним новый документ Flash в файле. И приступим...

Временна́я шкала

Прежде всего, обратим внимание на верхнюю половину окна документа Flash. Как нам уже известно, там отображается панель **Timeline**. (Если вы почему-то не видите эту панель, проверьте, не выключен ли пункт-выключатель **Timeline** меню **Window**, и включите его.) В левой части этой панели находится уже знакомый нам по *главе 5* список слоев. Правой же части этой панели мы пока не касались.

А находится там так называемая *временна́я шкала* (рис. 11.2) — набор последовательностей кадров (*дорожек*), из которых состоит фильм. Кадры фильма отображаются в виде прямоугольников с точкой в их нижней части. В настоящее время наш фильм содержит только один кадр, т. е. фактически представляет собой статичное изображение.

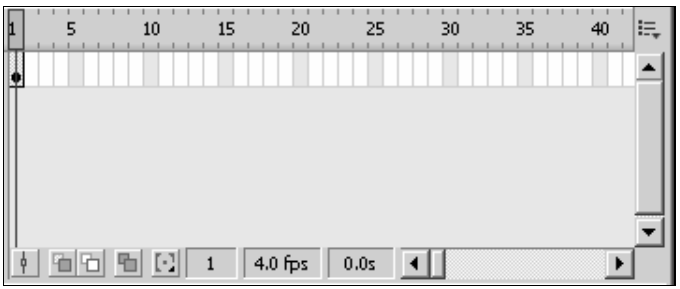


Рис. 11.2. Временна́я шкала в правой части панели **Timeline**

Остальное пространство временно́й шкалы занимают пустые позиции, в которые могут быть помещены другие кадры. Пустые позиции обозначаются либо белыми, либо светло-серыми (каждый пятый кадр) прямоугольниками без точки.

На рис. 11.2 хорошо видно, что каждому слою, присутствующему в изображении, соответствует своя дорожка кадров. Это понятно, ведь каждый слой может содержать свою собственную графику, независимую от графики, располагающейся на других слоях. Так что мы можем помещать на разных слоях разные анимированные персонажи, которые будут "работать" одновременно. Так, кстати, и делают.

Над самой верхней дорожкой находится *шкала кадров* — серая линейка, проградуированная в кадрах. Взглянув на нее, мы всегда можем выяснить номер того или иного кадра. По шкале кадров перемещается *бегунок* — вертикальная красная полоса, показывающая, какой кадр фильма в данный момент воспроизводится, или, как еще говорят, является *текущим*.

Бегунок этот можно перемещать мышью. Также можно просто щелкнуть по нужному кадру дорожки, чтобы его выделить. Выделенный кадр сразу же станет текущим.

Здесь нужно обязательно сказать вот что. На рабочем листе в данный момент отображается только содержимое текущего кадра. Содержимое остальных кадров при этом увидеть невозможно. (Забегая вперед, скажем, что способы увидеть содержимое всех кадров анимации все же имеются, и мы обязательно о них поговорим.)

Выше было сказано, что кадр обозначается небольшим прямоугольником с черной точкой в нижней его части. К этому нужно добавить, что прямоугольник-кадр может быть разных цветов. Так, кадр, имеющий содержимое (графику), закрашивается Flash светло-серым цветом. Если же кадр не имеет содержимого, он отобразится белым. Выделенный кадр всегда обозначается черным прямоугольником с белой точкой.

Мы также можем выделить любую пустую позицию на дорожке. Выделенная пустая позиция будет отображена в виде черного прямоугольника без точки (!).

В нижней части панели **Timeline** находится небольшая строка статуса временно́й шкалы. В этой строке отображается различная справочная информация, разнесенная по трем секциям (рис. 11.3). В левой секции выводится номер текущего кадра, в средней — частота кадров, заданная в параметрах документа, а в правой — продолжительность воспроизведения анимации с ее начала до текущего кадра. А если дважды щелкнуть на средней секции, на экране появится диалоговое окно **Document Properties**.

Рис. 11.3. Справочная информация, отображаемая в строке статуса панели **Timeline**

Также в строке статуса находится несколько небольших кнопок. Эти кнопки служат для задания некоторых дополнительных параметров; мы рассмотрим их далее в этой главе.

Наконец, временна́я шкала содержит две полосы прокрутки и кнопку вызова дополнительного меню, расположенную в ее правом верхнем углу. Для чего служат полосы прокрутки, понятно без всяких объяснений, пункты же дополнительного меню и их назначение мы рассмотрим позже, когда приступим собственно к созданию анимации.

Создание кадров анимации

Итак, мы хотим создать последовательность кадров, т. е. покадровую анимацию, фильм. Давайте же создадим ее.

Из *главы 1* мы помним, что каждый кадр покадровой анимации — ключевой. Мы уже говорили об этом, так что сейчас не будем повторяться.

Один ключевой кадр у нас уже есть — это первый кадр, созданный самим Flash при создании документа. Поскольку этот кадр является текущим (других-то кадров все равно нет), то нарисованное нами изначальное изображение готовой к делению клетки окажется именно в нем и станет его содержимым. Теперь нам нужно добавить к этому кадру другие, содержащие другие фазы деления клетки.

Ниже приведена последовательность действий по созданию покадровой анимации.

1. Создание нового ключевого кадра.
2. Выделение вновь созданного кадра.
3. Рисование изображения для нового кадра.
4. Если не конец фильма, то переход к первому шагу.

Сначала нам нужно создать новый ключевой кадр. Этот кадр будет находиться на следующей после первого кадра пустой позиции дорожки. Выделим ее щелчком мыши и выберем пункт **Keyframe** подменю **Timeline** меню **Insert** или пункт **Insert Keyframe** контекстного меню дорожки. Flash после этого создаст новый ключевой кадр и сам сделает его текущим.

Но на этом забота Flash о нас, пользователях, не заканчивается. При создании нового ключевого кадра он сам скопирует в него содержимое предыдущего ключевого кадра. Часто это бывает полезно — нам не придется рисовать каждый кадр "с нуля", будет достаточно только слегка подправить предыдущее изображение. Но мы собираемся нарисовать в новом кадре что-то принципиально новое (например, новую сцену фильма), сначала придется удалить его изначальное содержимое.

Специально для таких случаев Flash предоставляет возможность создать "пустой" ключевой кадр. Для этого достаточно выбрать пункт **Blank Keyframe** подменю **Timeline** меню **Insert** или пункт **Insert Blank Keyframe** контекстного меню дорожки. Такой кадр не будет содержать никакой графики.

Ну все, мы выяснили, как создаются ключевые кадры. Теперь остается только создать содержимое этих кадров — и первый наш фильм готов!

Чтобы "разделить" клетку пополам, мы можем выделить поочередно сначала одну, потом вторую ее половины и "разнести" их по разным концам листа. Для этого лучше всего использовать выделение перетаскиванием мыши. По-

сле этого останется исправить форму получившихся "половинок", чтобы они стали похожими на клетки. Все это выполняется с помощью изученных нами еще в *главе 3* инструментов.

И напоследок — один совет. Не следует допускать слишком резких перемещений изображений на разных кадрах. Это вызовет дерганье персонажа фильма при просмотре, очень, кстати сказать, неприятное.

Использование растянутых кадров

При создании фильма часто бывает нужно, чтобы какое-то изображение присутствовало на экране в течение нескольких кадров анимации. Как это сделать?

Самый простой и очевидный способ — создать соответствующее количество ключевых кадров, содержащих это самое неменяющееся изображение. Когда эти кадры будут воспроизводиться, на экране будет присутствовать одна и та же "картинка", что нам и нужно.

Да, это просто и очевидно, но не слишком экономно. Ведь все эти одинаковые кадры, содержащие одно и то же изображение, переносятся в результирующий файл Shockwave/Flash и увеличивают его размеры. А за эти самые размеры результирующего файла иной раз кипят нешуточные баталии.

Поэтому лучше сделать по-другому. А именно, использовать так называемые *растянутые кадры*. Растянутый кадр — это обычный ключевой кадр, но занимающий не одну, а сразу несколько позиций дорожки (*подкадров*). Фактически растянутый кадр — это замена нескольким обычным ключевым кадрам, содержащим одинаковые изображения. И замена отличная, ведь подкадры растянутого кадра практически не занимают место в результирующем файле Shockwave/Flash.

Чтобы создать растянутый кадр, сначала нужно выделить ключевой кадр, который мы желаем "растянуть", и выполнить любое из следующих действий:

- выбрать пункт **Frame** подменю **Timeline** меню **Insert**;
- выбрать пункт **Insert Frame** контекстного меню выделенного кадра;
- нажать клавишу <F5>.

В результате выделенный ключевой кадр "растянется" на один подкадр, сдвинув все последующие кадры дорожки вправо.

Часто растянутый кадр занимает больше двух позиций временно́й шкалы. Поэтому нам придется выполнить любую из перечисленных выше операций несколько раз, чтобы создать соответствующее количество подкадров. Проще и быстрее всего для этого использовать клавишу <F5>.

Растянутый кадр отображается на дорожке в виде прямоугольника белого или темно-серого цвета, в зависимости от того, содержит этот кадр графику или нет (рис. 11.4). Хорошо видно, что точка, являющаяся признаком ключевого кадра, находится в самом первом его подкадре, а в последнем хорошо заметен светлый прямоугольник — *конечный маркер* растянутого кадра.

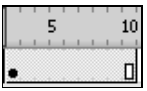


Рис. 11.4. Растянутый кадр

Выделить растянутый кадр можно, щелкнув мышью на любом его подкадре.

НА ЗАМЕТКУ

Фактически подкадры растянутого кадра — это промежуточные кадры, о которых мы говорили в *главе 1*. Они формируются самим проигрывателем Flash на основании изображения, содержащегося в ключевых кадрах.

Просмотр созданной анимации

Вот и готов наш первый фильм Flash. Давайте, что ли, посмотрим его?..

Простейший способ воспроизвести созданный фильм прямо в среде Flash — нажать клавишу <Enter> или выбрать пункт **Play** меню **Control**. Flash тотчас воспроизведет наше творение. При этом бегунок будет перемещаться по шкале кадров, показывая воспроизводимый в данный момент кадр.

Воспроизведение, запущенное таким образом, всегда начинается с текущего кадра, но только если это не последний кадр дорожки. Если же мы сделали текущим именно последний кадр дорожки, Flash начнет воспроизведение с первого кадра. Закончившись на последнем кадре, воспроизведение остановится.

Если мы захотим прервать воспроизведение раньше, чем фильм дойдет до конца, то должны будем нажать еще раз клавишу <Enter> или выбрать в меню **Control** пункт **Stop**.

Чтобы *зациклить* фильм (сделать так, чтобы он воспроизводился бесконечно), мы включим пункт-выключатель **Loop Playback** в меню **Control**. После этого, когда воспроизведение дойдет до последнего кадра фильма, бегунок перескочит на первый кадр, и фильм начнет воспроизводиться с начала.

Мы также можем перемещаться по фильму вперед (к концу) и назад (к началу) по кадрам. Для этого следует пользоваться клавишами <точка> и <запятая> соответственно. Также можно выбирать пункты **Step Forward One Frame** и **Step Backward One Frame** меню **Control**. Ну и, конечно, можно просто мышью перетаскивать бегунок в нужном направлении.

Чтобы быстро переместить бегунок на первый кадр фильма ("перемотать" фильм в начало), мы выберем пункт **Rewind** меню **Control** или нажмем комбинацию клавиш <Ctrl>+<Alt>+<R>. А пункт **Go to End** позволит нам переместить бегунок на последний кадр фильма ("перемотав" его в конец).

Для управления воспроизведением фильма мы можем пользоваться одним из инструментариев Flash — *пультом* (рис. 11.5). Чтобы вывести его на экран, нужно включить пункт-выключатель **Controller** подменю **Toolbars** меню **Window**.

Рис. 11.5. Пульт Flash

Перечислим все кнопки пульта в порядке слева направо:

- остановка воспроизведения;
- "перемотка" в начало, т. е. перемещение бегунка на первый кадр фильма;
- перемещение на кадр назад, т. е. к началу фильма;
- запуск воспроизведения;
- перемещение на кадр вперед, т. е. к концу фильма;
- "перемотка" в конец, т. е. перемещение бегунка на последний кадр фильма.

Будет ли пульт вам полезен — решать вам самим. Честно говоря, автор так ни разу им и не воспользовался. Ему вполне хватает пунктов меню **Control** и связанных с ними комбинаций клавиш.

Еще одно средство Flash будет нам полезно, но не сейчас, а в дальнейшем, когда мы начнем создавать фильмы, содержащие интерактивные элементы (например, кнопки). Это режим *воспроизведения фильма в отдельном окне*. Чтобы просмотреть фильм в отдельном окне, достаточно выбрать пункт **Test Movie** меню **Control** или нажать комбинацию клавиш <Ctrl>+<Enter>.

Что в этом случае сделает Flash? А вот что...

1. Выполнит экспорт нашего документа в формат Shockwave/Flash.
2. Откроет новое окно — *окно просмотра фильма*.
3. Загрузит получившийся файл Shockwave/Flash в это окно и запустит его воспроизведение.

Для управления воспроизведением фильма в отдельном окне мы можем пользоваться уже знакомыми нам пунктами меню **Control** (это меню доступно и в окне просмотра): **Play**, **Stop**, **Rewind**, **Step Forward One Frame** и **Step Backward One Frame**.

В окне просмотра фильм всегда воспроизводится зацикленным. Чтобы избежать этого, мы отключим пункт-выключатель **Loop** меню **Control** этого окна.

Находясь в окне просмотра фильма, можно также задавать качество и масштаб его отображения. Для задания качества служат пункты-переключатели, находящиеся в подменю **Quality** меню **View**: **Low** (низкое), **Medium** (среднее) или **High** (высокое; выбран по умолчанию). Для задания масштаба следует воспользоваться уже знакомыми нам пунктами меню **View**: **Zoom In** (увеличение масштаба), **Zoom Out** (уменьшение масштаба) — и пунктами подменю **Magnification**.

Насладившись своим творением, закроем окно просмотра, щелкнув на кнопке закрытия или выбрав пункт **Close** меню **File** этого окна.

Правка анимации

Любое, даже, на первый взгляд, совершенное творение может быть еще немного усовершенствовано. А поскольку в мире нет ничего совершенного, переделывать уже сделанное нам придется очень часто.

Фильмы Flash — не исключение из этого правила. Посмотрим, что предлагает нам Flash для того, чтобы улучшить уже сделанное.

Работа с кадрами

Не может быть, чтобы мы в процессе работы над нашим первым фильмом не наделали ошибок. Как их исправить?

Исправить содержимое какого-либо кадра очень просто. Щелкаем мышью по нужному кадру — и Flash выводит его содержимое на рабочем листе.

(Для выделения растянутого кадра достаточно щелкнуть по любому его подкадру.) А как исправить изображение на рабочем листе, мы давно знаем.

В некоторых случаях нужно выделить сразу несколько кадров. Для этого мы щелкнем на первом из кадров, которые нужно выделить, нажмем клавишу <Ctrl> и продолжим щелкать на остальных нужных кадрах, не отпуская этой клавиши. Если мы хотим выделить группу идущих подряд кадров, щелкнем мышью на первом из них, нажмем клавишу <Shift> и, не отпуская ее, щелкнем на последнем кадре.

Чтобы выделить сразу все кадры дорожки, воспользуемся пунктом **Select All Frames** подменю **Timeline** меню **Edit** или одноименным пунктом контекстного меню временно́й шкалы. Также мы можем нажать комбинацию клавиш <Ctrl>+<Alt>+<A>. При этом все кадры данной дорожки будут выделены, но на рабочем листе будет отображаться содержимое только кадра, на котором установлен бегунок.

Мы можете перемещать выделенные кадры мышью на другое место временно́й шкалы. При этом кадры будут перемещены вместе со всем их содержимым. Таким образом можно перемещать кадры не только внутри дорожки, но и с дорожки на дорожку, т. е. на другой слой документа.

При этом нужно иметь в виду вот что. Если мы выделим группу идущих подряд кадров и перетащим ее вправо, Flash автоматически растянет кадр, находящийся левее этой группы. Flash, как и природа, не терпит "пустоты" в середине дорожки и старается ее заполнить всеми доступными ему способами.

Создавать новые кадры мы уже умеем. Для этого достаточно выделить первую пустую позицию дорожки правее последнего ключевого кадра и выбрать пункт **Keyframe** подменю **Timeline** меню **Insert** или пункт **Insert Keyframe** контекстного меню этой пустой позиции. Созданный таким образом кадр будет содержать изображение, взятое с предыдущего ключевого кадра. Если же нужно создать пустой ключевой кадр, то следует выбрать пункт **Blank Keyframe** подменю **Timeline** меню **Insert** или пункт **Insert Blank Keyframe** контекстного меню.

Создать ключевой кадр в середине дорожки несколько сложнее. Для этого придется сначала освободить для него место, перетащив все кадры, расположенные правее места, где будет находиться новый кадр, вправо на одно деление шкалы кадров. Flash растянет предыдущий кадр, добавив к нему один подкадр. После этого нам останется выделить получившийся подкадр и создать ключевой кадр любым описанным выше способом.

Еще мы можем преобразовать любой подкадр растянутого кадра в ключевой кадр. Выделим нужный подкадр и выберем в подменю **Timeline** меню **Modify** или контекстном меню выделенного подкадра пункт **Convert to Keyframes**

или нажмем клавишу <F6>. Flash преобразует этот подкадр в ключевой кадр, сохранив его содержимое. А пункт **Convert to Blank Keyframes** и клавиша <F7> заставят Flash преобразовать подкадр в "пустой" ключевой кадр, в котором мы в дальнейшем сможем нарисовать любое изображение.

Обратное преобразование — из ключевого кадра в подкадр, являющийся частью предыдущего ключевого кадра (который соответственно растянется), — также возможно. Выделим ключевой кадр и выберем пункт **Clear Keyframe**) подменю **Timeline** меню **Insert** или контекстного меню или нажмем комбинацию клавиш <Shift>+<F6>.

Разумеется, описанные выше операции можно проделывать сразу над группой кадров. Как выделить группу кадров, мы уже знаем.

Мы можем выполнить дублирование любого ключевого кадра, создав его точную копию. Для этого установим курсор мыши на кадр, который хотим продублировать; нажмем левую кнопку мыши и, не отпуская ее, нажмем клавишу <Alt>; удерживая ее, перетащим кадр на нужное место и отпустим клавишу и кнопку мыши. Так же мы можем копировать и группы выделенных кадров.

Мы можем менять длительность растянутого кадра, перетаскивая мышью его конечный маркер и удерживая при этом нажатой клавишу <Ctrl>. Таким образом можно "ужать" растянутый кадр до размеров обычного, занимающего одно-единственное деление шкалы кадров. Нужно только иметь в виду, что при этом Flash заполнит "опустевшее" пространство на дорожке пустым растянутым ключевым кадром, который сам создаст.

Разумеется, мы можем удалять ненужные кадры. Для этого выделим кадр (кадры), который хотим удалить, и выберем пункт **Remove Frames** подменю **Timeline** меню **Edit** или контекстного меню кадра или нажмем комбинацию клавиш <Shift>+<F5>. Выделенный нами кадр (кадры) будет удален, а все кадры, расположенные правее него, сдвинутся влево, заполняя "вакантное" место.

> ***Внимание!***
>
> Не пользуйтесь для удаления кадров клавишей <Del>. Она удаляет содержимое выделенного кадра, но не сам кадр.

Пункты **Cut Frames** и **Copy Frames** подменю **Timeline** меню **Edit** (и соответствующие им клавиатурные комбинации <Ctrl>+<Alt>+<X> и <Ctrl>+<Alt>+ +<C>) позволят нам, соответственно, вырезать и скопировать выделенные кадры в буфер обмена Windows. А пункт **Paste Frames** того же подменю **Timeline** меню **Edit** (и комбинация клавиш <Ctrl>+<Alt>+<V>) выполняет вставку кадров из буфера обмена в выделенную позицию дорожки. Если пе-

ред вставкой какой-то кадр был выделен, он и все последующие кадры будут отодвинуты вправо, чтобы вместить вставляемые кадры.

В меню **Edit** имеется также пункт **Clear Frames** (комбинация клавиш <Alt>+<Backspace>). Выбрав его, мы удалим все содержимое выделенного кадра. (Не вполне понятно, зачем разработчики Flash ввели этот пункт — есть же пункт **Clear** меню **Edit** и многофункциональная клавиша <Del>, которые работают точно так же.)

Все пять вышеперечисленных пунктов присутствуют также в контекстном меню выделенного кадра.

Мы можем *инвертировать* всю дорожку или ее фрагмент, т. е. повернуть ее задом наперед. Для этого выделим нужные кадры (или сразу всю дорожку) и выберем пункт **Reverse** подменю **Timeline** меню **Modify** или пункт **Reverse Frames** контекстного меню выделенных кадров.

Для пробы давайте ради смеха заставим нашу клетку сначала делиться, как обычно, а потом, наоборот, сливаться воедино. Делается это очень просто. Ниже приведен алгоритм действий (предположим, что фильм "Деление клетки" полностью готов).

1. Выделяем всю последовательность кадров нашего фильма и копируем их в буфер обмена. (Пункт **Copy Frames** подменю **Timeline** меню **Edit**.)
2. Выделяем первую пустую позицию правее последнего кадра дорожки и вставляем кадры из буфера обмена. (Пункт **Paste Frames** подменю **Timeline** меню **Edit**.) Это будет вторая, "экзотическая" половина фильма, описывающая процесс слияния двух клеток в одну и следующая сразу же за первой, "бытовой", частью фильма.
3. Выделяем все только что вставленные кадры ("экзотическую" часть) и инвертируем их. (Пункт **Reverse Frames** подменю **Timeline** меню **Modify**.) Все!

Дополнительные возможности работы с кадрами

Flash предоставляет несколько дополнительных возможностей работы с кадрами анимации. Они связаны с весьма забавной и зачастую полезной функцией, позволяющей видеть на рабочем листе содержимое сразу нескольких кадров (так называемое *наложение*). Это может понадобиться, например, для того, чтобы выяснить, не содержит ли наша анимация резких рывков, которые отнюдь ее не украсят.

Чтобы включить наложение, нажмем кнопку-выключатель **Onion Skin** (), находящуюся в строке статуса временной шкалы. На рабочем листе появится изображение, похожее на показанное на рис. 11.6.

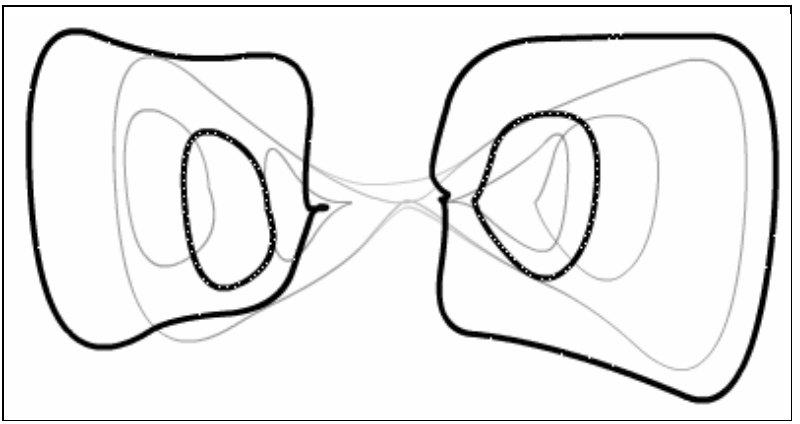


Рис. 11.6. Вид рабочего листа при включенном наложении

Видно, что содержимое текущего кадра отображается обычными линиями в обычных цветах. Содержимое же нескольких других кадров выводится тонкими серыми линиями в приглушенных цветах.

Режим наложения предусматривает также возможность вывода содержимого невыделенных кадров в виде одних контуров, без заливок (*наложение контуров*). Содержимое текущего кадра будет при этом выводиться полностью, со всеми заливками. Это может быть полезно, если требуется ускорить перерисовку изображения на медленных компьютерах. Также это может помочь, если наше изображение содержит много заливок, и мы хотим временно скрыть их, чтобы рассмотреть и изменить контуры фигур.

Чтобы включить режим наложения контуров, мы нажмем кнопку-выключатель **Onion Skin Outlines** (). Эта кнопка находится также в строке статуса временной шкалы, правее уже знакомой нам кнопки **Onion Skin**.

В обоих рассмотренных случаях можно править только содержимое текущего кадра — ограничение введено для того, чтобы мы случайно не изменили другие кадры анимации. Но если мы все-таки хотим иметь возможность править содержимое всех кадров, выведенных на рабочем листе, то должны будем включить кнопку-выключатель **Edit Multiple Frames** (), находящую-

ся, опять же, в строке статуса временно́й шкалы. Только в этом случае придется работать очень внимательно.

Теперь обратим внимание на временну́ю шкалу (режим наложения должен быть включен). То, что мы должны увидеть, показано на рис. 11.7. Хорошо заметно, что все кадры, чье содержимое отображается на рабочем листе, охвачены своего рода скобками. Это значит, что данные кадры входят в так называемый *диапазон наложения*; центром этого диапазона является текущий кадр (точнее, бегунок). А ограничивающие диапазон скобки называются *маркерами диапазона*.

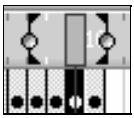


Рис. 11.7. Диапазон кадров

Диапазон наложения по умолчанию привязан к бегунку. Это значит, что если мы переместим бегунок на другую позицию шкалы кадров, диапазон наложения переместится вместе с ним.

Мы можем изменять размер диапазона наложения, перемещая мышью его маркеры. При этом левый и правый маркеры перемещаются независимо друг от друга; так что можно свободно увеличивать или уменьшать диапазон в любую сторону.

Есть и другая возможность задать размер диапазона кадров — воспользоваться пунктами меню, которое появится на экране при нажатии кнопки **Modify Onion Markers** (рис. 11.8). Эта кнопка также находится в строке статуса временно́й шкалы.

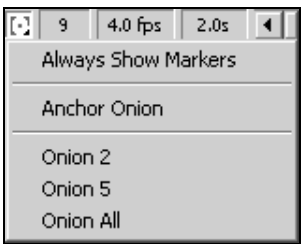


Рис. 11.8. Кнопка **Modify Onion Markers** с раскрытым меню

Для задания размера диапазона наложения это меню предоставляет нам три пункта-переключателя:

- **Onion 2** — по два кадра с обеих сторон бегунка;
- **Onion 5** — по пять кадров с обеих сторон бегунка;
- **Onion All** — все кадры дорожки.

Кроме этих трех, меню кнопки **Modify Onion Markers** содержит еще два пункта-выключателя. Рассмотрим их.

Пункт-выключатель **Always Show Markers**, будучи включенным, заставляет Flash показывать диапазон кадров даже тогда, когда режим наложения отключен (не включены ни кнопка **Onion Skin**, ни кнопка **Onion Skin Outlines**). Зачем это нужно, автору непонятно.

Выше говорилось, что диапазон наложения по умолчанию привязан к бегунку и перемещается вместе с ним. Пункт-выключатель **Anchor Onion** при включении фиксирует диапазон наложения на том месте, где он сейчас находится, в то время как бегунок может свободно перемещаться по шкале кадров.

Дополнительные возможности временно́й шкалы

Уже неплохо знакомая нам временна́я шкала Flash таит в себе несколько сюрпризов. Если их использовать с умом, они станут приятными сюрпризами. Давайте же познакомимся с ними.

Наш первый фильм оказался довольно коротким — всего 10 кадров. Но недалек тот день, когда наши фильмы будут содержать многие десятки и сотни кадров. В таких случаях Flash не хватит ширины панели **Timeline**, чтобы отобразить все эти кадры на временно́й шкале, и нам придется пользоваться горизонтальной полосой прокрутки для их просмотра. В этом случае весьма поможет одна небольшая возможность, предусмотренная во Flash.

Предположим, что мы поместили бегунок на какой-либо кадр, а потом прокрутили временну́ю шкалу так, что потеряли его из виду. Тогда нажмем кнопку **Center Frame** ( ), расположенную в строке статуса временно́й шкалы. Flash прокрутит временну́ю шкалу таким образом, чтобы кадр, помеченный бегунком, и, соответственно, сам бегунок оказались в ее середине.

Кроме того, мы можем изменять ширину прямоугольников, обозначающих кадры. Для этого нам придется воспользоваться дополнительным меню временно́й шкалы. Это меню можно открыть, нажав кнопку, расположенную в

правом верхнем углу панели **Timeline**, чуть выше вертикальной полосы прокрутки. Эта кнопка и ее раскрытое меню показаны на рис. 11.9.

Рис. 11.9. Дополнительное меню временно́й шкалы и кнопка его открытия (находится в правом верхнем углу панели **Timeline**)

Для задания ширины прямоугольников-кадров служат пять пунктов-переключателей: **Tiny** (самые узкие прямоугольники), **Small** (узкие), **Normal** (значение по умолчанию), **Medium** (средние) и **Large** (самые широкие прямоугольники). Единственное: со слишком узкими кадрами трудно работать, а слишком широкие занимают много места на экране.

В ту же группу, что и перечисленные выше пять пунктов, входят еще два пункта-переключателя: **Preview** и **Preview in Context**. Они включают режим *просмотра кадров на месте*. Давайте выясним, что это такое.

Если включен один из пунктов-выключателей, задающих размер кадров, Flash показывает кадры в виде простых прямоугольников. Чтобы просмотреть содержимое какого-либо кадра, нам придется выделить его, а чтобы просмотреть содержимое сразу нескольких кадров, нужно будет воспользоваться описанным в предыдущем разделе режимом наложения.

Если же задействовать режим просмотра кадров на месте, для чего достаточно включить пункт **Preview**, кадры на временно́й шкале примут вид достаточно крупных прямоугольников, в которых отобразится их содержимое (рис. 11.10). Причем, как видно на рис. 11.10, в кадрах выводится только собственно изображение, а все свободное пространство рабочего листа отрезается.

Пункт **Preview In Context** отличается от пункта **Preview** тем, что в каждом прямоугольнике-кадре показывается весь рабочий лист (рис. 11.11). Пустое

пространство рабочего листа не отрезается, так что мы всегда сможем увидеть, в каком месте на рабочем листе находится изображение, и оценить его относительные размеры.

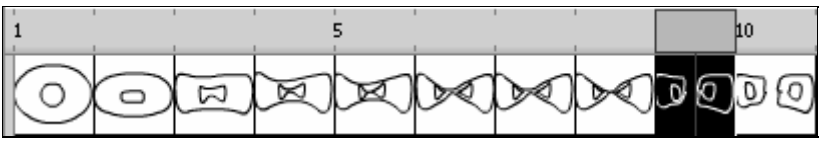


Рис. 11.10. Вид временно́й шкалы при включенном пункте **Preview** ее дополнительного меню

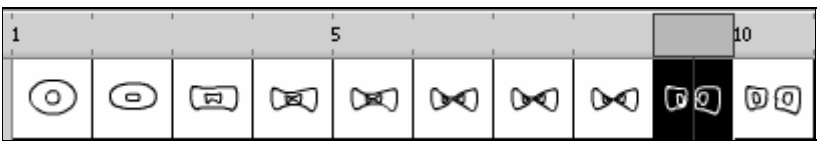


Рис. 11.11. Вид временно́й шкалы при включенном пункте **Preview In Context** ее дополнительного меню

Платой за наглядность служит серьезное увеличение размера прямоугольников-кадров при включенном режиме просмотра на месте. Поэтому оба вышеописанных режима лучше использовать только время от времени, когда без них действительно не обойтись. В остальное время лучше выбрать обычный режим отображения кадров в виде простых прямоугольников, для чего достаточно включить один из пунктов, задающих размер прямоугольников-кадров (**Tiny**, **Small**, **Normal**, **Medium** или **Large**).

Если мы отключим пункт-выключатель **Tinted Frames**, также находящийся в дополнительном меню, прямоугольники-кадры будут отображаться немного по-другому. Прежде всего, они всегда будут закрашиваться белым цветом, независимо от того, содержат они графику или нет. При этом кадр, содержащий графику, будет помечен синей точкой, а "пустой" кадр — белой. Вероятно, кому-то так нравится...

Последний, еще не рассмотренный нами пункт-выключатель дополнительного меню временно́й шкалы — **Short**. Если его включить, размеры прямоугольников-кадров уменьшатся по вертикали примерно наполовину. Это бывает очень полезно, если наш фильм содержит много слоев, которые не помещаются в список.

Использование сцен

Одной из интереснейших возможностей Flash является возможность разделения фильма на *сцены*. Каждая такая сцена представляет собой набор дорожек, содержащий логически законченный фрагмент фильма, более или менее независимый от остальных последовательностей. Например, в нашем случае в качестве отдельных сцен могут быть оформлены название фильма и сам фильм. (Хотя наш фильм весьма короток, поэтому разбивать его на сцены совсем не обязательно.)

Для работы со сценами служит панель **Scene** (рис. 11.12). Чтобы вывести ее на экран, нужно включить пункт-выключатель **Scene** подменю **Other Panels** меню **Window** или нажать комбинацию клавиш <Shift>+<F2>.

Рис. 11.12. Панель **Scene**

Бóльшую часть этой панели занимает список уже созданных сцен. Мы можем выделить, или сделать *текущей*, любую сцену, щелкнув на ней мышью. После этого мы сможем произвести над сценой различные предусмотренные Flash манипуляции.

Нужно сказать сразу, что на временнóй шкале отображаются только те дорожки, что входят в текущую сцену. Просмотреть содержимое остальных сцен, не делая их последовательно, одна за другой, текущими, мы не сможем.

Кстати, имя текущей сцены выводится в левой части инструментария документа (рис. 11.13).

Чтобы создать новую пустую (не содержащую кадров) сцену, следует нажать кнопку **Add scene** ( ), расположенную в нижней части панели **Scene**. Также можно выбрать пункт **Scene** меню **Insert**. Новая сцена появится в

списке позицией ниже текущей, после чего сама станет текущей. Так что мы можем начать наполнять ее полезным содержимым — кадрами — сразу же после создания.

Рис. 11.13. Имя текущей сцены выводится в инструментарии документа

Каждая сцена фильма должна иметь уникальное имя. По умолчанию Flash дает сценам имена вида `Scene <Порядковый номер>`. Изначально же фильм состоит из одной-единственной сцены, сформированной самим Flash при создании документа и имеющей имя `Scene 1`.

Вновь созданную сцену желательно сразу же переименовать, дав ей имя, отражающее содержимое (например, `Название` или `Фильм`). Для этого достаточно дважды щелкнуть мышью на имени нужной сцены в списке панели **Scene**. Вместо него появится небольшое поле ввода; введем в него новое имя и нажмем клавишу <Enter> для его сохранения или клавишу <Esc> для отмены.

Есть еще один способ создания новых сцен — дублирование существующих. Для этого служит кнопка **Duplicate scene** ( ), расположенная также в нижней части панели **Scene**. Выделим в списке панели **Scene** сцену, которую нужно продублировать, и нажмем эту кнопку. Новая сцена появится позицией ниже и получит имя вида `<Изначальная сцена> copy`.

Мы можем перемещать сцены в списке мышью, изменяя порядок их воспроизведения. Сцены воспроизводятся последовательно, в порядке сверху вниз.

Для удаления ненужной сцены мы должны будем выделить ее и нажать кнопку **Delete scene** ( ) в нижней части панели **Scene**. Flash выдаст предупреждение, говорящее о том, что сцена будет удалена безвозвратно. Нажмем кнопку **OK**, чтобы удалить сцену, или **Cancel**, если мы еще не набрались решимости. Чтобы сразу же удалить ненужную сцену без вывода этого предупреждения, можно щелкнуть по кнопке **Delete scene**, удерживая нажатой клавишу <Ctrl>.

Теперь вернемся в окно документа. В правой части инструментария документа находится кнопка **Edit Scene**, при нажатии которой на экране появляется меню созданных в документе сцен (рис. 11.14). Чтобы сделать какую-

либо сцену текущей, достаточно просто выбрать соответствующий пункт-переключатель этого меню.

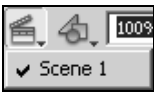


Рис. 11.14. Кнопка **Edit Scene** и раскрытое меню сцен

Кстати, эти же пункты-переключатели присутствуют в нижней части подменю **Go to** меню **View**.

В том же подменю **Go to** меню **View** находятся пункты, которыми мы можем пользоваться для "путешествия" по сценам. Этих пунктов четыре:

- **First** — переход к самой первой сцене в списке панели **Scene**;
- **Previous** — переход к предыдущей сцене;
- **Next** — переход к следующей сцене;
- **Last** — переход к последней сцене.

При воспроизведении готового фильма в среде Flash (его, как мы помним, запускает пункт **Play** меню **Control** и клавиша <Enter>) нужно иметь в виду, что по умолчанию воспроизводится только содержимое текущей сцены. Чтобы просмотреть весь фильм (в смысле, все его сцены), нужно будет сначала включить пункт-выключатель **Play All Scenes** меню **Control**.

Напротив, при воспроизведении фильма в отдельном окне (пункт **Test Movie** меню **Control** или комбинация клавиш <Ctrl>+<Enter>) фильм всегда воспроизводится полностью, со всеми сценами. (Кстати, точно так же ведет себя и проигрыватель Flash.) Если же мы пожелаем просмотреть в отдельном окне только текущую сцену, выберем пункт **Test Scene** меню **Control** или нажмем комбинацию клавиш <Ctrl>+<Alt>+<Enter>.

И еще кое-что о сценах, точнее — о необходимости их использования.

Сцены Flash — инструмент, предназначенный, в основном, только для разработчика фильмов. Сцены позволяют разделить длинный фильм на несколько частей и работать потом с этими частями по отдельности. Проигрыватель Flash, судя по всему, никак сцены не обрабатывает; более того, похоже, он вообще не знает об их существовании. Так, если мы загрузим в него файл Shockwave/Flash с фильмом, содержащим сцены, они будут воспроизведены последовательно, одна за другой.

Осталось сказать совсем немногое. А именно: нумерация кадров фильма при использовании сцен остается сквозной, проходящей через все сцены. Например, если в фильме имеются три сцены по десять кадров каждая, то пер-

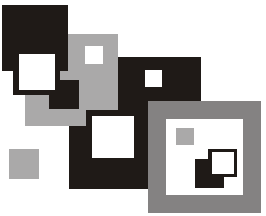
вая сцена будет содержать кадры с 1 по 10, вторая — с 11 по 20, а третья — с 21 по 30. Это еще раз доказывает, что проигрыватель Flash, в отличие от самой программы Flash, даже не подозревает о существовании сцен.

Что дальше?

Вот мы и создали нашу первую анимацию в среде Flash. И была она покадровой.

В следующей главе мы выясним, как создается во Flash трансформационная анимация. Мы также узнаем все о двух разновидностях трансформационной анимации — движении и морфинге: о том, как их создавать и как получить нужный нам результат. И, конечно же, мы продолжим наше творчество на кинематографической ниве.

Глава 12



Трансформационная анимация

А теперь настала пора познакомиться с трансформационной анимацией. То есть с тем, в чем так силен Macromedia Flash.

Подробно о трансформационной анимации рассказывалось в *главе 1*. Давайте вспомним, что это такое. Мы рисуем всего два ключевых кадра последовательности — начальный и конечный, — задающих положение анимируемого элемента до и после его перемещения, и определяем кое-какие параметры. Flash на основе всего этого создает необходимое количество промежуточных кадров, помещает их между ключевыми — и анимация готова. Просто, не правда ли?

Кроме простоты создания, трансформационная анимация имеет еще одно преимущество перед покадровой: в результирующем файле Shockwave/Flash она занимает меньше места. В самом деле, вместо того, чтобы хранить все кадры фильма, программа записывает в файл только несколько чисел — параметры функции, реализующей трансформационную анимацию. Сравним это с десятком изображений, пусть даже и векторных, из которых состоит покадровая анимация — это сравнение уж точно будет не в пользу последней.

Пожалуй, единственный недостаток трансформационной анимации — ограниченность ее возможностей. В самом деле, с ее помощью мы можем создать только:

- различные перемещения элементов изображения по рабочему листу;
- изменение формы элемента изображения с применением трансформаций (вращения, сдвига, изменения размеров и пр., в общем, всего того, что было описано в *главе 4*);
- изменение цвета элемента изображения с помощью цветовых эффектов (изменения прозрачности, оттенка и пр.; все они были описаны в *главе 8*);

- изменение параметров привязанных к элементу изображения фильтров (*см. главу 9*).

Негусто... Хоть Flash и предлагает нам достаточно мощные средства "оживить" изображения "малой кровью", красивые и сложные мультфильмы из одних трансформаций не сделаешь. Хотя для интернет-графики, которая обычно не отличается запредельной сложностью, этого более чем достаточно.

Что ж, пора за дело. Займемся трансформационной анимацией и посмотрим, что все-таки с ее помощью можно сделать. Flash, как всегда, нам поможет.

Два вида трансформационной анимации

Flash позволяет создавать трансформационную анимацию двух видов: движение и морфинг. Оба этих вида имеют практически одинаковые возможности, но применяются к разным видам элементов изображения.

Первый вид трансформационной анимации — *движение* — самый простой в реализации. Движение можно применить только к следующим разновидностям фрагментов изображения:

- экземплярам образцов (*см. главу 8*);
- группам (*см. главу 3*);
- текстовым блокам (*см. главу 7*).

К обычной графике, не являющейся ни тем, ни другим, ни третьим, движение неприменимо.

Как уже говорилось, движение создается очень просто — это его достоинство. Второе достоинство движения — в результирующем файле Shockwave/Flash оно занимает очень мало места. Недостаток же всего один — бедность изобразительных средств.

При создании движения нужно иметь в виду вот что. Каждый анимируемый элемент обязательно должен быть помещен на свой собственный слой. (О слоях *см. главу 5*.) Причем, кроме анимируемого экземпляра, на этом слое не должно находиться больше никакой графики. Иначе результат будет непредсказуемым.

Второй вид трансформационной анимации — *морфинг*. Он, в отличие от движения, может быть применен только к обычной графике. К экземплярам, группам и текстовым блокам он неприменим.

Недостатков у морфинга два. Во-первых, морфинг занимает в файле Shockwave/Flash больше места, чем движение (что понятно, ведь в этом случае приходится сохранять больше данных). Во-вторых, зачастую при морфинге бывает очень трудно получить нужный результат.

Выяснив все о двух поддерживаемых Flash типах трансформационной анимации, посмотрим теперь, как они создаются. Начнем с движения, как самого легкого в реализации.

Движение

В этом разделе мы выясним, как создается простейший вид трансформационной анимации — движение. Во время выяснения будем экспериментировать с созданным нами в *главе 11* фильмом, демонстрирующим процесс деления клетки. Мы добавим к нему начальные титры, показывающие его название.

Создание простейшего движения

Откроем документ, содержащий созданный нами фильм "Деление клетки". И разобьем его на две сцены — так нам будет проще (о сценах *см. главу 11*).

Первая сцена будет содержать титры с названием фильма и носить "говорящее" название `Название`. Для создания этих титров мы используем обычный текстовый блок, который потом анимируем (как мы помним, движение применимо и к текстовым блокам). Этот текстовый блок будет плавно "выезжать" сверху и останавливаться в центре рабочего листа. Отведем под этот процесс пять кадров — больше и не требуется.

Небольшое отступление. В *главе 2* говорилось, что любая графика, выходящая за пределы рабочего листа, отбрасывается при экспорте в формат Shockwave/Flash. Так вот, в случае анимированных экземпляров, групп и текстовых блоков дело обстоит не так — они сохраняются. Этим можно воспользоваться для того, чтобы скрыть до поры до времени какой-либо элемент изображения.

Вторая же сцена нашего фильма будет содержать собственно покадровую анимацию, показывающую процесс деления клетки. Мы назовем ее `Фильм`. Здесь мы ничего менять не будем.

Создадим пустую пока сцену `Название` и проследим, чтобы она шла в фильме первой. (Название фильма должно появляться в его начале, не так ли?) Дадим второй (имевшейся изначально) сцене имя `Фильм`. Сделаем сцену `Название` текущей и выделим единственный пока находящийся в ней ключевой кадр. Создадим в центре рабочего листа текстовый блок, напишем в нем текст `Деление клетки`. Потом сместим его точно вверх, пока он не выйдет за границы рабочего листа. Теперь все готово к созданию движения.

Первый ключевой кадр, содержащий скрытый за пределами рабочего листа текстовый блок, у нас есть. Нужно создать второй. Для этого выделим пус-

тую позицию дорожки, в которой движение нашего текстового блока должно закончиться. Это будет позиция № 5, как мы договорились ранее. Создадим там новый ключевой кадр, выбрав пункт **Keyframe** подменю **Timeline** меню **Insert** или пункт **Insert Keyframe** контекстного меню выделенной позиции. Первый ключевой кадр тотчас растянется, заполняя все свободное пространство до второго ключевого кадра.

Создав второй ключевой кадр, выделим его и переместим текстовый блок с названием фильма на его законное место — в центр рабочего листа. Воспользуемся панелью **Align**, чтобы расположить его точно в центре.

Все — оба ключевых кадра готовы. Настало время создать саму анимацию.

Выделим любой подкадр первого, растянутого, ключевого кадра. После этого посмотрим на панель **Properties**. В ней находится не рассмотренный еще нами раскрывающийся список **Tween**, с помощью которого и создается трансформационная анимация — и движение, и морфинг.

Изначально в этом списке выбран пункт **None**. Это значит, что для выделенного кадра не задана никакая трансформационная анимация. Чтобы задать движение, мы выберем пункт **Motion**. В панели **Properties** тотчас появится множество других элементов управления, задающих специальные параметры созданной нами анимации. Эти элементы управления и соответствующие им параметры мы рассмотрим далее в этой главе.

Все! Мы создали простейшее движение. При этом Flash преобразовал все подкадры первого, растянутого, ключевого кадра в промежуточные кадры. Эти кадры, как мы помним из *главы 1*, будут генерироваться самим проигрывателем Flash на основании содержимого ключевых кадров.

Щелкнем мышью на рабочем листе или временно́й шкале, чтобы убрать фокус ввода с панели **Properties**, и запустим воспроизведение анимации нажатием клавиши <Enter>. Наша анимация работает!

А теперь давайте посмотрим на временну́ю шкалу, точнее, на растянутый ключевой кадр (рис. 12.1). Хорошо видно, что он стал голубым (на рис. 12.1 — серым), более того — по нему проходит тонкая черная стрелка, указывающая вправо. Именно так Flash обозначает примененное к кадру движение.

Рис. 12.1. Растянутый кадр временно́й шкалы, для которого было задано движение

Если же мы захотим удалить созданное движение, то должны будем выделить любой промежуточный кадр задающей это движение последовательно-

сти и выбрать в раскрывающемся списке **Tween** панели **Properties** пункт **None**.

Что ж, наша первая трансформационная анимация готова. Давайте ее немного усложним.

Более сложное движение

Flash-аниматоры, как начинающие, так и опытные, очень любят такой эффект: какой-либо фрагмент изображения постепенно исчезает или, наоборот, появляется на экране. Это похоже на то, как на фотографии, опущенной в раствор проявителя, постепенно возникает изображение. Применим и мы этот прием. Пусть название фильма, закончив движение, постепенно исчезнет с экрана — как бы выцветет.

Для создания этого эффекта также используется трансформационная анимация, а именно — движение. Как мы помним, движение включает в себя не только собственно движение, но и трансформации, и цветовые эффекты. А с помощью задания цветового эффекта прозрачности можно заставить наш текстовый блок исчезнуть с экрана.

Здесь возникает одна небольшая проблема. Дело в том, что задавать цветовые эффекты можно только для экземпляров. А у нас — обычный текстовый блок! Как выйти из положения?

Выделим первый ключевой кадр нашей анимации и преобразуем текстовый блок в образец (о том, как это сделать, было рассказано в *главе 8*). Дадим образцу имя `Название` и присвоим тип "графический образец". После этого Flash создаст образец и преобразует текстовый блок в его экземпляр. Половина дела сделана.

Теперь выделим второй ключевой кадр нашей анимации и удалим текстовый блок. Растянутый ключевой кадр станет таким, как показано на рис. 12.2, — стрелка, обозначающая анимацию, превратится в пунктирную линию. Это значит, что наша анимация теперь фактически не имеет второго ключевого кадра (*"оборванная" анимация*), и Flash не знает, как ее закончить. (Кстати, такая же картина будет наблюдаться, если удалить второй ключевой кадр из последовательности кадров.)

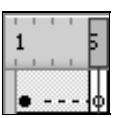


Рис. 12.2. "Оборванная" анимация

Чтобы в буквальном смысле положить конец такой анимации, нужно поместить на рабочий лист второго ключевого кадра экземпляр созданного нами ранее образца `Название`. После этого выровняем его точно по центру рабочего листа. Вот теперь все нормально.

Теперь заставим название фильма исчезнуть с экрана, для чего добавим к дорожке еще одну последовательность кадров. Эта последовательность будет следовать вслед за первой, и первым ее ключевым кадром станет второй ключевой кадр первой последовательности. Пусть вторая последовательность также состоит из пяти кадров — этого будет достаточно, чтобы зритель успел прочитать название фильма, прежде чем оно исчезнет.

Так, первый ключевой кадр новой последовательности у нас есть. Второй ее ключевой кадр мы поместим на позиции № 10 временно́й шкалы. Выделим любой из подкадров растянувшегося первого ключевого кадра и выберем в раскрывающемся списке **Tween** панели **Properties** пункт **Motion**. Результат этих действий показан на рис. 12.3.

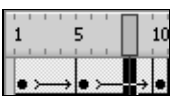


Рис. 12.3. Две следующие друг за другом последовательности кадров, реализующие различные анимации

Теперь выделим экземпляр образца `Название` (название фильма) и выберем пункт **Alpha** в раскрывающемся списке **Color** панели **Properties**. В поле ввода с регулятором, расположенным правее этого списка, введем значение 0% — полная прозрачность. Все — название нашего фильма теперь стало невидимым.

Проверим, что у нас получилось. Ура, работает!

Подобным образом мы можем создавать любые достаточно сложные анимации, просто добавляя во временну́ю шкалу последовательности кадров, одну за другой. Анимированные элементы могут "гулять" по всему экрану, исчезать и снова появляться, менять цвета, отбрасывать тень, искажаться и восстанавливать свой первоначальный вид. Попробуйте поэкспериментировать — это не только позволит вам приобрести опыт, но и просто позабавит.

Однако для начала все-таки стоит узнать о движении еще кое-что.

Параметры движения

Настала пора рассмотреть поближе панель **Properties** и все ее элементы управления, доступные в том случае, если выделен любой из кадров анимации (рис. 12.4).

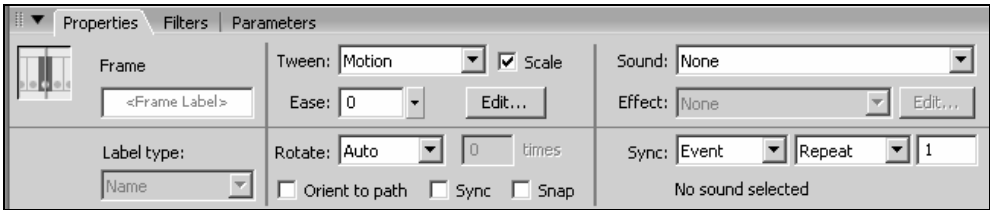


Рис. 12.4. Панель **Properties** после выделения промежуточного кадра движения

В верхнем левом углу панели **Properties** находится поле ввода **Frame**. С его помощью мы можем задать уникальное имя или примечание для какого-либо кадра. Имя кадра будет, в основном, полезно только для программирования сценариев и некоторых других возможностей, но мы можем воспользоваться этим полем, чтобы пометить нужный нам кадр.

Если мы зададим для кадра имя, он будет выглядеть так, как показано на рис. 12.5, т. е. заданное для него имя будет написано прямо на нем. Правда, это идеальный случай, поскольку кадр растянут, и имя помещается в нем целиком. Если же кадр занимает только одну позицию дорожки, в нем будет виден только красный флажок — знак того, что кадр имеет имя.

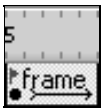


Рис. 12.5. Кадр, имеющий имя `frame`

В качестве *примечания* к кадру мы можем записать какую-либо дополнительную информацию, например, напоминание, что нужно с этим кадром сделать. Примечание также вводится в поле ввода **Frame**, но его должны предварять два символа `/`.

Кадр, содержащий примечание, выглядит так, как показано на рис. 12.6. Обратим внимание на два зеленых символа `/` — именно так обозначается кадр с примечанием.

Рис. 12.6. Кадр, содержащий примечание `frame`

Примечания не экспортируются в результирующий файл Shockwave/Flash, поэтому мы можем сделать их достаточно развернутыми, не опасаясь, что результирующий файл получится слишком большим.

Под полем ввода **Frame** в панели **Properties** (см. рис. 12.4) находится раскрывающийся список **Label type**. В нем задается тип значения в поле ввода **Frame**: имя кадра (пункт **Name**), примечание (пункт **Comment**) или имя якоря (пункт **Anchor**). При этом, если выбрать пункт **Comment**, Flash сам добавит в начало значения, введенного в поле ввода **Frame**, два символа `/`.

Да, но что такое *якорь*? Так называется кадр, являющийся своего рода "зацепкой" для проигрывателя Flash. Если фильм Flash воспроизводится в окне Web-обозревателя, пользователь может пользоваться кнопками **Forward** (Вперед) и **Back** (Назад) этой программы, чтобы "прыгать" от одного якоря к другому. Так что в фильмах, предназначенных для распространения через Интернет, якоря могут пригодиться.

Пример якоря можно увидеть на рис. 12.7. Он обозначается желтым значком якоря, видимым, даже если кадр занимает одну позицию дорожки.

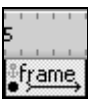


Рис. 12.7. "Якорь", имеющий имя `frame`

Флажок **Scale** нужно включить, если наш анимированный элемент при движении должен менять свои размеры. (Впрочем, этот флажок включен по умолчанию.) Если же его снять, размеры анимированного элемента меняться не будут.

С помощью поля ввода с регулятором **Ease** задается степень замедления движения анимированного элемента в конце анимации. Если мы введем в это поле значение от 1 до 100, элемент сначала будет двигаться быстро, а потом его движение замедлится. Если же, наоборот, мы введем значение от −1 до −100, сначала его движение будет медленным, а потом ускорится. Если нам нужно, чтобы элемент всегда двигался равномерно, мы оставим в этом поле ввода значение по умолчанию — 0.

Раскрывающийся список **Rotate** задает направление вращения анимированного элемента, если, конечно, во время движения он должен вращаться. Всего в этом списке четыре пункта:

- **None** — элемент не будет вращаться вообще, даже если на втором ключевом кадре он был повернут;
- **Auto** — направление вращения элемента выберет сам Flash (значение по умолчанию);

- **CW** — элемент всегда будет вращаться в направлении по часовой стрелке;
- **CCW** — элемент всегда будет вращаться в направлении против часовой стрелки.

Если в списке **Rotate** были выбраны пункты **CW** или **CCW**, то становится доступным поле ввода **times**, расположенное правее этого списка. В нем задается количество дополнительных поворотов, которые сделает анимированный элемент, прежде чем займет свое конечное положение. Значение по умолчанию — 0, т. е. элемент сразу повернется на заданный угол, не совершая никаких дополнительных поворотов.

Если правее кнопки **Edit** (ее мы рассмотрим чуть позже) находится кнопка , это значит, что с нашей анимацией что-то не в порядке. При нажатии этой кнопки на экране появится окно-предупреждение, описывающее проблему, с которой столкнулся Flash при создании анимации.

Остальные элементы управления, показанные на рис. 12.4, мы опишем в следующих главах этой книги.

Дополнительные средства управления движением

Шкатулка сюрпризов, припасенная нам Flash 8, еще далеко не пуста. Очередной сюрприз для аниматоров, припасенный им, — это дополнительные средства для управления движением. С их помощью мы сможем, например, управлять скоростью движения анимированного элемента или насыщенностью его тени на каждом участке его траектории.

Чтобы задать параметры движения, нам будет нужно сначала выделить любой промежуточный кадр в последовательности, задающей это движение. После этого нажмем упомянутую ранее кнопку **Edit** панели **Properties**. На экране появится диалоговое окно **Custom Ease In / Ease Out** (рис. 12.8).

Рассмотрение этого окна мы начнем с раскрывающегося списка **Property**. Пять его пунктов позволят нам задать, чем именно мы собираемся управлять:

- **Position** — местонахождением анимированного элемента на траектории его движения (если он должен двигаться);
- **Rotation** — углом его поворота (если он должен вращаться);
- **Scale** — его размерами (если они должны меняться);
- **Color** — примененными к анимированному элементу цветовыми эффектами;
- **Filters** — примененными к анимированному элементу фильтрами.

Если включен флажок **Use one setting for all properties** (а он включен по умолчанию), то все заданные нами настройки будут применены сразу ко всему: и к местонахождению, и к углу поворота, и к размерам, и к цветовому эффекту, и к фильтрам.

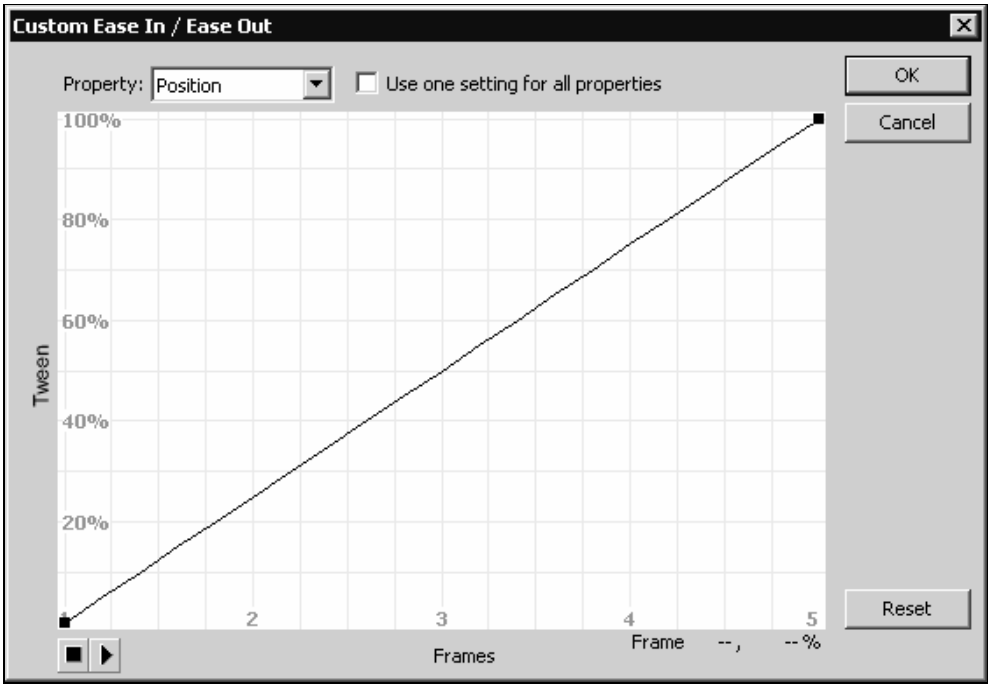


Рис. 12.8. Диалоговое окно **Custom Ease In / Ease Out**

Бóльшую часть окна **Custom Ease In / Ease Out** занимает *график*, задающий изменение выбранного нами в списке **Property** параметра с течением времени. Горизонтальная ось этого графика проградуирована в кадрах анимации и задает время. Вертикальная же его ось проградуирована в процентах и задает степень приложения заданного нами для анимированного элемента эффекта. Так, если наш элемент должен двигаться, то эта шкала задает его местоположение на траектории движения, а если он должен менять свою прозрачность — то степень изменения прозрачности.

Как хорошо видно на рис. 12.8, изначально график представляет собой прямую линию, начинающуюся в точке с координатами (0,0), а заканчивающуюся в точке (<*последний кадр анимации*>,100%). Это значит, что по умолчанию все заданные нами изменения эффекта, примененного к анимированному элементу, происходят плавно. Чтобы задать другие измене-

ния, мы должны будем изменить сам график, скажем, превратив его линию в кривую.

Как это сделать? Исключительно просто! Помещаем курсор мыши на нужную точку графика, нажимаем левую кнопку и буксируем мышь, а когда график примет нужную форму, отпускаем левую кнопку. В точке, где мы щелкнули мышью, появится точка искривления и касательная, которой мы можем пользоваться для изменения формы графика. В общем, делается это так же, как и при работе с инструментом "перо" (*см. главу 3*).

Мы можем выделять точки искривления, щелкая по ним мышью. При этом на выделенной точке искривления появится касательная, а под графиком Flash выведет номер кадра и значение выбранного нами в списке **Property** параметра, относящиеся к этой точке.

Еще мы можем перетаскивать мышью созданные ранее точки искривления, чтобы исправить график. А чтобы удалить ненужную точку искривления, мы должны будем щелкнуть по ней, удерживая нажатой клавишу <Alt>.

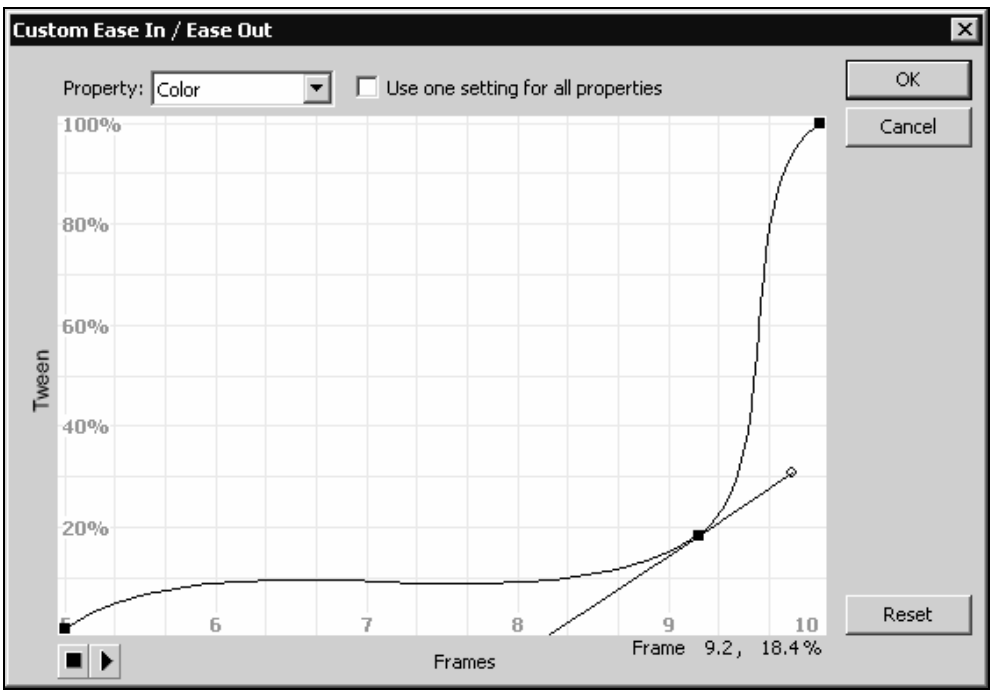


Рис. 12.9. График, задающий ускоренное исчезновение названия нашего фильма

Давайте, например, заставим название фильма пропадать с экрана сначала медленно, а потом все быстрее и быстрее. Для этого сначала выберем в раскрывающемся списке **Property** пункт **Color**, а потом исправим график таким образом, чтобы он походил на показанный на рис. 12.9.

Задав нужный нам график, нажмем кнопку **OK**, чтобы применить его последовательности кадров, промежуточный кадр которой мы выделили ранее. Чтобы отказаться от управления движением, мы нажмем кнопку **Cancel**.

Осталось сказать немного. В левом нижнем углу окна **Custom Ease In / Ease Out** находится группа из двух кнопок, схожих с кнопками пульта (см. рис. 11.5). Правая кнопка позволит нам запустить воспроизведение дорожки, кадр которой мы выделили. Правда, нужно иметь в виду, что при этом будет воспроизведена вся дорожка, а не только последовательность кадров, которой принадлежит выделенный кадр. А левая кнопка останавливает воспроизведение.

Кнопка **Reset** окна **Custom Ease In / Ease Out** позволит нам вернуться к изначальному графику изменения выбранного в списке **Property** параметра (т. е. к прямой линии).

Еще раз об "оборванной" анимации

Заканчивая разговор о движении, укажем на "подводные камни", которые могут встретиться на пути начинающего Flash-аниматора. И, разумеется, поговорим о том, как эти "камни" обойти.

Как мы помним, "оборванная" анимация получается в том случае, когда Flash не знает, как закончить созданную нами анимацию. Такое может случиться по двум причинам:

- если мы забыли создать второй ключевой кадр анимации или случайно его удалили;
- если мы удалили во втором ключевом кадре анимации с рабочего листа анимируемый экземпляр.

В любом из указанных случаев Flash отобразит "оборванную" анимацию так, как показано на рис. 12.2. И, конечно же, он не сможет ее нормально воспроизвести.

Какой имеется выход из этого положения?

Если в нашей анимации нет второго ключевого кадра, мы должны будем его создать. После этого выделим его и поместим на рабочий лист экземпляр того образца, который находится в первом ключевом кадре. Это важно: если анимируемые экземпляры, находящиеся в разных ключевых кадрах, порождены разными образцами, Flash не сможет создать движение, и анимация так и останется "оборванной".

Во втором случае выход также очевиден: удаляем все с рабочего листа во втором ключевом кадре и помещаем на него экземпляр соответствующего образца. После этого Flash должен успешно создать нужное движение.

И последнее. Если мы выполнили операцию смены образца, на котором основан экземпляр, находящийся в первом ключевом кадре, то должны будем сделать то же самое и с экземпляром, который находится во втором ключевом кадре. Иначе наша анимация станет "оборванной".

Морфинг

Итак, с движением мы разобрались. Пора приступать к морфингу.

Как делается морфинг? В сущности, так же, как движение. Мы создаем ключевые кадры — начало и конец анимации, наполняем их содержимым и при желании задаем параметры анимации. А остальное, как говорится, дело техники.

Создание морфинга

Создание морфинга мы будем изучать на другом примере. Давайте создадим новый документ Flash и, воспользовавшись инструментом "эллипс", нарисуем на рабочем листе что?.. правильно, эллипс. Заливку для него создавать не будем, чтобы изменение его формы было нагляднее.

Содержимое первого ключевого кадра нашей новой анимации готово. Теперь нужно создать второй ключевой кадр и — самое главное — дать понять Flash, чего мы от него хотим.

Создадим новый ключевой кадр, скажем, в позиции № 10 временно́й шкалы. Flash автоматически скопирует во второй ключевой кадр содержимое первого. А теперь превратим наш эллипс в подобие математического знака "бесконечность", как показано на рис. 12.10.

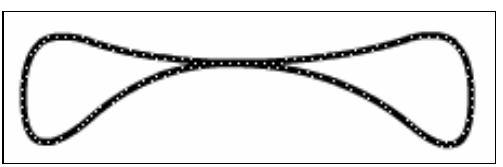


Рис. 12.10. Результат изменения формы эллипса в результате морфинга

Ну, теперь можно создать морфинг. Выделим любой из подкадров первого ключевого кадра будущей анимации и выберем в раскрывающемся списке **Tween** панели **Properties** пункт **Shape**. Все — анимация создана!

Созданный нами морфинг обозначается на временно́й шкале почти так же, как движение (см. рис. 12.1), — в виде тонкой черной стрелки, пересекающей кадры. Только сами кадры будут уже не голубыми, а светло-зелеными.

Остается проверить результат. Нажмем клавишу <Enter>, предварительно убрав фокус ввода из панели **Properties**. Нарисованный нами эллипс прямо на глазах сплющивается и превращается в знак бесконечности.

Но, как говорил один из персонажей мультфильма "История игрушек", бесконечность — не предел. Мы можем заставить наш эллипс во время изменения формы менять еще и свой цвет. Для этого выделим последний ключевой кадр нашего морфинга, выделим все содержимое рабочего листа и изменим его цвет, воспользовавшись селектором цвета линий панели **Properties**. И посмотрим, что получится. Эх-х-х, красота!

Чтобы удалить созданный морфинг, мы выделим любой промежуточный кадр задающей этот морфинг последовательности и выберем в раскрывающемся списке **Tween** панели **Properties** пункт **None**.

А теперь мы можем попробовать создать, так сказать, римейк фильма о делении клетки исключительно средствами морфинга. Для этого создадим еще один новый документ Flash. И приступим...

Нарисуем первый ключевой кадр. Как он выглядит, мы знаем. Чтобы нарисовать второй ключевой кадр, воспользуемся одним из трюков Flash, позволяющих разделять изображение на части. (Собственно, мы уже им пользовались.) Сначала выделяем перетаскиванием мыши левую часть нашей "клетки" и относим ее влево, потом выделяем оставшуюся, правую, часть и относим вправо. Осталось только придать обеим половинам "клетки" вид их "родителя" — и второй ключевой кадр готов.

Теперь создаем сам морфинг и проверяем результат. Вот так штука! Наша "клетка" просто разваливается пополам, а вовсе не делится, как это делают настоящие, живые клетки. Да, Flash не так уж и интеллектуален, как мы о нем думали...

Увы, человеческие руки и навыки живого художника не может заменить ни одна, даже самая "интеллектуальная" программа. Даже Flash бессилен сделать что-то, требующее подлинного творчества! Что ж, человеку — человеческое, а компьютеру — компьютерово...

И все же морфинг — неплохая альтернатива покадровой анимации. По крайней мере, простые фильмы можно создавать только с его помощью. А вы как считаете?

Параметры морфинга

Если мы выделим любой промежуточный кадр морфинга, в панели **Properties** появится группа элементов управления (рис. 12.11). С их помощью мы можем задать некоторые параметры созданного нами морфинга.

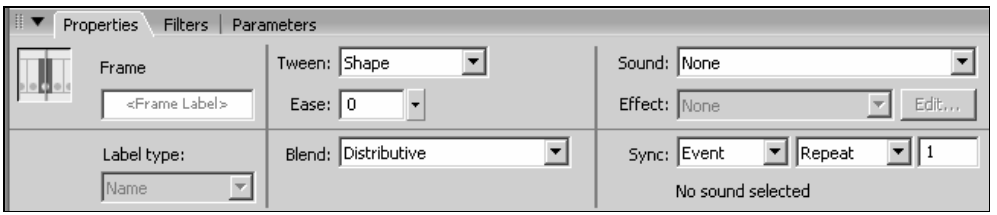


Рис. 12.11. Панель **Properties** после выделения промежуточного кадра морфинга

Большинство этих элементов управления нам уже знакомо. Мы рассматривали их, когда говорили о параметрах движения. Поэтому сейчас речь пойдет только об элементах управления, задающих специфичные для морфинга параметры.

А такой элемент всего один — раскрывающийся список **Blend**. Он позволяет сделать фигуры, расположенные на промежуточных кадрах анимации более или менее угловатыми. В этом списке всего два пункта:

- **Distributive** — промежуточные фигуры будут по возможности сглаженными (значение по умолчанию);
- **Angular** — промежуточные фигуры могут быть угловатыми, если изначальный фрагмент содержал углы и прямые линии.

Нужно иметь, однако, в виду, что заданные с помощью списка **Blend** настройки справедливы только тогда, когда контур, подвергаемый морфингу, содержит углы и прямые линии. В других случаях эти настройки не действуют.

Также заметим, что для морфинга доступно гораздо меньше параметров, чем для движения. Так, мы не можем управлять морфингом с помощью диалогового окна **Custom Ease In / Ease Out**, как управляли движением.

Маркеры морфинга и их использование

Когда мы создаем морфинг в среде Flash, эта своевольная программа часто творит, что хочет, наплевав на наши творческие замыслы. Однако есть средство все-таки заставить Flash делать то, чего хотим мы, а не он. Это *маркеры морфинга*.

Маркер морфинга позволяет нам выбрать точку на анимируемом фрагменте графики и жестко задать ее местоположение в начале и конце анимации. Таких маркеров мы можем создать до 26 штук включительно. Каждый маркер при этом помечается маленькой латинской буквой от `a` до `z`.

Давайте испробуем маркеры морфинга в действии. Воспользуемся для этого предыдущим фильмом, основанным на морфинге и демонстрирующим превращение эллипса в математический знак бесконечности. Пусть наш эллипс в процессе превращения еще и переворачивается — вот такое у нас возникло желание.

Выделим первый ключевой кадр анимации и создадим в нем первый маркер морфинга. Для этого выберем пункт **Add Shape Hint** подменю **Shape** меню **Modify** или нажмем комбинацию клавиш <Ctrl>+<Shift>+<H>. На рабочем листе появится небольшой красный кружок с латинской буквой в центре; поскольку это первый маркер, который мы создали, в его центре будет стоять буква `a`.

> ***Внимание!***
>
> По умолчанию маркеры морфинга не отображаются на рабочем листе. Чтобы вывести их на экран, включите пункт-выключатель **Show Shape Hints** меню **View** или нажмите комбинацию клавиш <Ctrl>+<Alt>+<H>.

Итак, первый маркер морфинга мы создали. Но он не привязан ни к одной точке на контуре нашей фигуры. Давайте исправим это. "Захватим" только что созданный маркер мышью и "приклеим" его к нужной точке контура, скажем, к правой стороне изначального эллипса. Все — мы задали первую контролируемую нами точку контура (рис. 12.12).

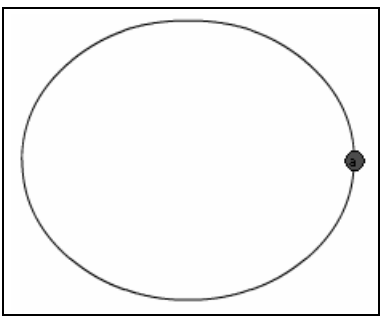


Рис. 12.12. Маркер морфинга, "приклеенный" к контролируемой им точке контура фигуры (небольшой кружок справа)

Далее создадим еще один маркер морфинга (он будет обозначен буквой `b`) и "приклеим" его к другой контролируемой точке эллипса.

Пусть это будет левая его сторона. У нас должно получиться то, что показано на рис. 12.13.

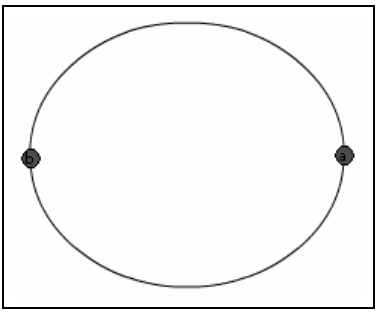


Рис. 12.13. Два маркера морфинга, "приклеенные" к точкам контура фигуры

Итак, начальное положение маркеров морфинга мы задали. Теперь нужно задать их конечное положение.

Выделим второй ключевой кадр анимации. Мы увидим, что на рабочем листе находятся два точно таких же маркера, помеченные буквами `a` и `b`. Собственно, это те же самые маркеры, что были созданы на первом ключевом кадре, и они задают конечное положение отмеченных нами точек контура фигуры.

Поскольку наш эллипс во время превращения в знак бесконечности должен переворачиваться, мы "приклеим" их на те же точки конечной фигуры, только в обратном порядке. То есть маркер `a` должен быть помещен на левую сторону фигуры, которой станет эллипс, а маркер `b` — на правую сторону.

После того как мы "приклеим" маркеры морфинга на втором ключевом кадре на свои места, они станут зелеными. Если же теперь мы вернемся к первому ключевому кадру, то увидим, что "тамошние" маркеры стали желтыми. Это значит, что контролируемые нами точки заданы и на первом, и на втором ключевых кадрах анимации.

Запустим теперь готовую анимацию. Что ж, маркеры помогли — помеченные ими точки оказались как раз на нужных нам местах. Сравним рис. 12.14 и рис. 12.15: на первом при включенном режиме наложения показана наша анимация, выполненная без использования маркеров морфинга, на втором — с использованием этих маркеров. Результат, как говорится, налицо.

Мы можем добавлять новые маркеры морфинга, выбрав пункт **Add Shape Hint** в подменю **Shape** меню **Modify** или нажав комбинацию клавиш <Ctrl>+<Shift>+<H>. Также можно щелкнуть на уже существующем марке-

ре правой кнопкой мыши и выбрать в появившемся на экране контекстном меню пункт **Add Hint**.

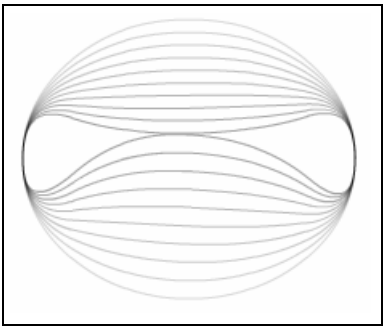


Рис. 12.14. Анимация, созданная без использования маркеров морфинга

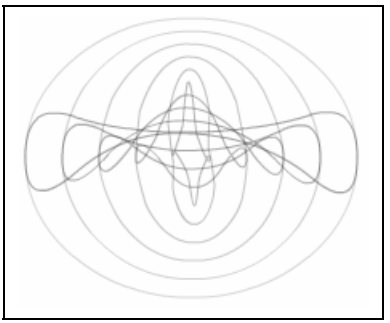


Рис. 12.15. Анимация, созданная с использованием маркеров морфинга. Хорошо заметно, как эллипс переворачивается

На заметку

При добавлении новых маркеров старайтесь, чтобы в первом ключевом кадре они располагались в порядке по часовой стрелке. Впоследствии вам будет проще отследить, откуда и куда они перемещались.

Мы можем перемещать маркеры мышью, "приклеивая" их к разным местам контура. (Собственно, мы уже этим занимались.) А удалить ненужный маркер можно двумя разными способами. Во-первых, мы можем щелкнуть на нужном маркере правой кнопкой мыши и выбрать в контекстном меню пункт **Remove Hint**. Во-вторых, мы можем просто "утащить" ненужный маркер прочь с рабочего листа.

Быстро удалить сразу все созданные нами маркеры морфинга можно выбором либо пункта **Remove All Hints** в контекстном меню любого маркера, либо одноименного пункта подменю **Shape** меню **Modify**.

Мы можем временно скрыть на рабочем листе все маркеры морфинга. Для этого достаточно отключить пункт-выключатель **Show Shape Hints** меню **View** или нажать комбинацию клавиш <Ctrl>+<Alt>+<H>. Еще можно отключить пункт-выключатель **Show Hints** в контекстном меню любого маркера. (Непонятно только, как его там снова включить. Еще одна загадка Flash...)

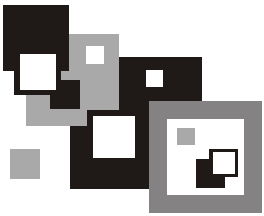
Кстати, используя маркеры морфинга, можно попытаться создать римейк фильма "Деление клетки", используя только морфинг. Скорее всего, нам придется задать несколько маркеров морфинга и поэкспериментировать с ними, помещая в разные точки контуров и внимательно следя за результатами. Получится, конечно, коряво, но хотя бы попробовать стоит. Пусть это будет вашим домашним заданием.

Что дальше?

Ну вот, с трансформационной анимацией мы покончили. Мы узнали, что она может быть двух видов — движение и морфинг, — и выяснили, как она создается. И даже немного попрактиковались.

Однако разговор о создании анимации еще далеко не закончен. Мы научились создавать только самые простые фильмы. А этого, согласитесь, недостаточно. Поэтому следующие две главы будут посвящены более сложным вопросам: а именно, составной анимации и использованию для ее создания слоев.

Глава 13



Создание составной анимации

Ух, как хорошо мы поанимировали в *главах 11* и *12*! Все у нас теперь движется, плавает, ползает и совершает прочие телодвижения, подчиняясь нашему замыслу. Воистину, Flash 8 — король анимации.

А теперь на время перестанем им восторгаться и представим себе такую ситуацию. Мы создали анимированный прямоугольник, который должен двигаться из левого нижнего в правый верхний угол рабочего листа и при этом поворачиваться на некоторый угол (рис. 13.1). Для этого мы переместили его с помощью инструмента "стрелка" и повернули с помощью инструмента "трансформатор" (об этих инструментах *см. главы 3* и *4*). Все работает, все замечательно и все довольны.

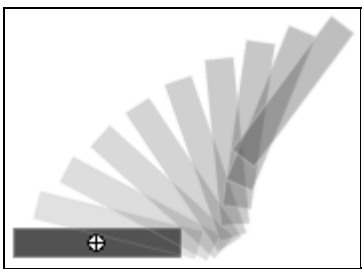


Рис. 13.1. Изначальная анимация с движущимся и вращающимся прямоугольником

Но злой рок в лице начальства не дремлет. Мы получили задание переделать эту анимацию так, чтобы наш прямоугольник не только двигался и не только вращался, но и при движении разваливался надвое. Средствами движения этого достичь очень трудно, да и морфинг в этом случае не бог весть какой помощник. Что делать?

Конечно, можно обратиться к покадровой анимации. Там этой проблемы нет: мы сможем нарисовать все, что угодно, и Flash корректно это воспроизведет. Но каждый кадр такой анимации придется рисовать вручную. А начальство ждет готовую работу через пять минут...

В такой ситуации проще всего будет создать составную анимацию. Вот о ней-то мы сейчас и поговорим.

Дело в том, что Flash позволяет создавать не только статичные образцы. Образец может содержать анимацию (так называемый *анимированный образец*). Делается это очень просто: мы запускаем один из режимов правки образца и прямо в нем создаем любую нужную нам анимацию, покадровую или трансформационную.

А когда мы создадим на основе анимированного образца экземпляр (*анимированный экземпляр*) и анимируем еще и его, две анимации — экземпляра и образца — будут воспроизводиться одновременно и независимо друг от друга. Таким образом мы сможем создать сколь угодно сложную анимацию. И решить проблему разваливающегося на части прямоугольника, о которой говорилось выше. И пусть только начальство попробует что-нибудь сказать!!!

Анимированные образцы могут, в свою очередь, содержать другие образцы, в том числе тоже анимированные. Представляете, какие это открывает возможности!

Такая анимация, включающая анимированные экземпляры, называется *составной* или *многоуровневой*. Каждый анимированный экземпляр, вложенный в другой экземпляр или помещенный в сам документ, станет *уровнем анимации*. Анимация же, содержащаяся в анимированном экземпляре (точнее, в образце, на основе которого от создан), называется *внутренней*.

Именно о составной анимации мы поговорим в этой главе. А также познакомимся с новым для нас типом образцов — образцом-клипом — и выясним, чем же образцы-клипы могут быть нам полезны.

Ну что же, кажется, с теорией покончено. Переходим к практике!

Создание составной анимации

Давайте создадим составную анимацию, демонстрирующую распад прямоугольника на части. Но сначала создадим сам "предмет распада", если еще этого не сделали. Нарисуем на рабочем листе прямоугольник и преобразуем его в обычный графический образец. Не мудрствуя лукаво, назовем его `Прямоугольник`.

Теперь нам нужно заставить его разделяться надвое. Пусть этот процесс длится на протяжении 10 кадров. Откроем только что созданный образец

`Прямоугольник` в режиме правки (неважно, каком именно) и приступим к созданию внутренней анимации.

Воспользуемся для этого хорошо знакомыми нам по *главе 12* средствами. Создадим второй ключевой кадр на десятой позиции шкалы кадров. Выделим только что созданный второй ключевой кадр, выберем инструмент "стрелка", выделим перетаскиванием правую половину прямоугольника и перетащим ее немного вверх и вправо (или еще куда-нибудь). Теперь нам останется выделить любой из подкадров первого, растянутого, ключевого кадра и выбрать в списке **Tween** панели **Properties** пункт **Shape**. У нас должна получиться картина, показанная на рис. 13.2.

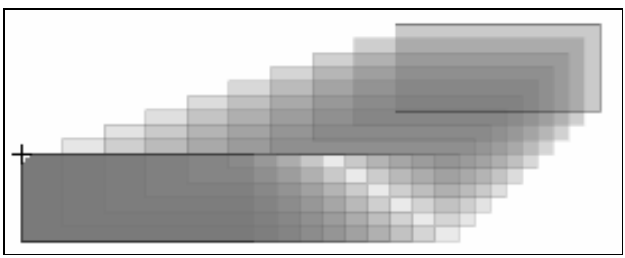


Рис. 13.2. Готовый анимированный образец `Прямоугольник` (включен режим наложения)

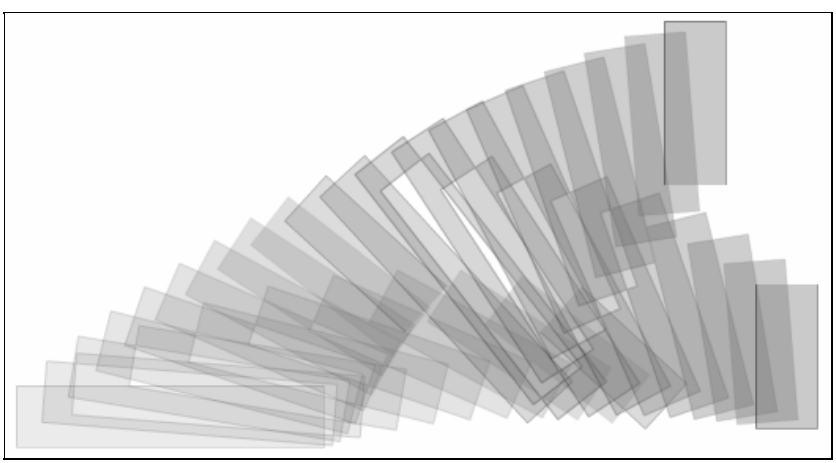


Рис. 13.3. Готовая анимация с движущимся, вращающимся и распадающимся на части прямоугольником (включен режим наложения)

Теперь создадим анимацию в самом документе (*внешнюю анимацию*). Для этого переключимся в режим правки документа. Поместим в нижнем левом

углу рабочего листа экземпляр образца `Прямоугольник`. Создадим анимацию, перемещающую его в верхний правый угол рабочего листа, продолжительностью как минимум вдвое большей, чем продолжительность анимации самого образца (в нашем случае — 20 кадров). И запустим готовую анимацию. Результат показан на рис. 13.3.

Мы только что создали составную двухуровневую анимацию, т. е. сложную анимацию, состоящую их двух простых, вложенных друг в друга. Все средства, использованные нами для этого, стандартны и знакомы нам по предыдущим главам этой книги.

Использование образцов-клипов

А теперь давайте сделаем вот что. Запустим воспроизведение созданной нами составной анимации в отдельном окне, выбрав пункт **Test Movie** меню **Control**. После этого в окне просмотра фильма отключим пункт-выключатель **Loop** в меню **Control**, чтобы убрать зацикливание, включенное по умолчанию. Затем дождемся, пока наш прямоугольник докувыркается до финиша, и посмотрим, что будет с ним дальше.

А ничего не будет. Анимация воспроизведется до конца и остановится. Причем это произойдет с обеими анимациями: и с внешней, и с внутренней. Всесильный режиссер Flash скажет в конце "Стоп!" им обоим.

Выходит, что внутренняя анимация, созданная нами в графическом образце, не такая уж и независимая. Она воспроизводится только вместе с внешней, до тех пор пока не кончится последовательность кадров последней. Такая внутренняя анимация называется *зависимой*.

Конечно, в рассмотренном случае такая зависимость удобна. Но что делать, если нам нужно, чтобы внутренняя анимация воспроизводилась в любом случае, даже после прекращения внешней анимации или даже при ее отсутствии? Может ли Flash и в этом случае нам помочь?

Конечно! Для решения проблемы нам всего-навсего нужно будет использовать образцы-клипы.

Внутренняя анимация, созданная в образце-клипе, будет воспроизводиться в любом случае. Даже если внешняя анимация закончилась. Даже если мы вообще не создали внешнюю анимацию! Такая анимация называется *независимой*.

Единственное "но": Flash не воспроизводит внутреннюю анимацию клипов в окне документа. То есть если мы запустим анимацию, содержащую анимированный клип, анимация клипа не воспроизведется. Чтобы просмотреть ее, нам придется запустить воспроизведение фильма в отдельном окне, выбрав в меню **Control** пункты **Test Movie** или **Test Scene**.

Давайте откроем созданный нами ранее документ, содержащий анимированный прямоугольник, и попробуем применить к делу полученные знания. Включим пункт-выключатель **Library** меню **Window**, чтобы вывести на экране панель **Library**. Выделим в списке этой панели образец `Прямоугольник` и выберем в подменю **Type** контекстного меню пункт **Movie Clip**. Все — мы поменяли тип этого образца на "образец-клип".

Теперь нужно выделить первый ключевой кадр внешней анимации, выделить на рабочем листе экземпляр этого образца и выбрать в раскрывающемся списке, находящемся в верхнем левом углу панели **Properties**, пункт **Movie Clip**. Таким образом мы поменяем тип экземпляра этого образца на "образец-клип". (Как мы помним из *главы 8*, типы образца и порожденного им экземпляра могут быть разными.) И не забудем сделать то же самое во втором ключевом кадре анимации.

Теперь нажмем клавишу <Enter>, чтобы запустить воспроизведение готовой анимации. Как говорилось ранее, прямоугольник просто переместится на новое место и повернется; его собственная анимация — "распад" прямоугольника на две части — воспроизведена не будет. А вот если мы запустим ее воспроизведение в отдельном окне и отключим пункт-выключатель **Loop** в меню **Control** окна просмотра, чтобы убрать зацикливание, то увидим результаты наших трудов во всей красе. Внешняя анимация дойдет до конца, но внутренняя все равно будет воспроизводиться — прямоугольник будет разваливаться на части, пока мы не закроем окно просмотра.

Образцы-клипы очень часто применяются для создания такой вот независимой анимации. (У них есть и другие сферы применения, но мы рассмотрим их в *части IV* этой книги, когда будем говорить о программировании в среде Flash.) Если же нам не нужна независимая анимация, лучше использовать обычные графические образцы, благо в результирующем файле Shockwave/Flash они занимают заметно меньше места, чем образцы-клипы.

Параметры внутренней анимации

Теперь давайте выделим на рабочем листе экземпляр нашего анимированного образца и посмотрим, что нам предлагает панель **Properties**. Если это экземпляр графического образца, эта панель примет вид, показанный на рис. 13.4.

В данный момент нас интересуют только находящаяся в центре панели **Properties** группа, содержащая раскрывающийся список, и поле ввода **First**. Все остальные элементы управления, показанные на рис. 13.4, нам уже знакомы.

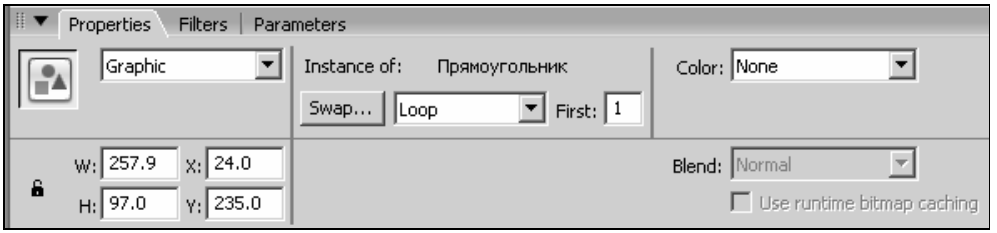


Рис. 13.4. Панель **Properties** после выделения экземпляра анимированного графического образца

Раскрывающийся список данной группы позволяет указать, какая часть внутренней анимации образца, на основе которого был создан выделенный экземпляр, будет воспроизведена как часть общей анимации. Список содержит три пункта:

- **Loop** — внутренняя анимация образца будет воспроизводиться бесконечно (зациклится). Причем воспроизведение ее начнется с кадра, номер которого введен в поле ввода **First**. Этот пункт списка выбран по умолчанию;
- **Play Once** — внутренняя анимация образца будет воспроизведена только один раз, после чего остановится, даже если внешняя анимация еще не закончилась. При этом она будет воспроизведена, начиная с кадра, номер которого введен в поле ввода **First**. Пример такой анимации показан на рис. 13.5;

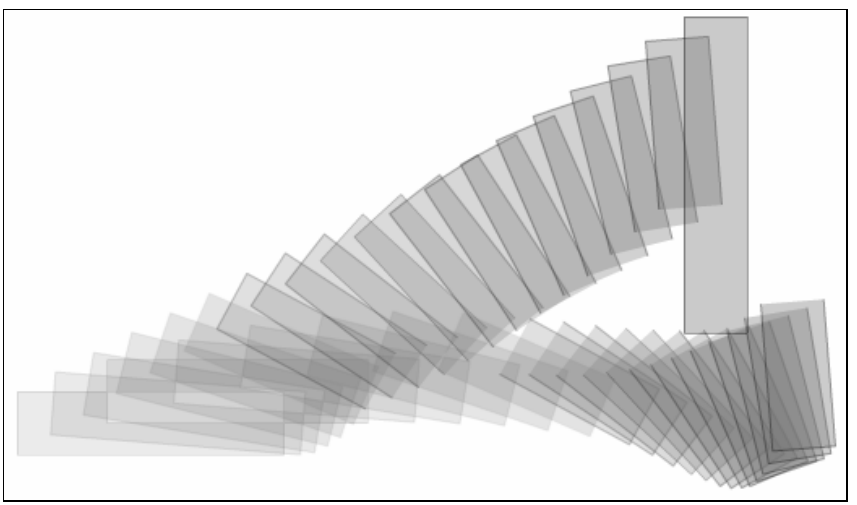


Рис. 13.5. Внутренняя анимация экземпляра воспроизводится один раз, начиная с четвертого кадра

- **Single Frame** — внутренняя анимация образца застынет на кадре, номер которого введен в поле ввода **First**. Пример такой анимации можно увидеть на рис. 13.6.

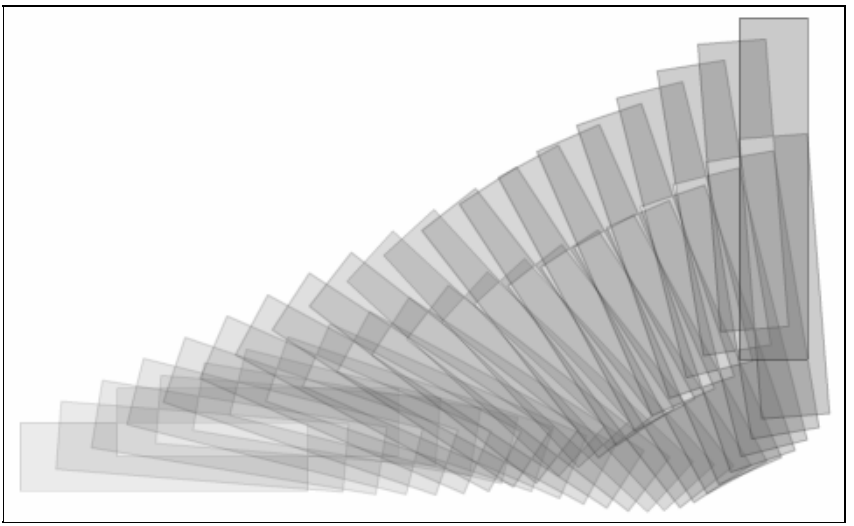


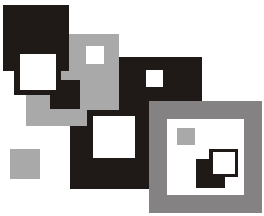
Рис. 13.6. Внутренняя анимация экземпляра "застыла" на четвертом кадре

В случае анимированного клипа все эти элементы управления недоступны. Как уже говорилось выше, внутренняя анимация клипов воспроизводится независимо от внешней анимации, и управлять ей придется другими способами — с помощью сценариев ActionScript (*см. часть IV*).

Что дальше?

На этом разговор о составной анимации можно закончить. Тема исчерпана.

Но это не означает, что мы узнали все о сложной анимации. В следующей главе будет рассказано о том, как использовать для создания сложной анимации слои. Мы узнаем, как анимировать элементы, находящиеся на разных слоях, а также выясним все о специальных слоях и о том, как они могут нам помочь.



Глава 14

Использование слоев для создания анимации

Здесь мы продолжим начатый в *главе 5* разговор о слоях и их использовании. А поскольку данная часть полностью посвящена анимации, то и разговор пойдет о создании анимированных изображений с помощью слоев.

Как мы помним из *главы 12*, трансформационная анимация может быть применена только к экземпляру, находящемуся на своем собственном слое. На первый взгляд, это серьезное ограничение: невозможно анимировать сразу два или более экземпляров. Но его легко преодолеть, если разнести эти экземпляры по нескольким слоям и анимировать их по отдельности (*многослойная анимация*).

Кроме того, для создания сложной анимации Flash предоставляет две разновидности так называемых *специальных слоев*, имеющих особое назначение. Это направляющие и маскирующие слои, и мы их также рассмотрим.

С помощью направляющего слоя можно заставить какой-либо экземпляр двигаться по сложной траектории. Делается это так: мы создаем слой, рисуем на нем линию, представляющую нужную нам траекторию, привязываем к ней этот экземпляр и создаем анимацию. Остальное, как и во многих других случаях, берет на себя Flash.

Маскирующий слой используется для того, чтобы частично скрыть лежащее под ним изображение, показав только его небольшую часть. Анимировав содержимое этого слоя, можно создать "световое пятно", "ползающее" по изображению и "высвечивающее" разные его части.

Многослойная анимация

Ну что ж, начнем с самого простого — с многослойной анимации, т. е. анимации, состоящей из нескольких анимированных элементов, находящихся на разных слоях.

Создание многослойной анимации

Создадим новый документ Flash и нарисуем на рабочем листе круг. Преобразуем его в графический образец и дадим ему имя `Круг`. В результате этих действий мы получим на рабочем листе экземпляр этого образца.

Теперь можно приступать к созданию анимации. Давайте заставим наш экземпляр двигаться по горизонтали, слева направо. Продолжительность анимации пусть составляет 10 кадров. Проверим готовую анимацию в действии и убедимся, что она работает.

Теперь давайте добавим на рабочий лист еще один экземпляр образца `Круг` и заставим его двигаться по вертикали, сверху вниз. Для этого сначала создадим новый слой, а уже на него и поместим новый экземпляр. (О создании слоев *см. главу 5*.) Анимируем его и запустим воспроизведение готовой анимации. Вот и все.

Таким образом мы можем создавать сколько угодно слоев, помещать на них экземпляры и анимировать их. (Главное только — не переусердствовать, не то у зрителей зарябит в глазах. Но это так, шутка...)

Создание фоновых изображений

Очень часто бывает необходимо, чтобы анимированный элемент изображения располагался поверх какого-либо *фонового изображения* (*фона*). В частности, такие фоны часто применяются в художественных фильмах.

Как сделать фон во Flash? Очень просто!

Откроем документ, созданный нами чуть раньше, когда мы учились делать многослойную анимацию. Создадим в нем новый слой и поместим его ниже тех двух слоев, на которых находятся уже созданные нами анимированные элементы. Импортируем какое-либо большое растровое изображение и поместим его на этот слой.

А теперь очень важный момент. Единственный ключевой кадр, содержащий фоновое изображение, нужно растянуть так, чтобы его длина сравнялась с длиной анимации, которая его использует. Это нужно, чтобы созданный нами фон присутствовал на всех кадрах анимации.

ВНИМАНИЕ!

Не забывайте о том, что было сказано выше! Ключевой кадр, содержащий фон, должен быть достаточной длины, чтобы покрыть всю использующую этот фон анимацию.

Все! Если мы теперь запустим воспроизведение созданной ранее анимации, наши элементы будут двигаться на красивом графическом фоне. Неплохо, правда?

Разумеется, фоном может служить не только импортированное растровое изображение, но и любая векторная графика, нарисованная в самом Flash. Фон также может быть анимированным, но это уже, на взгляд автора, перебор.

Ускорение отображения фрагментов фона

Как говорилось выше, фон может содержать векторную графику, нарисованную средствами самого Flash. Также фон может включать экземпляры — графические или клипы. Вот о клипах (в смысле, экземплярах образцов-клипов), используемых для создания фона, мы и поговорим.

Клипы — замечательная штука. Мы можем применить к ним фильтры (*см. главу 9*), чтобы наше кино выглядело эффектнее. Но в таком случае возникнет одна проблема, о которой нужно поговорить.

Если фильм включает в себя множество анимированных фрагментов, проигрывателю Flash может не хватить процессорных ресурсов, чтобы отобразить анимацию плавно, без рывков. В самом деле, при выводе каждого кадра фильма ему придется извлечь из файла Shockwave/Flash описания каждого анимированного образца, вычислить его координаты и нарисовать его в той позиции, в которой он должен быть в данный момент. А ведь анимированные элементы могут перекрывать друг друга, быть полупрозрачными, обрабатываться с помощью фильтров и даже управляться сценариями ActionScript. И, кроме проигрывателя Flash, на компьютере могут выполняться другие программы, тоже требующие "внимания" со стороны процессора. Так что ничего нет удивительного в том, что воспроизводимая анимация будет постоянно дергаться, а то и застывать на несколько секунд.

Но Flash 8 предоставляет способ хоть как-то решить эту проблему. Он позволяет включить *кэширование* выбранных клипов в оперативной памяти компьютера. То есть проигрыватель Flash после считывания образцов-клипов из файла Shockwave/Flash, где хранится фильм, и помещения их экземпляров на рабочий лист поместит их также в оперативную память. Тогда, если ему понадобится нарисовать какой-либо клип в другой позиции, он возьмет его именно из оперативной памяти вместо того, чтобы снова лезть в файл Shockwave/Flash.

Преимущество у этого подхода одно, и оно очевидно — ускорение воспроизведения анимации. А вот недостатков несколько:

- кэшироваться в оперативной памяти могут только клипы, не содержащие встроенной анимации (хотя такие клипы могут быть анимированы);
- при кэшировании множества клипов сильно возрастает потребность в системных ресурсах;
- проигрыватель Flash в любой момент может посчитать, что клип не должен кэшироваться (например, если он слишком велик или если не хватает оперативной памяти).

Включить кэширование клипа очень просто. Выделяем на рабочем листе нужный клип и обращаемся к панели **Properties**. В нижней ее части, прямо под раскрывающимся списком наложений, находится отключенный по умолчанию флажок **Use runtime bitmap caching**. Включаем этот флажок — и клип будет кэшироваться при выводе.

> ***НА ЗАМЕТКУ***
>
> Слово "bitmap" в подписи к этому флажку говорит о том, что клип кэшируется в растровом виде. Именно за счет этого и достигается ускорение его вывода на экран.

Использование специальных слоев

Наш разговор о многослойной анимации получился коротким. В самом деле, о чем там особо рассказывать!.. Все и так ясно.

А теперь поговорим о специальных слоях. Это слои-направляющие и маскирующие слои.

Слои-направляющие

При создании анимации часто бывает нужно, чтобы какой-либо элемент совершал не прямолинейное перемещение, а двигался по достаточно сложному пути (или, как говорят аниматоры, *траектории*). Эта траектория может быть кривой или ломаной линией, окружностью или сложным контуром какой-то фигуры.

Средствами, изученными в *главе 12*, добиться этого невозможно или слишком трудоемко. Поэтому специально для таких случаев Flash предлагает особую разновидность специальных слоев — *слои-направляющие*. И сейчас мы выясним, как их создать и как использовать.

Создадим новый документ Flash, поместим на рабочий лист прямоугольник и преобразуем его в образец по имени `Прямоугольник`. После этого создадим простейшую анимацию, перемещающую экземпляр этого образца слева направо по рабочему листу. Как это сделать, мы уже знаем.

Теперь нужно заставить этот прямоугольник двигаться по какой-либо траектории. Для этого мы создадим слой-направляющую, нарисуем на нем траекторию и привяжем к ней наш прямоугольник.

Чтобы создать слой-направляющую и привязать к нему анимированный элемент, сначала выделим в списке слоев панели **Timeline** тот, что содержит анимированный элемент. Далее выберем пункт **Motion Guide** подменю **Timeline** меню **Insert**. Также можно выбрать пункт **Add Motion Guide** контекстного меню выделенного слоя. Созданный нами слой-направляющая появится над выделенным слоем (рис. 14.1) и будет иметь имя вида `Guide: <Имя выделенного слоя>`. Также он будет помечен особой пиктограммой.

Рис. 14.1. Слой-направляющая в списке слоев панели **Timeline**

Теперь выделим слой-направляющую и нарисуем в нем нужную нам траекторию. Для этого можно пользоваться инструментами "карандаш", "перо", "линия", "прямоугольник", "эллипс" и "кисть". Нужно только проследить, чтобы слой-направляющая не содержал больше никакой графики, кроме траектории.

Осталось привязать наш анимированный элемент к нарисованной траектории. Для этого выделим первый ключевой кадр нашей анимации и включим флажок **Snap** панели **Timeline**. Точка вращения анимированного элемента после этого должна "приклеиться" к траектории. Переместим анимированный элемент в самое начало траектории, если он почему-либо оказался в ее середине. Наконец, выделим второй ключевой кадр анимации и переместим анимированный элемент в конец траектории.

Все! Можно проверить готовую анимацию (рис. 14.2).

Единственное, что портит картину — сама линия траектории. Нам нужно скрыть содержащий ее слой-направляющую, для чего достаточно щелкнуть в соответствующей строке списка слоев по колонке, помеченной значком "глаз".

В панели **Properties** есть еще один полезный нам флажок — **Orient to path**. Если его включить, Flash будет автоматически ориентировать анимирован-

ный элемент по линии траектории. На рис. 14.3 показана анимация, использующая автоматическое ориентирование по траектории.

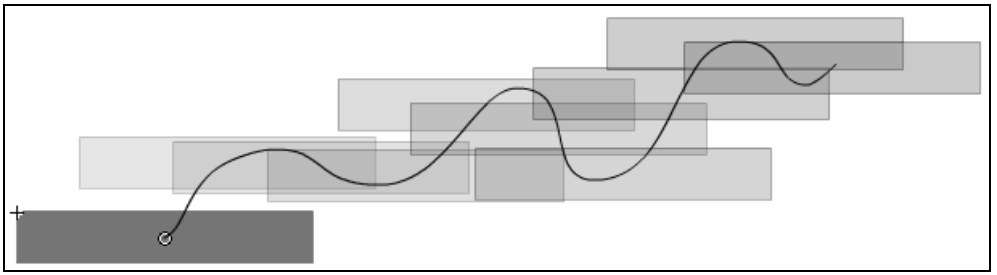


Рис. 14.2. Готовая анимация, использующая слой-направляющую (включен режим наложения)

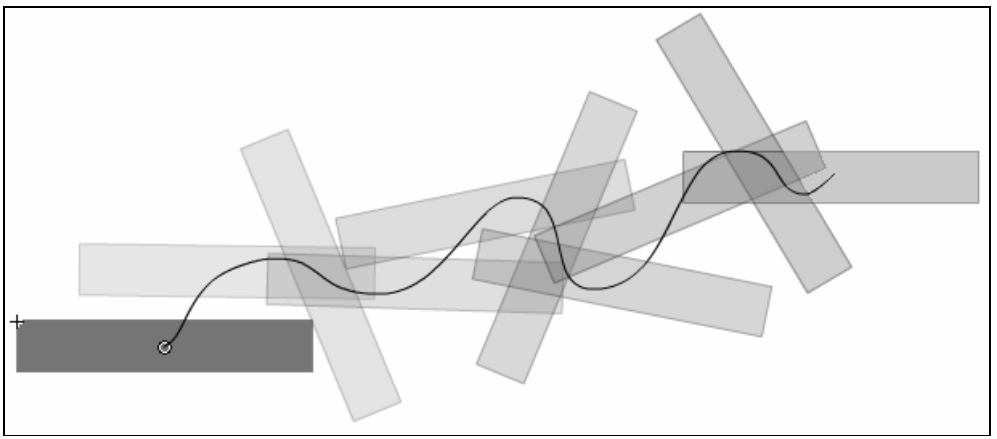


Рис. 14.3. Готовая анимация, использующая слой-направляющую и автоматическое ориентирование по траектории

Создать слой-направляющую можно и другим способом. Создадим обычный слой, выделим его, нарисуем на нем траекторию и выберем пункт-выключатель **Guide** контекстного меню этого слоя. Еще можно выбрать пункт **Properties** в контекстном меню, включить в появившемся на экране диалоговом окне **Layer Properties** (см. рис. 5.2) переключатель **Guide** и нажать кнопку **OK**. Обратим внимание, что слой-направляющая, к которому не привязан ни один слой, обозначен значком в виде молотка.

Привязать к вновь созданному слою-направляющей слой с анимацией можно тремя способами. Во-первых, можно просто перетащить слой с анимацией и "бросить" под слоем-направляющей. Во-вторых, можно выделить ка-

кой-либо слой, уже привязанный к слою-направляющей, и создать новый слой. (Как мы знаем, новый слой помещается прямо над выделенным — значит, в нашем случае он также будет привязан к слою-направляющей.) В-третьих, можно выбрать пункт **Properties** в контекстном меню, включить в появившемся на экране диалоговом окне **Layer Properties** переключатель **Guided** и нажать кнопку **OK**.

Кстати, мы можем привязать к одному слою-направляющей несколько слоев с анимацией. При этом все анимированные элементы будут двигаться по одной траектории.

"Отвязать" же слой с анимацией от слоя-направляющей можно также тремя способами. Первый способ: перетащить слой с анимацией так, чтобы он оказался выше слоя-направляющей. Второй способ: выделить слой с анимацией, выбрать пункт **Properties** в контекстном меню, включить в появившемся на экране диалоговом окне **Layer Properties** переключатель **Normal** и нажать кнопку **OK**. Третий способ: отключить флажки **Snap** и **Orient to path**, "отклеив" тем самым анимированный элемент от линии траектории.

Чтобы превратить слой-направляющую в обычный слой, нужно отключить пункт-выключатель **Guide** в контекстном меню этого слоя. Можно также вызвать на экран диалоговое окно **Layer Properties**, включить переключатель **Normal** и не забыть нажать кнопку **OK**.

Маскирующие слои

Маскирующий слой позволяет скрыть содержимое лежащего под ним *маскируемого слоя*, показав только его часть. Он содержит особое изображение, называемое *маской*; она задает размеры и форму фрагмента маскируемого слоя, который будет виден. Мы можем представить себе, что маска — это "дырка" в маскирующем слое, сквозь которую видно содержимое маскируемого слоя. Эта "дырка" может быть анимирована, что позволит получить весьма интересные и забавные эффекты, вроде "светового пятна", высвечивающего разные фрагменты маскируемого слоя.

Как же создать и использовать маскирующий слой? Очень просто — даже проще, чем слой-направляющую.

Создадим новый документ Flash. По умолчанию он будет содержать единственный слой — его-то мы и превратим в маскируемый. Нарисуем на нем что-нибудь, а еще лучше — импортируем любое достаточно большое растровое изображение и поместим его на этот слой.

Так... Маскируемый слой у нас готов. А как насчет маскирующего?

Создадим над маскируемым слоем новый пустой слой. В нем нарисуем изображение, которое станет нашей маской.

При этом будем иметь в виду следующее:

- любой контур и любая заливка станут прозрачными частями маски;
- пустое пространство на рабочем листе станет непрозрачной частью маски;
- цвета и стили линий и заливок будут проигнорированы. Так что не стоит увлекаться красивыми градиентными и графическими цветами — никакого эффекта это не даст, а результирующий файл Shockwave/Flash заметно увеличится в размерах.

Пусть наша маска имеет вид круга. Преобразуем ее в образец, который так и назовем — `Маска`. Заставим ее перемещаться, скажем, из левого верхнего угла рабочего листа в правый нижний. Растянем единственный ключевой кадр слоя, содержащего маскируемое изображение, так, чтобы он "покрыл" всю анимацию. Теперь все готово к созданию маскирующего слоя.

Выделим в списке слой, содержащий маску, и выберем пункт **Mask** в контекстном меню этого слоя. Также можно выбрать пункт **Properties** в контекстном меню, включить в появившемся на экране диалоговом окне **Layer Properties** переключатель **Mask** и нажать кнопку **OK**. После всех произведенных действий список слоев будет иметь вид, показанный на рис. 14.4.

Рис. 14.4. Маскирующий и маскируемый слои в списке слоев панели **Timeline**

Как видим, кроме всего прочего, и маскирующий, и маскируемый слои были заблокированы. Блокировка обоих слоев — важное условие их нормальной работы во Flash. Если мы не заблокируем хотя бы один слой, маскирование работать не будет. Если же нам понадобится что-то изменить, то мы сначала разблокируем нужный слой, изменим, что хотим, а потом не забудем заблокировать его снова.

Все — наша анимация готова! Можно запускать ее воспроизведение.

Мы можем маскировать одним маскирующим слоем несколько слоев, содержащих графику. Для этого мы создадим нужные слои, наполним их графикой, перетащим их и "бросим" под маскирующим слоем. Мы также можем выбрать пункт **Properties** контекстного меню слоя, который нужно сделать маскируемым, включим в появившемся на экране диалоговом окне **Layer Properties** переключатель **Masked** и нажмем кнопку **OK**.

"Вынуть" же слой с графикой из-под маскирующего слоя можно двумя способами. Первый способ: перетащить слой с маскируемой графикой так, что-

бы он оказался выше маскирующего слоя. Второй способ: выделить слой с графикой или маскирующий слой, выбрать пункт **Properties** в контекстном меню, включить в появившемся на экране диалоговом окне **Layer Properties** переключатель **Normal** и нажать кнопку **OK**. В результате маскирующий или маскируемый слой станет обычным слоем.

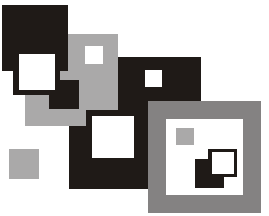
К сожалению, мы не можем маскировать уже маскированный слой, т. е. накладывать маску на маску. Flash не предоставляет для этого стандартных средств. Для этого нам придется использовать составную анимацию, описанную в *главе 13*. Последовательность действий в этом случае будет такая:

1. Создаем графический образец или образец-клип с внутренней анимацией, использующей маскирующий и маскируемый слои. Маскирующий слой этого образца станет маской первого уровня, а маскируемый слой будет содержать собственно маскируемую графику.
2. Возвращаемся в сам документ. Помещаем экземпляр созданного на первом шаге образца-клипа в слой. Создаем еще один слой и делаем его маскирующим — он станет маской второго уровня.
3. Создаем всю анимацию и смотрим готовый фильм.

Что дальше?

На этом разговор о применении слоев в непростом труде Flash-аниматора закончен.

Наш опыт работы во Flash пока еще очень невелик. И нам совсем не помешала бы сейчас любая помощь. Хотя бы помощь от самого Flash. Давайте выясним, может ли он помочь нам хотя бы создать за нас пару-тройку "красивостей" вроде анимированной тени. И займемся мы этим в следующей главе.



Глава 15

Использование эффектов и преобразований

В *главе 9* мы изучили фильтры и наложения, которые можно использовать для создания различного рода специальных эффектов (тени, размытия и пр.). Все они призваны помочь на первых порах начинающим пользователям. Опытные Flash-художники, в принципе, могут создать то же самое и вручную, но для этого им придется мобилизовать свои знания и опыт. А вот у начинающих пользователей Flash знаний и опыта как раз и не хватает.

В этой главе мы изучим эффекты и преобразования Flash. *Эффекты* аналогичны уже знакомым фильтрам и позволяют, в частности, заставить фрагмент графики "вырастать" на глазах или "взрываться". *Преобразования* — это различные элементарные анимации, которые могут быть применены к любому фрагменту изображения и позволяют в результате анимации переместить его, повернуть, изменить размеры, задать цветовой эффект или сделать все это одновременно.

Есть одно ограничение — мы можем применить к выделенному фрагменту только один эффект или преобразование. Так что комбинировать эффекты (преобразования), как фильтры, мы не сможем.

Эффекты и преобразования Flash могут быть применены к любому фрагменту нашего изображения. Даже если выделить на рабочем листе обычную фигуру, не являющуюся ни группой, ни текстовым блоком, ни экземпляром, мы сможем применить к ней любой эффект или преобразование. Flash сам преобразует этот фрагмент в образец и создаст на его основе экземпляр.

Но будут ли нам действительно полезны все эти эффекты и преобразования? В конце концов, мы уже имеем кое-какой опыт работы с Flash и даже успели создать несколько фильмов. Кто знает... Так или иначе, Flash нам их предлагает, и грех отказаться от такого предложения.

Простейшие эффекты

Начнем мы с создания самых простых эффектов. Всего их два: клонирование выделенного экземпляра по ячейкам воображаемой сетки и его анимированное копирование.

Клонирование по сетке ("Copy to Grid")

Клонирование по сетке — самый простой из эффектов, который мы можем применить к выделенному фрагменту изображения. Данный эффект даже не анимирован.

Чтобы применить этот эффект к выделенному фрагменту изображения, нужно выбрать пункт **Copy to Grid** подменю **Assistants** подменю **Timeline Effects** меню **Insert**. (Вот как далеко он спрятался!) На экране появится диалоговое окно **Copy to Grid** (рис. 15.1).

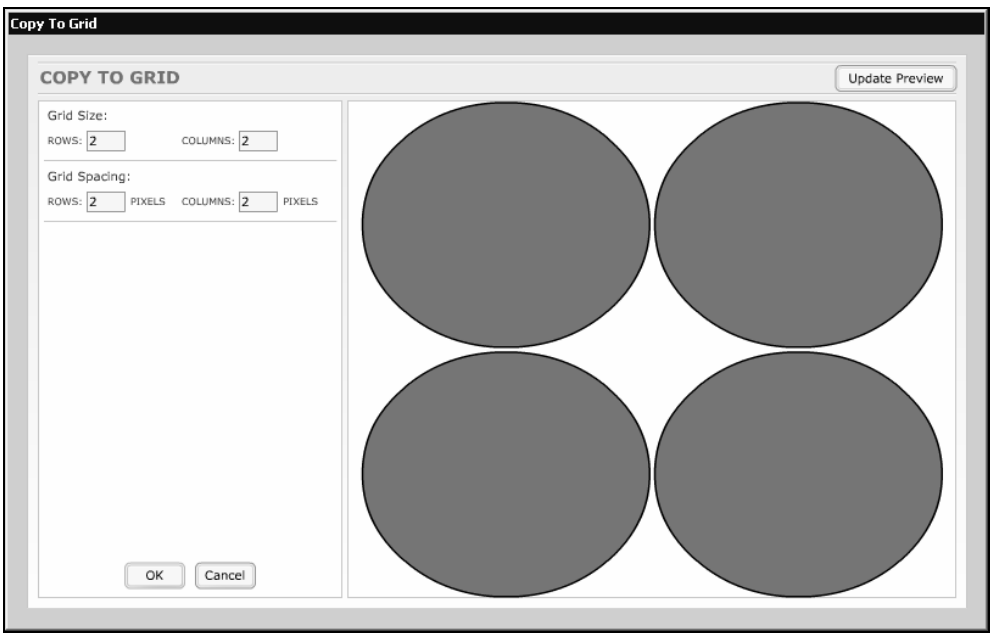


Рис. 15.1. Диалоговое окно **Copy to Grid**

Интересно это окно тем, что выполнено средствами самого Flash. Да-да, пакет Flash включает в себя, в том числе, и модули, написанные на нем самом!

Группа элементов управления **Grid Size** позволяет задать количество строк и столбцов в воображаемой сетке, где будут помещены "клоны" выделенного нами фрагмента. Количество строк указывается в поле ввода **Rows**, а количество столбцов — в поле ввода **Columns**.

Группа элементов управления **Grid Spacing** служит для задания расстояния между ячейками этой воображаемой сетки. Расстояние по вертикали задается в поле ввода **Rows**, а расстояние по горизонтали — в поле ввода **Columns**. Все эти расстояния измеряются в пикселах.

Всю правую половину окна **Copy to Grid** занимает большая область предварительного просмотра. Так что мы всегда сможем увидеть, что случится с нашим фрагментом после задания различных параметров эффекта. Кнопка **Update Preview** заставляет Flash обновить содержимое этой области; рекомендуется нажимать эту кнопку после изменения любого параметра.

Задав нужные параметры, следует нажать кнопку **OK**. Кнопка **Cancel** позволит нам отказаться от создания эффекта.

Анимированное копирование ("Distributed Duplicate")

Этот эффект позволит создать несколько копий выделенного нами фрагмента изображения с разными эффектами. Так, различные копии фрагмента могут иметь разный цвет или быть повернутыми на разные углы. Такое копирование может быть анимированным; в этом случае копии будут создаваться прямо на глазах зрителя.

Чтобы применить этот эффект к выделенному фрагменту изображения, нужно выбрать пункт **Distributed Duplicate** подменю **Assistants** подменю **Timeline Effects** меню **Insert**. На экране появится диалоговое окно **Distributed Duplicate** (рис. 15.2), также выполненное средствами Flash.

В поле ввода **Number of Copies** задается количество копий выделенного нами фрагмента.

Группа элементов управления **Offset Distance** служит для задания расстояния между копиями. Расстояние по горизонтали вводится в поле ввода **X**, расстояние по вертикали — в поле ввода **Y**. Все эти величины вводятся в пикселах.

Поле ввода **Offset Rotation** служит для задания угла поворота копий в градусах.

В поле ввода **Offset Start Frame** задается количество кадров, в течение которых происходит создание каждой копии. Если ввести в это поле значение 0,

эффект будет статичным. Если же ввести количество кадров, отличное от нуля, эффект будет анимированным.

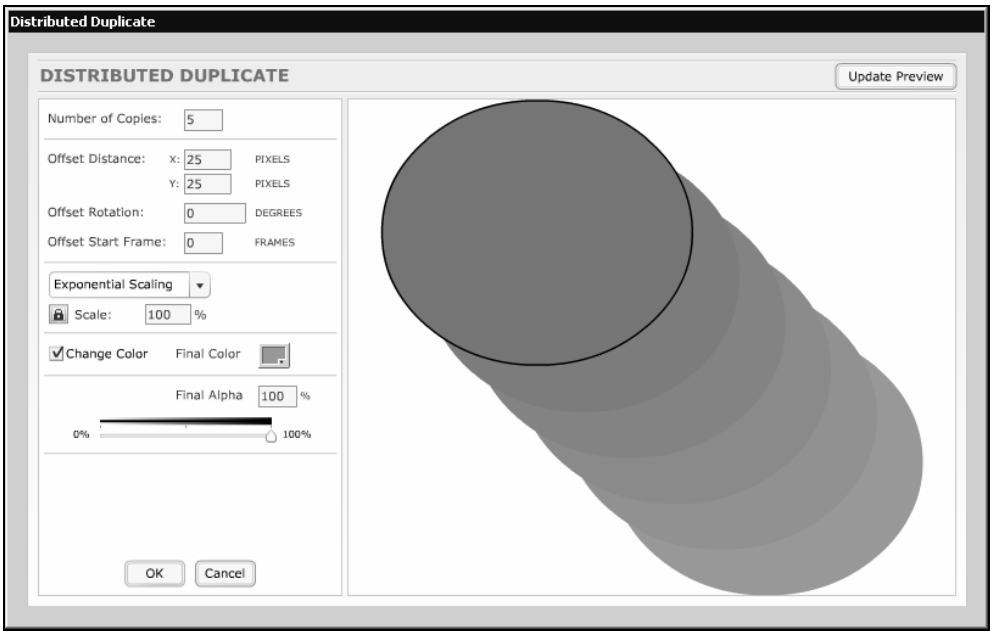


Рис. 15.2. Диалоговое окно **Distributed Duplicate**

Раскрывающийся список ниже поля ввода **Offset Start Frame** служит для задания метода масштабирования копий. В нем находятся два пункта: **Exponential Scaling** (экспоненциальное масштабирование) и **Linear Scaling** (линейное масштабирование).

Ниже этого списка находится группа элементов управления, предназначенная для задания степени масштабирования новых копий. Если небольшая кнопка содержит изображение закрытого замка, то в расположенное правее поле ввода можно будет ввести значение масштаба, которое будет использоваться для масштабирования копий и по горизонтали, и по вертикали. Если же эта кнопка содержит изображение открытого замка, значение масштаба можно будет задать отдельно для горизонтального (поле ввода **X**) и вертикального (поле ввода **Y**) направления. Переключить кнопку-"замок" из открытого состояния в закрытое и наоборот можно щелчком по ней.

Флажок **Change Color**, будучи включенным, предписывает Flash плавно изменять цвет получающихся копий выделенного фрагмента. Конечный цвет можно задать с помощью селектора цвета **Final Color**.

Поле ввода **Final Alpha** и расположенный под ним регулятор служат для задания конечного уровня прозрачности последней копии. Все предыдущие копии будут иметь уровни прозрачности, постепенно приближающиеся к заданному.

Опять же, правую половину окна **Distributed Duplicate** занимает область предварительного просмотра и кнопка **Update Preview**, о которой не следует забывать.

Задав нужные параметры, мы нажмем кнопку **OK**. Кнопка **Cancel** позволит нам отказаться от создания эффекта.

Более сложные эффекты

Закончив с эффектами простейшими, перейдем к более сложным и более, если так можно сказать, эффектным. Их побольше — целых четыре.

Анимированное размытие ("Blur")

Этот эффект заставит выделенный нами фрагмент постепенно "размываться". Как гласит само его название, он всегда создается анимированным.

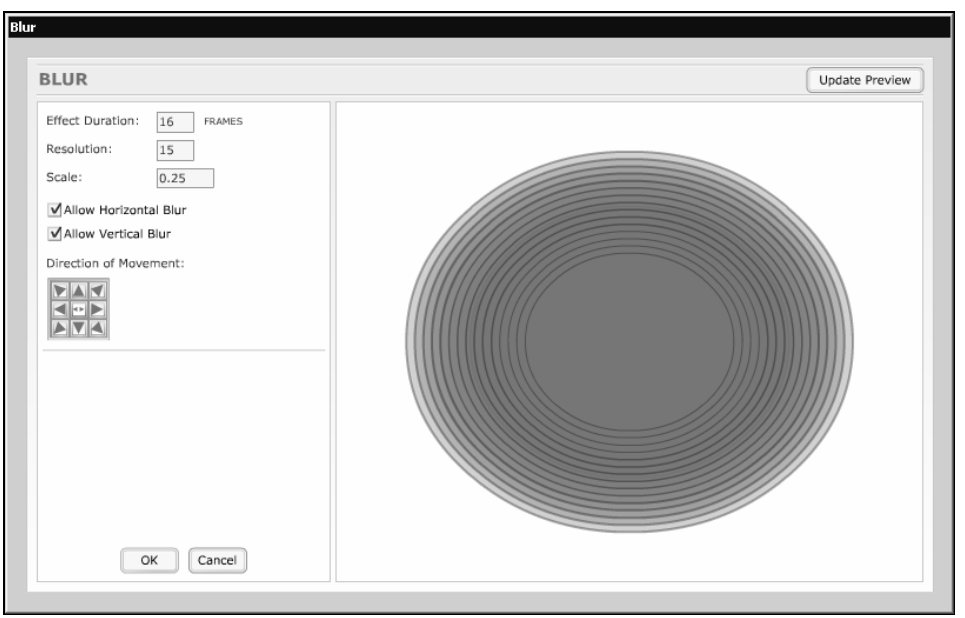


Рис. 15.3. Диалоговое окно **Blur**

Чтобы применить этот эффект к выделенному фрагменту изображения, нужно выбрать пункт **Blur** подменю **Effects** подменю **Timeline Effects** меню **Insert**. На экране появится диалоговое окно **Blur** (рис. 15.3).

В поле ввода **Effect Duration** задается продолжительность эффекта в кадрах.

Поле ввода **Resolution** служит для задания количества "размытий" (копий выделенного фрагмента, создаваемых для имитации его размытия). А в поле ввода **Scale** вводится относительный масштаб этих копий.

Флажки **Allow Horizontal Blur** и **Allow Vertical Blur** служат для включения и отключения размытия по горизонтали и вертикали соответственно.

Набор кнопок **Direction of Movement** задает направление, в котором будет "размываться" выделенный фрагмент. Каждая кнопка этого набора задает направление, в котором указывает находящаяся на ней стрелка. Центральная же кнопка заставляет "размываться" во все стороны.

Правую половину окна **Distributed Duplicate** занимают область предварительного просмотра и кнопка **Update Preview**.

Задав нужные параметры, мы нажмем кнопку **OK**. Кнопка **Cancel** позволит нам отказаться от создания эффекта.

Отображение тени ("Drop Shadow")

С помощью этого эффекта мы можем добавить к выделенному фрагменту тень. Правда, она будет статичной, не анимированной.

Чтобы применить этот эффект к выделенному фрагменту изображения, нужно выбрать пункт **Drop Shadow** подменю **Effects** подменю **Timeline Effects** меню **Insert**. На экране появится диалоговое окно **Drop Shadow** (рис. 15.4).

Селектор цвета **Color** задает цвет создаваемой тени. А поле ввода **Alpha Transparency** и расположенный под ним регулятор служат для задания уровня прозрачности тени.

Группа элементов управления **Shadow Offset** служит для задания смещения тени относительно выделенного фрагмента. Смещение по горизонтали задается в поле ввода **X**, а смещение по вертикали — в поле ввода **Y**. Обе эти величины задаются в пикселах.

Задав нужные параметры, мы нажмем кнопку **OK**. Кнопка **Cancel** позволит нам отказаться от создания эффекта. Об области предварительного просмотра и кнопке **Update Preview** упоминать уже, вероятно, излишне.

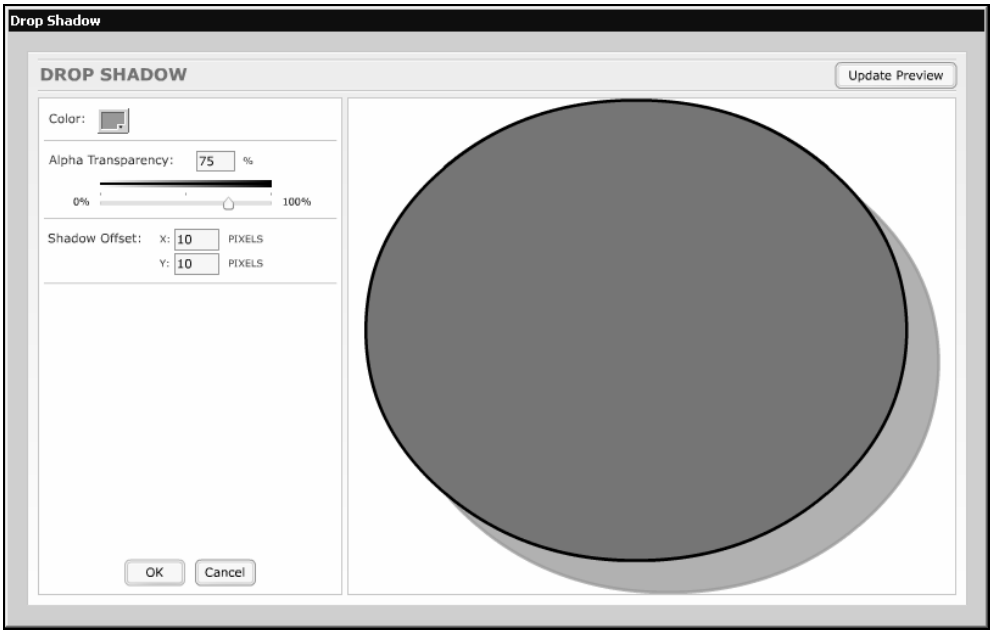


Рис. 15.4. Диалоговое окно **Drop Shadow**

Расширение ("Expand")

Этот эффект заставляет выделенный фрагмент изображения постепенно "вырастать" или "съеживаться", в зависимости от заданных нами параметров.

Чтобы применить этот эффект к выделенному фрагменту изображения, нужно выбрать пункт **Expand** подменю **Effects** подменю **Timeline Effects** меню **Insert**. На экране появится диалоговое окно **Expand** (рис. 15.5).

В поле ввода **Effect Duration** задается продолжительность эффекта в кадрах.

Расположенная ниже группа переключателей позволит нам задать, что именно будет происходить с выделенным фрагментом. Всего в этой группе три переключателя.

- **Expand** — выделенный фрагмент будет "вырастать".
- **Squeeze** — выделенный фрагмент будет "съеживаться".
- **Both** — выделенный фрагмент сначала "вырастет", а потом "съежится" до первоначальных размеров.

Уже знакомый нам по другим окнам набор кнопок **Direction of Movement** служит для указания изначального направления, в котором выделенный фрагмент будет "вырастать" или "съеживаться". Каждая кнопка этого набора задает направление, в котором указывает находящаяся на ней стрелка; реально активны только две кнопки, задающие движение по горизонтали, влево и вправо. Что касается центральной кнопки, то она, судя по всему, никакого влияния на движение фрагмента не оказывает.

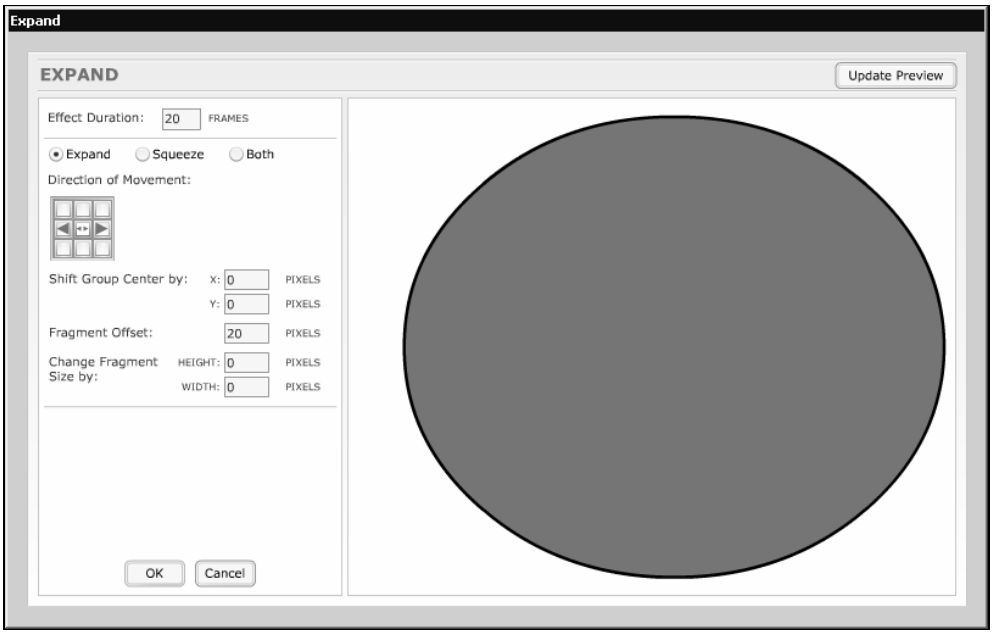


Рис. 15.5. Диалоговое окно **Expand**

Группа элементов управления **Shift Group Center by** задает дополнительное смещение выделенного фрагмента при его движении. Дополнительное смещение по горизонтали задается в поле ввода **X**, а смещение по вертикали — в поле ввода **Y**. Обе эти величины задаются в пикселах.

Значение в поле ввода **Fragment Offset** определяет основное смещение выделенного фрагмента при движении.

Группа элементов управления **Change Fragment Size by** предназначена для указания величины, на которую будет изменяться размер выделенного фрагмента на каждом шаге его движения. Величина изменения размеров по горизонтали задается в поле ввода **X**, по вертикали — в поле ввода **Y**. Обе эти величины задаются в пикселах.

Задав нужные параметры и проверив результат в области предварительного просмотра, мы нажмем кнопку **OK**. Кнопка **Cancel** позволит нам отказаться от создания эффекта.

"Взрыв" ("Explode")

С помощью этого впечатляющего эффекта мы можем "взорвать" выделенный нами фрагмент графики.

Чтобы применить этот эффект к выделенному фрагменту, нужно выбрать пункт **Explode** подменю **Effects** подменю **Timeline Effects** меню **Insert**. На экране появится диалоговое окно **Explode** (рис. 15.6).

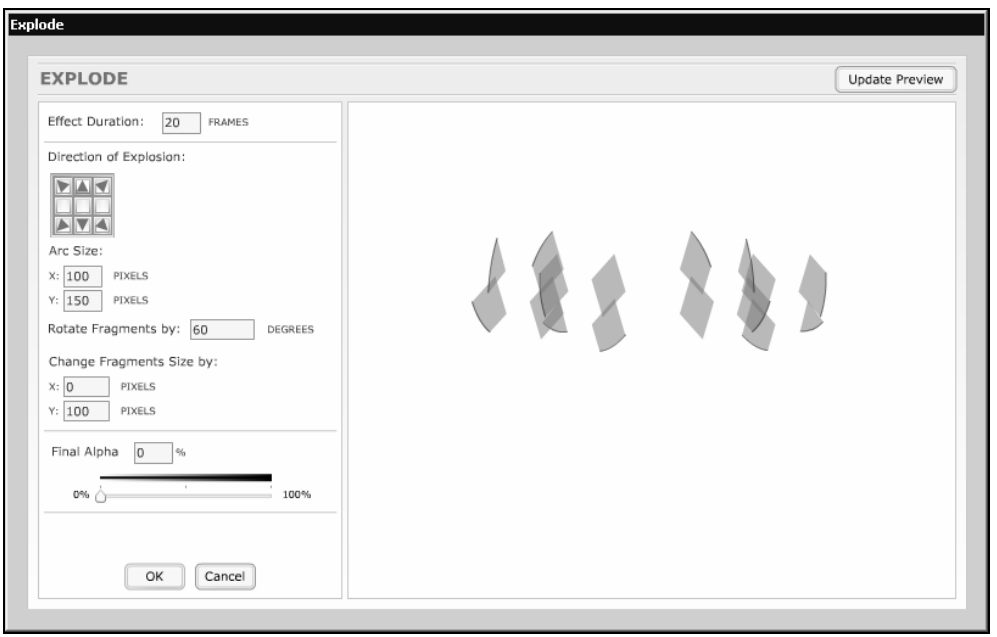


Рис. 15.6. Диалоговое окно **Explode**

В поле ввода **Effect Duration** задается продолжительность эффекта в кадрах.

Набор кнопок **Direction of Explosion** задает направление, в котором будут разлетаться "осколки" "взорванного" фрагмента изображения.

Группа элементов управления **Arc Size** задает размер воображаемой дуги "разлета" "осколков" выделенного фрагмента. Размер дуги по горизонтали задается в поле ввода **X**, размер по вертикали — в поле ввода **Y**. Обе эти величины задаются в пикселах.

Поле ввода **Rotate Fragments by** указывает угол поворота "осколков" при разлетании их в стороны. Этот угол задается в градусах.

Группа элементов управления **Change Fragment Size by** предназначена для указания величины, на которую будет изменяться размер "осколков" выделенного фрагмента на каждом шаге их разлетания. Величина изменения размеров по горизонтали задается в поле ввода **X**, по вертикали — в поле ввода **Y**. Обе эти величины задаются в пикселах.

Поле ввода **Final Alpha** и расположенный под ним регулятор служат для задания конечного уровня прозрачности "осколков". Таким образом, при "разлете" их прозрачность будет меняться от изначальной до указанной в вышеупомянутом поле ввода.

Задав нужные параметры и проверив результат в области предварительного просмотра, мы нажмем кнопку **OK**. Кнопка **Cancel** позволит нам отказаться от создания эффекта.

Преобразования

Преобразований, предусматриваемых Flash, всего два. Однако с их помощью можно задать весьма сложное движение выделенного нами фрагмента графики.

Обычное движение ("Transform")

Это самое обычное движение фрагмента графики, наподобие тех, что мы создавали в *главе 12*. С помощью этого преобразования мы можем менять местоположение, размеры, угол поворота выделенного фрагмента и задавать для него цветовые эффекты.

Чтобы применить это преобразование к выделенному фрагменту, нужно выбрать пункт **Transform** подменю **Transform/Transition** подменю **Timeline Effects** меню **Insert**. На экране появится диалоговое окно **Transform** (рис. 15.7).

В поле ввода **Effect Duration** задается продолжительность движения выделенного фрагмента в кадрах.

Расположенная ниже группа элементов управления позволяет указать, как должны быть заданы новые координаты выделенного фрагмента, и сами эти координаты. Раскрывающийся список, находящийся в этой группе, содержит два пункта:

- **Change Position by** — выделенный элемент должен сместиться по горизонтали на заданное в поле ввода **X**, а по вертикали — на заданное в поле ввода **Y** количество пикселов;

- **Move to Position** — выделенный фрагмент должен переместиться в точку, чьи координаты в пикселах заданы в полях ввода **X** (горизонтальная) и **Y** (вертикальная).

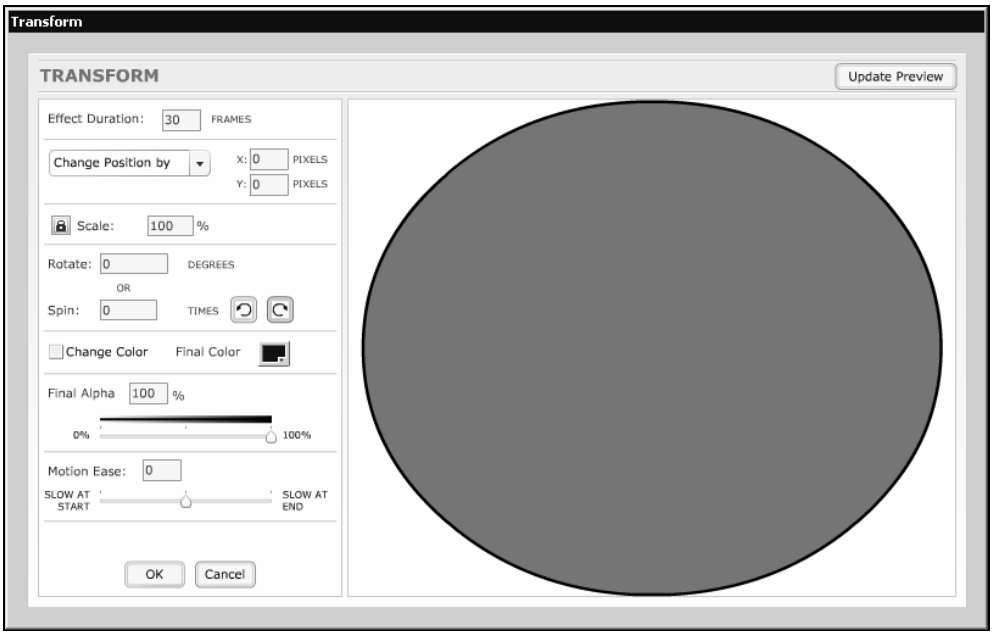


Рис. 15.7. Диалоговое окно **Transform**

Ниже этой группы элементов управления находится другая, предназначенная для задания степени масштабирования выделенного фрагмента при движении. Если небольшая кнопка содержит изображение закрытого замка, то в расположенное правее поле ввода можно будет ввести значение масштаба, которое будет использоваться для масштабирования фрагмента и по горизонтали, и по вертикали. Если же эта кнопка содержит изображение открытого замка, значение масштаба можно будет задать отдельно для горизонтального (поле ввода **X**) и вертикального (поле ввода **Y**) направления. Переключить кнопку-"замок" из открытого состояния в закрытое и наоборот можно щелчком по ней.

Еще ниже находится группа элементов управления, задающая угол поворота выделенного фрагмента или количество оборотов, которые он должен сделать при движении. Угол поворота в градусах задается в поле ввода **Rotate**. А в поле ввода **Spin** указывается количество оборотов. В последнем случае можно задать направление вращения с помощью пары кнопок-переклю-

чателей, расположенных правее поля ввода **Spin**; левая кнопка задает вращение против часовой стрелки, правая — по часовой стрелке.

Флажок **Change Color**, будучи включенным, предписывает Flash изменить цвет выделенного фрагмента в результате движения. Конечный цвет указывается с помощью селектора цвета **Final Color**.

Поле ввода **Final Alpha** и расположенный под ним регулятор служат для задания конечного уровня прозрачности выделенного фрагмента.

В поле ввода **Ease** задается степень замедления движения выделенного фрагмента в конце анимации. Если мы введем в это поле значение от 1 до 100, элемент сначала будет двигаться быстро, а потом его движение замедлится. Если же, наоборот, мы введем значение от −1 до −100, сначала его движение будет медленным, а потом ускорится. Если нам нужно, чтобы элемент всегда двигался равномерно, мы оставим в этом поле ввода значение по умолчанию — 0. Для задания степени замедления мы также можем воспользоваться регулятором, расположенным ниже поля ввода **Ease**.

Задав нужные параметры и проверив результат в области предварительного просмотра, мы нажмем кнопку **OK**. Кнопка **Cancel** позволит нам отказаться от создания этого преобразования.

Проявление ("Transition")

А это преобразование будет для нас более полезным, чем обычное движение. Оно заставляет выделенный фрагмент графики постепенно проявляться на экране (путем уменьшения его прозрачности) или создает эффект опускающегося занавеса.

Чтобы применить это преобразование к выделенному фрагменту, нужно выбрать пункт **Transition** подменю **Transform/Transition** подменю **Timeline Effects** меню **Insert**. На экране появится диалоговое окно **Transition** (рис. 15.8).

В поле ввода **Effect Duration** задается продолжительность проявления выделенного фрагмента в кадрах.

Набор переключателей **Direction** позволяет задать, будет ли выделенный фрагмент постепенно появляться на экране (переключатель **In**) или, наоборот, пропадать (переключатель **Out**).

Флажок **Fade** заставляет выделенный элемент менять свою прозрачность, создавая эффект постепенного появления или исчезновения. А флажок **Wipe** предписывает выделенному элементу создать эффект поднимающегося или опускающегося занавеса.

Если включен флажок **Wipe**, становится доступным расположенный правее него набор кнопок. Он служит для указания направления движения воображаемого занавеса.

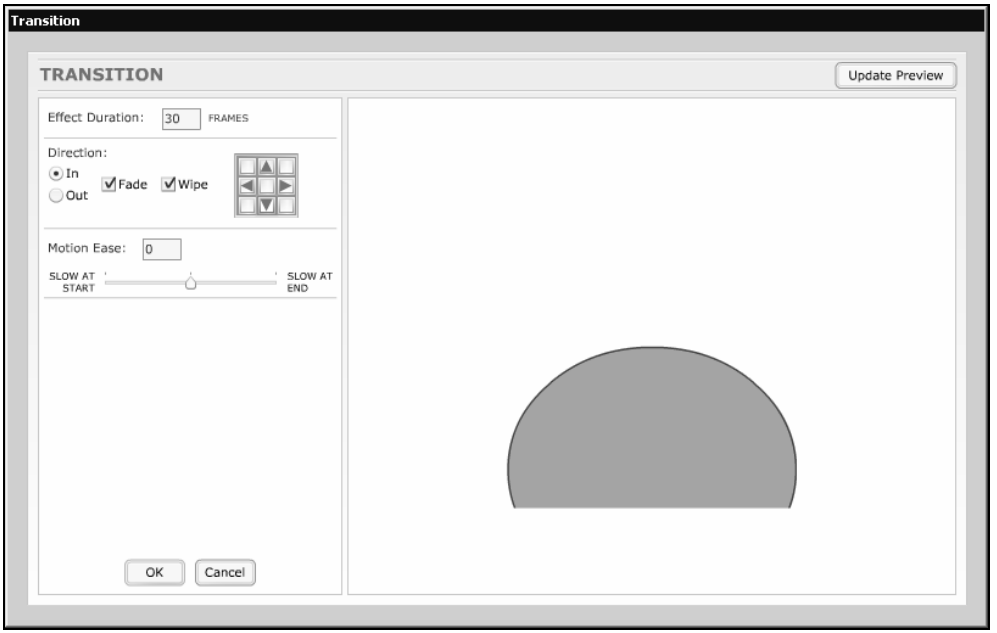


Рис. 15.8. Диалоговое окно **Transition**

В поле ввода **Ease** задается степень замедления движения выделенного фрагмента в конце анимации. Для этого мы также можем воспользоваться расположенным ниже регулятором.

Задав нужные параметры и проверив результат в области предварительного просмотра, мы нажмем кнопку **OK**. Кнопка **Cancel** позволит нам отказаться от создания этого преобразования.

Работа с эффектами и преобразованиями

Вот и все эффекты и преобразования, доступные во Flash 8. Их немного, но при умелом их применении они могут заметно оживить любой фильм.

> ***Внимание!***
>
> Не забывайте, что к одному и тому же фрагменту изображения можно применить только один эффект (преобразование).

Теперь давайте выясним, что мы можем делать с уже созданным для какого-либо выделенного фрагмента графики эффектом или преобразованием.

Прежде всего, мы можем исправить параметры эффекта или преобразования. Для этого достаточно выделить фрагмент, к которому был применен этот эффект или преобразование, и выбрать пункт **Edit Effect** подменю **Timeline Effects** меню **Modify**. На экране появится одно из рассмотренных нами в этой главе диалоговых окон, соответствующих данному эффекту (преобразованию), в котором мы сможем задать новые параметры. После их задания нужно будет не забыть нажать кнопку **OK** этого окна.

Удалить ненужный эффект (преобразование) еще проще. Выделяем на рабочем листе нужный фрагмент графики и выбираем пункт **Remove Effect** подменю **Timeline Effects** меню **Modify**. Эффект (преобразование), примененный нами ранее к этому фрагменту, будет тотчас удален. Возможно, при этом придется подождать, так как Flash будет удалять все созданные им вспомогательные образцы и дорожки анимации.

ВНИМАНИЕ!

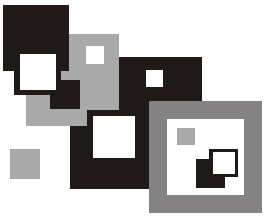
Как говорилось в начале этой главы, если мы решим применить эффект или преобразование к фрагменту графики, не являющемуся ни экземпляром, ни группой, ни текстовым блоком, Flash автоматически создаст на основе этого фрагмента образец и поместит на рабочий лист его экземпляр. Если же мы потом удалим примененный к этому фрагменту эффект или преобразование, он (фрагмент) так и останется экземпляром созданного Flash образца.

Что дальше?

Что ж, об эффектах и преобразования все! Применять их в своих фильмах или нет — дело ваше.

В *главе 10* мы говорили об импорте графики, созданной в других программах (в том числе и растровой), и ее использовании в изображениях Flash. В следующей главе мы займемся импортированием и использованием видеоклипов. И создадим, наверно, самые большие, в смысле занимаемого ими на диске места, документы Flash. Ведь видео — штука весьма увесистая...

Глава 16



Импорт видео

Полноценно использовать в фильмах импортированные видеоклипы стало возможно еще во Flash MX. А уж во Flash 8 средства по работе с импортированным видео достигли и вовсе заоблачных высот. Нет, автор говорит совершенно серьезно!

> ***На заметку***
>
> Надо сказать, что проигрыватели Flash версий 3—5 поддерживали воспроизведение внешних видеоклипов Apple QuickTime. Подробнее об этом мы поговорим далее в этой главе.

Во-первых, мы теперь можем импортировать понравившийся клип в свой фильм, сделав его частью документа Flash, а значит, и результирующего файла Shockwave/Flash. Это самый простой способ позаимствовать чужое творение в качестве основы для собственного.

Во-вторых, мы можем написать несложный сценарий, который во время воспроизведения нашего фильма заставит проигрыватель Flash загрузить и воспроизвести внешний видеоклип. Это полезно, если мы хотим дать зрителю возможность просмотреть какой-либо длинный клип, но не желаем помещать его в свой фильм, чтобы не увеличивать его размеров (ведь зритель может и не заинтересоваться внешним клипом). Конечно, говорить о сценариях ActionScript пока рановато — мы будем изучать их в *части IV*. Но кое-какую подготовительную работу выполним прямо сейчас.

И, наконец, разработчики фирмы Macromedia припасли для нас еще один сюрприз. Встречайте — новый видеоформат Macromedia Flash Video! Быстрый, качественный, компактный и — самое главное! — для воспроизведения его нам не понадобятся никакие дополнительные программы, кроме самого проигрывателя Flash. Да это просто мечта компьютерного киношника!..

Единственное "но"... Помните, о чем мы говорили в конце *главы 10*? Правильно, об авторских правах на импортируемую графику и об их соблюде-

нии. Так вот, все сказанное там справедливо и в случае импортируемого видео. Не забывайте об этом!

Ну что, пора начинать? Но сначала давайте немного поговорим о компьютерном видео и освежим наши знания, полученные в *главе 1*. Заодно узнаем кое-что новое.

Сжатие видео во Flash. Кодеки On2 VP6 и Sorenson Spark

В *главе 1* мы уже говорили о видео. Давайте кратко повторим все, что уже успели узнать и, возможно, уже забыли.

Итак, видеоинформация, хранящаяся в файле, практически всегда сжимается. Иначе и не получится: данные, содержащие видеофильм, имеют огромный размер, и нормально работать с ними практически невозможно. Для сжатия видео в абсолютном большинстве случаев применяется сжатие с потерями, т. е. часть информации при сжатии отбрасывается, отчего размер видеоданных становится значительно меньше. Чем меньше информации было отброшено при сжатии видео, тем выше его качество и тем больше размер файла, в котором оно хранится.

Для работы со сжатым видео требуются две программы — кодер и декодер. Кодер сжимает видео по заданному алгоритму, а декодер распаковывает и отправляет программе, которая его использует (выводит на экран, как Windows Media Player, или помещает в документ, как Flash). И практически всегда пара кодер-декодер представляет собой одну программу — кодек (сокращение от "кодер-декодер").

Трудолюбивое человечество напридумывало преизрядное количество алгоритмов (и соответствующих им кодеков) для сжатия видео. Некоторые из них получили широкую популярность (их мы рассмотрели, опять же, в *главе 1*), другие применяются в очень узкой области, большинство же кануло в Лету. Поэтому сейчас, для того чтобы просмотреть практически любой фильм, нужно иметь на своем компьютере установленными пять кодеков: Indeo, MPEG I, MPEG II, DivX и Microsoft Media Video. Все эти кодеки, кроме DivX, поставляются в составе современных версий Windows, да и установить DivX не составляет никакого труда.

Но для сжатия фильмов, импортированных во Flash, применяются совсем другие кодеки. Их два, и сейчас мы о них поговорим.

- *Sorenson Spark*. Использовался для сжатия импортированного в предыдущих версиях Flash — MX и MX 2004, также сохранился во Flash 8. Обеспечивает неплохое качество изображения при хорошей степени сжатия и невысокой загрузке центрального процессора.

- *On2 VP6.* Один из самых современных кодеков видео. Используется во Flash 8 для сжатия импортированного видео по умолчанию. Обеспечивает более высокое качество изображения и более сильное сжатие, чем Sorenson Spark, но нагрузка на центральный процессор при просмотре сжатого с его помощью видео заметно выше.

При импорте видео в документ Flash 8 оно автоматически перекодируется с использованием кодека On2 VP6 или Sorenson Spark и именно в таком виде хранится в файле документа. Когда готовый фильм публикуется в формате Shockwave/Flash, импортированное видео переносится в результирующий файл без изменений.

При импорте видео мы можем задать его параметры, влияющие на его качество и размер. (Эти параметры будут подробно описаны чуть позже.)

Но почему импортируемое во Flash видео обязательно нужно перекодировать в какие-то другие кодеки? Почему его нельзя оставить "как есть"? А все ради той же универсальности, о которой говорилось в *главе 1.* Кодеки (точнее, декодеры) On2 VP6 и Sorenson Spark встроены в проигрыватель Flash, и пользователю, просматривающему сделанный нами фильм Shockwave/Flash, не понадобятся никакие дополнительные программы, кроме самого этого проигрывателя.

Но программисты фирмы Macromedia пошли еще дальше. Они придумали еще и новый формат хранения видео в файле — *Macromedia Flash Video.* Он используется для хранения видеоклипов, которые будут впоследствии импортированы в документ Flash. Удобно, правда?

Параметры цифрового видео

А теперь настала пора поговорить о параметрах цифрового видео. Таких параметров существует три.

Первый параметр, который мы рассмотрим, — это *ширина потока данных.* (Распространен также термин *"битрейт"* — калька с английского *bitrate.*) Это самый, наверно, важный параметр — он определяет, какой объем памяти отводится под хранение данных сжатого видео, и измеряется в килобитах или мегабитах в секунду (Кбит/с или Мбит/с).

Фактически ширина потока данных показывает, сколько места будет отведено для хранения оставшейся, неотброшенной и уже сжатой информации. Она напрямую определяет качество фильма: чем она больше, тем качественнее "картинка", но и тем больше размер данных, содержащих цифровое видео.

Для сжатия фильмов, распространяемых через Интернет, используются низкие значения ширины потока данных ("низкие битрейты"): от 50 до 250 Кбит/с. Значения от 700 до 900 Кбит/с ("средние битрейты") используются для сжатия пиратских копий фильмов с применением кодека DivX; такой фильм как раз помещается на один CD. В формате DVD-Video хранятся высококачественные фильмы, сжатые с шириной потока данных в несколько Мбит/с ("высокие битрейты").

> ***НА ЗАМЕТКУ***
>
> Ширина потока данных имеет смысл только в случае сжатия с потерями. Если фильм сжимается без потерь, то этот параметр не имеет никакого практического смысла, ведь в этом случае сколько кодеку понадобится места для хранения сжатого видео, столько он и займет.

Второй по значимости параметр — это уже знакомая нам частота кадров. Разумеется, чем она ниже, тем хуже качество клипа, но и тем меньше размеры его файла. Значения частоты кадров, используемые в различных случаях, приведены в табл. 1.2.

Третий параметр цифрового видео определяет геометрические *размеры изображения*, т. е. его ширину и высоту, измеряемые в пикселах. Опять же, чем меньше эти величины, тем хуже качество клипа и тем меньше размеры его файла. Своего рода стандартами де-факто являются значения 160×120, 192×144, 320×240, 352×288, 384×288, 640×480 и 720×576 пикселей.

Полезные советы по подготовке видео для импорта

Напоследок сто́ит привести несколько полезных советов, которые помогут нам правильно подготовить видеоклипы для импорта их во Flash. Они особенно полезны, если мы готовим материалы, предназначенные для распространения через Интернет, когда размер результирующего видеофайла имеет значение.

Особенность современных алгоритмов сжатия видео (в том числе On2 VP6 и Sorenson Spark) такова, что неподвижные или малоподвижные сцены сжимаются ими лучше, чем быстро меняющиеся. Поэтому по возможности не следует выбирать для импорта в фильм Flash видеоклипы, содержащие большое количество динамичных сцен.

Обязательно нужно удалять из видеоклипов шумы. Мало того, что "зашумленные" клипы плохо сжимаются, так они еще и выглядят не очень презентабельно. И пользователь, даже если он и дождется окончания загрузки нашего фильма, все равно будет разочарован.

Как правило, "зашумленными" часто оказываются самые края изображения, особенно если это изображение было получено оцифровкой телетрансляции или аналоговой видеозаписи с кассеты. Поэтому следует обращать внимание на края изображения и при необходимости обрезать их (выполнять кадрирование) или закрывать "зашумленные" края черными рамками.

Также следует позаботиться о том, чтобы "картинка" видеоклипа имела приемлемые значения яркости, контрастности, насыщенности цветов и пр. Вряд ли пользователь захочет смотреть чересчур темный фильм.

Существует довольно много программ, предназначенных для работы с цифровым видео. Все предоставляют возможность очистить клип от шумов, отрегулировать его яркость, контрастность и цветовую насыщенность, изменить его размеры, а также выполнить другие действия — например, добавить спецэффекты. Описание работы с этими программами выходит за рамки данной книги.

И, конечно же, нужно задать подходящие значения трех рассмотренных нами выше параметров. А какие их значения подразумеваются под "подходящими", мы сейчас выясним.

Сначала, разумеется, о ширине потока данных. В большинстве случаев должно хватить 50—250 Кбит/с, поскольку импортированные клипы в фильмах Flash обычно играют вспомогательную роль. Более высокие значения ширины потока данных используются очень редко, если нужно получить высокое качество импортированного видео любой ценой.

Для выбора нужной частоты кадров можно воспользоваться табл. 1.2. А при выборе размеров изображения правило одно: они должны быть такими, чтобы "вписаться" в создаваемый нами фильм Flash.

Разумеется, во всем нужно знать меру. Если мы сделаем очень качественный фильм, то получим слишком большой результирующий файл Shockwave/Flash. Если же мы переусердствуем, ухудшая свой фильм до предела, его никто не захочет смотреть. Так что лучше поискать золотую середину, для чего, скорее всего, придется поэкспериментировать.

Какие форматы видео поддерживает Flash

А теперь выясним, какие форматы цифрового видео поддерживает Flash 8. Все они приведены в табл. 16.1. Там же указано, какое дополнительное программное обеспечение необходимо для того, чтобы импортировать файл каждого из этих форматов.

Таблица 16.1. *Список видеоформатов, которые можно импортировать во Flash*

Название формата	Расширение файлов	Требуется Apple QuickTime 7*	Требуется Microsoft DirectX 9*
Apple QuickTime	mov	+	
AVI	avi	+	+
Microsoft Windows Media	wmv, asf		+
MPEG	mpeg, mpe, mpg, dat	+	+
Видео с цифровой видеокамеры (формат Digital Video)	dv	+	

* Данная версия или более новая.

ВНИМАНИЕ!

В некоторых случаях для видеофайлов, содержащих звуковое сопровождение, может быть импортировано только видео, но не звук. Это может произойти из-за того, что Flash не поддерживает данный формат сжатия и записи звуковых данных. В таком случае следует преобразовать звуковую дорожку видеоклипа к одному из совместимых форматов, воспользовавшись программами редактирования видео. (Все поддерживаемые Flash форматы звука перечислены в табл. 17.1).

Импорт видео

Нужно сразу сказать, что импортировать видеоклип в документ Flash можно двумя разными способами. Сейчас мы их рассмотрим.

Первый способ — собственно импорт, или *внедрение*. Файл с видеоклипом открывается во Flash, тут же автоматически перекодируется с использованием кодека On2 VP6 или Sorenson Spark и помещается в файл документа Flash. Все — теперь импортированный нами клип стал неотъемлемой частью нашего фильма.

Преимущество внедрения очевидно — как говорится, "все свое несу с собой". Результирующий файл Shockwave/Flash будет содержать импортированный нами клип, и никаких проблем с его воспроизведением возникнуть не должно. Недостатков всего два. Во-первых, поскольку импортированный клип целиком помещается в файл Shockwave/Flash, размер увеличивается, и зачастую сильно. Во-вторых, как мы давно знаем, внедряемый клип при

импорте перекодируется кодеком On2 VP6 или Sorenson Spark, а это довольно долгий процесс.

Второй способ — *привязка* видеоклипа к документу Flash. Импортируемый клип не становится частью документа Flash; он будет загружен отдельно при воспроизведении готового фильма. Привязать можно только клип в формате Apple QuickTime, причем поддерживает привязку только проигрыватель Flash версий 3-5.

Достоинств у привязки два. Первое — импортированный клип не хранится в файле Shockwave/Flash, в результате чего последний не увеличивается в размерах. Второе — если мы исправили импортированный клип, не нужно выполнять его повторного импорта. Недостатков также два: нужно позаботиться о том, чтобы пользователь, просматривающий фильм Shockwave/Flash, имел доступ к привязанному к нему клипу, невозможность привязки клипов любых форматов, кроме Apple QuickTime, и необходимость просмотра фильма, содержащего привязанный клип, в проигрывателе Flash версии 3—5.

Разобравшись с внедрением и привязкой, давайте приступим к собственно импорту. Но сначала поищем по закоулкам жесткого диска различные видеофайлы, лучше всего — разных форматов.

Импорт видеоклипа в документ Flash выполняется очень просто. Выбираем пункт **Import to Stage** подменю **Import** меню **File** или нажимаем комбинацию клавиш <Ctrl>+<R>, чтобы сразу поместить импортированный клип на рабочий стол, или пункт **Import to Library** подменю **Import** меню **File**, чтобы выполнить импорт только в библиотеку. В обоих этих случаях на экране появится стандартное диалоговое окно открытия файла Windows, в котором мы должны будем выбрать файл, содержащий нужный клип, и нажать кнопку открытия.

Мы также можем выбрать пункт **Import Video** подменю **Import** меню **File**. В этом случае стандартного диалогового окна открытия файла на экране не появится, и мы должны будем выбрать нужный файл потом.

В любом случае на экране появится диалоговое "окно-мастер" **Import Video**, первая страница которого показана на рис. 16.1.

Это окно содержит два переключателя, задающих местоположение импортируемого нами клипа. Изначально включен переключатель **On your computer**, задающий в качестве местоположения наш собственный компьютер. В поле ввода **File path** отображается полное имя выбранного нами файла; справа от него находится кнопка **Browse**, после нажатия которой на экране появится диалоговое окно открытия файла Windows. Это может пригодиться, если мы открыли окно **Import Video** выбором пункт **Import Video** подменю **Import** меню **File**.

Мы также можем выбрать для импорта файл, уже находящийся на каком-либо Web-сервере. Для этого достаточно включить переключатель **Already**

deployed to a web server, Flash Video Streaming Service, or Flash Communication Server и ввести интернет-адрес этого файла в поле ввода **URL**.

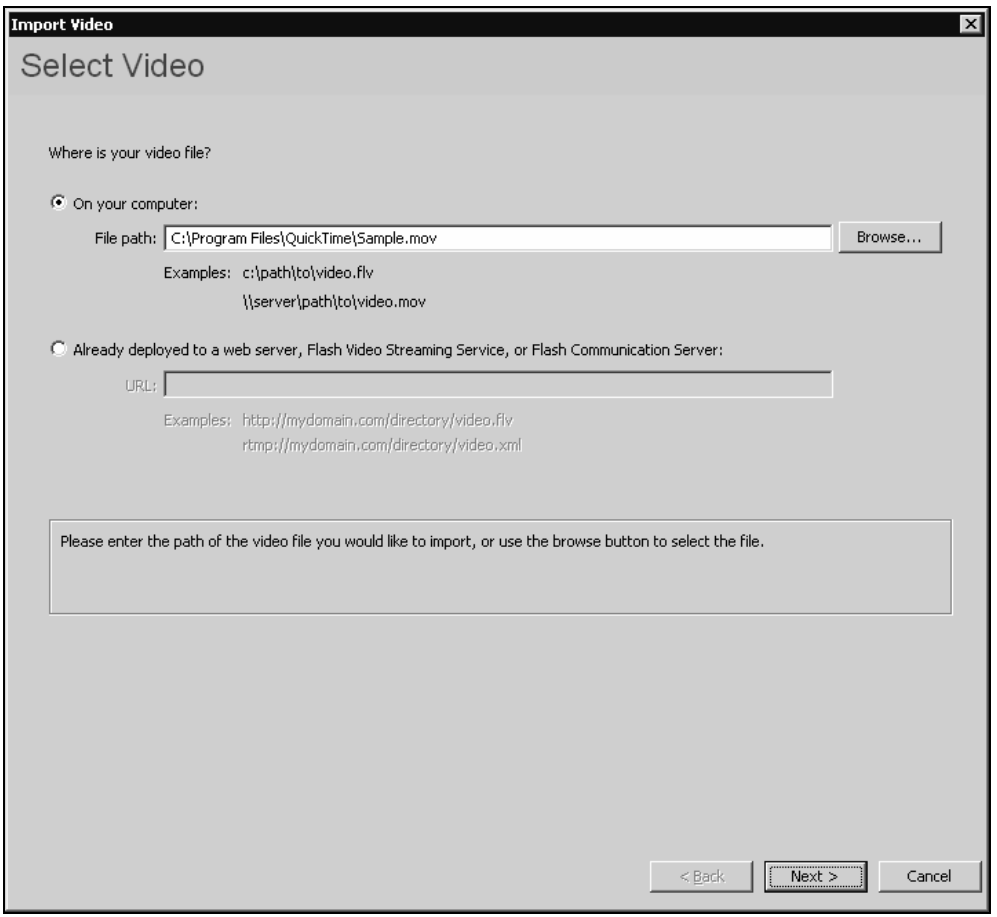


Рис. 16.1. Диалоговое окно **Import Video** (первая страница)

Во всех уважающих себя "окнах-мастерах" для перехода на следующую страницу нужно нажать кнопку **Next**. Если нажать кнопку **Cancel**, фильм не будет импортирован.

Вторая страница окна **Import Video** показана на рис. 16.2. Здесь мы надолго не задержимся.

Чтобы выполнить внедрение клипа, нам будет нужно выбрать переключатель **Embed video in SWF and play in timeline**. После этого можно нажимать кнопку **Next** и отправляться на следующую страницу окна **Import Video**.

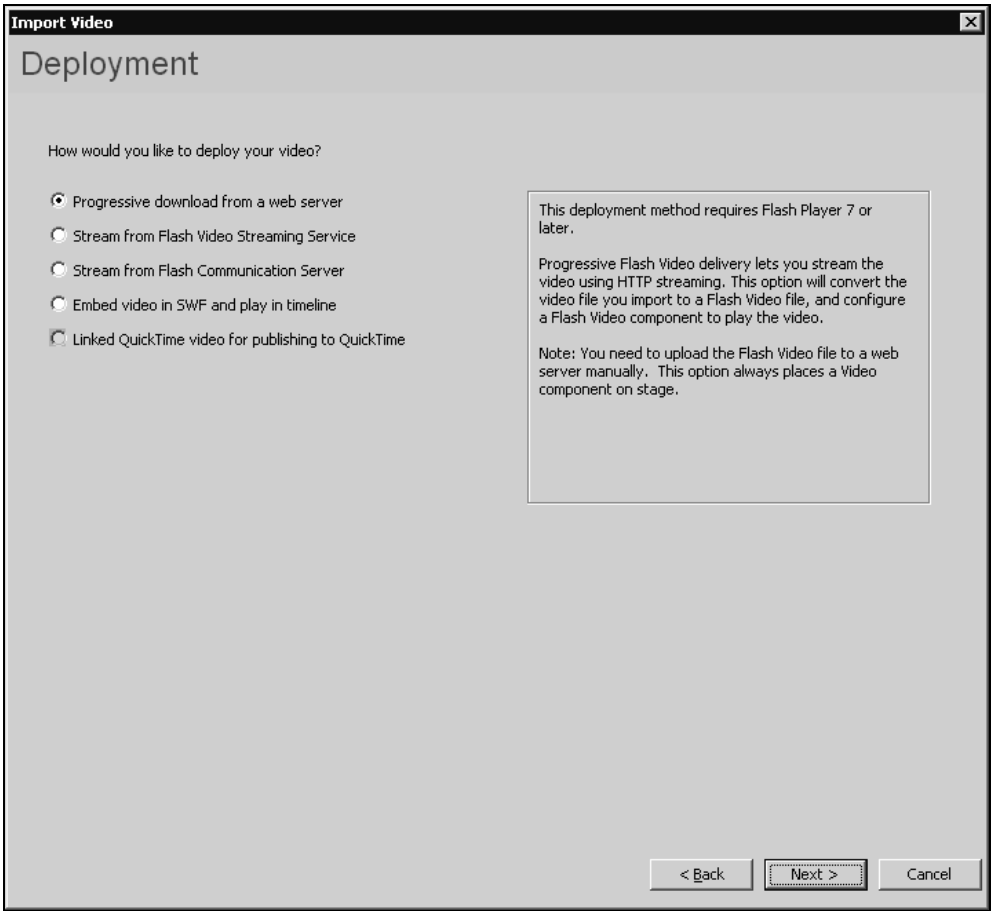


Рис. 16.2. Диалоговое окно **Import Video** (вторая страница)

Если же мы хотим выполнить привязку клипа, то включим переключатель **Linked QuickTime video for publishing to QuickTime**. При этом если в параметрах публикации документа мы не указали создание файла Shockwave/Flash для версий 3—5 проигрывателя Flash, на экране появится окно-предупреждение, предлагающее нам это сделать. (О параметрах публикации *см. главу 21.*) Сделав это, можно нажать кнопку **Finish** — и процесс привязки клипа закончен.

На заметку

Как видно на рис. 16.2, на второй странице окна **Import Video** присутствуют также переключатели **Progressive download from a web server**, **Stream from Flash Video Streaming Service** и **Stream from Flash Communication Server**.

Эти переключатели указывают Flash создать на рабочем листе компонент-проигрыватель видео, в котором будет воспроизводиться импортируемый клип. Использование компонентов выходит за рамки данной книги; желающие могут найти их описание в интерактивной справке Flash.

Третья страница окна **Import Video** показана на рис. 16.3. Она, как и все последующие страницы этого окна, появится, только если мы выполняем внедрение клипа.

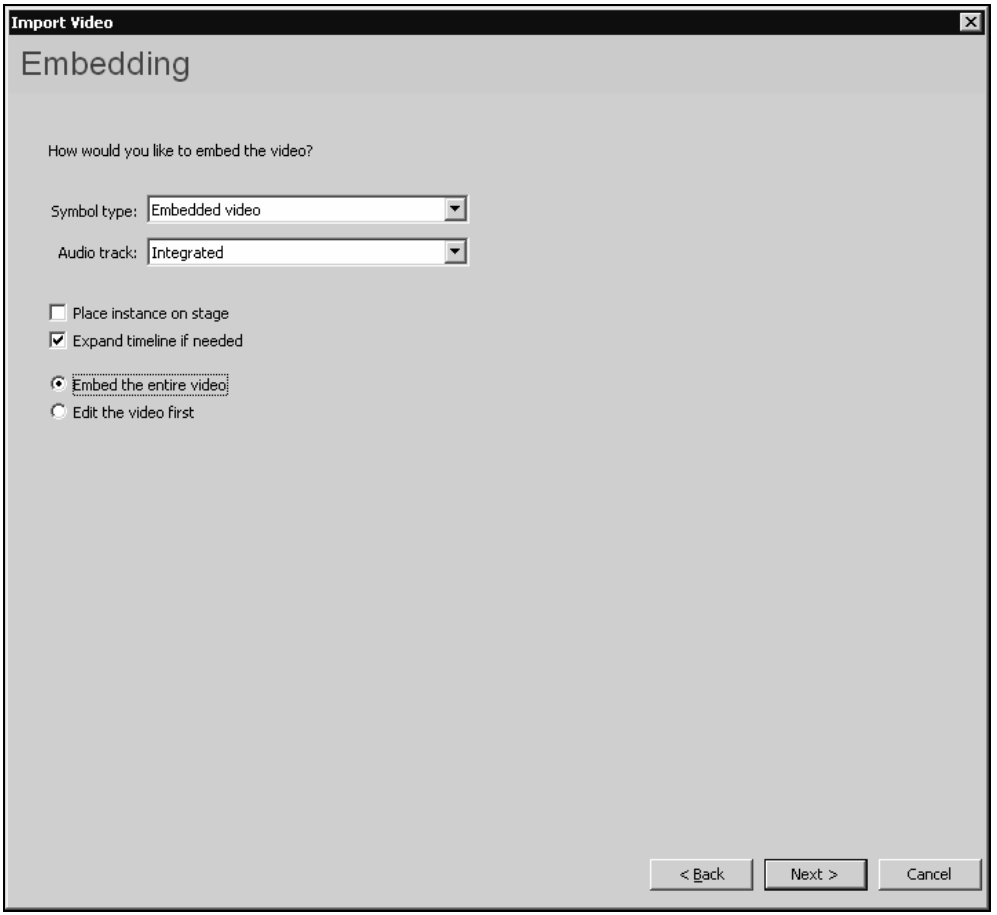


Рис. 16.3. Диалоговое окно **Import Video** (третья страница)

Раскрывающийся список **Symbol type** позволит нам выбрать тип образца, в который будет преобразован импортируемый клип. Здесь доступны три

пункта: **Embedded video** (образец-импортированный клип; этот пункт выбран по умолчанию), **Movie clip** (образец-клип) и **Graphic** (графический образец).

Другой раскрывающийся список — **Audio track** — позволит нам выбрать, импортировать ли вместе с клипом его звуковую дорожку (если она есть). Выбранный по умолчанию пункт **Integrated** заставляет Flash импортировать звуковую дорожку клипа, а пункт **Separate** позволяет нам отказаться от этого (например, если мы хотим потом, после прочтения *главы 17*, задать для клипа свое звуковое сопровождение).

Флажок **Place instance on stage**, будучи включенным, указывает Flash сразу же после импорта клипа и создания на его основе образца поместить его экземпляр на рабочий лист. А флажок **Expand timeline if needed** при включении заставляет Flash при необходимости расширить дорожку, чтобы в ней поместились все кадры импортируемого клипа. Последний флажок включен по умолчанию, и лучше его не отключать.

Последняя группа переключателей позволит нам перед импортом немного отредактировать клип. Для этого нужно будет включить переключатель **Edit the video first**. Но нам сейчас не до этого, так что давайте включим переключатель **Embed the entire video** (впрочем, он и так включен по умолчанию), чтобы импортировать клип без всякой редактуры.

Нажмем в очередной раз кнопку **Next**. И перейдем на четвертую по счету страницу окна **Import Video** (рис. 16.4).

Здесь мы можем выбрать профиль сжатия импортируемого клипа. *Профиль* — это именованный набор предопределенных параметров, относящихся в данном случае к сжатию видео. (Flash также поддерживает профили публикации, о которых мы поговорим в *главе 21*.)

Для выбора профилей служит раскрывающийся список **Please select a Flash Video encoding profile**. Его пункты имеют "говорящие" названия, так что описывать их излишне. В самом деле, если пункт имеет название **Flash 8 - Medium Quality (400kbps)**, то сразу понятно, что он задает сжатие видео с помощью кодека On2 VP6 (для проигрывателя Flash 8) с шириной потока данных 400 Кбит/с, в результате чего достигается среднее (в терминологии Flash) качество.

Ниже списка **Please select a Flash Video encoding profile** находится большое текстовое поле. В нем отображается более подробная информация о выбранном профиле. В частности, мы можем увидеть там сведения о параметрах сжатия звуковой дорожки импортируемого клипа.

Кнопка **Show Advanced Settings** выводит в том же окне **Import Video** большой набор элементов управления, предназначенных для более точного задания параметров сжатия видео. Эти элементы управления пригодятся нам, если мы не удовлетворимся ни одним профилем, а захотим задать свои параметры. Во всех подробностях они будут описаны в конце этой главы.

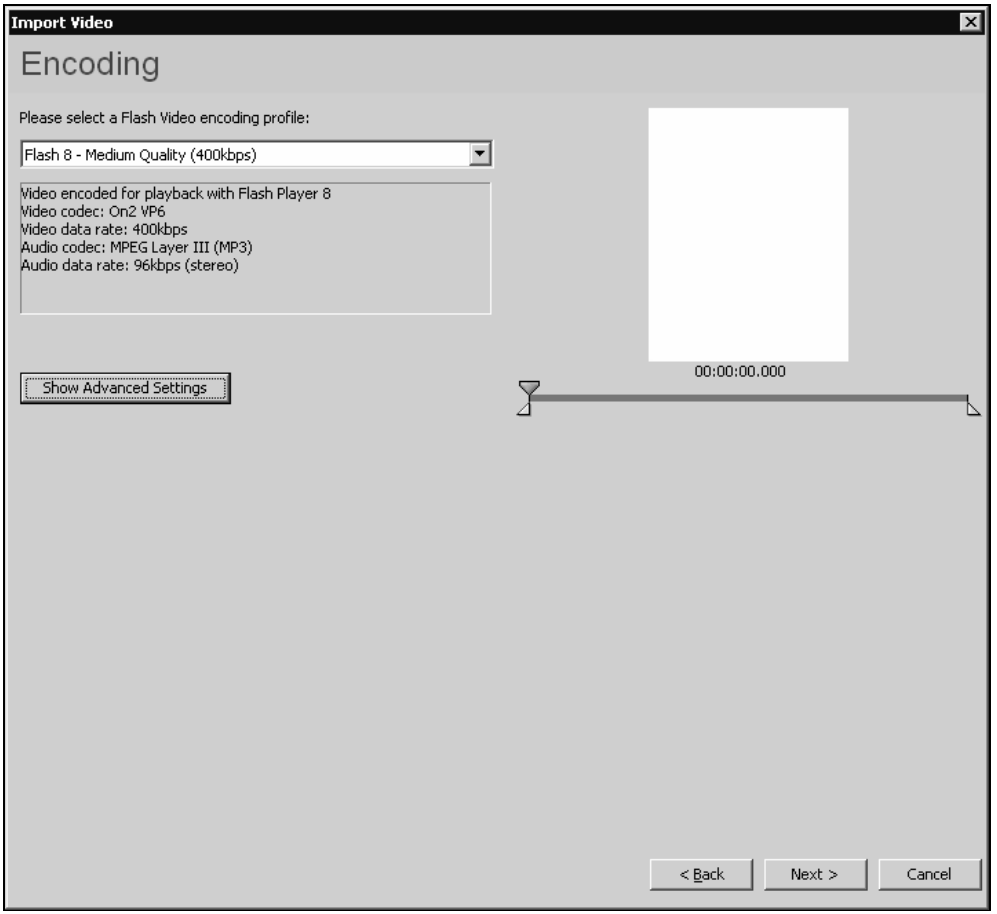


Рис. 16.4. Диалоговое окно **Import Video** (четвертая страница)

А еще на четвертой странице окна **Import Video** находятся элементы управления, которые позволят нам просмотреть импортируемый клип и, если нужно, отрезать у него начало и конец, не несущие смысловой нагрузки. Это, прежде всего, область предварительного просмотра, находящаяся в правом верхнем углу этого окна. Под ней находится временна́я шкала с бегунком. Бегунок помечает текущий кадр клипа, содержимое которого отображается в области предварительного просмотра. Позиция, в которой находится бегунок, показывается ниже области предварительного просмотра, правда, в единицах времени, а не в кадрах.

На концах временно́й шкалы находятся два пока еще незнакомых нам маркера. Это *маркер начала* и *маркер конца клипа*. Они обозначают начало и конец импортируемого фрагмента клипа соответственно и пригодятся для

того, чтобы удалить из импортируемого клипа не несущие содержания фрагменты из его начала и конца. Мы можем перетаскивать эти маркеры мышью.

Очередное нажатие кнопки **Next** — и перед нами пятая, информационная, страница окна **Import Video** (рис. 16.5).

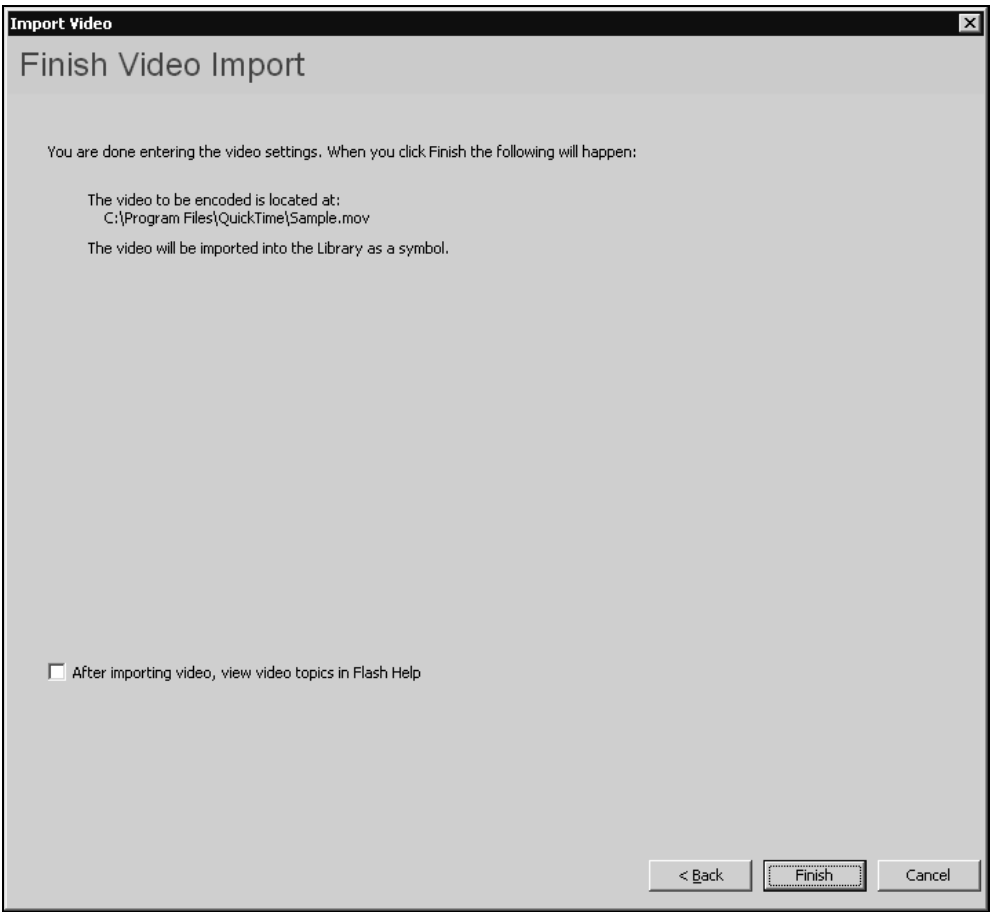


Рис. 16.5. Диалоговое окно **Import Video** (пятая страница)

Здесь мы можем прочитать сведения об импортируемом клипе и краткую сводку всех заданных нами параметров. И нажмем кнопку **Finish**, чтобы завершить, наконец, процесс импорта.

Если перед нажатием кнопки **Finish** мы включим флажок **After importing video, view video topics in Flash Help**, Flash после выполнения импорта клипа

выведет в панели **Help** все топики, так или иначе относящиеся к импорту видео и его использованию в фильмах Flash. Впрочем, можно и не включать этот флажок — как вывести на экран панель **Help** и найти в ней нужный нам топик, мы узнали еще в *главе 2*.

Во время перекодирования клипа Flash будет показывать небольшое окно с индикатором степени ее выполнения. Процесс этот весьма длительный, так что придется набраться терпения.

В результате всех этих мучений мы получим новый образец в библиотеке и, возможно, на рабочем листе (если на третьей странице окна **Import Video** (см. рис. 16.3) включен флажок **Place instance on stage**).

Если в раскрывающемся списке **Symbol type** третьей страницы окна **Import Video** был выбран пункт **Embedded video**, мы получим образец-импортированный клип. В графе **Kind** списка образцов панели **Library** будут стоять слова `Embedded Video`.

Экземпляры таких образцов имеют одну особенность, которая знакома нам по графическим образцам. Дело в том, что они воспроизводятся как бы внутри дорожки анимации, в которой созданы (т. е., здесь мы получим случай зависимой анимации, о которой рассказывалось в *главе 13*). Поэтому дорожка должна содержать такое количество кадров, чтобы клип смог воспроизвестись полностью. Проще всего это сделать, растянув ключевой кадр, в котором этот экземпляр создан. (Впрочем, если, наряду с флажком **Place instance on stage** третьей страницы окна **Import Video** мы включим флажок **Expand timeline if needed**, Flash сам добавит в дорожку нужное количество подкадров.)

Если же в раскрывающемся списке **Symbol type** третьей страницы окна **Import Video** мы выбрали пункты **Movie clip** или **Graphic**, на основе импортированного клипа будет создан, соответственно, образец-клип и графический образец. Как обращаться с ними, мы давно знаем.

Использование импортированных клипов

Мы можем создавать экземпляры образцов-импортированных клипов таким же образом, как экземпляры всех остальных знакомых нам образцов. И точно так же мы можем применять к этим экземплярам какие угодно трансформации, используя все инструменты, описанные в *главе 4*. Мы даже можем их анимировать, для чего достаточно создать второй ключевой кадр и задать нужную анимацию. (Это будет забавно: видеоклип на рабочем столе одновременно воспроизводится и прыгает туда-сюда.)

Если мы создадим на основе образца-импортированного клипа экземпляр, Flash сразу же выведет на экран окно-предупреждение (рис. 16.6), предла-

гающее нам сразу создать дорожку кадров такой длительности, чтобы этот клип воспроизвелся полностью. Так, на этом рисунке Flash предлагает нам создать дорожку из 59 кадров — именно такую длину имеет импортированный клип.

Рис. 16.6. Окно-предупреждение, предлагающее создать дорожку кадров для воспроизведения импортированного клипа

Зачем это нужно, мы уже знаем. Анимация, сохраненная в импортированном клипе, является зависимой, как и внутренняя анимация графического образца. Поэтому для ее воспроизведения нам понадобится дорожка, содержащая достаточное количество кадров.

Как Flash создает эту дорожку? Очень просто. Он берет ключевой кадр, в котором мы создали этот экземпляр, и растягивает его на нужное количество подкадров. Как говорится, дешево и сердито.

Конечно, мы можем отказаться от автоматического создания требуемой дорожки кадров и создать ее самим. Правда, для этого придется каким-то образом выяснить, сколько кадров содержит импортированный клип. В этом нам могут помочь программы, которые позволяют увидеть параметры клипа, например, GSpot (найти ее можно на сайте **http://gspot.headbands.com**).

Кстати, мы можем воспроизвести экземпляр импортированного клипа не целиком, а частично, точнее, какую-то его начальную часть. Для этого нужно просто сократить созданную Flash дорожку до нужного числа кадров. К сожалению, воспроизвести такими средствами произвольный фрагмент клипа, начинающийся с любого его кадра, а не только с первого, невозможно — нам придется править сам файл, где хранится этот клип, а потом импортировать его заново.

Также можно выполнить смену образца-импортированного клипа, на основе которого был создан данный экземпляр, на другой. Для этого выделим на рабочем листе нужный клип и нажмем кнопку **Swap** в панели **Properties**. На экране появится диалоговое окно **Swap Embedded Video** (рис. 16.7).

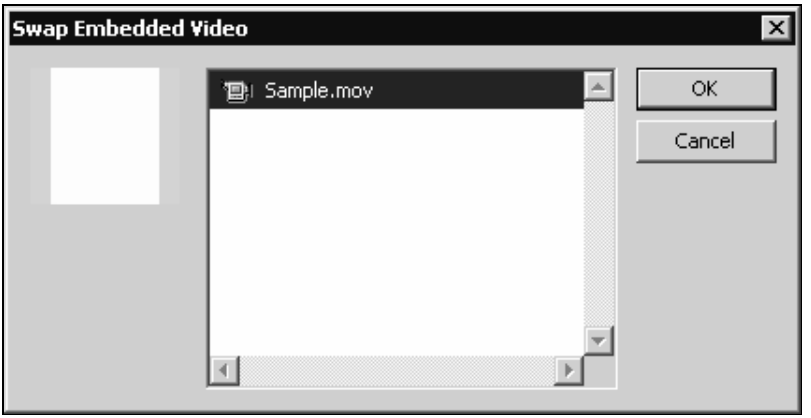


Рис. 16.7. Диалоговое окно **Swap Embedded Video**

В списке, занимающем бóльшую часть этого окна, перечислены все импортированные в текущий документ клипы. Выбираем требуемый клип, после чего в небольшой области предварительного просмотра, расположенной в левом верхнем углу этого окна, будет показан его первый кадр. Далее нам останется только нажать кнопку **OK**, чтобы выполнить смену образца, или кнопку **Cancel** для отказа от этого.

> ***Внимание!***
>
> Если вы выполнили смену образца экземпляра в первом ключевом кадре трансформационной анимации, не забудьте сделать то же самое и во втором кадре. Иначе анимация будет "оборвана". (О трансформационной анимации *см. главу 12*.)

К сожалению, Flash 8 не позволяет менять внедренный клип на привязанный. Нам придется полностью удалить старый экземпляр и создать на рабочем листе новый, уже на основе нового образца.

Работа с образцами-импортированными клипами

Мы можем проделывать с образцами-импортированными клипами то же самое, что и с образцами любых других типов: превращать их в разделяемые, задавать для них различные параметры, помещать в папки и удалять. Как все это делается, описывалось в *главе 8*. Здесь же мы рассмотрим лишь немногочисленные специфичные для импортированных клипов действия.

А выполняются эти действия в диалоговом окне **Video Properties**. Вызвать его на экран можно, выделив нужный образец-импортированный клип в списке панели **Library** и выбрав пункт **Properties** контекстного или дополнительного меню этой панели. На рис. 16.8 показано окно **Video Properties**, отображающее параметры внедренного клипа.

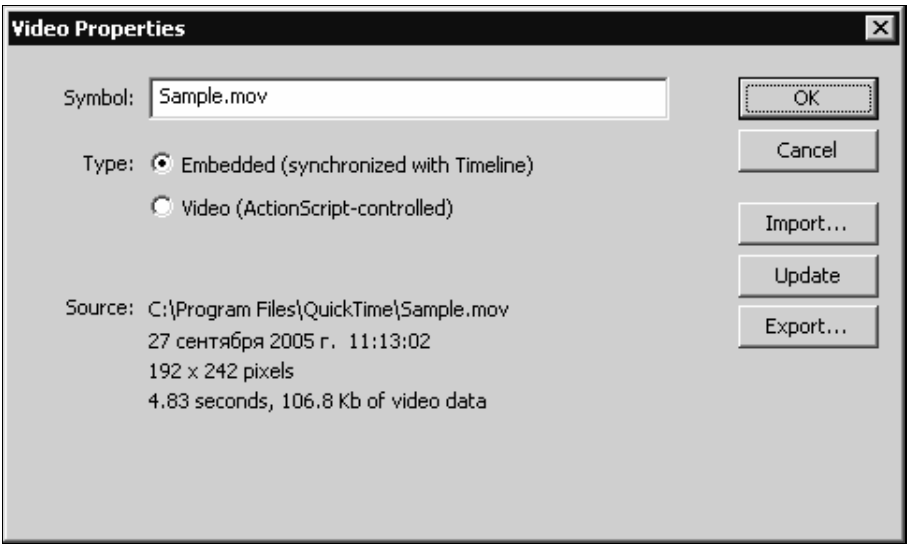


Рис. 16.8. Диалоговое окно **Video Properties**, показывающее параметры внедренного клипа

В поле ввода **Symbol** вводится имя образца. Это нам уже знакомо по *главе 8*.

Группа переключателей **Type** задает, будет ли этот образец внедренным (переключатель **Embedded (synchronized with Timeline)**) или управляться через компонент (переключатель **Video (ActionScript-controlled)**). Поскольку мы в этой книге не будем изучать компоненты, оставим включенным первый переключатель.

А вот информация, представленная в нижней половине этого окна, будет нам полезна. Это (в порядке сверху вниз) полное имя файла, из которого был импортирован клип, дата и время его импорта, размер его изображения в пикселах, продолжительность в единицах времени (часы, минуты и секунды, но — вот досада! — не кадры) и размер данных этого клипа.

С помощью кнопки **Import** мы можем импортировать на место существующего клипа другой. При нажатии этой кнопки на экране появится стандартное диалоговое окно открытия файла Windows. Находим нужный файл и нажимаем кнопку открытия.

Кнопка **Update** служит для повторного импорта (обновления) этого клипа. Это может быть полезно, если мы отредактировали изначальный клип в какой-либо программе для работы с цифровым видео. (Встроенные во Flash возможности правки клипов довольно бедны; с их помощью, например, нельзя наложить титры.)

После задания новых параметров клипа и выполнения всех манипуляций с ним нужно нажать кнопку **OK**. Кнопка **Cancel** позволит нам отказаться от ввода новых параметров.

Все описанное выше касалось внедренных клипов. В случае же привязанных клипов набор доступных операций намного меньше — см. рис. 16.9. Мы можем только сменить имя образца и имя файла, где хранится привязанный клип. Чтобы сделать это, нужно нажать кнопку **Set Path**. На экране появится стандартное диалоговое окно открытия файла Windows, в котором нам останется выбрать нужный файл.

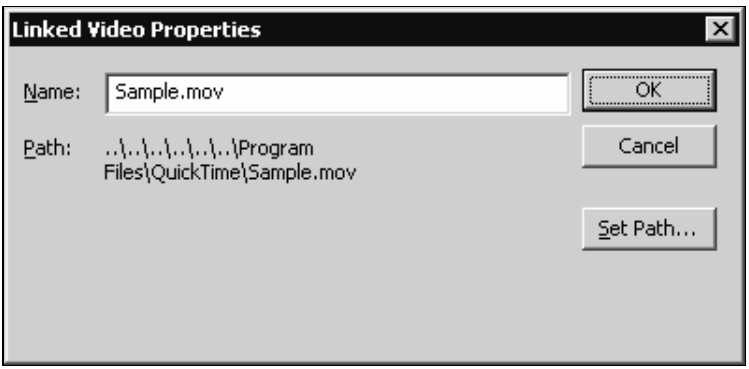


Рис. 16.9. Диалоговое окно **Video Properties**, показывающее параметры привязанного клипа

Правка импортируемого клипа средствами Flash

Мы уже знаем, что прямо перед импортом можем выполнить обрезку клипа, задав точки его начала и конца на четвертой странице диалогового окна **Import Video** (см. рис. 16.4). Для этого предназначена временна́я шкала с соответствующими маркерами, описанная в начале этой главы.

Но на самом деле возможности Flash по правке клипа перед импортом гораздо больше. Так, мы можем разрезать клип на несколько более мелких

клипов (*подклипов*), которые будут импортированы как отдельные образцы. Давайте выясним, как это сделается, поскольку такие "хирургические" операции приходится выполнять довольно часто, а далеко не у всех на компьютере установлена программа для работы с видео.

Чтобы выполнить правку, импортируемый клип нужно включить на третьей странице окна **Import Video** (см. рис. 16.3) переключатель **Edit the video first** и нажать кнопку **Next**. Окно **Import Video** примет вид, показанный на рис. 16.10.

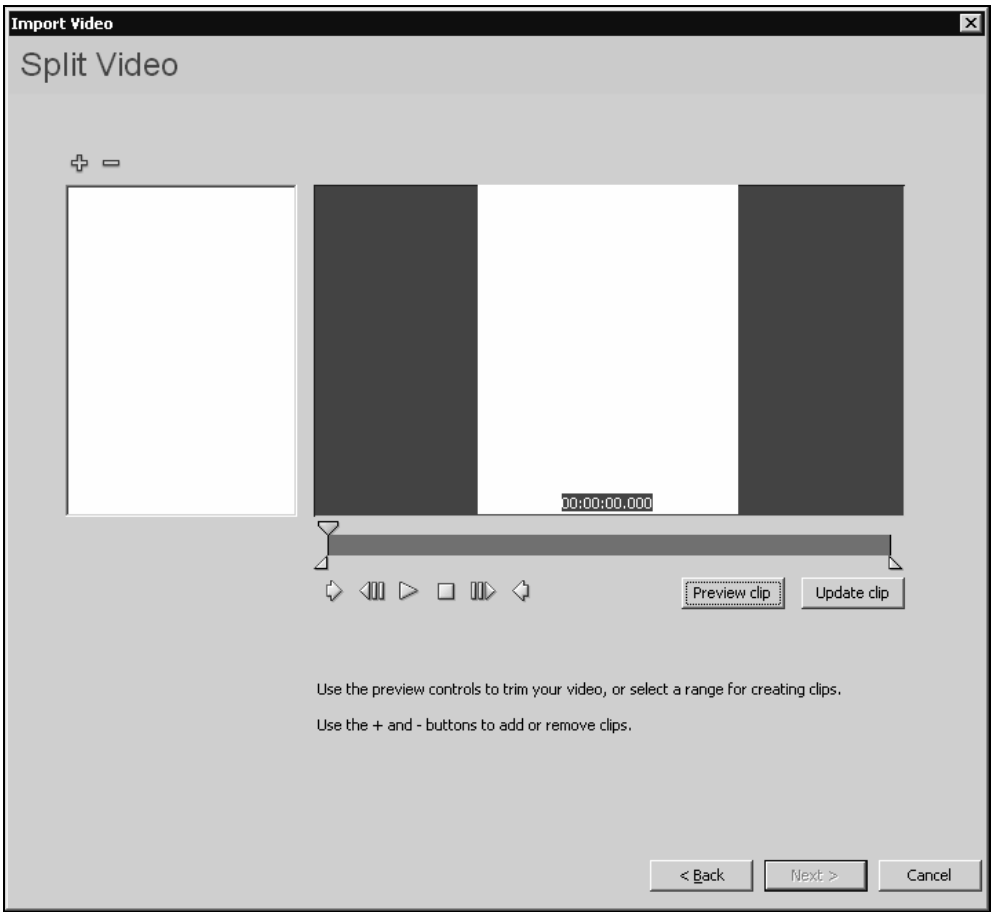


Рис. 16.10. Диалоговое окно **Import Video** в режиме правки импортируемого клипа

Бóльшую часть этого окна занимает область предварительного просмотра, на которой воспроизводится содержимое клипа. Под ней находится времен-

на́я шкала клипа с бегунком. Бегунок помечает текущий кадр клипа, содержимое которого видно в области предварительного просмотра. Позиция, в которой находится бегунок, выводится в нижней части области предварительного просмотра, правда, в единицах времени, а не в кадрах. Также на временно́й шкале присутствуют маркеры начала и конца клипа. Это нам уже знакомо.

Ниже и левее временно́й шкалы находится набор кнопок, управляющих воспроизведением и положением маркеров начала и конца клипа. Перечислим их в порядке слева направо:

- перемещение маркера начала клипа в позицию, в которой находится бегунок;
- перемещение бегунка на кадр назад;
- воспроизведение клипа с позиции, в которой находится бегунок, до изначального конца клипа (то, где в данный момент находится маркер конца клипа, значения не имеет);
- остановка воспроизведения;
- перемещение бегунка на кадр вперед;
- перемещение маркера конца клипа в позицию, в которой находится бегунок.

Мы можем задать начало и конец клипа, установив на нужные позиции соответствующие маркеры, и этим ограничиться. Но это можно сделать и на четвертой странице окна **Import Video**. А разрезать клип на подклипы мы можем только здесь. Давайте выясним, как это делается.

Чтобы создать подклип, мы зададим его начало и конец с помощью соответствующих маркеров и нажмем кнопку со знаком "плюс", находящуюся над списком, что располагается левее области предварительного просмотра клипа. Созданный нами подклип появится в этом списке. Точнее, сначала появится небольшое поле ввода, в котором мы сможем ввести имя создаваемого подклипа, после чего должны будем нажать клавишу <Enter>. Если мы не хотим вводить новое имя для вновь созданного подклипа, Flash даст ему имя вида *<Имя изначального файла клипа> <Порядковый номер>*.

> ***Внимание!***
>
> Если вы разобьете клип на подклипы, импортированы будут только они. Все остальное содержимое клипа, не вошедшее в подклипы, будет отброшено Flash при импорте.

Чтобы просмотреть содержимое какого-либо подклипа, нужно выделить его в списке и нажать кнопку **Preview clip**.

Мы можем дать уже имеющемуся подклипу новое содержимое. Для этого выделим в списке нужный подклип, зададим для него новые начало и конец и нажмем кнопку **Update clip**.

Также мы можем переименовать любой подклип. Для этого дважды щелкнем на его имени в списке подклипов, введем в появившемся вместо его имени поле ввода новое имя и нажмем клавишу <Enter>.

Чтобы удалить ненужный подклип, достаточно выделить его и нажать кнопку со знаком "минус". Она также находится над списком подклипов.

Закончив правку клипа, нажмем кнопку **Next**, и окно **Import Video** переключится на четвертую страницу (см. рис. 16.4). Если мы нажмем кнопку **Back**, то вернемся на третью страницу этого окна (см. рис. 16.3).

Вот и все о правке видеоклипов в среде Flash. Да, много таким способом не направишь, и для того чтобы сделать в клипе какие-то серьезные изменения, нам придется воспользоваться специализированной программой. Но для простейших правок встроенных возможностей Flash вполне достаточно.

Управление качеством импортируемого клипа

Кроме правки импортируемого клипа прямо во время импорта мы можем задавать параметры его качества. Выполняется это на четвертой странице диалогового окна **Import Video** (см. рис. 16.4).

Чтобы получить доступ к элементам управления, задающим параметры импортируемого клипа, нужно нажать кнопку **Show Advanced Settings**. Все эти элементы управления появятся в нижней половине окна **Import Video** (рис. 16.11) в "блокноте", содержащем две вкладки. Рассмотрим их по порядку.

Начнем рассмотрение с вкладки **Encoding**. Эта вкладка содержит элементы управления, предназначенные для задания параметров сжатия видео и звука.

Флажок **Encode video** при включении указывает Flash импортировать видеодорожку клипа. Он включен по умолчанию, и отключать его не стоит (в самом деле, кому нужен видеоклип без видеодорожки?..).

Если флажок **Encode video** включен, становятся доступными все элементы управления, расположенные ниже его. Они задают параметры видеодорожки импортируемого клипа.

В раскрывающемся списке **Video codec** задается кодек видео, с помощью которого будет сжата видеодорожка клипа. В нем доступны пункты **On2 VP6** (выбран по умолчанию) и **Sorenson Spark**. Назначение этих пунктов должно быть понятно из их названий.

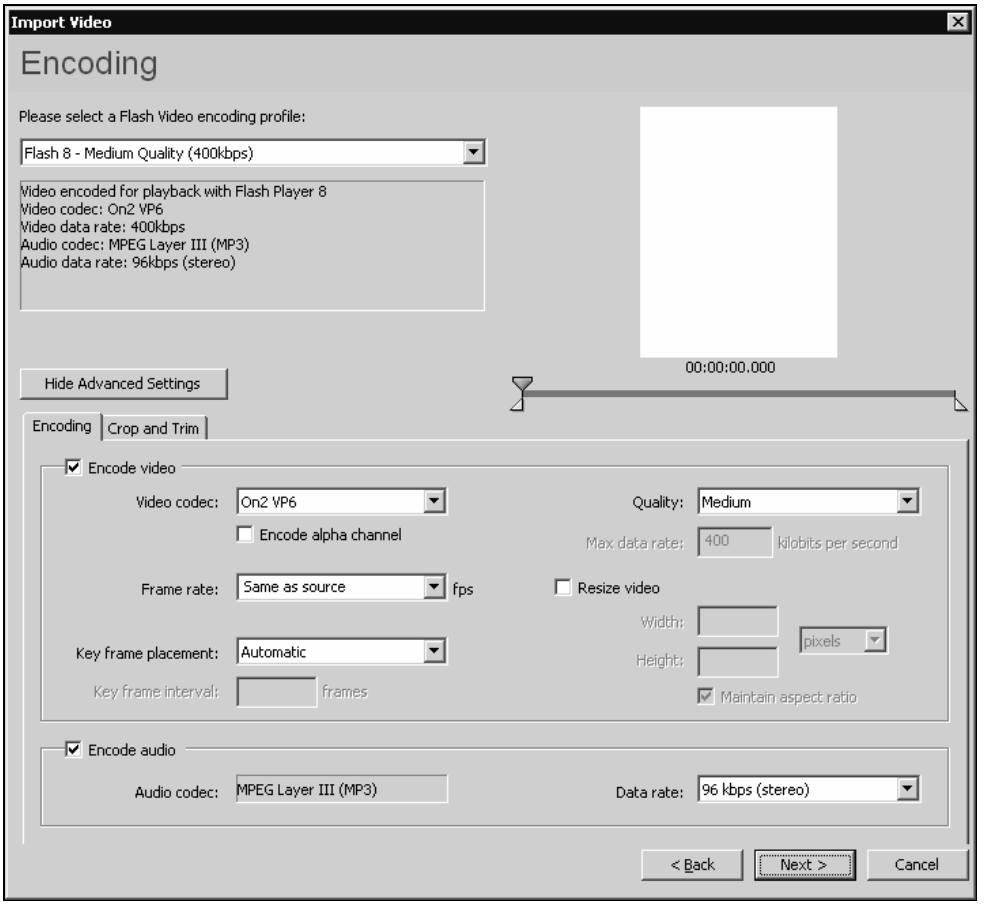


Рис. 16.11. Диалоговое окно **Import Video** (четвертая страница, нажата кнопка **Show Advanced Settings**)

В большинстве случаев следует оставить выбранным пункт **On2 VP6**. Пункт **Sorenson Spark** мы выберем только в том случае, если собираемся создавать фильм для предыдущих версий проигрывателя Flash, поддерживающих воспроизведение внедренного видео (это версии 6 и 7).

Если в списке **Video codec** выбран пункт **On2 VP6**, становится доступным флажок **Encode alpha channel**. При его включении Flash поместит в импортированную видеодорожку канал прозрачности оригинального клипа (если таковой есть).

Раскрывающийся список **Frame rate** служит для выбора частоты кадров импортированной видеодорожки. По умолчанию выбран пункт **Same as source** (Частота кадров та же, что у оригинального клипа); также доступны пункт

Same as FLA (Частота кадров та же, что у документа Flash) или любой из пунктов **10**, **12**, **15**, **24**, **25** или **30**, задающих числовое значение частоты кадров. Нужное значение частоты кадров можно ввести в этот список и вручную.

Как мы помним из *главы 1*, алгоритмы, реализующие сжатие видео, анализируют каждый кадр и сохраняют в результирующем файле только данные о различиях между соседними кадрами. Так вот, чтобы иметь возможность вычислить это различие, какие-то кадры кодек должен сохранить целиком — это будут своего рода образцы для сравнения с последующими кадрами. Такие кадры, сохраненные целиком, называются ключевыми — этот термин знаком нам из трансформационной анимации.

Раскрывающийся список **Key frame placement** как раз и задает расположение ключевых кадров видеодорожки. Выбранный по умолчанию пункт **Automatic** отдает расстановку ключевых кадров на откуп кодеку. Также можно выбрать пункт **Custom** и ввести интервал между ключевыми кадрами в поле ввода **Key frame interval**; этот интервал задается в кадрах.

Раскрывающийся список **Quality** задает качество импортированного видео после сжатия. В нем доступны следующие пункты: **Low** (Низкое качество), **Medium** (Среднее), **High** (Высокое) и **Custom**. Если выбран последний пункт, становится доступным поле ввода **Max data rate**, в котором можно ввести ширину потока данных в Кбит/с.

Флажок **Resize video** при включении заставляет Flash при перекодировке выполнить изменение размеров "картинки" импортируемого клипа. Новые размеры задаются в полях ввода **Width** (Ширина) и **Height** (Высота). Раскрывающийся список, расположенный правее этих полей, задает единицу измерения размеров "картинки"; в нем доступны пункты **pixels** (пикселы) и **percent** (проценты). Если включен флажок **Maintain aspect ratio**, Flash при вводе одного из значений размеров будет пересчитывать значение другого таким образом, чтобы сохранить пропорции "картинки".

С параметрами видеодорожки мы покончили. Перейдем к элементам управления, задающим параметры звуковой дорожки клипа. И начнем с флажка **Encode audio**, который при включении указывает Flash импортировать аудиодорожку клипа. Он также включен по умолчанию.

Если флажок **Encode audio** включен, становятся доступными немногочисленные элементы управления, расположенные ниже его. Они задают параметры аудиодорожки импортируемого клипа.

Собственно, элемент управления здесь всего один — раскрывающийся список **Data rate**. Он задает ширину потока данных звуковой дорожки и количество каналов: один (монофонический звук) или два (стереофонический). Пункты этого списка имеют "говорящие" названия; так, пункт **96 kbps (stereo)** обозначает ширину потока данных в 96 кбит/с и двухканальный (стереофонический) звук.

ВНИМАНИЕ!

Звуковое сопровождение импортируемых клипов всегда перекодируется с использованием кодека MPEG I level 3 (MP3).

Перейдем теперь к вкладке **Crop and Trim**. Эта вкладка содержит элементы управления, позволяющие нам выполнить обрезку "картинки" клипа, чтобы, например, удалить присутствующие по ее краям помехи.

На этой вкладке присутствуют четыре поля ввода с регулятором, расположенные на сторонах воображаемого прямоугольника, представляющего "картинку" клипа. Каждое из этих полей ввода задает количество пикселов, которые будут отрезаны, с соответствующей стороны этого прямоугольника. В общем, здесь все достаточно наглядно.

При вводе значений в любое из этих полей в области предварительного просмотра клипа в правом верхнем углу окна **Import Video** (см. рис. 16.11) появится пунктирный прямоугольник, обозначающий остающуюся после обрезки область "картинки" клипа. Так что мы всегда сможем оценить, не "отстригли" ли чего лишнего.

Вот и все о задании параметров импортируемого клипа. Да и вообще, с импортом видео мы, пожалуй, закончили. Почти...

Macromedia Flash Video

Macromedia Flash Video — это формат файлов, разработанный фирмой Macromedia специально для хранения видеоклипов, предназначенных для импорта в документы Flash. Файлы Flash Video содержат сжатые с помощью алгоритмов On2 VP6 или Sorenson Spark видео и, возможно, звук и имеют расширение flv.

Преобразовать нужный нам клип в формат Macromedia Flash Video можно двумя способами. Во-первых, можно воспользоваться поставляемой в составе Flash 8 и очень простой в использовании утилитой Macromedia Flash 8 Video Encoder. Во-вторых, имеется возможность экспортировать в этот формат любой видеоклип, уже импортированный в документ Flash.

Чтобы сохранить импортированный клип в формате Macromedia Flash Video, нужно сначала выбрать в списке панели **Library** соответствующий ему образец-импортированный клип. Далее следует выбрать пункт **Properties** контекстного или дополнительного меню и в диалоговом окне **Video Properties** (см. рис. 16.8) нажать кнопку **Export**. На экране появится стандартное диалоговое окно сохранения файла Windows. Вводим в этом окне имя результирующего файла Flash Video и нажимаем кнопку сохранения. Через некоторое время файл Flash Video будет создан.

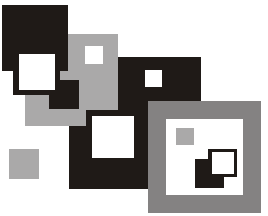
Впоследствии мы можем импортировать файл Flash Video в другой документ Flash. Импортируется он так же, как файлы других форматов. Единственное исключение: диалоговое окно **Wizard** при этом не появляется, так как файл Flash Video не нужно перекодировать при импорте.

Есть еще один способ импортировать клип Flash Video в библиотеку (но не на рабочий лист). Откроем панель **Library** и выберем пункт **New Video** в ее дополнительном меню. В списке появится новый, пока пустой образец-импортированный клип с именем вида `Embedded Video <Порядковый номер>`. Выделим его и выберем в контекстном или дополнительном меню пункт **Properties**, чтобы вызвать на экран диалоговое окно **Video Properties**. В поле ввода, расположенном в верхней части окна, введем имя создаваемого образца. После этого нажмем кнопку **Import**. На экране появится стандартное диалоговое окно открытия файла Windows. Найдем нужный файл Flash Video и нажмем кнопку открытия.

Что дальше?

Итак — все нужное для работы видео импортировано и тихо лежит в библиотеке, дожидаясь своего часа. А чем нам заняться теперь?

Звуком. Аудиоклипами. Пора научиться импортировать в свои Flash-документы звук. В самом деле, что это за фильм без звукового сопровождения!..



Глава 17

Работа со звуком

Как и кинематограф, Интернет прежде был "великим немым". Все Web-странички, присутствовавшие в нем, молчали, как рыбы. Да, они содержали много текстов и рисунков, несущих зачастую очень полезную информацию, но... Тишина, мертвая тишина, не нарушаемая ни одним звуком, ни одним словом, ни одной нотой, царила в великой Всемирной сети, пока в нее не вторглось, ломая все устоявшиеся стандарты и привычные представления, то, что мы называем мультимедиа.

И началась мультимедийная революция. Компьютеры обзавелись звуковыми картами, а пользователи — программами проигрывателей и огромными коллекциями музыки и фильмов, буквально пожравшими их жесткие диски. В программы, предназначенные для создания интернет-документов, в спешном порядке добавлялись средства поддержки мультимедийных данных. А звукозаписывающие компании подняли шум по поводу вопиющего нарушения авторских прав. (Впрочем, это уже совсем другая история...)

Разумеется, Flash 8 также поддерживает импорт и использование звуковых клипов. Так что мы сможем добавить к нашим кинематографическим шедеврам и звуковое сопровождение. Кому сейчас нужно немое кино!..

> ***Внимание!***
>
> Автор в третий раз призывает вас не забывать об авторских правах.

Нужно сразу же сказать одну очень важную вещь. Дело в том, что Flash, хоть он и способен импортировать звуковые клипы, не имеет совершенно никаких инструментов для их создания. Поэтому для подготовки звука нам придется использовать сторонние программы. Их сейчас довольно много, как платных, так и бесплатных, так что проблем с выбором подходящей программы у нас не возникнет.

Что ж, мы готовы к погружению в океан цифрового звука. Так погрузимся же!

Введение в цифровой звук

Начнем, как всегда, с теории. А именно: выясним, как звуковые данные кодируются и сохраняются в файлах и какие форматы записи звука чаще всего используются.

Три способа кодирования звука

Звук может быть закодирован в цифровой формат с использованием трех принципиально различных методов. (Они различаются так же, как покадровая и трансформационная анимация.) Давайте все о них разузнаем.

Прямая оцифровка

Первый метод, применяемый при кодировании звука, — самый простой и очевидный. Источник звука подключается к линейному входу звуковой карты компьютера, после чего звук переписывается на компьютер и сохраняется в файле. Этот метод называется *прямой оцифровкой звука*.

Для кодирования звука методом прямой оцифровки используется способ, называемый *PCM* (Pulse Coded Modulation — импульсно-кодовая модуляция). Суть его в том, что звуковая карта с частотой много тысяч раз в секунду замеряет уровень звукового сигнала и записывает получившиеся значения в массив данных. Этот массив данных и представляет собой оцифрованный звук.

Преимущество у способа PCM одно — звук копируется очень точно, со всеми нюансами. Именно поэтому прямая оцифровка — самый популярный путь к тому, чтобы дать звуку новую, цифровую, жизнь. Все цифровые звуковые данные — компакт-диски, файлы MP3 — записаны именно таким образом.

Недостаток у прямой оцифровки, пожалуй, тоже один — очень большой объем результирующего массива данных. (В самом деле, сколько значений получится, если замерять уровень сигнала, скажем, 44 100 раз в секунду!) Поэтому, чтобы уменьшить его, практически всегда применяется сжатие звука — как правило, с потерями. Для этого используются специальные алгоритмы (и кодеки), "заточенные" под звук.

Алгоритмов сжатия звука в мире существует, по крайней мере, не меньше, чем кодеков, предназначенных для сжатия видео. Давайте рассмотрим самые распространенные из них.

- *ADPCM*. Был разработан фирмой Microsoft для сжатия звуковых файлов WAV еще в начале 90-х годов прошлого века. Обеспечивает не очень сильное сжатие, зато без потерь. Сейчас применяется только для сжатия

очень коротких звуковых клипов, в основном — системных "сигналов" операционной системы Windows.

- MPEG I level 3 (MP3). Был разработан университетом Фраунгофера в Германии в середине 90-х годов для сжатия звуковых файлов. Является версией известного алгоритма сжатия видео MPEG I (*см. главу 1*), специально предназначенной для сжатия звука. В настоящее время — самый популярный формат сжатия звука.
- RealMedia. Был разработан фирмой RealNetwork в середине 90-х годов для распространения видео и музыки через Интернет. Собственно, мы уже рассматривали его в *главе 1*. В настоящее время — один из популярнейших алгоритмов сжатия звука.
- *OGG Vorbis*. Был разработан сообществом независимых программистов в конце 90-х годов как замена MPEG I level 3. В настоящее время потихоньку завоевывает популярность. Программное обеспечение для сжатия звука поставляется бесплатно с открытыми исходными текстами.
- *WMA* (Windows Media — Audio). Еще один конкурент MPEG I level 3. Был разработан фирмой Microsoft в конце 90-х годов. В настоящее время постепенно завоевывает популярность.

Многие из перечисленных выше форматов поддерживаются Flash.

Командное кодирование

Второй метод кодирования звука основан на том принципе организации данных, при котором в файлах хранятся не оцифрованные значения уровня звукового сигнала, а особые команды, указывающие, какой звук и какой продолжительности нужно сгенерировать. При воспроизведении эти команды читаются и выполняются либо специальной программой-проигрывателем, либо соответствующим образом оснащенной звуковой картой компьютера. Это так называемое *командное кодирование звука*.

Для записи файлов, содержащих закодированный таким образом звук, используются специальные программы. Практически всегда методом командного кодирования записывается музыка; команды содержат номер инструмента, высоту и продолжительность ноты. Записать, скажем, речь, шум, старую фонограмму или "живое" исполнение по способу командного кодирования невозможно — для этого нужно использовать прямую оцифровку.

Достоинство командного кодирования — компактность получающихся файлов. В самом деле, чтобы записать пару байтов, составляющих команду, много места не нужно. Поэтому такие файлы очень часто используются для распространения музыки через Интернет.

Недостатков у командного кодирования два. Первый нам уже знаком — не всякий звук можно записать этим способом. Второй недостаток — очень

часто разные программы воспроизводят закодированный таким способом звук по-разному.

Командное кодирование используется, в основном, для записи и тиражирования музыки и имеет достаточно узкое распространение. А форматов записи закодированного командами звука всего два, и вот каких.

- *MIDI.* Расшифровывается как Music Instruments Digital Interface — цифровой интерфейс музыкальных инструментов. Был разработан фирмой Yamaha еще в начале 80-х годов. Используется, в основном, для распространения музыки через Интернет и для создания фонового музыкального сопровождения Web-сайтов.
- *KAR.* Используется для распространения записей караоке. Файлы этого формата практически такие же, как файлы MIDI, но, кроме команд воспроизведения звука, содержат тексты песен. Этот формат довольно популярен, по крайней мере, в русском сегменте Интернета.

Ни один из этих форматов не поддерживается Flash.

Командно-сэмплерное кодирование

Третий метод — своего рода комбинация первого и второго. Он называется *командно-сэмплерным кодированием звука.*

Заключается он в том, что в звуковом файле, как при командном кодировании, сохраняются команды для воспроизведения звука, содержащие номер инструмента, высоту и продолжительность ноты. Но, кроме того, файл хранит также небольшие фрагменты звучания того или иного инструмента, записанные способом прямой оцифровки. Такие фрагменты называются *сэмплами.* Фактически номер инструмента, содержащийся в каждой команде воспроизведения звука, указывает именно на сэмпл.

Проигрыватель, воспроизводя файл, читает команду и извлекает из нее номер сэмпла. Далее он считывает из файла нужный сэмпл и использует его для воспроизведения звука.

> ***НА ЗАМЕТКУ***
>
> В случае командного кодирования для воспроизведения инструментов в настоящее время также используются сэмплы. Однако эти сэмплы принадлежат не самому звуковому файлу, а программе-проигрывателю. Существует много разных проигрывателей, использующих свои специфические сэмплы, поэтому звук, воспроизводимый ими, зачастую различается.

Достоинств у командно-сэмплерного кодирования два. Первое — файлы имеют размер меньший, чем файлы, содержащие оцифрованный напрямую звук (но бóльший, чем звуковой файл, полученный командным кодированием). Второе — звук воспроизводится одинаково разными программами.

Как ни странно, но командно-сэмплерное кодирование звука не снискало большой популярности. Сейчас его используют только особые "малобюджетные" программы создания музыки, называемые *трекерами*. Файлы, создаваемые с помощью трекеров, называются *трекерными модулями*. Трекеры были особенно популярны в 80-х и начале 90-х годов прошлого века; сейчас же они доживают свои дни.

Программ, поддерживающих трекерные модули (кроме, собственно, самих трекеров), очень немного. Flash к ним не относится.

Параметры цифрового звука

Настала пора поговорить о параметрах цифрового звука. Если цифровое видео характеризуется тремя параметрами, то в случае цифрового звука их насчитывается четыре.

> ***Внимание!***
>
> Дальнейший разговор затрагивает только звук, кодированный способом прямой оцифровки. Способы командного и командно-сэмплерного кодирования мы рассматривать не будем — в случае Flash они не актуальны. (Хотя для сэмплов они все же имеют значение, так как сэмплы — суть звуки, оцифрованные напрямую.)

Главный параметр цифрового звука нам уже знаком — это ширина потока данных. Она показывает, сколько места будет отведено для хранения массива звуковых данных, и значит, определяет качество аудиоклипа и размер файла, где он хранится.

Поскольку звуковые данные занимают значительно меньше места, чем данные видео, то значения ширины потока данных для звука меньше, чем для фильмов. Так, для распространения аудиоклипов через Интернет используются значения 96—128 Кбит/с. Если нужно получить клипы небольших размеров, пожертвовав его качеством, используются значения 24—64 Кбит/с. Если же важно именно качество, а размер не имеет особого значения, звук сжимают с шириной потока данных 256—320 Кбит/с, а то и бóльшей, если такие значения поддерживаются кодеком.

Второй по значимости параметр — *частота оцифровки* звука, измеряемая в герцах. Она определяет, сколько раз в секунду звуковая карта измеряет уровень сигнала, и, соответственно, также задает качество цифрового звука.

Для оцифровки высококачественного звука используются значения 44,1 или 48 кГц — это так называемое "CD-качество" (качество компакт-диска). Если нужно получить файл меньшего размера, звук оцифровывают с частотой 22,05 или 24 кГц, а иногда — даже с меньшей. В студиях звукозаписи при

необходимости получить максимальное качество применяют оцифровку с частотой 96 КГц ("профессиональное качество").

Третий параметр — *разрядность* — определяет, какое количество битов отводится под хранение каждого значения уровня сигнала. В настоящее время практически всегда под это отводится 16 бит (два байта); иногда, но очень редко также используется значение 8 бит (один байт). В студиях звукозаписи звук оцифровывают с разрядностью 24 бита (три байта).

Четвертый и последний параметр — *количество каналов*. Оно определяет количество отдельных массивов звуковых данных — *каналов звука*, хранящихся в одном файле. Так, стереофонический звуковой файл, скопированный нами с CD, содержит два таких канала. Разумеется, чем больше каналов звука хранится в файле, тем он больше, и наоборот.

Чаще всего для распространения через Интернет в программировании и Web-дизайне используется одноканальный (монофонический) или двухканальный (стереофонический) цифровой звук. На DVD записывается шестиканальный звук (так называемый "стандарт 5.1"), а самые современные звуковые карты поддерживают воспроизведение даже восьмиканального звука ("стандарт 7.1"). Но это не наш случай, так как Flash поддерживает только одно- и двухканальный звук.

Полезные советы по подготовке звука для импорта

Советы по подготовке цифрового звука практически совпадают с рекомендациями, данными в *главе 16* для цифрового видео.

- ❒ Удаляйте шумы из фонограммы.
- ❒ Выставьте приемлемое значение громкости.
- ❒ Обрежьте ненужные "хвосты". Оставьте только то, что действительно нужно.
- ❒ Задайте подходящие значения параметров окончательной фонограммы. Подберите их, исходя из того, что было сказано выше.
- ❒ Соблюдайте меру!

Программ для работы со звуком существует сейчас предостаточно — наверное, не меньше, чем программ для работы с видео. Среди них встречаются и коммерческие "монстры", и небольшие бесплатные утилиты, зачастую ничем не уступающие первым. Все эти программы вы можете найти в Интернете, в любом файловом архиве; там же вы можете отыскать документацию

к ним. И посетите сайт **http://faqs.org.ru** — там помещено много сборников часто задаваемых вопросов (FAQ, Frequently Asked Questions — часто задаваемые вопросы) по цифровому звуку.

Какие форматы звука поддерживает Flash

А теперь выясним, какие форматы звука поддерживает Flash. Все они перечислены в табл. 17.1.

Таблица 17.1. *Список форматов звука, поддерживаемых Flash*

Название формата	Расширение файлов	Требуется Apple QuickTime 4.0*
AIFF	aif	+
Apple QuickTime	mov	+
MP3	mp3	
Sun AU	au	+
WAV	wav	

* Данная версия или более новая.

Все эти форматы предназначены для записи звука, полученного методом прямой оцифровки. Давайте немного поговорим о некоторых из них.

Формат WAV

Формат *WAV* (от английского *wave* — волна) был создан фирмой Microsoft еще в начале 90-х годов прошлого века в качестве стандартного формата хранения звуковых данных в операционной системе Windows. Сейчас используется также для записи звука, предназначенного для последующей обработки, для хранения сэмплов, помещения звуковых данных в различные программы, на Web-сайты и т. п. Для распространения музыки применяется очень редко. Поддерживается абсолютно всеми программами записи и обработки звука.

Файл формата WAV имеет "говорящее" расширение wav. Звуковые данные могут быть сжаты с использованием практически любого кодека; иногда, если аудиоклип невелик по размерам, их вообще не сжимают.

Судя по всему, формат WAV будет применяться и в дальнейшем. Пока что никаких причин от него отказаться нет.

Формат MP3

Формат MP3 был создан для распространения музыкальных файлов, сжатых кодеком MPEG 1 level 3. В настоящее время — самый популярный формат распространения музыки через Интернет, и не только. Поддерживается абсолютно всеми программами записи и обработки звука, за исключением некоторых узкоспециализированных.

Файл этого формата имеет "говорящее" расширение mp3. Звуковые данные всегда сжимаются с помощью кодека MPEG 1 level 3; использование других кодеков не допускается. Файл MP3 также может хранить дополнительные данные, называемые *тегами MP3*. В частности, с помощью тегов сохраняются имя исполнителя, название композиции, название альбома, год записи альбома, стиль музыки и пр.

Этому формату, похоже, уготована долгая жизнь. Несмотря на то, что многие фирмы — распространители музыкальной продукции настроены против него, MP3 полностью завоевал Интернет и сдавать завоеванные позиции не собирается. Больше того, этот формат не собирается уступать и потенциальным конкурентам (OGG Vorbis, MS WMA и др.), хотя последние, пожалуй, ему не уступят.

Импорт звука

Ну что ж, теория закончилась. Теперь можно уделить время практике — импортировать что-нибудь. И посмотреть, что из этого получится.

Для того чтобы импортировать в документ Flash звуковой файл, достаточно выбрать в подменю **Import** меню **File** один из хорошо знакомых нам пунктов **Import to Stage** или **Import to Library**. Вместо выбора первого пункта также можно нажать комбинацию клавиш <Ctrl>+<R>. На экране появится стандартное диалоговое окно открытия файла Windows; выбираем в нем нужный файл и нажимаем кнопку открытия.

После довольно продолжительного времени файл будет импортирован. Разумеется, он будет помещен в библиотеку в виде образца-звука, имя которого будет совпадать с именем изначального файла. Но на рабочем листе ничего нового не появится — нам придется добавить звук к своей анимации самостоятельно.

Использование импортированного звука

Импортированный звук можно "оприходовать" точно так же, как и любой другой образец, созданный во Flash или импортированный из внешнего файла. Достаточно поместить экземпляр нужного образца-звука в отдельный слой, создать дорожку кадров нужной длины, задать кое-какие дополнительные параметры — и все!

Давайте поместим в готовую анимацию только что импортированный звук.

Создадим новый слой и ничего рисовать на нем не будем. Поместим в этот слой экземпляр образца-звука и растянем единственный ключевой кадр слоя до нужной длины. Дорожка кадров станет такой, как показано на рис. 17.1. Назовем ее *дорожкой звука*.

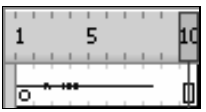


Рис. 17.1. Дорожка звука (дорожка кадров, в которую помещен экземпляр образца-звука)

На рис. 17.1 хорошо видно, что дорожка звука отличается от обычной дорожки, создающей анимацию, тем, что прямо в растянутом кадре, составляющем эту дорожку, отображается небольшой график звука. Чтобы рассмотреть его получше, мы можем увеличить размер прямоугольников, которыми обозначаются отдельные кадры (как это сделать, было описано в *главе 11*).

Кстати, помещение звука в отдельный слой и создание для него отдельной дорожки кадров — обычная практика во Flash-анимации. Так нам самим будет проще работать со звуками, импортированными в фильм.

Теперь можно запустить воспроизведение готовой анимации. Импортированный нами звук будет проигран (если, конечно, на нашем компьютере установлена звуковая карта).

Чтобы временно приглушить звук, мы можем включить пункт-выключатель **Mute Sounds** меню **Control**. Сами звуки при этом никуда не денутся и при отключении этого пункта снова начнут воспроизводиться.

Теперь давайте выделим единственный кадр дорожки звука и посмотрим на панель **Properties**. В правой ее части появится набор элементов управления, с помощью которых мы сможем задать дополнительные параметры звука (рис. 17.2). Рассмотрим их.

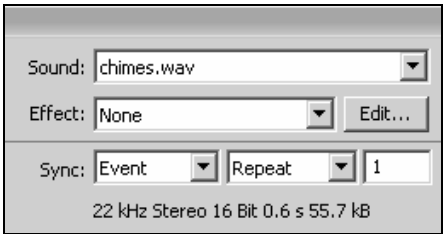


Рис. 17.2. Элементы управления панели **Properties**, позволяющие задать параметры звука

В раскрывающемся списке **Sound** образец-звук, который должен воспроизводиться в данной дорожке. Выбор звука в этом списке — альтернатива созданию экземпляра звукового образца перетаскиванием из панели **Library**, удобная в случае, если эта панель скрыта. Чтобы отменить воспроизведение звука в последовательности кадров, достаточно выбрать в этом списке пункт **None**.

С помощью раскрывающегося списка **Effect** можно задать эффект, который будет применен Flash к воспроизводимому звуку (*звуковой эффект*). Этот список доступен только тогда, когда в списке **Sound** выбран какой-либо образец-звук, и содержит восемь пунктов, о которых мы сейчас поговорим.

Если выбран пункт **None**, никаких звуковых эффектов применено не будет, и звук воспроизведется "как есть".

При выборе пунктов **Left channel** или **Right channel** звук будет воспроизведен соответственно только в левом или только в правом канале. Это значит, что звучать он будет только в левой или правой колонке компьютера.

Если выбран пункт **Fade left to right**, то звук при воспроизведении будет плавно "перетекать" из левого канала в правый. Пункт **Fade right to left** вызывает обратное "перетекание" — из правого канала в левый. Это может быть забавным, если фрагмент изображения перемещается по рабочему листу слева направо или справа налево — звук будет "перемещаться" за ним следом.

Если выбран пункт **Fade in**, громкость звука будет плавно увеличиваться во время воспроизведения. Это может быть полезно, например, в заставке фильма. Пункт **Fade out** выполняет обратное действие — плавное уменьшение громкости звука, что может пригодиться при завершении фильма.

Выбор последнего пункта — **Custom** — позволит нам самим задать, как будет изменяться громкость и местоположение источника звука в пространстве (*панорамирование*) во время его воспроизведения. Как это сделать, мы выясним чуть позже.

С помощью раскрывающегося списка **Sync** задается метод *синхронизации* звука и изображения, т. е. когда звук начнет и закончит воспроизводиться. В этом списке содержатся четыре пункта, которые перечислены ниже.

Если выбран пункт **Event**, звук начнет воспроизводиться, как только Flash выведет на экран содержимое первого кадра дорожки, содержащей этот звук. Звук будет воспроизведен до конца, вне зависимости от того, кончилась ли содержащая его дорожка кадров или нет. Такие звуки в терминологии Flash называются *сигналами* (event).

Часто бывает так, что в одно и то же время воспроизводятся несколько сигналов. В этом случае Flash выполнит их смешивание, или, как говорят, профессиональные звукорежиссеры, *микширование*. Сигналы будут воспроизведены одновременно и, опять же, до конца.

> ***Внимание!***
>
> Воспроизведение сигнала начнется только тогда, когда будет полностью загружен массив его данных. Поэтому, если вы собираетесь выкладывать свои фильмы в Интернет, не делайте сигналы слишком длинными, иначе пользователю, решившему просмотреть ваш фильм, придется очень долго ждать.

Если выбран пункт **Start**, то звук будет воспроизведен так же, как и при выбранном пункте **Event**, за одним исключением: если другой такой же звук уже воспроизводится в другой дорожке кадров, то первый воспроизводиться не будет. Фактически это такой же сигнал, но воспроизводимый в данный момент времени только один раз.

Выбор пункта **Stop** останавливает в данном кадре воспроизведение любого звука. Его можно использовать, чтобы заставить какой-нибудь сигнал "замолчать". Для этого достаточно создать новый пустой ключевой кадр сразу же за последовательностью, где воспроизводится этот сигнал, выделить его, выбрать в списке **Sound** нужный звук, а списке **Sync** — пункт **Stop**.

Сигналы очень часто используются для звукового оформления каких-либо событий. Таким событием может быть переход к какому-то кадру анимации, нажатие кнопки (о кнопках *см. главу 18*) и пр.

Если выбран пункт **Stream**, звук будет воспроизводиться синхронно с анимацией. Когда анимация закончится (все дорожки кадров будут воспроизведены до конца), закончится воспроизведение и этого звука. В терминологии Flash такой звук называется *потоковым* (stream).

Воспроизведение потокового звука начинается сразу же, как только проигрыватель Flash загрузит достаточное для этого количество звуковых данных. Остальная их часть будет загружена уже в процессе воспроизведения. Таким образом, зрителю не придется ждать, когда массив данных потокового звука будет полностью загружен, в отличие от сигнала, который перед началом воспроизведения обязательно должен загрузиться полностью.

Укажем еще на одно отличие потоковых звуков от сигналов. Оно заключается в том, что потоковые звуки микшируются при экспорте готового фильма в формат Shockwave/Flash. Flash "знает", когда именно будет воспроизведен тот или иной потоковый звук, поэтому сможет смикшировать потоковые звуки уже при экспорте, объединив их в новый массив звуковых данных. Проигрывателю Flash останется только воспроизвести готовый, смикшированный звук.

В отличие от потоковых звуков, сигналы могут быть воспроизведены в любой момент времени. В самом деле, кто знает, когда посетитель нажмет кнопку, к которой привязан сигнал, и нажмет ли вообще. Поэтому сигналы включаются в результирующий файл Shockwave/Flash в виде отдельных массивов данных и микшируются самим проигрывателем Flash при просмотре готового фильма.

В виде потоковых звуков в фильмы Flash включают фоновую музыку, голоса персонажей и пр. Обычно размеры таких звуков довольно велики, поэтому не стоит оформлять их в виде сигналов — фильм будет загружаться слишком долго, и посетитель потеряет терпение, ожидая, когда начнется "кино".

С помощью раскрывающегося списка, расположенного правее списка **Sync**, можно задать, какое число раз будет воспроизведен данный звук. Пункт **Loop** этого списка заставляет звук воспроизводиться снова и снова, пока не кончится дорожка звука (зацикливает звук). При выборе же пункта **Repeat** правее этого списка появится поле ввода, в которое мы сможем ввести нужное количество повторений этого звука.

Очень хорошая идея — для создания длинного звука взять короткий фрагмент и заставить его воспроизводиться нужное число раз. Таким образом можно сильно уменьшить размер результирующего файла Shockwave/Flash.

И последнее. Не стоит зацикливать потоковые звуки. Иначе Flash автоматически добавит к дорожке столько кадров, сколько длится каждое повторение этого звука. Результат — сильное увеличение размеров файла Shockwave/Flash, что вряд ли пойдет на пользу фильму.

Параметры звука

Flash содержит средства для задания характера изменения громкости и панорамирования звука в процессе его воспроизведения. Мы, собственно, уже познакомились с простейшими из них — это раскрывающийся список **Effect** панели **Properties** (см. рис. 17.2). Настала пора рассмотреть более сложные и мощные средства.

Давайте выберем в списке **Effect** пункт **Custom** или нажмем кнопку **Edit**, расположенную правее этого списка **Effect**. На экране появится диалоговое окно **Edit Envelope** (рис. 17.3).

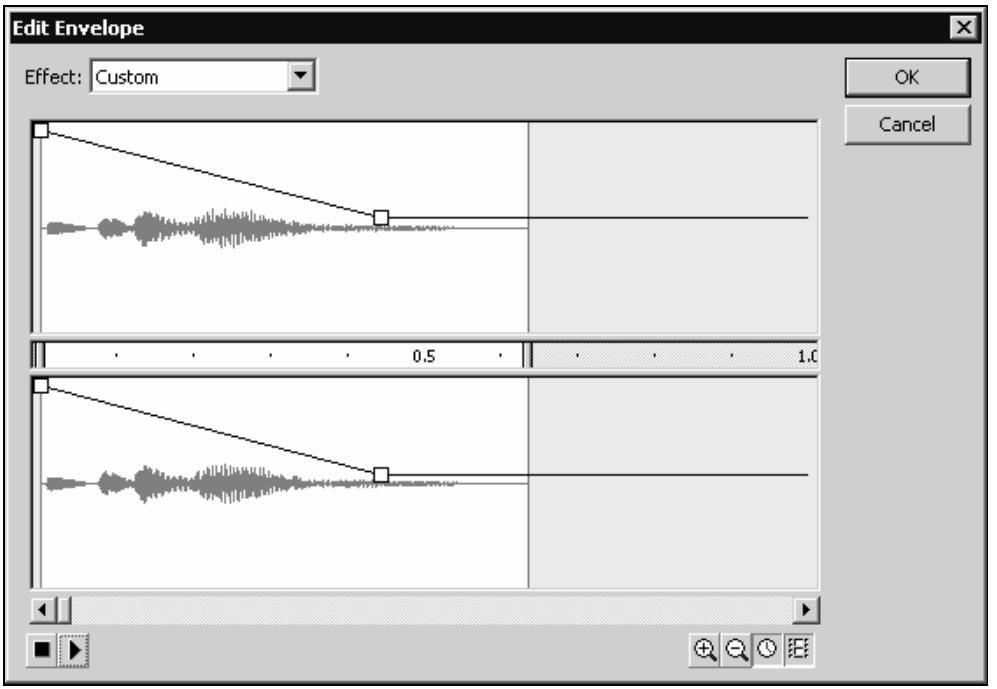


Рис. 17.3. Диалоговое окно **Edit Envelope**

Правка звука выполняется с помощью двух больших графиков, расположенных соответственно в верхней и нижней частях этого диалогового окна. Верхний график позволяет задать характер изменения громкости в левом канале звука, а нижний — в правом канале. Кроме того, в левом и правом нижнем углах этого диалогового окна находятся наборы небольших кнопок, которые весьма помогут нам в работе.

Между этими графиками находится временна́я шкала, проградуированная в секундах. Можно, однако, изменить градуировку с секунд на кадры, включив четвертую слева (последнюю) кнопку-переключатель в группе, расположенной в нижнем правом углу окна (см. рис. 17.3), эта кнопка помечена изображением кинопленки. Чтобы вернуть градуировку в секундах, достаточно включить третью слева кнопку-переключатель (с изображением часов). Первая и вторая слева кнопки, расположенные в этой же группе, позволяют соответственно увеличить или уменьшить масштаб графика.

Над кнопками, ниже графика панорамирования находится полоса прокрутки. С ее помощью можно одновременно прокручивать оба графика по горизонтали. Серые области на графике слева и справа обозначают соответственно начало и конец звучания.

В верхнем левом углу диалогового окна **Edit Envelope** находится раскрывающийся список **Effect**. Он дублирует своего "коллегу" из панели **Properties**.

Задание характера изменения громкости на каком-либо графике выполняется с помощью огибающей. Это тонкая черная линия, хорошо заметная на каждом графике. С ее помощью задаются "взлеты" или "падения" громкости в соответствующем канале нашего звука.

Чтобы переместить какую-либо точку огибающей на графике, нужно щелкнуть на ней мышью. В месте щелчка появится довольно крупный белый квадрат — *маркер огибающей*. Достаточно просто перетащить его на нужное место графика, и огибающая тотчас изменит свою форму.

С помощью огибающей мы можем заставить звук воспроизводиться с постоянно повышающейся или понижающейся громкостью. Также мы можем задавать изменение позиционирования звука — для этого достаточно заставить громкость звука понижаться в одном канале и повышаться — в другом. В общем, здесь все достаточно просто и наглядно.

Теперь еще раз посмотрим на временну́ю шкалу. В крайних левой и правой ее точках хорошо заметны небольшие серые *маркеры начала* и *конца* звука. С помощью этих маркеров мы можем задать точки, где будет соответственно начато и закончено его воспроизведение. И эти точки совсем не обязаны совпадать с изначальными началом и концом импортированного звукового файла.

Пользуясь маркерами начала и конца звука, мы можем воспроизвести наш звук не с начала, а, скажем, с середины, и не до конца, а с обрывом, не производя в нем никаких изменений и не пользуясь внешними программами. В частности, с помощью этих маркеров мы сможем убрать ненужную тишину в начале и конце звучания.

В нижнем левом углу диалогового окна **Edit Envelope** находятся две кнопки, позволяющие воспроизвести полученный в результате правки звук. Правая кнопка (с направленной вправо треугольной стрелкой) запускает воспроизведение звука, а левая (с черным квадратом) — останавливает его.

Конечно же, после окончания работы нужно нажать кнопку **OK**. Если же мы не собираемся сохранять заданные нами параметры, нажмем кнопку **Cancel**.

Работа с образцами-звуками

Мы можем переименовывать и удалять образцы-звуки так же, как и любые другие образцы. Также мы можем преобразовать любой образец-звук в раз-

деляемый для совместного использования. Как это делается, описывалось в *главе 8*.

Также мы можем задать параметры любого образца-звука в диалоговом окне **Sound Properties** (рис. 17.4). Для этого достаточно выделить нужный образец-звук в списке панели **Library** и выбрать пункт **Properties** в контекстном или дополнительном меню этой панели.

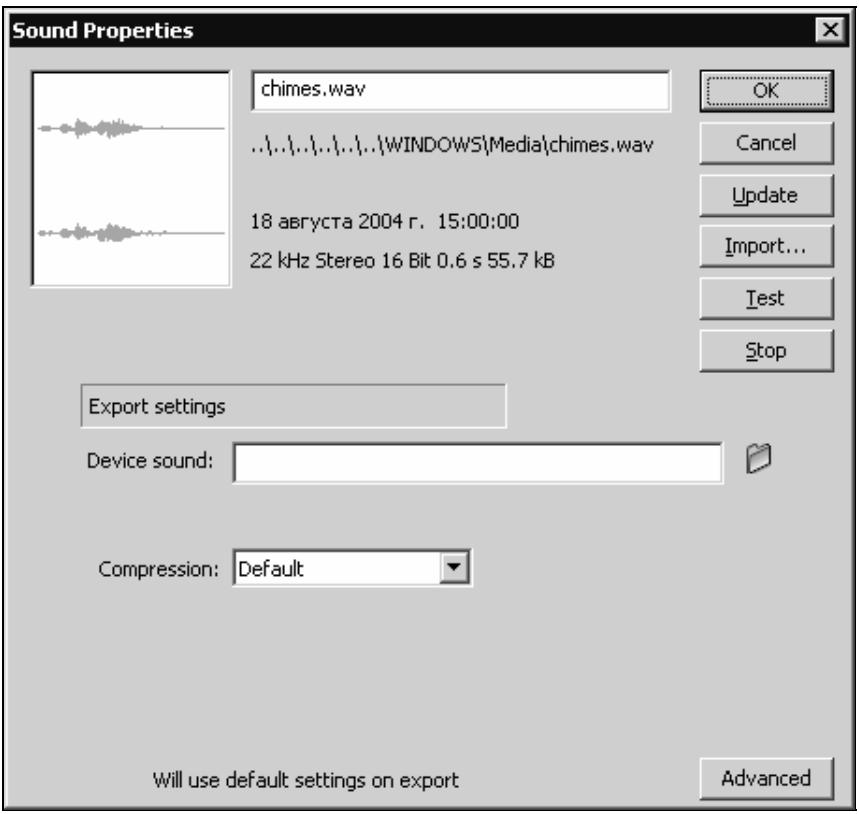


Рис. 17.4. Диалоговое окно **Sound Properties**

Мы можем изменить имя образца-звука, введя его в поле ввода, расположенное в верхней части данного диалогового окна. (Изначально, как мы помним, оно совпадает с именем импортированного звукового файла.)

С помощью кнопки **Import** можно импортировать на место существующего звука другой. При нажатии этой кнопки на экране появится стандартное диалоговое окно открытия файла Windows. Выберем в нем нужный файл и нажмем кнопку открытия.

Нажав кнопку **Test**, мы можем прослушать импортированный звук. Заметим, что он будет воспроизведен в своем изначальном виде. Чтобы остановить воспроизведение, нужно нажать кнопку **Stop**.

В средней части диалогового окна **Sound Properties**, ниже поля ввода имени образца, приведены полное имя файла, из которого был импортирован этот звук, дата создания этого файла и параметры импортированного звука: частота дискретизации, количество каналов (моно или стерео), разрядность, продолжительность в секундах и размер его данных.

Чтобы обновить импортированный звук после его правки во внешней программе, нужно выделить в списке панели **Library** нужный образец-звук и в контекстном или дополнительном меню этой панели выбрать пункт **Update**. На экране появится диалоговое окно **Update Library Items** (см. рис. 8.20). Установим флажок против соответствующего файла в списке образцов этого окна и нажмем кнопку **Update**. После обновления образцов закроем окно **Update Library Items**, нажав кнопку **Close**.

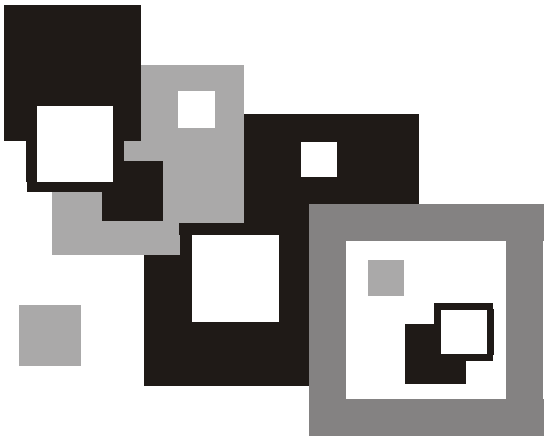
Кстати, то же самое можно выполнить, нажав кнопку **Update** диалогового окна **Sound Properties** (см. рис. 17.4). В этом случае окно **Update Library Items** на экране не появится, и Flash выполнит обновление образца-звука немедленно.

Что дальше?

На этом рассказ об импорте и использовании звука в фильмах закончен. Собственно, на этом можно заканчивать и с анимацией. Третья часть книги о Flash 8 подошла к концу.

Четвертая же ее часть будет полностью посвящена программированию. Мы узнаем о том, что такое сценарии, поговорим о привязке их к кадрам фильма, клипам и кнопкам, а также пройдем краткий курс языка программирования ActionScript. И конечно, мы будем сами писать сценарии, управляющие различными элементами нашего фильма.

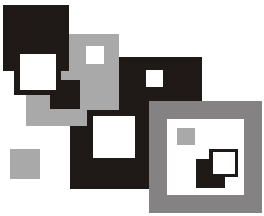
Конечно, углубляться в мир Flash-программирования мы не будем — эта книга посвящена исключительно графике и анимации Flash. Но знать самые начала программирования нам не помешает.



ЧАСТЬ IV

Интерактивность

Глава 18



Основы программирования во Flash

Вот мы и добрались до программирования. Путь был долог и тяжел, но мы его преодолели, расправившись по дороге со статичной и анимированной графикой, импортированным видео и звуком. Перед нами высятся неприступные стены и грозные башни крепости под названием ActionScript. Могучие воины-действия натачивают мечи, а грозные маги-объекты извлекают из тайников древние артефакты... Однако пора заканчивать со стратегическими играми!

В четвертой части книги мы научимся создавать *сценарии* — небольшие программы, являющиеся неотъемлемой частью документа Flash и описывающие поведение какого-либо элемента изображения Flash или самого этого изображения. Мы научимся также привязывать сценарии к кадрам анимации, клипам и кнопкам и, разумеется, научимся создавать сами кнопки.

Сценарии Flash пишутся на языке программирования *ActionScript*, разработанном в фирме Macromedia специально для пакета Macromedia Flash. Этот язык очень прост в освоении и, вместе с тем, имеет достаточно средств для того, чтобы добавить фильмам Flash "жизни". На нем можно даже писать настоящие программы — так называемые Flash-приложения. И те из вас, кто продолжит изучение Flash 8 по его интерактивной справке или другим книгам, смогут стать настоящими Flash-программистами.

С помощью сценариев с фильмом и его элементами можно сделать все, что душе угодно. Например, можно остановить воспроизведение на каком-то кадре, а продолжить только после нажатия кнопки. Можно заставить фрагмент изображения двигаться по рабочему листу, не пользуясь стандартными средствами создания анимации, описанными в *части III* этой книги. Можно заставить Flash открыть при щелчке на кнопке какую-либо Web-страницу. Можно... Да много чего можно сделать с помощью сценариев!

Начнем мы, как уже повелось, с теории. Нам нужно поближе познакомиться со сценариями и выяснить, как (или, точнее — когда) они работают. А заодно познакомиться с понятием события.

Как работают сценарии. События

Любой сценарий, который мы создаем в документе Flash, должен быть *привязан* к какому-то его элементу. Привязать сценарий во Flash можно:

- к кадру анимации (внешней или внутренней, принадлежащей какому-либо образцу);
- к клипу (в смысле — к экземпляру образца-клипа). О клипах и их создании *см. главу 8*;
- к кнопке (к экземпляру образца-кнопки). О кнопках будет рассказано далее в этой главе.

Чтобы привязать к элементу изображения, нужно сначала выделить этот элемент. Далее мы просто открываем панель **Actions** и пишем в ней текст (или, как говорят профессиональные программисты, — код) сценария. Так, если был выделен кадр анимации, то написанный в панели **Actions** сценарий будет привязан к нему. Если же был выделен элемент, к которому нельзя привязать сценарий (например, "обычная" графика или экземпляр графического образца), панель **Action** будет недоступна.

Все сценарии Flash, привязанные к кнопке или клипу, выполняются в ответ на какое-либо *событие*, произошедшее в фильме Flash или операционной системе. Таким событием может быть щелчок на кнопке, "касание" мышью клипа и пр.; набор событий, к которым можно привязать сценарий, весьма обширен. Сценарии, привязанные к событиям, называются также *обработчиками событий*.

Сценарии же, привязанные к кадру анимации, выполняются при воспроизведении данного кадра. Поэтому они называются просто сценариями.

А теперь давайте выясним, как написать сценарий и привязать его к нужному событию нужного элемента документа Flash. Первый наш сценарий будет привязан к кадру фильма; заодно мы познакомимся с панелью **Actions** и научимся с ней работать. А потом перейдем к более сложным случаям — привязке сценариев к клипам и кнопкам. Ну и, конечно, узнаем, как создаются эти самые кнопки.

Привязка сценариев к кадрам анимации

Привязать сценарий к кадру проще простого. Откроем фильм, демонстрирующий деление клетки, который мы создали в *главе 11*, и выделим второй ключевой кадр второй сцены. Мы привяжем к этому кадру сценарий, запускающий воспроизведение данной сцены с первого кадра, и таким образом зациклим наш фильм.

> ***НА ЗАМЕТКУ***
>
> При воспроизведении фильма в отдельном окне Flash по умолчанию его зацикливает. Но фильм, сохраненный в формате Shockwave/Flash, может и не быть зациклен. Поэтому для зацикливания фильмов часто применяют сценарии, наподобие того, какой мы сейчас напишем.

Этот небольшой сценарий будет выглядеть вот так:

```
gotoAndPlay(1);
```

Здесь `gotoAndPlay` — особая команда языка ActionScript, выполняющая переход на кадр текущей сцены, номер которого указан в скобках, и запускающая воспроизведение фильма уже с этого кадра. В нашем случае она выполняет переход на кадр № 1.

Команды такого типа воздействуют на фильм Flash или часть изображения на рабочем листе, причем результат этого воздействия проявляется немедленно. В терминологии Flash они называются *действиями* (actions). А номер кадра, указанный в скобках, называется *параметром*.

Очень часто действия принимают не один параметр, а несколько. В этом случае значения этих параметров разделяются запятыми.

В совокупности действие `gotoAndPlay` с параметром составляют *выражение* — минимальную единицу сценария ActionScript. Символ "точка с запятой" служит признаком конца выражения ActionScript и является обязательным.

Наш первый сценарий очень короток — в нем всего одно выражение. Сложные же сценарии могут содержать тысячи, а то и десятки тысяч выражений.

Знакомство с панелью *Actions*

Теперь нам осталось привязать наш сценарий к кадру. А значит, нам понадобится панель **Actions**. Чтобы вывести ее на экран, нужно:

- либо включить пункт-выключатель **Actions** меню **Window**;
- нажать клавишу <F9>;
- нажать кнопку , расположенную в правой части панели **Properties**.

Панель **Actions** показана на рис. 18.1. Она разделена на три части, которые мы сейчас рассмотрим.

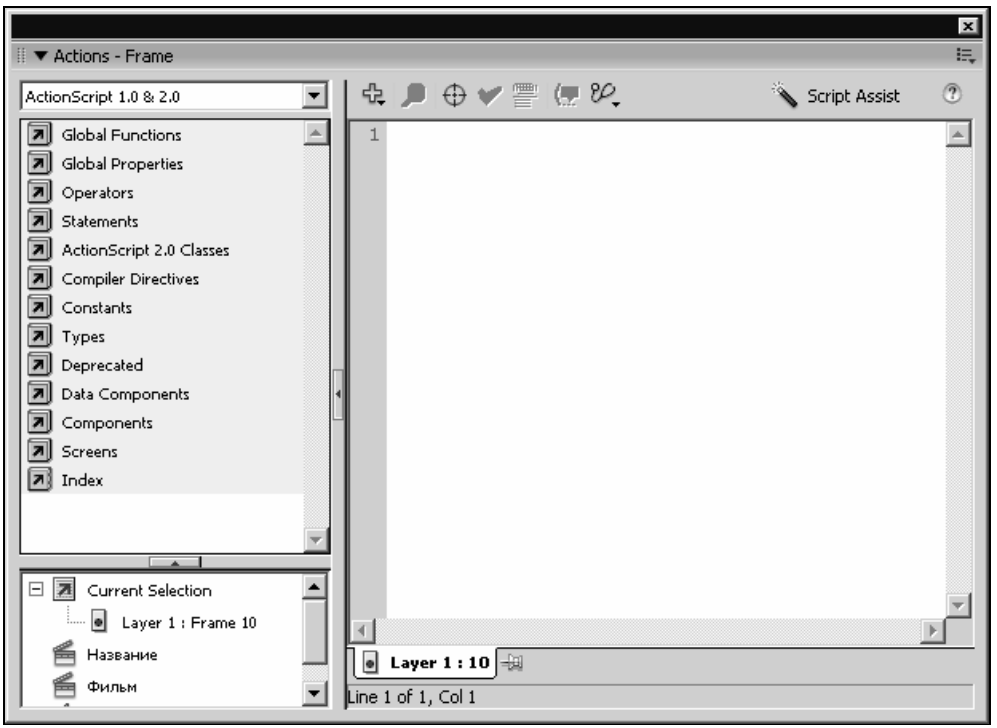


Рис. 18.1. Панель **Actions**

В левой части панели **Actions** находится большой иерархический список доступных в языке ActionScript команд (действий, операторов, методов и пр.; с ними мы познакомимся в *главе 19*), которые мы можем использовать в своих сценариях. Таких команд очень много — так много, что они разбиты на несколько групп, и каждой такой группе соответствует определенное "дерево" иерархического списка. Если в сценарии нельзя использовать какую-либо команду, соответствующий ей пункт списка показан серым цветом и недоступен для выбора.

В данный момент панель **Actions** показывает все сценарии, привязанные к тому элементу фильма (кадру, кнопке или клипу), который сейчас выделен. Чтобы выяснить, какие сценарии привязаны к другому элементу фильма, достаточно просто выделить его на рабочем листе (для кнопки или клипа) или временно́й шкале (для кадра).

Есть и другой способ "путешествовать" по элементам фильма и привязанным к ним сценариям — использование другого иерархического списка, расположенного в левом нижнем углу панели **Actions**. Этот список представляет все элементы страницы, к которым уже были привязаны сценарии. Чтобы просмотреть код сценария, привязанный к тому или иному элементу, достаточно просто выбрать в списке соответствующий ему пункт.

В правой части панели находится большая область редактирования, в которую вводится собственно код сценария. При этом мы можем пользоваться всеми приемами, знакомыми нам по работе с программами текстовых редакторов, такими как перемещение курсора, выделение текста, работа с буфером обмена Windows и пр.

Начинающие программисты смогут получить более подробную информацию о командах языка ActionScript. Для этого им сначала нужно будет либо выбрать соответствующий пункт в иерархическом списке доступных команд (верхняя левая часть панели **Actions**), либо в области редактирования кода установить на нужную команду курсор мыши. После этого им останется нажать кнопку , расположенную в правом верхнем углу панели **Actions**, или выбрать в дополнительном меню панели пункт **Help**. Также они могут щелкнуть по нужному пункту иерархического списка команд правой кнопкой мыши и выбрать в появившемся на экране контекстном меню пункт **View Help**. После этого на экране появится панель **Help** (см. рис. 2.24) с текстом справки по выбранной команде.

Левая (оба иерархических списка) и правая (область редактирования кода) части панели **Actions** отделены друг от друга тонкой серой линией. Мы можем перетаскивать ее мышью, меняя относительные размеры частей. Можно также дважды щелкнуть на этой линии или нажать расположенную на ней небольшую кнопку с изображением черной стрелки, чтобы убрать с экрана оба списка. Повторный щелчок на этой кнопке восстановит левую часть панели.

Кроме того, толстой серой линией отделены друг от друга оба иерархических списка, расположенные в левой части панели. При двойном щелчке на линии или при нажатии расположенной на ней кнопки с изображением стрелки будет убран верхний список (список доступных команд).

Написание сценария

Итак, с панелью **Actions** мы вкратце познакомились. Конечно, полностью ее возможности (кстати, весьма широкие) мы не рассмотрели, но сейчас нам важнее написать наш первый сценарий, чем выяснять, чем богата эта панель. Позже мы займемся ей более подробно.

Выше уже говорилось, что для перемещения бегунка на другой кадр нужно использовать действие `gotoAndPlay`. Все действия находятся в "дереве" **Global Functions** иерархического списка доступных команд (см. рис. 18.1). В этом "дереве" нам нужно раскрыть "ветвь" **Timeline Control** (в ней содержатся действия, управляющие воспроизведением анимации) и найти пункт **gotoAndPlay**, "отвечающий" за одноименное действие. Теперь нам нужно добавить это действие в код нашего сценария, который пока что пуст.

Чтобы добавить какую-либо из перечисленных в иерархическом списке команд языка ActionScript в код сценария, следует выполнить любое из перечисленных ниже действий:

- перетащить соответствующий пункт иерархического списка мышью в область редактирования кода. Нужно только проследить, чтобы он попал в нужное место;
- дважды щелкнуть на пункте списка;
- выбрать пункт **Add to Script** в контекстном меню выбранного пункта;
- нажать расположенную над областью редактирования кода кнопку со знаком "плюс", раскрытое меню которой показано на рис. 18.2, "проследовать" по иерархии подменю и выбрать соответствующий пункт;
- нажать комбинацию клавиш, соответствующую требуемому пункту упомянутого выше меню (рис. 18.2).

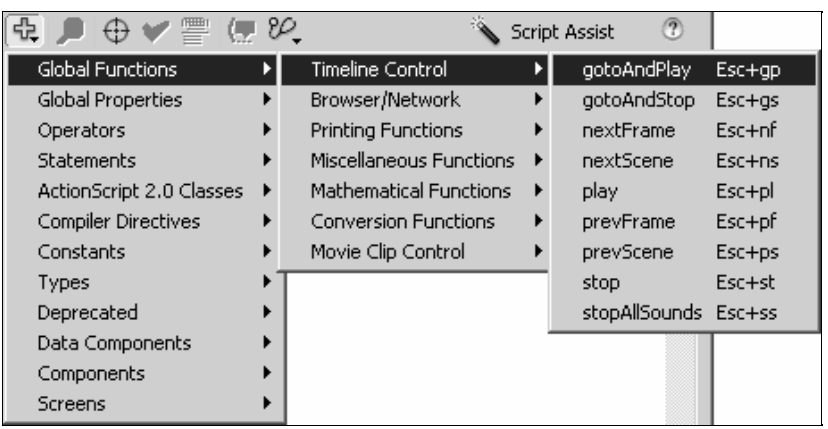


Рис. 18.2. Кнопка добавления команды в код сценария с раскрытым меню

О комбинациях клавиш, служащих для добавления в сценарий команд языка ActionScript, нужно рассказать подробнее. Эти комбинации состоят из трех клавиш: двух алфавитно-цифровых и клавиши <Esc>. Нажимать их нужно

так: сначала — клавишу <Esc>, затем — первую клавишу, потом — вторую. И еще: перед тем как пользоваться этими комбинациями, необходимо переключиться в область редактирования кода сценария — в обоих иерархических списках панели **Actions** эти комбинации не действуют.

Комбинации клавиш — очень полезная вещь. Только по умолчанию они почему-то не отображаются в иерархическом списке доступных команд — наверно, разработчикам Flash это показалось излишним. Чтобы все-таки заставить Flash отображать эти комбинации прямо в этом списке, достаточно включить пункт-выключатель **Esc Shortcut Keys** в дополнительном меню панели **Actions**.

Итак, мы выбрали действие `gotoAndPlay` и добавили его в код сценария одним из описанных выше способов. Это действие тотчас появится в области редактирования кода:

```
gotoAndPlay()
```

Осталось ввести значение параметра действия `gotoAndPlay`, равное 1, возможно, поставить точку с запятой — и первый наш сценарий готов. Он будет таким:

```
gotoAndPlay(1);
```

А кадр, к которому мы привязали сценарий, будет выглядеть так, как показано на рис. 18.3. Хорошо заметна буква `a` — она обозначает, что кадр содержит сценарий, который будет выполнен при достижении этого кадра.

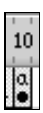


Рис. 18.3. Кадр анимации, к которому был привязан сценарий

В этом примере мы поместили свой сценарий в тот же слой, в котором находится наша анимация. Но если таких сценариев много, рекомендуется помещать их в отдельный слой под названием `Сценарии` или `Actions`. Так нам будет проще управляться со всеми созданными нами сценариями, которых в сложном фильме может быть не один десяток.

На этом создание нашего первого сценария можно считать законченным. Проверим его в действии.

Воспроизведение фильмов, кадры которых содержат сценарии

Есть одна тонкость, которую необходимо учитывать при воспроизведении фильмов, кадры которых содержат сценарии. Дело в том, что Flash способен

выполнять такие сценарии прямо во время воспроизведения, и это обстоятельство может очень помочь при их отладке. Правда, наше замечание касается только простейших сценариев, наподобие того, что мы только что написали.

Если мы сейчас начнем воспроизведение нашего фильма, выбрав пункт **Play** меню **Control** или нажав клавишу <Enter>, то Flash корректно обработает наш сценарий и выполнит переход на первый кадр сцены `Фильм` по достижении ее последнего кадра. Если же он этого не сделает, нужно проверить, включен ли пункт-выключатель **Enable Simple Frame Actions** меню **Control**. Как раз этот пункт при включении предписывает Flash выполнять сценарии, привязанные к кадрам.

Привязка сценариев к клипам

Привязать сценарий к кадру, как мы только что убедились, несложно. А как насчет клипа?

Чтобы привязать сценарий к клипу, нам будет нужно дать ему имя. Для этого достаточно его выделить и ввести нужное имя в поле ввода, расположенное в левой части панели **Properties**, под раскрывающимся списком доступных типов экземпляра.

Для экспериментов возьмем фильм, в котором демонстрировались возможности составной анимации (*см. главу 13*). (Помните этот фильм: прямоугольник, кувыркаясь, летит по рабочему листу?) Загрузим содержащий его документ. И выполним подготовительные действия.

1. Преобразуем экземпляр графического образца в экземпляр-клип, если не сделали этого раньше. Для чего выберем в раскрывающемся списке, расположенном в верхнем левом углу панели **Properties**, пункт **Movie Clip**.
2. Дадим клипу имя, например, `clip`.

Теперь все готово для написания сценария.

Давайте сделаем так, чтобы при щелчке мышью на вложенном клипе он останавливал свое воспроизведение, переставая крутиться. Конечно, пользы от такой интерактивности никакой, но, по крайней мере, это будет наш первый опыт привязки сценария к клипу. Мы ведь пока только учимся писать сценарии...

Итак, к делу! Выделим клип `clip` на рабочем листе. И вспомним то, что говорилось в начале этой главы о событиях.

В случае кадра нам было достаточно просто выделить его и написать сценарий, который выполнится при достижении этого кадра. Клип же поддерживает множество событий: щелчок мышью, "касание" мышью, буксировка и др.

Так что нам придется указать, к какому из поддерживаемых клипов событий мы привязываем наш сценарий. Это указание производится с помощью особого действия `onClipEvent`, в качестве параметра которого задается событие, в ответ на которое будет выполняться сценарий.

Полный "шаблон" для написания сценария-обработчика события будет иметь такой вид:

```
onClipEvent(<Обозначение события>)
{
  <Код сценария-обработчика>
}
```

Первая строка этого "шаблона" задает событие, на которое "откликается" обработчик, и называется *заголовком*. Сам же код обработчика помещается в фигурных скобках после заголовка и называется *телом*.

Давайте же, поглядывая на приведенный выше "шаблон", напишем свой сценарий. Он будет таким:

```
onClipEvent(mouseDown)
{
  stop();
}
```

В качестве параметра мы передаем действию `onClipEvent` обозначение события `mouseDown`. Это событие возникает, когда посетитель щелкает на клипе мышью. А действие `stop` вызывает немедленную остановку воспроизведения анимации клипа, к которому привязан содержащий это действие сценарий.

Здесь нужно сказать вот еще что. Все действия, помещенные нами в привязанные к клипу сценарии, будут управлять анимацией именно этого клипа. Поэтому действие `stop` в приведенном выше сценарии остановит внутреннюю анимацию клипа `clip`, а не анимацию самого фильма.

Теперь выделим наш клип, если еще не сделали этого, и вызовем на экран панель **Actions**. Введем в область редактирования кода приведенный выше код сценария-обработчика события. И запустим готовый фильм на воспроизведение.

Выше говорилось, что Flash при воспроизведении фильмов может выполнять простейшие сценарии, привязанные к кадрам. Сейчас же мы привязали сценарий к клипу, а не к кадру, а такие сценарии Flash выполнить не сможет. Поэтому нам придется выбрать пункт **Test Movie** меню **Control** или нажать комбинацию клавиш <Ctrl>+<Enter>, чтобы запустить воспроизведение фильма в отдельном окне.

Давайте, после того как начнется воспроизведение фильма, попробуем щелкнуть мышью на вращающемся прямоугольнике. Если нам все-таки удастся это сделать, он тотчас прекратит "разваливаться" пополам и "застынет" в таком виде, пока анимация не закончится.

Ура, наш первый обработчик события работает! Давайте его немного усложним. Снова выделим клип `clip`, откроем панели **Actions** и допишем в сценарий еще одно выражение. Наш сценарий примет такой вид (добавленное выражение выделено полужирным шрифтом):

```
onClipEvent(mouseDown)
{
  stop();
  _parent.stop();
}
```

Специальная команда языка ActionScript `_parent` обозначает клип, в который вложен клип, к которому привязан этот сценарий — в нашем случае это будет основной фильм. (Такие команды, кстати, называются *ключевыми словами*.) Ну, а записанное после этого слова и отделенное от него точкой действие `stop` нам уже знакомо.

Что же мы получим в результате? А вот что. Проигрыватель Flash прочитает сначала ключевое слово `_parent`, а потом — действие `stop`. И поймет, что действие `stop` нужно распространить на основной фильм — ведь именно в него вложен клип `clip`. Так что добавленное нами выражение остановит внешнюю анимацию.

Привязка сценариев к кнопкам

Кнопки Flash мы пока еще подробно не рассматривали, не считая краткого упоминания в *главе 8*. Настала пора поговорить о них подробно.

Кнопки (или экземпляры образцов-кнопок) Flash ведут себя подобно обычным кнопкам, которые мы встречаем практически во всех приложениях Windows. Они могут быть нажаты, в результате чего всегда выполняется какой-либо сценарий.

Используются кнопки обычно для того, чтобы предоставить зрителю (в частности, посетителю Web-сайта) возможность выполнить какое-то действие: запустить или остановить воспроизведение анимации, открыть Web-обозреватель и загрузить в него какую-то Web-страницу и пр. В интерактивных фильмах Flash кнопки встречаются очень часто.

Создание кнопок

Прежде чем привязывать к кнопке сценарий, ее нужно создать. Займемся этим.

Давайте продолжим эксперименты с фильмом, показывающим вращающийся прямоугольник. А именно, добавим к нему кнопку, останавливающую анимацию. Ведь остановить анимацию, щелкнув по неподвижной кнопке, проще, чем пытаться "поймать" мышью движущийся клип.

Сначала нам нужно создать образец-кнопку. Делается это уже знакомым нам из *главы 8* способом. Выберем пункт **New Symbol** меню **Insert**, в поле **Name** диалогового окна **Create New Symbol** (см. рис. 8.2) введем имя создаваемого образца-кнопки (дадим ему имя `Кнопка`), включим переключатель **Button** в группе **Behavior** и нажмем кнопку **OK**. После этого Flash откроет вновь созданный, пока еще "пустой" образец в режиме правки. Нам останется только нарисовать нашу кнопку.

Но не будем спешить это делать. Посмотрим лучше на временну́ю шкалу (рис. 18.4). Она содержит всего четыре позиции, и весьма странных — все они имеют имена: `Up`, `Over`, `Down` и `Hit`. В позиции `Up` уже создан ключевой кадр.

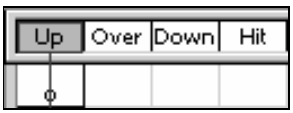


Рис. 18.4. Временна́я шкала образца-кнопки

Дело в том, что эти в этих позициях создаются ключевые кадры, которые содержат четыре разных изображения, используемых Flash для отображения различных состояний кнопки. Давайте их перечислим:

- изображение, находящееся в кадре `Up`, выводится, когда кнопка не нажата и курсор мыши не установлен над ней;
- изображение, находящееся в кадре `Over`, выводится, когда курсор мыши установлен над кнопкой, но сама кнопка не нажата;
- изображение, находящееся в кадре `Down`, выводится в тот момент, когда пользователь нажимает кнопку;
- изображение, находящееся в кадре `Hit`, задает область, которая будет откликаться на касания и щелчки мышью (*"горячую область"* кнопки). Это изображение никогда не выводится на экран.

Заметим, что обязательным является только изображение в кадре `Up`. Без остальных можно обойтись. Их отсутствие восполнят следующие правила.

- Если отсутствует кадр `Hit`, "горячую область" определяет изображение в кадре `Up`.
- Если отсутствует кадр `Down`, для отображения нажатой кнопки Flash возьмет изображение из кадра `Over`.
- Если отсутствует кадр `Over`, для отображения кнопки, которой "коснулся" курсор мыши, Flash позаимствует изображение из кадра `Up`.

Кнопки Flash имеют еще одну интересную особенность. Дело в том, что "горячая" область кнопки совсем не обязательно должна совпадать с самой кнопкой. Так, иная кнопка может сработать, если пользователь щелкнет совсем в другом месте рабочего листа. Это обстоятельство позволяет создавать весьма забавные и полезные вещи вроде подсказок, "всплывающих" при наведении мыши на какой-либо фрагмент изображения, и пр.

Но давайте все-таки нарисуем нашу кнопку. Не станем особенно мудрить — нарисуем в кадре `Up` обычный синий прямоугольник. На первый раз этого будет достаточно.

Следующий шаг — рисование изображения в кадре `Over`. Пусть это будет красный прямоугольник. Создадим в делении `Over` новый ключевой кадр; Flash автоматически поместит туда изображение из кадра `Up`, и нам останется только перекрасить его в красный цвет.

Нарисуем содержимое остальных двух кадров кнопки. В кадре `Down` прямоугольник пусть будет темно-синим. А цвет изображения в кадре `Hit` роли не играет.

Все — образец-кнопка готов. Вернемся в режим правки фильма и откроем панель **Library**. Создадим новый слой и назовем его `Кнопка` — именно в него мы и поместим нашу кнопку, иначе анимация в первом слое окажется "оборванной". Найдем в списке образцов только что созданную нами кнопку и поместим в нижний правый угол рабочего листа ее экземпляр. И, наконец, проверим ее в действии.

Воспроизведение фильмов, содержащих кнопки

Воспроизведение фильмов, содержащих кнопки, также имеет некоторые особенности. Flash способен сделать так, что мы сможем нажимать кнопки прямо во время работы над фильмом (конечно, привязанные к ним сценарии работать не будут). Это поможет проверить правильность всех наших действий.

Чтобы задействовать режим обработки кнопок на рабочем листе, нам нужно будет включить пункт-выключатель **Enable Simple Buttons** меню **Control**. Также можно нажать комбинацию клавиш <Ctrl>+<Alt>+<B>. После этого,

если мы поместим курсор мыши над кнопкой, она сменит свой цвет на красный. Мы даже сможем ее нажать (правда, привязанный к кнопке сценарий в этом случае не будет выполнен).

Единственный недостаток такой кнопки: мы не сможем ее выделить. Для этого нам придется отключить пункт-переключатель **Enable Simple Buttons** меню **Control**.

Естественно, при воспроизведении фильма в отдельном окне кнопки будут работать всегда. Также в этом случае будут работать и все привязанные к ним сценарии.

Добавление звуков к кнопкам

Мы имеем возможность, кроме всего прочего, привязывать к кнопкам звуки. Звуки могут быть привязаны к разным состояниям кнопки, в результате чего, например, при наведении курсора мыши на кнопку будет воспроизведен один звук, а при щелчке на кнопке — другой. Сейчас мы выясним, как это делается.

Прежде всего, импортируем требуемые звуки в библиотеку. (О том, как это делается, было подробно рассказано в *главе 17*.) Дадим получившимся в результате импорта образцам-звукам "говорящие" имена, например: `Касание` и `Нажатие`. Все — можно приступать к "озвучке" кнопки.

Откроем нужный образец-кнопку в режиме правки. Создадим новый слой для звуков и назовем его `Звук-касание`. После этого выделим деление `Over` временно́й шкалы, создадим в нем ключевой кадр и поместим на рабочий лист экземпляр образца-звука `Касание`. Результат этих действий показан на рис. 18.5.

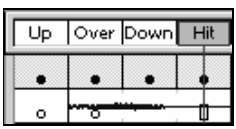


Рис. 18.5. Звук, привязанный к состоянию `Over` кнопки

Чтобы привязать звук к состоянию `Down` кнопки, нам нужно будет создать еще один слой — `Звук-нажатие`. После этого останется только создать ключевой кадр в делении `Down` и поместить в него звук `Нажатие`.

Теперь можно вернуться в основной фильм и проверить "озвученную" кнопку в действии. Проще всего это сделать, запустив воспроизведение фильма в отдельном окне.

Написание сценариев, привязанных к кнопкам

Уф! Наконец-то кнопка создана. Теперь можно заняться собственно привязкой к ней сценария.

Сценарии привязываются к кнопке точно так же, как и к клипу. Мы выделяем на рабочем листе нужную кнопку, даем ей имя (скажем, `button`), воспользовавшись панелью **Properties**, вызываем на экран панель **Actions** и пишем в ее области редактирования код нужного сценария. Различия имеются только в наименовании событий и синтаксисе заголовков их обработчиков.

Так, "шаблон" для написания сценария-обработчика события кнопки имеет следующий вид:

```
on(<Обозначение события>)
{
  <Код сценария-обработчика>
}
```

Обратим внимание, что вместо действия `onClipEvent` в этом случае используется действие `on`. Это первое отличие. Второе же отличие заключается в том, что для обозначения события нажатия кнопки нам будет нужно использовать слово `press`, а не `mouseDown`.

Учитывая сказанное, напишем сценарий для кнопки. Он будет таким:

```
on(press)
{
  stop();
}
```

Выражение, записанное в теле этого сценария, нам давно знакомо. Оно остановит анимацию в основном фильме, так как сценарии, привязанные к кнопке, действуют на тот клип, в котором эта кнопка находится (в нашем случае — это основной фильм). Это третье отличие сценариев, привязанных к кнопкам, от сценариев, являющихся "собственностью" клипов.

Введем этот сценарий в область редактирования кода панели **Actions** и запустим воспроизведение фильма в отдельном окне. После этого щелкнем по кнопке, и вся анимация — и внешняя, и внутренняя — остановится.

Работа с панелью *Actions*

С панелью **Actions** мы уже знакомы. (Еще бы — сколько сценариев с ее помощью мы уже написали!.. Целых четыре!) Давайте посмотрим, что она нам еще может предложить, продолжим знакомство, так сказать.

Как уже говорилось, мы можем использовать при вводе кода в область редактирования этой панели любые приемы, знакомые нам по работе в программах редактирования текста, в частности — операции с буфером обмена Windows, "откат" последнего совершенного действия и "откат отката".

Кроме того, панель **Actions** предоставляет нам дополнительные инструменты, которые сильно облегчат наш труд. Рассмотрим самые полезные из них.

Подсказка по коду. Когда мы открываем круглую скобку, готовясь ввести значение параметра какого-либо действия (например, `gotoAndPlay`), чуть ниже текстового курсора появляется небольшое светло-желтое окошко с текстом, описывающим назначение этого параметра (рис. 18.6). После того как мы введем значение этого параметра и поставим запятую, подсказка выдаст описание следующего параметра, и т. д. Щелкая на небольших стрелках, находящихся в окне подсказки и направленным вправо и влево, можно просмотреть описания соответственно следующего и предыдущего параметров.

Рис. 18.6. Подсказка по коду

Подсказки-списки. Если мы откроем скобку, чтобы задать параметр, значения которого должны выбираться из ограниченного списка, на экране появится этот самый список со значениями (рис. 18.7). Нам останется только выбрать в списке нужное значение либо щелчком мыши, либо выделив его клавишами-стрелками и нажав клавишу <Enter>.

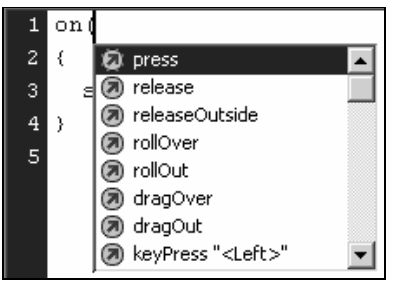


Рис. 18.7. Подсказка-список

Любую из этих подсказок можно вызвать на экран, поставив в нужное место кода текстовый курсор и нажав кнопку , расположенную над областью редактирования кода. Также можно выбрать пункт **Show Code Hint** в дополнительном меню панели или, что проще всего при работе с клавиатурой, нажать комбинацию клавиш <Ctrl>+<пробел>.

Автоматическое создание отступов при вводе строк кода. (Отступы очень часто применяются, например, для выделения тела обработчика события.) Если мы ввели строку с отступом от ее начала, то при нажатии клавиши <Enter> новая строка будет иметь такой же отступ. Отступы формируются также, если предыдущая строка заканчивается круглой или фигурной скобкой.

Можно установить отступ вручную, для чего достаточно целиком выделить нужные строки кода и нажать клавишу <Tab>. Чтобы убрать отступ, нужно также выделить строки и нажать комбинацию клавиш <Shift>+<Tab>.

Проверка набранного кода на предмет синтаксических ошибок. Чтобы включить проверку, нужно нажать кнопку , расположенную над областью редактирования кода. Также можно выбрать пункт **Check Syntax** в дополнительном меню панели или нажать комбинацию клавиш <Ctrl>+<T>. Flash проверит введенный нами код на правильность и выведет окно-предупреждение с текстом "The script contains no errors" (если сценарий не содержит ошибок) или "This script contains errors. The errors encountered are listed in Output Panel" (если сценарий содержит ошибки).

Если сценарий содержит ошибки, Flash также выведет на экран небольшую панель **Output**, в которой будут приведены описания всех найденных ошибок (рис. 18.8). Правда, эти описания будут даны по-английски...

Встроенные средства поиска и замены подстрок в коде сценария. Рекомендуются к использованию в случае, когда сценарий очень велик, а нам нужно быстро найти и заменить параметр какого-либо действия.

Для выполнения поиска подстроки нужно нажать кнопку , расположенную над областью редактирования кода. Можно также выбрать в дополнительном меню панели пункт **Find and Replace** или нажать комбинацию клавиш <Ctrl>+<F>. В результате этого на экране появится диалоговое окно **Find and Replace** (рис. 18.9).

В поле ввода **Find what** зададим искомую подстроку, включим флажок **Match case**, если желаем, чтобы Flash учитывал при поиске регистр символов, и нажмем кнопку **Find Next**. Flash выделит первую строку кода, в которой имеется заданная нами подстрока. Если подстрока так и не будет найдена, Flash выведет окно-предупреждение с текстом "Cannot find the string '<*Искомая подстрока*>'".

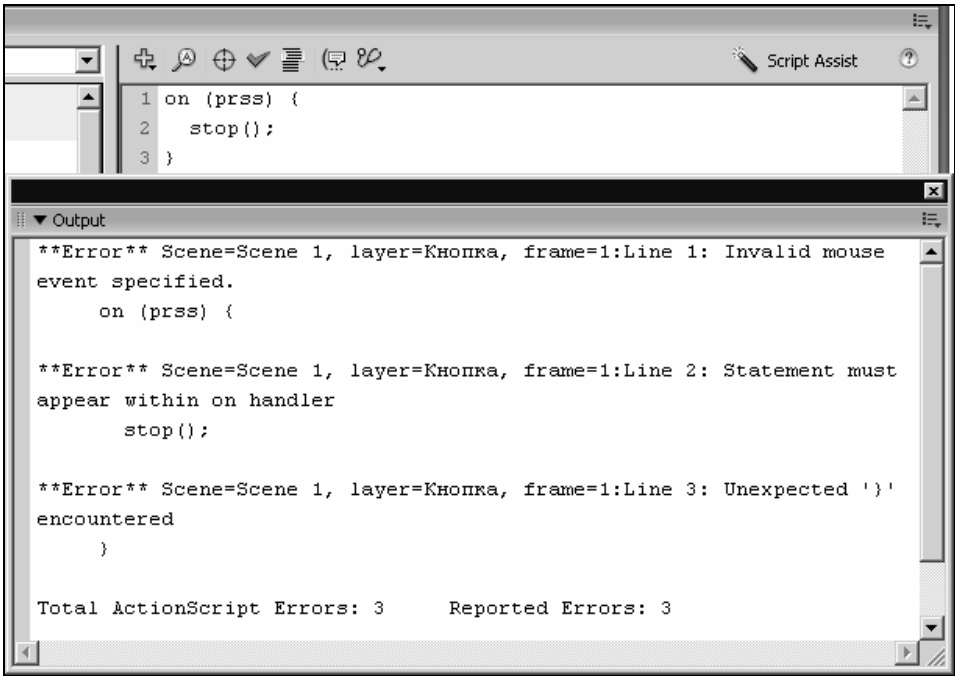


Рис. 18.8. Панель **Output** с описанием найденных ошибок (неправильно задано наименование события в заголовке обработчика)

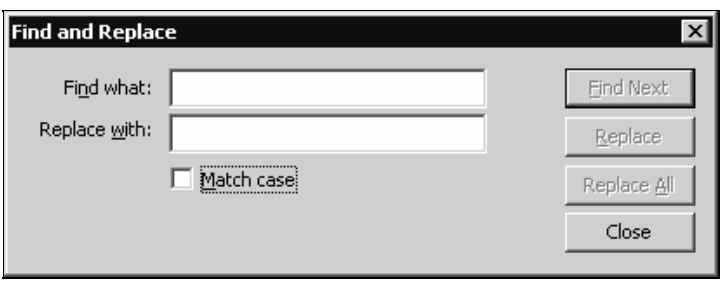


Рис. 18.9. Диалоговое окно **Find**

Чтобы продолжить поиск введенной подстроки далее по тексту сценария, мы снова нажмем кнопку **Find Next** диалогового окна **Find**. Если мы уже закрыли это окно (для чего достаточно нажать кнопку **Close**), то выберем в дополнительном меню панели пункт **Find Again** или нажмем клавишу <F3>.

Для выполнения замены искомой подстроки на другую служит все то же окно **Find and Replace**. В этом случае после задания искомой подстроки нужно будет ввести в поле **Replace with** подстроку, которая заменит иско-

мую. Опять же, можно установить флажок **Match case**, чтобы Flash учитывал при поиске регистр символов. Замена запускается нажатием кнопки **Find Next**. Flash выделит первую строку кода, в которой встретилась искомая подстрока. После этого мы можем нажать кнопку **Replace**, чтобы выполнить замену. Для продолжения поиска нужно снова нажать кнопку **Find Next**.

Мы можем выполнить замену сразу всех найденных в коде подстрок. Для этого сразу же после ввода значений в поля ввода окна **Replace** следует нажать кнопку **Replace All**.

Автоматическая нумерация строк сценария. Чтобы воспользоваться этим средством, достаточно включить пункт-выключатель **Line Numbers** в дополнительном меню панели или нажать комбинацию клавиш <Ctrl>+ +<Shift>+<L>.

Быстрый переход на строку кода с заданным номером. Выбираем пункт **Go to Line** дополнительного меню панели или нажимаем комбинацию клавиш <Ctrl>+<G>, вводим номер нужной строки в поле ввода **Line number** диалогового окна **Go to Line** (рис. 18.10) и нажимаем кнопку **OK**.

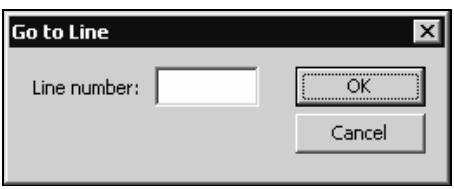


Рис. 18.10. Диалоговое окно **Go to Line**

Автоматический перенос слишком длинных строк. От него можно избавиться, отключив пункт-выключатель **Word Wrap** дополнительного меню панели или нажав комбинацию клавиш <Ctrl>+<Shift>+<W>. Правда, в этом случае нам придется часто пользоваться горизонтальной полосой прокрутки, чтобы просмотреть слишком длинные строки кода.

Печать всех сценариев, созданных в документе. Выбираем пункт **Print** дополнительного меню панели. На экране появится стандартное диалоговое окно задания параметров печати Windows; задаем эти параметры и нажимаем кнопку **OK**.

Поведения Flash и их использование

Специально для начинающих программистов Flash представляет набор так называемых *поведений* (behaviors). Это набор сценариев ActionScript, которые

мы можем привязать к кадру, клипу или кнопке без необходимости набирать их вручную в панели **Actions**. Таких поведений в составе Flash поставляется довольно много; они служат для управления анимацией, клипами, звуком и пр.

Работа с поведениями — создание, правка и удаление — выполняется с помощью панели **Behaviors** (рис. 18.11). Чтобы вывести эту панель на экран, нужно включить пункт-выключатель **Behaviors** меню **Window** или нажать комбинацию клавиш <Shift>+<F3>.

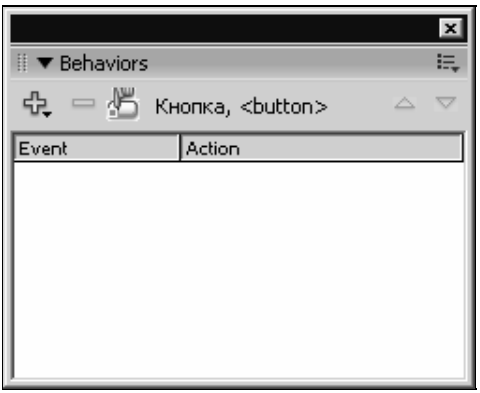


Рис. 18.11. Панель **Behaviors**

Бóльшую часть этой панели занимает список поведений, уже привязанных к выделенному элементу фильма: кадру, клипу или кнопке. (На рис. 18.11 этот список пуст.) Он организован в виде таблицы из двух колонок: **Event** (событие, которое обрабатывает поведение) и **Action** (поведение, выполняемое в ответ на событие).

Чтобы создать поведение, привязанное к какому-либо элементу фильма, нужно выделить этот элемент (его имя, равно как и имя образца, на основе которого он был создан, появятся над списком поведений) и нажать на кнопку с изображением знака "плюс" панели **Behaviors**. На экране появится небольшое меню (рис. 18.12), содержащее все доступные поведения, сгруппированные в *категории*. Категории представляют собой подменю, а сами поведения — пункты этих подменю.

После выбора пункта, соответствующего нужному поведению, на экране появится диалоговое окно, где нужно будет задать параметры этого поведения. Поведение будет создано после нажатия кнопки **OK** этого окна. А в списке панели **Behaviors** появится соответствующая этому поведению строка.

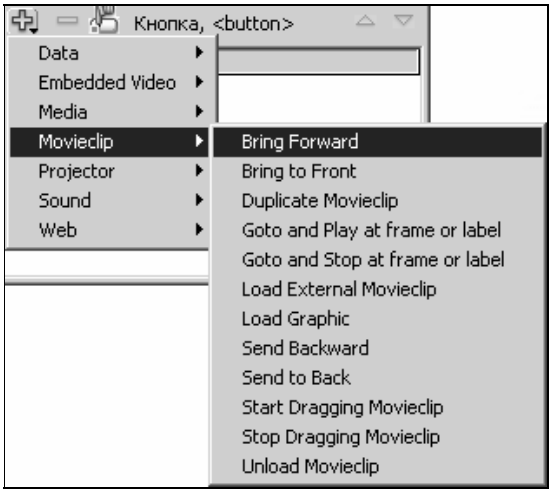


Рис. 18.12. Меню категорий панели **Behaviors**

Если поведение было привязано к клипу или кнопке, нужно будет задать событие, в ответ на которое будет выполняться сценарий этого поведения. Для этого достаточно выбрать нужную строку в списке панели **Behaviors** и щелкнуть внутри колонки **Event** этой строки. После этого там появится небольшой раскрывающийся список (рис. 18.13), в котором и выбирается нужное событие.

Рис. 18.13. Список событий панели **Behaviors**

Для правки параметров поведения достаточно дважды щелкнуть на соответствующей ей строке списка, после чего на экране появится диалоговое окно задания параметров этого поведения. А чтобы удалить ненужное поведение, следует выбрать соответствующую ему строку списка и нажать кнопку с изображением знака "минус".

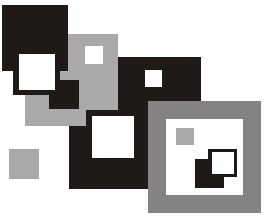
Хоть поведения и создавались для помощи начинающим программистам, но для понимания принципов их создания нужно все же познакомиться с языком программирования ActionScript. Так что мы не будем здесь подробно знакомиться со всеми этими поведениями, а лучше перейдем к рассмотрению этого языка. После знакомства с ActionScript понять, что делает и как создается то или иное поведение, не составит никакого труда.

Что дальше?

Ну что ж, мы сделали самый первый шаг в мире Flash-программирования. Мы усвоили новые термины, изучили новые инструменты и были потрясены открывшимися перед нами новыми возможностями. И мы написали целых четыре сценария! Хоть они и очень просты, но ведь это все-таки наш первый шаг...

Для продолжения занятий по программированию во Flash нам не хватает знания самого языка ActionScript. Поэтому всю следующую главу мы посвятим изучению этого весьма мощного языка. А попутно множество других сценариев, на этот раз — более сложных.

Глава 19



Язык ActionScript

В этой главе мы изучим язык ActionScript, используемый для написания сценариев Flash. Мы подробно разберем принципы написания сценариев и команды языка, которые для этого используются. Также мы познакомимся с механизмами, предоставляемыми Flash для управления различными элементами фильма: клипами, кнопками, полями ввода, звуком и пр.

Эта глава действительно велика. Так что не будем тратить время зря.

Начала языка ActionScript

В этом разделе мы рассмотрим основные понятия языка программирования ActionScript — то, без чего нельзя приступать к написанию полноценных сценариев.

Основные понятия ActionScript

Сценарий — это последовательный набор выражений, которые могут содержать действия — команды, воздействующие непосредственно на фильм. Также в выражениях могут быть ключевые слова — особые слова языка ActionScript, имеющие специальное значение, но не воздействующие на фильм непосредственно. Каждое выражение ActionScript должно завершаться знаком "точка с запятой". Все это нам уже знакомо из *главы 18*.

Вот типичный пример сценария:

```
on(press) {
  stop();
}
```

Это обработчик события press, которое возникает в кнопке при щелчке по ней. Он останавливает воспроизведение анимации.

А вот еще один обработчик события:

```
onClipEvent(mouseDown) {
  b = a + 2;
  gotoAndPlay(b);
}
```

Он выполняется в ответ на событие mouseDown, возникающее при щелчке мышью на клипе. Его второе выражение, содержащее действие gotoAndPlay, нам уже знакомо. (Правда, вместо номера кадра в качестве параметра действию передается нечто непонятное, но об этом потом.) Первое же выражение нам что-то напоминает... Математическая формула?

Да, это математическая формула, записанная на языке ActionScript в виде выражения. Подобные выражения называются *математическими выражениями*. Выражение b = a + 2, как можно предположить, вычисляет некое значение. Давайте рассмотрим его подробнее.

Очевидно, что данное выражение складывает две величины — об этом говорит знак +. Этим знаком обозначается оператор арифметического сложения ActionScript, выполняющий сложение двух величин. *Оператор* — это команда, выполняющая действия над данными сценария, а не над элементами фильма. Забегая немного вперед, скажем, что в ActionScript существуют также операторы арифметического вычитания, умножения и деления, обозначаемые соответственно знаками -, * и /.

Слева и справа от оператора сложения находятся *операнды* — значения, которыми манипулирует этот оператор. Второй операнд (число 2 в нашем примере) является *константой* — постоянной величиной. Ведь значение числа 2 никогда не изменяется, не так ли?

Первый же операнд в выражении какой-то непонятный — он обозначен буквой a. Что это такое? А это *переменная* — ячейка памяти, в которую можно на время поместить какое-либо значение, чтобы сохранить его до лучших времен. (Поэтому она и называется переменной, в отличие от константы.) Каждая переменная должна иметь уникальное имя.

Итак, что же делает описанное выражение? Оно берет из переменной a значение первого операнда, складывает его с константой 2 (второй операнд), а результат помещает в переменную b. Помещение результата в переменную b выполняется с помощью *оператора присваивания*, обозначаемого символом =.

Вот еще один пример арифметического выражения, на этот раз более сложного:

```
y = y1 * y2 + x1 * x2;
```

Здесь операторы выполняются в следующем порядке:

1. Значение переменной `y1` умножается на значение переменной `y2`.
2. Перемножаются значения переменных `x1` и `x2`.
3. Полученные на шагах 1 и 2 произведения складываются.
4. Полученная сумма присваивается переменной `y`.

Но почему на шаге 2 выполняется умножение `x1` на `x2`, а не сложение произведения `y1` и `y2` с `x1`. Дело в том, что каждый оператор имеет *приоритет* — своего рода номер в очереди их выполнения. Так вот, оператор умножения имеет более высокий приоритет, чем оператор сложения, поэтому умножение всегда выполняется перед сложением.

А вот еще одно выражение:

```
x = x + 3;
```

Оно абсолютно правильно с точки зрения ActionScript, хоть и выглядит нелепым. В нем сначала выполняется сложение значения переменной `x` и числа `3`, после чего результат сложения снова помещается в переменную `x`. Такие выражения используются в сценариях ActionScript очень часто.

Вот и все. Основные принципы программирования на ActionScript мы изучили. В дальнейшем мы вернемся к ним, чтобы пополнить и уточнить наши знания.

Данные ActionScript. Типы данных

Любая программа при своей работе оперирует некими данными. Такими данными могут быть координаты клипа на рабочем листе, номер кадра, имя и пароль пользователя, цена какого-нибудь товара в интернет-магазине, величина атмосферного давления и пр. Конечно, не составляют исключения и сценарии ActionScript; мы сами использовали номер кадра, когда писали наши первые сценарии (*см. главу 18*).

ActionScript может манипулировать данными, относящимися к разным видам, или, как, говорят программисты, *типам данных*. Тип данных описывает их возможные значения и набор применимых к ним операций. Давайте перечислим все типы данных, с которыми мы можем столкнуться.

Строковые данные (или *строки*) — это последовательности букв, цифр, пробелов, знаков препинания и прочих символов, заключенные в одинарные или двойные кавычки. Например, это могут быть такие строки:

```
"Flash 8"
"1234567"
'Строковые данные — это последовательности символов.'
```

Строки могут иметь любую длину (определяемую количеством составляющих их символов), ограниченную лишь объемом свободной памяти компьютера. Разумеется, теоретически существует предел в 2 Гбайт, но вряд ли в нашей практике встретятся столь длинные строки.

Здесь нужно сказать немного о символах, из которых состоят строки. Каждый символ в памяти компьютера представлен особым *кодом*, однозначно его определяющим. Поэтому все строки фактически представляют собой совокупный набор кодов составляющих их символов.

Набор всех доступных символов вместе с соответствующими им кодами называется *кодировкой*. Flash 8 для хранения строк использует кодировку *Unicode*, в которой каждый символ представлен двумя байтами (можно кодировать до 65 535 символов). Кодировка Unicode содержит практически все символы практически всех языков мира; это позволяет использовать в документах Flash тексты на любых языках мира.

Кроме букв, цифр и знаков препинания, строки могут содержать *специальные символы*, служащие для особых целей. Все специальные символы, поддерживаемые Flash, приведены в табл. 19.1.

Таблица 19.1. *Специальные символы, поддерживаемые Flash*

Символ	Описание	Код ASCII
`\f`	Прогон листа	12
`\n`	Прогон строки	10
`\r`	Возврат каретки	13
`\t`	Табуляция	9
`\"`	Двойная кавычка	22
`\'`	Одинарная кавычка	27
`\\`	Обратная косая черта	92
`\999`	Любой символ по его восьмеричному коду ASCII (обозначен как `999`)	–
`\xFF`	Любой символ по его шестнадцатеричному коду ASCII (обозначен как `FF`)	–
`\xFFFF`	Любой символ по его коду Unicode (обозначен как `FFFF`)	–

Таким образом, если нам требуется поместить в строку двойные кавычки, нужно записать ее так:

```
"\"Macromedia Flash\" - пакет векторной графики и анимации"
```

Числовые данные (или *числа*) — это обычные числа, над которыми можно производить все арифметические действия, извлекать из них квадратный корень и вычислять тригонометрические функции. Числа могут быть как целыми, так и дробными; в последнем случае целая и дробная части разделяются точкой (не запятой!).

Примеры чисел:

```
13756
454.7873
0.5635
```

Для записи дробных чисел может быть использована экспоненциальная форма вида `<Мантисса>E<Порядок>`. Вот примеры заданных таким образом чисел (в скобках дано традиционное математическое представление):

`1E-5` (10^{-5})

`8.546E23` ($8{,}546 \times 10^{23}$)

Логическая величина может принимать только два значения: true и false — "истина" и "ложь", — обозначаемые соответственно ключевыми словами `true` и `false`. Логические величины часто используются в операциях сравнения (о них будет рассказано далее в этой главе).

ActionScript также поддерживает два специальных типа. Эти типы обозначают отсутствие любых данных. Тип *null* обозначает отсутствие каких-либо данных и обозначается ключевым словом `null`. А тип *undefined* указывает на то, что переменной не было присвоено никакое значение, и обозначается ключевым словом `undefined`.

> ***Внимание!***
>
> `Undefined` — это не то же самое, что `null`!

Остальные типы данных, поддерживаемые ActionScript и не описанные здесь, мы рассмотрим позже.

Константы

Константы, как мы уже выяснили, — данные, значение которых никогда не меняется. Константами являются, в частности, все числа, строки, логические и специальные значения, записанные в соответствии с правилами конкретного типа данных:

```
26756
"Строка"
true
null
```

ActionScript также предоставляет несколько специальных констант, заданных ключевыми словами. Среди них, например, есть константа `Infinity`, представляющая значение ∞, возникающее при делении на ноль. А константа `-Infinity` представляет значение −∞.

Переменные

Для того чтобы успешно обрабатывать данные, сценарию нужно где-то их хранить. Под хранение данных выделяются специальные участки памяти, называемые переменными. Сценарий может обращаться к этим участкам памяти по имени, помещать в них данные любого типа и читать их.

В начале этой главы мы кое-что узнали о переменных. Сейчас настало время поговорить о них подробнее.

Именование переменных

Как мы уже знаем, каждая переменная должна иметь имя, которое однозначно ее идентифицирует. Другими словами, не должно быть двух переменных с одним и тем же именем (об исключениях из этого правила поговорим попозже). Также не допускается совпадение имени переменной с ключевым словом языка ActionScript.

В имени переменной могут присутствовать латинские и русские буквы, цифры, символы `$` (знак доллара) и `_` (знак подчеркивания), причем первый символ имени должен быть либо буквой, либо знаком доллара или подчеркивания. Например, `FrameNumber`, `_link`, `ИмяПользователя` — правильные имена переменных, а `678vasya` и `Имя пользователя` — неправильные.

Нужно сразу сказать, что язык ActionScript чувствителен к регистру символов, которыми набраны имена переменных. Это значит, что `framenumber` и `FrameNumber` — разные переменные.

При выборе имен переменных специалисты фирмы Macromedia рекомендуют следовать одному простому правилу: имя должно быть "говорящим". Это значит, что имя должно отражать назначение переменной. Например, переменную, в которой хранится номер кадра, лучше всего назвать `frameNumber` — так будет сразу понятно, зачем она нужна. Но не стоит слишком усердствовать: имена типа `lastVisitedFormFrameNumber` очень трудно набирать. Для *вре́менных переменных* (хранящих данные, необходимые в данный конкретный момент) лучше всего брать однобуквенные имена: `i`, `a`, `x`, `y` и т. п.

Объявление переменных

Перед использованием переменной в коде сценария обязательно нужно выполнить ее *объявление*. Для этого используется оператор объявления переменной `var`, после которого указывается имя переменной. Вот так:

```
var x;
```

Теперь объявленной переменной можно присвоить какое-либо значение:

```
x = 1234;
```

и использовать в сценарии:

```
y = x / 2 + 10;
```

Значение переменной также можно присвоить прямо при ее объявлении:

```
var x = 1234;
```

Нужно сказать, что объявлять переменные с помощью оператора `var` не обязательно. Мы можем просто присвоить переменной какие-либо значение, и Flash сам ее создаст. Просто явное объявление переменных оператором `var` считается хорошим стилем программирования.

> ***ВНИМАНИЕ!***
>
> Если обратиться к переменной, еще не получившей никакого значения, она вернет значение `undefined`.

Область видимости переменных

Теперь нам нужно поговорить об одном важном моменте в "жизни" любой переменной. Это область ее *видимости*, определяющая, какие выражения могут получить доступ к этой переменной, а какие — не могут.

Пусть мы объявили переменную в сценарии, привязанном к кадру анимации какого-либо клипа или внешней анимации. В этом случае к данной переменной получат доступ все сценарии, привязанные к кадрам, кнопкам и вложенным клипам, содержащимся в этом клипе (фильме). Разумеется, эту переменную нужно сначала объявить, а потом уже к ней обращаться.

А вот сценарии, содержащиеся во вложенных в эту анимацию клипах, не смогут получить доступ к этой переменной. Также ее не смогут использовать сценарии, находящиеся во внешнем по отношению к текущему клипе. Если же мы все-таки попытаемся извлечь значение этой переменной из одного из таких сценариев, то получим значение `undefined`, с которым каши не сваришь.

Такие переменные называются *переменными уровня клипа*. Говорят, что такие переменные "видны" только в "своем" клипе.

Но что делать, если нам все же нужно обратиться к переменной, находящейся в другом клипе? Да просто указать имя этого клипа перед именем переменной, разделив их точкой. Вот так:

```
someClip.someVar = 0;
```

Глобальные переменные "видны" абсолютно из всех сценариев, находящихся в данном фильме. Они объявляются с помощью *модификатора* (особого ключевого слова, изменяющего действие какой-либо команды ActionScript) `_global`, который добавляется перед именем переменной и отделяется от него точкой:

```
_global.someVar = 0;
```

Существует еще один тип видимости переменных — *локальные* переменные. Мы поговорим о них, когда начнем изучение функций.

Выше говорилось, что каждая переменная должна иметь уникальное имя, по которому к ней можно будет обратиться. Есть одно исключение из этого правила: переменные, имеющие разную область видимости, могут иметь одинаковые имена. При этом обращение происходит к переменной, имеющей более "узкую" область видимости.

Давайте рассмотрим такой пример:

```
_global.someVar:String = "Value";

var someVar:Integer = 3;

. . .

s = someVar;
```

Мы объявили две переменные `someVar`, одну — уровня клипа, другую — глобальную, и присвоили им разные значения. Если мы теперь запросим значение переменной `someVar`, то переменной `s` будет присвоено число 3 — значение переменной `someVar` уровня клипа как имеющей более "узкую" область видимости. Глобальная переменная `someVar` будет "замаскирована".

> ***Внимание!***
>
> Всегда проверяйте, к какой переменной вы обращаетесь. Ошибки доступа к переменным — самые трудноулавливаемые.

Но как получить значение глобальной переменной `someVar`? Очень просто — нужно использовать модификатор `_global`:

```
s = _global.someVar;
```

Операторы

Ну что ж, с данными, константами и переменными покончено. Пора приступать к рассмотрению операторов, предназначенных для манипуляций с ними.

Арифметические операторы

Арифметические операторы служат для выполнения арифметических действий над значениями констант и переменных. Все арифметические операторы, поддерживаемые ActionScript, перечислены в табл. 19.2.

Таблица 19.2. *Арифметические операторы ActionScript*

Оператор	Описание
-	Смена знака числа
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток от деления
++	*Инкремент* (увеличение на единицу)
--	Декремент (уменьшение на единицу)

Арифметические операторы делятся на две группы: *унарные* и *бинарные*. Унарные операторы выполняются над одним операндом; к ним относятся операторы смены знака, инкремента и декремента. Унарный оператор извлекает из переменной значение, изменяет его и снова помещает в ту же переменную. Приведем пример выражения с унарным оператором:

```
++r;
```

При выполнении этого выражения в переменной `r` окажется ее значение, увеличенное на единицу. А если записать вот так:

```
s = ++r;
```

то же значение будет помещено и в переменную `s`.

Операторы инкремента и декремента могут ставиться как перед операндом, так и после него. Если оператор инкремента стоит перед операндом, то значение операнда сначала инкрементируется, а уже потом используется в дальнейших вычислениях. Если же оператор инкремента стоит после опе-

ранда, то его значение сначала используется в других вычислениях, а уже потом инкрементируется. Точно так же ведет себя оператор декремента.

Бинарные операторы всегда имеют два операнда и помещают результат в третью переменную. Вот примеры выражений с бинарными операторами:

```
l = r * 3.14;
f = e / 2;
x = x + t / 3;
```

> ***ВНИМАНИЕ!***
>
> Операторы инкремента и декремента рекомендуется использовать, если значение какой-либо переменной нужно увеличить или уменьшить на единицу. Эти операторы выполняются быстрее, чем операторы сложения и вычитания.

Оператор объединения строк

Оператор объединения строк `+` позволяет соединить две строки в одну. Например, сценарий:

```
s1 = "Flash";
s2 = "8";
s = s1 + s2;
```

поместит в переменную `s` строку `"Flash8"`.

Операторы присваивания

Оператор простого присваивания `=` нам уже знаком. С его помощью переменной присваивается новое значение:

```
a = 2;
b = c = 3;
```

Второе выражение в приведенном примере выполняет присвоение значения `3` сразу двум переменным — `b` и `c`.

Кроме операторов присваивания, ActionScript поддерживает *операторы сложного присваивания*. Такие операторы позволяют выполнять операцию присваивания одновременно с другой операцией:

```
a = a + b;
a += b;
```

Два этих выражения эквивалентны по результату. Просто во втором был использован оператор сложного присваивания `+=`.

Все операторы сложного присваивания, поддерживаемые ActionScript, и их эквиваленты приведены в табл. 19.3.

Таблица 19.3. *Операторы сложного присваивания*

Оператор	Эквивалентное выражение
`a += b;`	`a = a + b;`
`a -= b;`	`a = a - b;`
`a *= b;`	`a = a * b;`
`a /= b;`	`a = a / b;`
`a %= b;`	`a = a % b;`
`a <<= b;`	`a = a << b;`
`a >>= b;`	`a = a >> b;`
`a >>>= b;`	`a = a >>> b;`
`a &= b;`	`a = a & b;`
`a ^= b;`	`a = a ^ b;`
`a |= b;`	`a = a | b;`

Операторы сравнения

Операторы сравнения сравнивают два операнда и возвращают логическое значение. Если условие сравнения выполняется, возвращается логическое значение "истина" (true), если не выполняется — "ложь" (false). Вот примеры выражений с операторами присваивания:

```
a1 = 2 < 3;
a2 = -4 > 0;
a3 = r < t;
```

Переменная `a1` получит значение `true` (2 меньше 3), переменная `a2` — значение `false` (число −4 по определению не может быть больше нуля), а значение переменной `a3` будет зависеть от значений переменных `r` и `t`.

Все поддерживаемые ActionScript операторы сравнения приведены в табл. 19.4.

Таблица 19.4. *Операторы сравнения*

Оператор	Описание
<	Меньше
>	Больше
==	Равно
<=	Меньше или равно

Таблица 19.4 *(окончание)*

Оператор	Описание
>=	Больше или равно
!=	Не равно
===	Строго равно
!==	Строго не равно

С первыми шестью операторами сравнения все понятно. Но на двух последних операторах — "строго равно" и "строго не равно" — нужно остановиться подробнее. Это операторы так называемого *строгого сравнения*. Обычные операторы "равно" и "не равно", если встречают операнды разных типов, пытаются преобразовать их к одному типу (о преобразованиях типов см. далее в этой главе). Операторы строгого равенства и строгого неравенства такого преобразования не делают, а в случае несовместимости типов операндов возвращают `false`.

Логические операторы

Логические операторы выполняют действия над логическими значениями. Все они приведены в табл. 19.5. А в табл. 19.6 и 19.7 показаны результаты выполнения этих операторов.

Таблица 19.5. *Логические операторы*

Оператор	Описание
!	НЕ (логическая инверсия)
&&	И (логическое умножение)
||	ИЛИ (логическое сложение)

Таблица 19.6. *Результаты выполнения операторов И и ИЛИ*

Операнд 1	Операнд 2	&& (И)	|| (ИЛИ)
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Таблица 19.7. *Результаты выполнения оператора НЕ*

Операнд	! (НЕ)	Операнд	! (НЕ)
`true`	`false`	`false`	`true`

Основная область применения логических операторов — выражения сравнения (о них см. далее в этой главе). Приведем примеры таких выражений:

```
a = (b > 0) && (c + 1 != d);
flag = !(status = 0);
```

Оператор *typeof*

Оператор `typeof` принимает в качестве операнда переменную или выражение и возвращает строку, описывающую тип данных операнда. Все значения, которые он может вернуть, перечислены в табл. 19.8.

Таблица 19.8. *Значения, возвращаемые оператором* `typeof`

Тип данных	Возвращаемое значение
Строковый	`"string"`
Числовой	`"number"`
Логический	`"boolean"`
Объект	`"object"`
Функция	`"function"`
Клип	`"movieclip"`
null	`"null"`
undefined	`"undefined"`

Оператор `typeof` может использоваться, например, так:

```
status = typeof(somevar);
```

Здесь мы присваиваем результат выполнения оператора `typeof` строковой переменной `status`. Впоследствии он может быть использован, например, в условном выражении.

Совместимость и преобразование типов данных

И напоследок рассмотрим еще два важных вопроса: *совместимость типов данных* и *преобразование* одного типа к другому.

Что получится, если сложить два числовых значения? Правильно — еще одно числовое значение. А если сложить число и строку? Трудно сказать. Тут Flash сталкивается с проблемой несовместимости типов данных и пытается сделать эти типы совместимыми, преобразуя один из них к другому. Сначала он пытается преобразовать строку в число и, если это удается, выполняет сложение. В случае неудачи число будет преобразовано в строку, и две полученные строки будут объединены. Например, в результате выполнения операторов

```
var a, b, c, d, e, f;
a = 11;
b = "12";
c = a + b;
d = "Flash";
e = 8;
f = d + e;
```

значение переменной `b` при сложении с переменной `a` будет преобразовано в числовой тип; таким образом, переменная `c` будет содержать значение `23`. Но так как значение переменной `d` не может быть преобразовано в число, значение `e` будет преобразовано в строку, и результат — значение `f` — станет равным `"Flash8"`.

Логические величины преобразуются либо в числовые, либо в строковые, в зависимости от конкретного случая. Значение `true` будет преобразовано в число `1` или строку `"1"`, а значение `false` — в `0` или `"0"`.

Видно, что Flash изо всех сил пытается правильно выполнить даже некорректно написанные выражения. Иногда это получается, но чаще все работает не так, как планировалось, и, в конце концов, выполнение сценария прерывается в связи с обнаружением ошибки совсем в другом его месте, на абсолютно верном операторе. Поэтому лучше всего не допускать подобных казусов, оперировать только переменными совместимых типов и самим выполнять нужные преобразования типов.

Для преобразования типов данных используется следующий синтаксис:

```
<Переменная нужного типа> = <Ключевое слово, соответствующее этому типу>
↳(<Преобразуемое значение>);
```

Ключевые слова, соответствующие тому или иному типу данных, приведены в табл. 19.8. Да, именно эти слова возвращает оператор `typeof` в качестве результата. Единственное — их нужно указывать с большой буквы.

Например, для того чтобы преобразовать строку в число, нужно написать такое выражение:

```
n = Number("2004");
```

Если же Flash не сможет выполнить заданное преобразование типов, будет возвращено значение `null`.

Приоритет операторов

Последний вопрос, который мы здесь рассмотрим, — это приоритет операторов. Как мы помним, приоритет влияет на порядок, в котором выполняются операторы в выражении.

Пусть имеется следующее выражение:

```
a = b + c - 10;
```

В этом случае сначала к значению переменной `b` будет прибавлено значение `c`, а потом из суммы будет вычтено `10`. Операторы этого выражения имеют одинаковый приоритет и поэтому выполняются строго слева направо.

Теперь рассмотрим такое выражение:

```
a = b + c * 10;
```

А в этом случае сначала будет выполнено умножение значения `c` на 10, а уже потом к полученному произведению будет прибавлено значение `b`. Оператор умножения имеет бóльший приоритет, чем оператор сложения, поэтому порядок "строго слева направо" будет нарушен.

Самый низкий приоритет имеет оператор присваивания `=`. Вот почему сначала вычисляется само выражение, а потом его результат присваивается переменной.

В общем, основной принцип выполнения всех операторов таков: сначала выполняются операторы с более высоким приоритетом, а уже потом — операторы с более низким. Операторы с одинаковым приоритетом выполняются в порядке их следования (слева направо).

В табл. 19.9 перечислены все изученные нами операторы в порядке убывания их приоритетов. Полную таблицу приоритетов можно найти в интерактивной справке Flash.

Таблица 19.9. *Приоритет операторов (в порядке убывания)*

Операторы	Описание
`++ -- - ! typeof`	Инкремент, декремент, смена знака, логическое НЕ, определение типа
`* / %`	Умножение, деление, взятие остатка
`+ -`	Сложение и объединение строк, вычитание
`< > <= >= = !=`	Операторы сравнения
`&&`	Логическое И

Таблица 19.9 *(окончание)*

Операторы	Описание
||	Логическое ИЛИ
= <Оператор>=	Присваивание, простое и сложное

> ***ВНИМАНИЕ!***
>
> Запомните эту таблицу. Неправильный порядок выполнения операторов может стать причиной трудновыявляемых ошибок, при которых внешне абсолютно правильное выражение дает неверный результат.

Но что делать, если нам нужно нарушить обычный порядок выполнения операторов? Воспользуемся скобками. При такой записи заключенные в скобки операторы выполняются первыми.

```
a = (b + c) * 10;
```

В этом случае сначала будет выполнено сложение значений переменных `b` и `c`, а потом получившаяся сумма будет умножена на 10.

Операторы, заключенные в скобки, также подчиняются приоритету. Поэтому часто используются многократно вложенные скобки:

```
a = ((b + c) * 10 - d) / 2 + 9;
```

Здесь операторы будут выполнены в такой последовательности:

1. Сложение `b` и `c`.
2. Умножение полученной суммы на 10.
3. Вычитание `d` из произведения.
4. Деление разности на 2.
5. Прибавление 9 к частному.

Если удалить скобки:

```
a = b + c * 10 - d / 2 + 9;
```

то порядок выполнения операторов будет таким:

1. Умножение `c` и 10.
2. Деление `d` на 2.
3. Сложение `b` и произведения `c` и 10.
4. Вычитание из полученной суммы частного от деления `d` на 2.
5. Прибавление 9 к полученной разности.

Получается совсем другой результат, не так ли?

Комментарии

Очень часто при написании сценариев возникает потребность поместить в код ActionScript какие-либо примечания для себя или коллег по работе. Для этого используются особые выражения языка ActionScript, так называемые *комментарии*. Комментарии не обрабатываются Flash и исключаются из сценариев при экспорте фильма, поэтому в них можно писать все что угодно.

Для вставки комментариев в код ActionScript предусмотрены два оператора комментария: `//` и `/*...*/`. Первый из них позволяет вставить в конец выражения однострочный комментарий:

```
// Строка комментария
a = b + c; // Это однострочный комментарий
```

Заметим, что комментарий ставится после точки с запятой, обозначающей конец выражения.

А оператор `/*...*/` позволяет вставить в код программы комментарий любого размера:

```
/*
В этом выражении мы складываем содержимое двух переменных
и помещаем результат в третью
*/
a = b + c;
```

Сложные выражения ActionScript

Рассмотрение основных понятий ActionScript мы закончили. Можно приступать к изучению более сложных вопросов. И начнем мы с написания так называемых сложных выражений.

Сложные выражения получили свое название благодаря тому, что все они составлены из нескольких простых выражений. Сложные выражения выполняются особым образом и служат для особых целей — в основном для управления процессом выполнения содержащихся в них простых выражений.

Блоки

ActionScript позволяет нам объединить несколько выражений в одно. Такое выражение называется *блочным выражением* или просто *блоком*. Для выделе-

ния блочного выражения составляющие его выражения заключаются в фигурные скобки, например:

```
{
  b = "12";
  c = a - b;
}
```

Как правило, блоки не используются сами по себе. Чаще всего они входят в состав других сложных выражений. Также с их помощью пишутся обработчики событий.

Условные выражения

Условное выражение позволяет нам выполнить одно из двух входящих в него выражений в зависимости от выполнения или невыполнения какого-либо *условия*. Существует также другая, "вырожденная" разновидность условного выражения, содержащая только одно выражение, которое выполняется при выполнении условия и пропускается, если условие не выполнено.

Что может служить условием в условном выражении? Значение логической переменной или результат вычисления логического выражения. В качестве же выражений, которые должны быть выполнены или не выполнены, в зависимости от условия, могут быть использованы блоки.

Условное выражение имеет следующий формат:

```
if (<Условие>) . . .
  <Блок "то">
else
  <Блок "иначе">
```

"Вырожденный" формат его выглядит так:

```
if (<Условие>) . . .
  <Блок "то">
```

Для написания условных выражений используются особые ключевые слова `if` и `else` (в приведенных выше примерах они выделены полужирным шрифтом). `Условие` — это и есть логическое выражение, в соответствии с которым Flash принимает решение, какой блок выполнить. Если условие имеет значение `true` ("истина"), то выполняется `блок "то"`. Если же условие имеет значение `false` ("ложь"), то выполняется `блок "иначе"` (если он присутствует в условном выражении). Если же `блок "иначе"` отсутствует, выполняется следующее выражение сценария.

Рассмотрим несколько примеров.

```
if (x == 1) {
  a = "Единица";
  b = 1;
}
 else {
  a = "Не единица";
  b = 22222;
}
```

Здесь мы сравниваем значение переменной `x` с единицей и в зависимости от результатов сравнения присваиваем переменным `f` и `h` разные значение. Обратим внимание на условие — именно так записывается оператор сравнения, в скобках.

Условие может быть довольно сложным:

```
if ((x == 1) && (y > 10))
  f = 3;
else
  f = 33;
```

Здесь мы использовали сложное условие, возвращающее значение `true` в случае, если значение переменной `x` равно единице и значение переменной `y` больше десяти. Заметим также, что мы подставили одиночные выражения, так как фрагменты кода слишком просты, чтобы оформлять их в виде блоков.

Если условное выражение совсем простое, мы можем записать его немного по-другому. А именно, воспользоваться *условным оператором* `?`:

```
<Условие> ? <Выражение "то"> : <Выражение "иначе">;
```

Достоинство этого оператора в том, что он может быть частью выражения. Например:

```
f = (x == 1 && y > 10) ? 3 : 33;
```

Фактически мы записали условное выражение из предыдущего примера, но в виде обычного арифметического выражения. Компактность кода налицо. Недостаток же оператора `?` в том, что с его помощью можно записывать только самые простые условные выражения.

Выражения выбора

Выражение выбора — это фактически несколько условных выражений, объединенных в одном.

Его формат таков:

```
switch (<Условие>) {
  case <Значение 1> :
    <Блок 1>
    [break;]
  [case <Значение 2> :
    <Блок 2>
    [break;]]
  <... Другие секции case>
  [default :
    <Блок, исполняемый для остальных значений>]
}
```

В выражениях выбора используются ключевые слова `switch`, `case` и `default` (выделены полужирным шрифтом).

Давайте выясним, как выполняется выражение выбора. Результат вычисления `условия` последовательно сравнивается со `значением 1`, `значением 2` и т. д. и, если такое сравнение увенчалось успехом, выполняется соответствующий блок кода (`блок 1`, `блок 2` и т. д.). Если же ни одно сравнение не увенчалось успехом, выполняется блок кода, находящийся в секции `default` (если, конечно, она есть).

Пример использования выражения выбора:

```
switch (a) {
  case 1 :
    out = "Единица";
    break;
  case 2 :
    out = "Двойка";
    break;
  case 3 :
    out = "Тройка";
    break;
  default :
    out = "Другое число";
}
```

Встретив действие `break`, Flash прерывает выполнение блока, в котором оно присутствует, и начинает выполнение кода, следующего за выражением выбора. Если его опустить, то будет выполнен следующий блок. Так, если зна-

чение `условия` совпало со `значением 1` и был выполнен `блок 1`, не содержащий действия `break`, будет также выполнен `блок 2`.

Давайте уберем все действия `break` в нашем примере:

```
switch (a) {
  case 1 :
    out = "Единица";
  case 2 :
    out = "Двойка";
  case 3 :
    out = "Тройка";
  default :
    out = "Другое число";
}
```

В этом случае все блоки будут выполняться последовательно, один за другим. И переменной `out` всегда будет присваиваться строка `"Другое число"`.

Циклы

Циклы — это особые выражения, позволяющие выполнить один и тот же блок кода несколько раз. Выполнение кода прерывается по наступлению некоего условия.

ActionScript предлагает программистам несколько разновидностей циклов. Рассмотрим их.

Цикл со счетчиком

Цикл со счетчиком используется, если какой-то код нужно выполнить определенное число раз. Это наиболее часто используемый вид цикла.

Для подсчета того, сколько раз был выполнен этот фрагмент кода, используется переменная, называемая *счетчиком цикла*. Перед каждым выполнением блока кода производится проверка значения счетчика и, если оно достигло предельного значения, блок выполняется. Сам же этот блок называется *телом цикла*.

Цикл со счетчиком записывается так:

```
for (<Выражение инициализации>; <Условие>; <Приращение>)
  <Тело цикла>
```

Для задания цикла со счетчиком используется ключевое слово `for`. Поэтому такие циклы часто называют "циклами for".

Выражение инициализации присваивает счетчику начальное значение. Далее проверяется условие цикла, и, если его значение истинно (true), выполняется тело цикла. После этого выполняется выражение приращения, изменяющее значение счетчика, затем снова проверяется условие, и т. д., пока условие не станет ложным (false), т. е. пока счетчик не дойдет до предельного значения.

Пример цикла со счетчиком:

```
for (i = 1; i < 11; i++) {
  a += 3;
  b = i * 2 + 1;
}
```

Этот цикл будет выполнен 10 раз. Мы присваиваем счетчику i начальное значение 1 и после каждого выполнения тела цикла увеличиваем его на единицу. Цикл перестанет выполняться, когда значение счетчика увеличится до 11, и условие цикла станет ложным.

Кроме того, допустимо использовать счетчик цикла в одном из выражений тела цикла — как это сделали мы. Счетчик i будет содержать последовательно возрастающие значения от 1 до 10, которые можно использовать в вычислениях.

Приведем еще два примера цикла со счетчиком:

```
for (i = 10; i > 0; i--) {
  a += 3;
  b = i * 2 + 1;
}
```

Здесь значение счетчика декрементируется. Начальное его значение равно 10. Цикл выполнится 10 раз и завершится, когда счетчик i будет содержать 0; при этом значения последнего будут последовательно уменьшаться от 10 до 1.

```
for (i = 2; i < 21; i += 2) b = i * 2 + 1;
```

А в этом примере начальное значение счетчика равно 2, а конечное — 21, но цикл выполнится опять же 10 раз. А все потому, что значение счетчика увеличивается на 2 и последовательно принимает значения 2, 4, 6... 20.

В особом "вырожденном" случае цикл for может даже не содержать тела. В этом случае "полезную нагрузку" цикла несет на себе выражение приращения.

Цикл с постусловием

Цикл с постусловием во многом похож на цикл со счетчиком, а именно, в том, что он выполняется до тех пор, пока остается истинным условие цикла.

Причем условие проверяется не до, а после выполнения тела цикла, отчего цикл с постусловием и получил свое название. Такой цикл выполнится хотя бы один раз, даже если его условие с самого начала ложно.

Формат цикла с постусловием:

```
do
  <Тело цикла>
while (<Условие>);
```

Для задания цикла с постусловием используются ключевые слова `do` и `while`. Поэтому такие циклы часто называют "циклами do-while".

Цикл с постусловием можно использовать различными способами. Например, так:

```
do {
  a = a * i + 2;
  i = ++i;
} while (a < 100);
```

В рассмотренном выше примере проверяется наступление некого отвлеченного условия.

А можно записать такой цикл так:

```
var a = 0, i = 1;
do {
  a = a * i + 2;
  i = ++i;
} while (i < 20);
```

Здесь мы используем счетчик, чье конечное значение ограничено. Хотя, конечно, в данном примере удобнее было бы использовать уже знакомый нам и специально предназначенный для таких случаев цикл со счетчиком.

Цикл с предусловием

Цикл с предусловием отличается от цикла с постусловием тем, что условие проверяется перед выполнением тела цикла. Так что, если оно (условие) изначально ложно, цикл не выполнится ни разу.

```
while (<Условие>)
  <Тело цикла>
```

Для создания цикла с постусловием используется ключевое слово `while`. Поэтому такие циклы называют еще "циклами while" (не путать с "циклами do-while"!).

Пример цикла с предусловием:

```
while (a < 100) {
  a = a * i + 2;
  i = ++i;
}
```

Прерывание и перезапуск цикла

Иногда бывает нужно прервать выполнение цикла. Для этого Flash предоставляет программистам действия `break` и `continue`.

Действие `break` позволяет *прервать* выполнение цикла и перейти к следующему за ним выражению.

```
while (a < 100) {
  a = a * i + 2;
  if (a > 50) break;
  i = ++i;
}
```

В этом примере мы прерываем выполнение цикла, если значение переменной `a` превысит 50.

Собственно, мы уже знакомы с действием `break`. Оно использовалось в выражении выбора, где вело себя точно так же.

Действие `continue` позволяет *перезапустить* цикл, т. е. оставить невыполненными все последующие выражения, входящие в тело цикла, и запустить выполнение цикла с самого его начала: проверка условия, приращение счетчика и выполнение тела и т. д.

```
while (a < 100) {
  i = ++i;
  if (i > 9 && i < 11) continue;
  a = a * i + 2;
}
```

Здесь мы пропускаем выражение вычисления `a` для всех значений `i` от 10 до 20.

Функции

Функция — это особым образом написанный и оформленный фрагмент кода ActionScript, который можно вызвать из любого места любого сценария. Фактически функция — это повторно используемый участок кода, который,

будучи написанным один раз, может быть вызван где угодно, кем угодно и сколько угодно раз. Так что, если какой-то фрагмент кода встречается в нескольких местах вашего сценария или сценариев, лучше всего оформить его в виде функции.

Собственно код, ради которого и была создана функция, называется *телом функции*. Каждая функция, кроме того, должна иметь уникальное имя, по которому к ней можно будет обратиться. Функция также может принимать один или несколько параметров и возвращать результат, который можно использовать в выражениях.

Объявление функций

Прежде чем функция будет использована где-то в сценарии, ее нужно объявить. Объявление функции выполняется с помощью ключевого слова `function`.

```
function <Имя>([<Список параметров, разделенных запятыми>])
  <Тело функции>
```

`Имя` функции, как уже говорилось, должно быть уникально в пределах ее области видимости. Для имен функций действуют те же правила, что и для имен переменных.

`Список параметров` представляет собой набор переменных, в которые при вызове функции будут помещены значения переданных ей параметров. Мы можем придумать для этих переменных любые имена — все равно они будут использованы только внутри `тела функции`. Это так называемые *формальные параметры* функции.

`Список параметров` функции помещается в круглые скобки, а сами параметры отделяются друг от друга запятыми. Если функция не требует параметров, скобки все равно нужно указать.

В пределах `тела функции` над принятыми ею параметрами (если они есть) и другими данными выполняются некоторые действия и, возможно, вырабатывается результат. Чтобы вернуть результат из функции в выражение, из которого она была вызвана, используется действие `return`:

```
return <Переменная или выражение>;
```

Здесь `переменная` должна содержать возвращаемое значение, а `выражение` должно его вычислять.

Пример объявления функции:

```
function divide(a, b) {
  var c;
```

```
  c = a / b;
  return c;
}
```

Данная функция принимает два параметра — `a` и `b`, — после чего делит `a` на `b` и возвращает частное от этого деления. При этом для хранения промежуточного результата она использует собственную (так называемую локальную) переменную `c`. Такие переменные "видимы" только внутри тела этой функции. При завершении выполнения функции значения всех локальных переменных теряются.

В случае локальных переменных также действует правило "своя переменная ближе к телу" (в данном случае — телу функции). То есть если существуют две переменные с одинаковыми именами, одна — уровня клипа, другая — локальная, то при обращении по этому имени будет получен доступ к локальной переменной. Одноименная переменная уровня клипа будет "замаскирована" своей локальной "тезкой".

Функции, объявленные в сценариях, привязанных к кадрам, кнопкам и вложенным клипам текущего клипа, "видимы" только в пределах этих сценариев (*функции уровня клипа*). Чтобы сделать функцию доступной во всех сценариях фильма (определить ее как *глобальную*), нужно использовать модификатор `_global`:

```
function _global.glDivide(a, b) { return a / b; }
```

Вызов функций

После того как мы объявили функцию, мы можем *вызвать* ее из любого места кода в пределах области видимости. Для этого используется следующий формат:

```
<Имя функции>([<Список фактических параметров, разделенных запятыми>])
```

Здесь указывается `имя` нужной функции и в круглых скобках перечисляются *фактические* параметры, над которыми нужно выполнить соответствующие действия. Функция вернет результат, который можно присвоить переменной или использовать в сложном выражении.

> ***Внимание!***
>
> При вызове функции подставляйте именно фактические параметры, а не формальные, использованные в объявлении функции.

Вот пример вызова объявленной нами выше функции `divide`:

```
d = divide(3, 2);
```

Здесь мы подставили в выражение вызова функции фактические параметры — константы 3 и 2.

```
s = 4 * divide(x, r) + y;
```

А здесь мы выполняем вызов функции с переменными в качестве фактических параметров.

Если функция не возвращает результата, то она вызывается вот так:

```
initVars(1, 2, 3, 6);
```

Более того, таким образом можно вызвать и функцию, возвращающую результат, который в этом случае будет отброшен. Такой способ вызова может быть полезен, если результат, возвращаемый функцией, не нужен для работы сценария.

При вызове функции простым указанием ее имени будет вызвана функция уровня клипа. Если же функции с этим именем не будет найдено, Flash вызовет глобальную функцию с таким же именем (если она есть). Чтобы вызвать функцию, находящуюся в другом клипе, нужно, как и в случае с переменной, указать имя этого клипа перед именем функции, отделив его точкой, например:

```
someClip.initVars(1, 2, 3, 6);
```

Рекурсия

И еще один важный вопрос, связанный с вызовом функций.

Мы уже узнали, что функции могут вызывать другие функции, конечно, если те уже определены. Но функции могут также вызывать и сами себя. Такой прием программирования называется *рекурсией* и иногда бывает очень полезен.

Если в теле функции написать команду ее вызова, то функция будет вызывать себя до бесконечности (так называемая *бесконечная рекурсия*). В конце концов, это приведет к аварийному завершению самого Flash, а то и к краху всей операционной системы. Поэтому функция, предназначенная для рекурсивного вызова, должна предусматривать возможность выхода из этого цикла вызовов. Стандартного способа осуществить это нет, поэтому решение для каждого конкретного случая нужно искать особо.

Приведем пример функции, написанной специально для применения ее в рекурсии:

```
function factorial(a)  {
  if (a == 0) {
    return 1;
    }
```

```
  else
    return (a * factorial(a - 1));
}
```

Эта функция вычисляет факториал числа `a`, переданного ей в качестве параметра. Она рекурсивно вызывает сама себя для того, чтобы получить факториал числа `a - 1`. Также в ее теле выполняется проверка условия равенства `a` нулю; если это условие истинно, возвращается единица, после чего дальнейших рекурсивных вызовов не производится, а все уже сделанные "хором" завершаются. Таким образом, данная функция имеет защиту от бесконечной рекурсии, ведь когда-нибудь она все равно получит в качестве аргумента ноль.

Массивы

Мы уже довольно много знаем о переменных и работе с ними. Но наши знания все еще неполны. Так, мы ничего пока не знаем о массивах — особом способе хранения данных, доступном в ActionScript. Давайте же выясним, что это такое.

Создание массивов и работа с ними

Массив — это пронумерованный набор переменных одного типа, называемых *элементами* массива. Доступ к нужному элементу массива выполняется по его порядковому номеру, называемому *индексом*. А общее число элементов массива называется его *размером*.

> ***ВНИМАНИЕ!***
>
> Нумерация элементов массива начинается с нуля.

Массивы идеально подходят в тех случаях, когда нужно хранить в одной переменной упорядоченный набор данных. Ведь массив фактически представляет собой одну переменную.

Чтобы создать массив, нужно просто присвоить любой переменной список его элементов, разделенных запятыми и заключенных в квадратные скобки:

```
var someArray;
someArray = [1, 2, 3, 4];
```

Здесь мы создали массив, содержащий четыре элемента, и присвоили его переменной `someArray`. После этого мы можем получить доступ к любому

из элементов по его индексу, указав его после имени переменной массива в квадратных скобках:

```
a = massive[2];
```

В данном примере мы получили доступ к третьему элементу массива. (Нумерация элементов массива начинается с нуля — помните об этом!)

Определять сразу все элементы массива необязательно:

```
someArray2 = [1, 2, , 4];
```

Здесь мы пропустили третий элемент массива, и он остался неопределенным (т. е. будет содержать значение `undefined`).

Если будет нужно, мы легко сможем добавить к массиву еще один элемент, просто присвоив ему требуемое значение. Вот так:

```
someArray[4] = 9;
```

При этом будет создан новый, пятый по счету, элемент массива с индексом 4 и значением 9.

Можно даже сделать так:

```
someArray[7] = 9;
```

В этом случае будут созданы четыре новых элемента, и восьмой элемент получит значение 9. Пятый, шестой и седьмой останутся неопределенными (`undefined`).

Мы можем присвоить любому элементу массива другой массив (или, как говорят опытные программисты, создать *вложенный* массив).

```
someArray[2] = ["n1", "n2", "n3"];
```

После этого можно получить доступ к любому элементу вложенного массива, указав последовательно оба индекса:

```
str = someArray[2][1];
```

Переменная `str` получит в качестве значения строку, содержащуюся во втором элементе вложенного массива, — `"n2"`.

Оператор `typeof` возвращает для массива строку `"object"`. Это значит, что массив имеет объектный тип (об объектах см. далее в этой главе).

Ссылки

Осталось рассмотреть еще один момент, связанный с организацией программного доступа к данным. Это так называемые *ссылки* — своего рода указатели на массивы и объекты, в действительности хранящиеся в соответствующих им переменных (мы обязательно столкнемся с ними, когда начнем работать с массивами и объектами).

Когда мы создаем массив, Flash выделяет под него область памяти и помещает в нее значения элементов этого массива. Но в переменную, которой мы присвоили вновь созданный массив, помещается не сама эта область памяти, а ссылка на нее. Если теперь обратиться к какому-либо элементу этого массива, Flash извлечет из переменной ссылку, по ней найдет нужную область памяти, вычислит местонахождение нужного элемента и вернет его значение.

Далее, если мы присвоим переменную массива другой переменной, будет выполнено присвоение именно ссылки. В результате получатся две переменные, ссылающиеся на одну область памяти, хранящую сам этот массив.

Рассмотрим такой пример:

```
var myArray = ["Flash", "MX 2004"];
var newArray = myArray;
```

Здесь создается массив `myArray` с двумя элементами и далее он присваивается переменной `newArray` (при этом данная переменная получает ссылку на массив). Если потом мы присвоим новое значение второму элементу массива `myArray`:

```
myArray[1] = "8";
```

и обратимся к нему через переменную `newArray`:

```
s = newArray[1];
```

то переменная `s` получит строку `"8"` — новое значение второго элемента этого массива. Т. е. фактически переменные `myArray` и `newArray` указывают на один и тот же массив.

Объекты

Итак, мы познакомились с типами данных, переменными, константами, операторами, действиями, простыми и сложными выражениями, функциями и массивами. Теперь настала пора узнать о самых сложных структурах данных ActionScript — объектах.

Понятия объекта и экземпляра

Объект — это сложный тип данных, включающий в себя множество переменных — *свойств* — и набор функций для манипулирования значениями этих переменных — *методов*. Здесь все достаточно просто: свойства хранят некие данные, а методы их обрабатывают, выдавая на выходе другие данные или совершая действия с самим объектом. Каждый объект должен иметь уникальное имя, по которому к нему можно обратиться.

Собственно, сам объект — это всего лишь тип данных, такой же, как числовой или строковый, т. е. некое абстрактное описание, включающее в себя набор свойств и методов. От объекта порождаются конкретные его "представители" — *экземпляры* объекта. Не правда ли, напрашивается аналогия образца и созданных на его основе экземпляров?

Объект можно представить себе как некий прибор, снабженный набором регуляторов и кнопок. Регуляторы задают значения свойств, а кнопки запускают на выполнение методы. Мы можем заполучить один из приборов данного типа (экземпляр), выставить нужные значения свойств с помощью регуляторов и нажать какую-либо кнопку, инициировав выполнение соответствующего метода. Внутреннее устройство этого прибора от нас скрыто, мы не знаем, что у него внутри, для нас важно только одно — чтобы он выполнял свою задачу.

Теперь давайте вернемся к нашему любимому Flash. Любой клип на его рабочем листе фактически является экземпляром объекта `movieClip` (об этом объекте мы поговорим далее в этой главе). Пусть имеется клип `someClip`, обладающий свойством `width` (ширина) и методом `gotoAndPlay`, запускающим воспроизведение анимации этого клипа с какого-то кадра. Тогда можно написать такой сценарий:

```
someClip.width = 200;
someClip.gotoAndPlay(3);
```

Здесь мы обратились к методу и свойству объекта `someClip`, использовав уже знакомый нам синтаксис "с точкой". А именно, отделили точкой имя объекта от имени свойства (метода).

Итак, с теорией мы разобрались. Приступим теперь к практической работе с объектами и их экземплярами.

Работа с объектами и их экземплярами

Перед тем как начать работу с экземпляром какого-либо объекта, его нужно создать. Создание экземпляра объекта выполняется с помощью оператора `new`:

```
<Переменная> = new <Имя объекта>([<Список параметров, разделенных
↳запятыми>])
```

После создания экземпляра объекта оператор `new` возвращает ссылку на него. Эта ссылка, как правило, присваивается какой-либо переменной.

Список параметров может как присутствовать, так и отсутствовать. Обычно он содержит значения, которые присваиваются свойствам экземпляра при его создании. А то, какие значения каким свойствам присваиваются, можно узнать из описания данного объекта в интерактивной справке Flash.

Вот пример создания экземпляра `obj` некоего объекта `someObject`:

```
var obj;
obj = new someObject(a, b);
```

Теперь мы можем обращаться к свойствам и методам созданного экземпляра, используя уже знакомый синтаксис "с точкой":

```
obj.prop1 = 0;
a = obj.prop2 + 2;
obj.method1();
```

Иногда бывает, что один объект содержит внутри себя другие объекты (так называемые *внутренние*). Для обращения к свойствам и методам внутренних объектов используется тот же синтаксис, но обладающий, скажем так, "повышенной этажностью":

```
outerObject.innerObject.prop = 0;
```

где `outerObject` — экземпляр *внешнего объекта* (имеющего в своем составе внутренние объекты), а `innerObject` — экземпляр внутреннего объекта.

Выражение, описывающее цепочку вложенных друг в друга объектов, называется *путем*. Иногда путь бывает очень длинным:

```
objectOuter.objectInner.objectInnerInner.prop = 10;
```

Как и в случае массива, мы можем присваивать ссылку на экземпляр объекта другой переменной:

```
obj2 = obj;
```

Чтобы удалить ненужный экземпляр, следует использовать оператор `delete`:

```
delete <Переменная, содержащая ссылку на экземпляр объекта>;
```

Например,

```
delete obj, obj2;
```

> ***ВНИМАНИЕ!***
>
> Закончив работу с экземпляром какого-либо объекта, всегда удаляйте его. Помните, что каждый экземпляр расходует ресурсы компьютера, которых всегда не хватает.

Оператор `typeof` возвращает для экземпляра объекта строку "`object`". Это значит, что экземпляр объекта имеет *объектный тип данных*.

> ***НА ЗАМЕТКУ***
>
> Очень часто Flash сам создает экземпляры каких-либо объектов и сам же их удаляет. В частности, именно так происходит с экземплярами объекта `movieClip`, т. е. клипами.

Несколько новых операторов и ключевых слов

А теперь самое время рассмотреть несколько новых операторов и действий, применяемых при работе с объектами и их экземплярами.

Оператор `instanceof` проверяет, является ли `экземпляр` экземпляром заданного `объекта`, и возвращает соответственно значение `true` или `false`. Формат его записи такой:

```
<Переменная> = <Экземпляр объекта> instanceof <Объект>;
```

В приведенном ниже примере `экземпляр` задается в виде переменной, содержащей ссылку на него, а `объект` — в виде имени объекта.

```
if (obj instanceof someObject) . . .
```

Ключевые слова `for` и `in` служат для организации так называемого *цикла просмотра*, позволяющего просмотреть все свойства экземпляра объекта. Он также называется "циклом for-in" и имеет следующий формат записи:

```
for (<Переменная-ссылка на свойство> in <Экземпляр объекта>)
  <Тело цикла>
```

`Переменная-ссылка на свойство` каждый раз получает значение очередного свойства `экземпляра`. Эту переменную можно использовать для получения доступа к найденному свойству.

```
for (k in obj) {
  k = ' ' + k + ' ';
}
```

Приведенный фрагмент сценария просматривает все свойства экземпляра `obj` и добавляет к значению каждого из них пробелы слева и справа. (Предполагается, что значения всех свойств этого экземпляра имеют строковый тип.)

Ключевое слово `with` позволяет значительно сократить длину выражений ActionScript, если в них используются свойства или методы какого-либо одного экземпляра объекта. Рассмотрим его применение на примере.

Напишем несколько выражений, обращающихся к свойствам и методам экземпляра какого-либо объекта:

```
someObject.prop1 = 1;
someObject.prop2 = 2;
someObject.prop3 = 3;
someObject.method1;
```

Какие длинные строки у этих выражений! А теперь перепишем их, использовав ключевое слово `with`:

```
with (someObject){
  prop1 = 1;
  prop2 = 2;
  prop3 = 3;
  method1;
}
```

Сценарий сразу стал компактнее. И быстрее, кстати говоря.

Встроенные объекты ActionScript

Встроенными называются объекты, реализованные в самом языке ActionScript. В этом разделе будут рассмотрены их краткие описания и даны примеры использования. Полное описание всех этих объектов можно найти в интерактивной справке Flash.

Интересной особенностью языка ActionScript является то, что он может представлять обычные типы данных — строковый, числовой, логический — как объекты. Поэтому со строками, числами и логическими величинами можно обращаться как с экземплярами соответствующих объектов, вызывать их методы и использовать их свойства. Львиная доля встроенных объектов ActionScript как раз и "отвечает" за объектное представление обычных типов данных.

Объект *String*

Объект `String` представляет собой обычную строку. Пользуясь его свойствами и методами, можно выполнять различные манипуляции над строками.

Создать экземпляр объекта `String` можно одним из двух способов, как показано в следующих примерах:

```
var s = new String("Flash");
var s = "Flash";
```

Как правило, чаще используется второй способ — он и компактнее в записи, и привычнее.

Свойство `length` объекта `String` позволяет получить (или, как говорят программисты, "возвращает") длину строки в символах. Например:

```
l = s.length;
```

Метод `charAt` объекта `String` возвращает символ строки, номер позиции которого в строке был передан в качестве единственного параметра метода:

```
ch = s.charAt(s.length - 1);
```

> ***Внимание!***
>
> Нумерация символов строки начинается с нуля.

После выполнения выражения из приведенного примера в переменной `ch` окажется предпоследний символ строки `s`.

Метод `indexOf` возвращает номер вхождения `подстроки` в `строку`. Если `подстрока` не найдена, возвращается −1. При этом можно вторым параметром передать в метод `номер вхождения`, с которого начнется поиск:

```
<Строка>.indexOf(<Подстрока>, [<Номер вхождения>])
```

Например:

```
s = "Macromedia Flash 8";
n = s.indexOf("a", 2);
```

После выполнения этого сценария в переменной `n` окажется число 13 — именно под таким номером стоит символ `"a"` в слове `"Flash"`.

Метод `lastIndexOf` схож с методом `indexOf`, только ищет подстроку не слева направо, а справа налево, т. е. с конца `строки`.

Методы `toLowerCase` и `toUpperCase` преобразуют все символы строки соответственно к нижнему и верхнему регистру.

Объект *Number*

Объект `Number` представляет собой обычное число. Пользуясь его свойствами и методами, мы можем выполнять различные манипуляции над числовыми данными.

Создается экземпляр объекта `Number` одним из двух способов:

```
var n = new Number(232);
var n = 232;
```

Второй способ нам уже знаком, поэтому будем пользоваться им и впредь.

Метод `toString` объекта `Number` возвращает строковое представление числа:

```
s = n.toString();
```

Помимо данного метода, объект `Number` имеет ряд свойств, возвращающих различные "специальные" значения. Так, свойства `MIN_VALUE` и `MAX_VALUE` позволяют узнать соответственно минимальное (примерно 5×10^{-324}) и максимальное (примерно $1{,}79\times10^{308}$) значения, которые могут быть присвоены

числовой переменной ActionScript. А свойства NEGATIVE_INFINITY и POSITIVE_INFINITY возвращают соответственно значения $-\infty$ и ∞.

Объект *Boolean*

Объект Boolean представляет собой обычную логическую величину. С его помощью можно выполнять различные манипуляции над логическими данными.

Создается экземпляр объекта Boolean одним из двух приведенных ниже способов:

```
var b = new Boolean(true);
var b = true;
```

Метод toString объекта Boolean возвращает строковое представление логической величины — "true" или "false" соответственно:

```
s = b.toString();
```

Объект *Date*

Объект Date предназначен для хранения значения даты и времени, закодированных особым способом в виде числа.

Экземпляр объекта Date создается так:

```
var <Имя переменной> = new Date([<Год>, <Месяц>, [<Число>
↳[, <Часы>[, <Минуты> [,<Секунды> [,<Миллисекунды>]]]]]])
```

Вот краткое описание приведенных выше параметров:

- Год может быть задан двумя или четырьмя цифрами. С четырьмя цифрами все просто; если же год задан двумя цифрами, то значение 0 соответствует 1900 году, а 99 — 1999 году;
- Месяц задается значением от 0 (январь) до 11 (декабрь);
- Дата задается значением от 1 до 31;
- Минуты и Секунды задаются значениями от 0 до 59;
- Миллисекунды задаются значением от 0 до 999.

Если же ни один из параметров не указан, в экземпляр объекта Date заносится текущая дата.

Объект Date имеет огромное количество методов, возвращающих или задающих различные "части" значения даты. Так, метод getMonth возвращает текущее значение месяца, а метод setMonth позволяет задать месяц, не меняя других "частей" даты. А уже знакомый нам метод toString возвращает строковое представление даты, используя региональные установки операционной системы.

Объект *Array*

Объект `Array` представляет собой массив. С его помощью можно выполнять над массивами различные манипуляции.

Создать массив можно одним из четырех приведенных ниже способов:

```
var arr = new Array();
var arr = new Array(<Размер>);
var arr = new Array(<Список элементов, разделенных запятыми>);
var arr = [<Список элементов, разделенных запятыми>];
```

Последний способ нам уже знаком — именно так мы создавали наши первые массивы. При использовании первого способа созданный массив будет иметь нулевой размер, т. е. не будет содержать ни одного элемента. При создании массива вторым способом все его элементы будут содержать значение `undefined`.

Свойство `length` объекта `Array` позволяет нам узнать размер массива.

```
l = arr.length;
```

Метод `reverse` объекта `Array` изменяет порядок следования элементов массива на противоположный. А метод `toString` возвращает строку, содержащую значения всех элементов массива, разделенные запятыми.

Объект *Math*

Объект `Math` предоставляет доступ к встроенным константам и математическим и тригонометрическим функциям языка ActionScript. Единственный экземпляр этого объекта создается самим Flash.

Методы `sin`, `cos` и `tan` объекта `Math` позволяют вычислить соответственно синус, косинус и тангенс угла, заданного в радианах. Метод `sqrt` вычисляет квадратный корень. Метод `pow(x, y)` возводит `x` в степень `y`.

Кроме того, объект `Math` имеет несколько свойств, возвращающих значение различных математических констант. Так, свойство `PI` возвращает значение числа π. А свойство `E` возвращает значение основания натурального логарифма.

Объект *Object*

Объект `Object` — простейший из объектов, предоставляемых языком ActionScript. Он обладает минимальным набором свойств и методов и служит для создания на его основе других объектов и различных вспомогательных целей.

Работа с клипами

Продолжаем изучение языка ActionScript. Поговорим о клипах (в смысле, экземплярах образцов-клипов) и выясним, какие инструменты для работы с ними предоставляет ActionScript. А таких инструментов немало...

Все находящиеся на рабочем листе клипы, для которых было задано имя, равно как и сам фильм, представляют собой экземпляры объекта `movieClip`. Их создает сам Flash; нам же остается только использовать их свойства и методы.

Все клипы имеют тип данных *"клип"*. Оператор `typeof` при передаче ему в качестве параметра клипа вернет строку `"movieclip"`.

Доступ к нужному клипу

Прежде чем приступить к изучению объекта `movieClip`, нам следует выяснить, как же получить доступ к его экземплярам, т. е. клипам и самому фильму.

В начале этой главы говорилось, что обращение к экземпляру объекта осуществляется по его имени. Имя клипа задается с помощью панели **Properties**, а именно, поля ввода, расположенного в ее левом верхнем углу.

Задав имя клипа, мы можем получить к нему доступ из сценария. Так, если на рабочем листе присутствует клип `car`, то для остановки вложенной в него анимации мы можем создать во внешней анимации такой сценарий, привязав его, скажем, к кнопке:

```
car.stop();
```

Здесь мы указываем имя соответствующего экземпляра объекта `movieClip` и, через точку, — метод `stop`, останавливающий воспроизведение анимации.

Если же мы хотим остановить вложенную анимацию клипа `car` из самого этого клипа, то нам нужно слегка изменить написанный нами сценарий:

```
this.stop();
```

и поместить его в этот клип. Модификатор `this` указывает на текущий экземпляр объекта, т. е. на клип `car`.

Мы можем вообще опустить модификатор `this` и записать наш сценарий коротко:

```
stop();
```

Но так делать не рекомендуется.

Если же нам нужно остановить воспроизведение вложенной анимации клипа `wheel`, являющегося составной частью клипа `car`, нам следует написать такой сценарий:

```
car.wheel.stop();
```

Здесь мы написали последовательно имена клипов `car` и `wheel`, разделив их точкой. Фактически мы осуществили доступ к внутреннему объекту.

Подведем итог. Чтобы получить доступ к методу или свойству какого-либо клипа, нужно перечислить имена всех клипов, в которые он вложен, через точку (создав таким образом путь), а уже потом указать нужный метод или свойство. Причем имена клипов указываются в порядке увеличения уровня вложенности в текущий клип (тот, который содержит сценарий). Если же нужно получить доступ к методу или свойству текущего клипа (к которому привязан данных сценарий), достаточно указать только этот метод или свойство.

Если нам нужно обратиться к методу или свойству анимации основного фильма из вложенного в нее клипа, мы воспользуемся модификатором `_root`. Так, мы можем поместить в клип `wheel` сценарий

```
_root.stop();
```

который остановит воспроизведение внешней анимации.

Чтобы получить доступ к внешнему объекту из внутреннего, используется модификатор `_parent`. Так, сценарий, помещенный в клип `wheel`:

```
_parent.stop();
```

остановит воспроизведение анимации клипа `car`, в который вложен клип `wheel`.

А теперь необходимо заметить следующее. Пути доступа к клипам могут быть записаны двумя различными способами. И при написании сценариев мы уже использовали оба этих способа.

Во-первых, доступ к клипу можно получить по его абсолютному пути. *Абсолютный* путь всегда записывается, начиная от внешней анимации, для доступа к которой используется модификатор `_root`. Вот пример абсолютного пути (выделен полужирным шрифтом):

```
_root.car.wheel.gotoAndPlay(1);
```

Этот сценарий будет работать в любом вложенном клипе и в самой внешней анимации, поскольку путь доступа к нужному клипу всегда один и тот же.

Абсолютный путь очень хорош в тех случаях, когда мы хотим обратиться из одного вложенного клипа к другому, находящемуся очень "далеко" от первого. В такой ситуации применение абсолютного пути позволит сделать выражение более компактным.

Во-вторых, мы можем задать относительный путь. *Относительный* путь записывается относительно текущего клипа, в котором находится сценарий. Так, если нам нужно запустить воспроизведение анимации клипа `car` из сценария, находящегося в клипе `wheel`, мы напишем такой сценарий (относительный путь выделен полужирным шрифтом):

```
_parent.gotoAndPlay(1);
```

Относительные пути применяются, если нужно обратиться к другому клипу, находящемуся не очень "далеко". В противном случае он может стать очень длинным.

Для примера напишем два выражения с разными видами путей и сравним их:

```
_parent.gotoAndPlay(1);
_root.car.gotoAndPlay(1);
```

Очевидно, что выражение, использующее относительный путь, компактнее того, которое использует абсолютный. А вот еще два разных выражения, находящихся в клипе `wheel` и останавливающих внешнюю анимацию:

```
_parent._parent.stop();
_root.stop();
```

Видно, что второе выражение, использующее абсолютный путь, компактнее.

Пути доступа к клипам могут быть весьма длинными, например:

```
_root.car.engine.electricPart.ignitor.spark();
```

Чтобы сократить их, можно использовать уже знакомое нам выражение с ключевым словом `with`:

```
with (_root.car.engine.electricPart) {
  headlightLeft.on();
  headlightRight.on();
  ignitor.spark();
}
```

Для вставки в код сценария путей доступа к клипам панель **Actions** предоставляет специальные средства, а именно, кнопку , расположенную над областью редактирования кода. При нажатии этой кнопки на экране появится диалоговое окно **Insert Target Path** (рис. 19.1).

Бóльшую часть этого окна занимает иерархический список клипов, вложенных в основной фильм. (Сам основной фильм там также присутствует.) Переключатели **Absolute** и **Relative**, расположенные под списком, позволяют выбрать способ записи пути, соответствующий их названиям. А в поле ввода, расположенном над списком, мы можем ввести путь доступа к клипу вручную. И, разумеется, после задания нужного пути следует нажать кнопку **OK**.

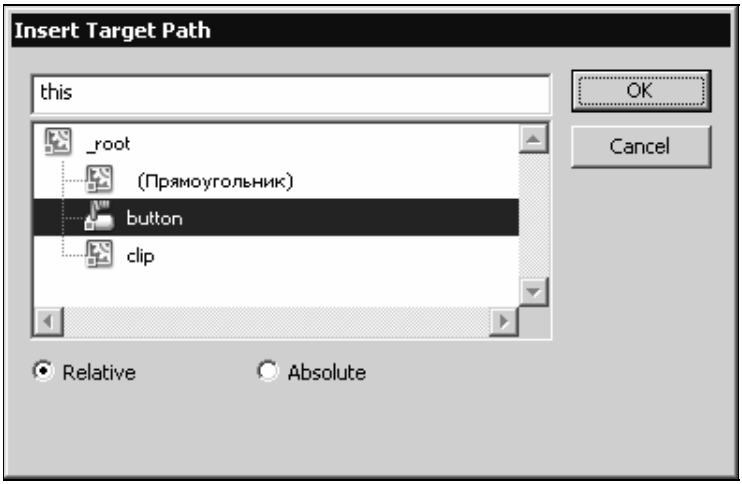


Рис. 19.1. Диалоговое окно **Insert Target Path**

Управление воспроизведением анимации

Обычно каждый клип начинает воспроизводиться сразу же после загрузки и воспроизводится до последнего кадра его анимации. Однако, используя сценарии, мы можем управлять этими процессами, в частности, останавливать и запускать воспроизведение, а также воспроизводить фильм, начиная с заданного кадра.

Мы уже знакомы с методом `stop` объекта `movieClip` — он останавливает воспроизведение клипа. А чтобы запустить воспроизведение клипа с того места, где он был остановлен, нужно воспользоваться методом `play`:

```
car.play();
```

Для осуществления перехода на какой-либо кадр клипа служат методы `gotoAndPlay` и `gotoAndStop`. Первый выполняет переход на заданный кадр и начинает воспроизведение клипа с него. Второй же метод выполняет переход на заданный кадр, но воспроизведение не инициирует. В качестве единственного параметра оба этих метода принимают номер или имя кадра; причем если значение этого параметра имеет числовой тип, то оно трактуется как номер кадра, если строковый — то как его имя. Пример с использованием числового параметра:

```
car.gotoAndPlay(10);
```

Для управления текущим клипом можно пользоваться также действиями `gotoAndPlay` и `gotoAndStop` объекта `movieClip`.

Формат записи их вызова таков:

```
gotoAndPlay|gotoAndStop([<Имя сцены>, ]<Номер или имя кадра>);
```

Здесь второй параметр задает, опять же, номер или имя кадра, на который будет выполнен переход. Первый же, необязательный, параметр задает имя сцены, содержащей этот кадр; если же параметр пропущен, то выполняется переход на кадр текущей сцены. Например:

```
gotoAndPlay(10);
gotoAndStop("Эпизод 1", x + 1);
```

Осталось упомянуть еще о двух методах объекта `movieClip`. Метод `prevFrame` перемещает бегунок на предыдущий кадр клипа, а метод `nextFrame` — на следующий. Ни один из этих методов не принимает параметров.

Методы `prevFrame` и `nextFrame` можно использовать, например, для создания средствами Flash презентации или слайд-шоу. Делается это так. Все изображения (слайды), которые должны войти в презентацию, помещают в отдельных кадрах фильма. К первому кадру фильма привязывается сценарий, содержащий одно-единственное выражение — `stop()`, которое остановит фильм на первом же кадре. После этого остается создать кнопки и привязать к ним сценарии, содержащие выражения `this.prevFrame()` и `this.nextFrame()`, которые будут выполнять переход соответственно на предыдущий и последующий кадр фильма, — и простейшее слайд-шоу готово!

Обработка событий, возникающих в клипах

Обработчики событий клипов имеют такой формат:

```
onClipEvent(<Событие>) {
  <Тело обработчика>
}
```

Собственно, нам он уже знаком из *главы 18*. Вот пример такого обработчика:

```
onClipEvent(mouseDown) {
  this.stop();
}
```

Этот обработчик выполняется в ответ на щелчок мышью на клипе и останавливает воспроизведение вложенной в него анимации. Его код должен быть привязан к самому клипу.

ВНИМАНИЕ!

При написании обработчиков событий клипов крайне желательно использование модификатора `this`, возвращающего ссылку на текущий клип.

Можно создать обработчик, выполняющийся в ответ сразу на несколько событий, перечислив их имена через запятую. Вот так:

```
onClipEvent(mouseDown, mouseMove) {
  stop();
}
```

Этот обработчик выполнится либо после щелчка мышью на клипе (событие `mouseDown`), либо после перемещения над клипом курсора мыши (событие `mouseMove`).

Теперь самое время выяснить, какие события поддерживаются объектом `movieClip`. Некоторые из них приведены в табл. 19.10; полный же список событий можно найти в интерактивной справке Flash.

Таблица 19.10. *Некоторые события, поддерживаемые объектом* `movieClip`

Событие	Описание
`data`	Возникает при приеме клипом внешних данных (если использовалось действие `loadVariables`) или загрузке фильма из внешнего файла (если использовалось действие `loadMovie`). О работе с внешними файлами будет рассказано далее в этой главе
`enterFrame`	Возникает при воспроизведении очередного кадра анимации клипа. Привязанный к нему обработчик выполняется перед всеми сценариями, привязанными к данному кадру
`load`	Возникает сразу после загрузки массива данных образца, на основе которого был создан вложенный клип, и появления на экране самого клипа
`mouseDown`	Возникает при нажатии левой кнопки мыши, когда ее курсор находится над клипом
`mouseMove`	Возникает при перемещении курсора мыши над клипом
`mouseUp`	Возникает при отпускании левой кнопки мыши, когда ее курсор находится над клипом
`unload`	Возникает после того, как клип пропадет с экрана, обычно — при воспроизведении следующего кадра, в котором уже нет этого клипа. Привязанный к нему обработчик выполняется перед всеми сценариями, привязанными к следующему кадру

Событие `enterFrame` возникает при воспроизведении очередного кадра анимации клипа. Это значит, что обработчик данного события будет выпол-

няться постоянно с частотой, равной частоте кадров фильма, до тех пор, пока фильм не закончится. Это можно использовать для выполнения различных задач, например, для отслеживания курсора мыши.

События `load` и `unload` можно использовать для запуска или останова воспроизведения других клипов. Например, при загрузке какого-либо клипа воспроизведение другого клипа может приостанавливаться, а после выгрузки — продолжаться. Также можно написать сценарий, загружающий после выгрузки одного клипа другой и запускающий его на воспроизведение.

События `mouseDown` и `mouseUp` можно, по идее, использовать для создания на рабочем листе подобия кнопок. Но для таких целей лучше использовать полноценные кнопки Flash, создание которых было описано в *главе 18*.

Интересная особенность клипов Flash состоит в том, что они могут обрабатывать события кнопки. Для этого достаточно привязать к клипу "кнопочный" обработчик события, код которого показан ниже:

```
on(press) {
  stop();
}
```

И клип будет вести себя как кнопка.

Полный список событий, поддерживаемых кнопками Flash, будет приведен далее в этой главе.

Управление клипами

А теперь рассмотрим средства, предоставляемые Flash для управления клипами, находящимися на рабочем листе.

Создание новых клипов

Flash предоставляет возможность создавать экземпляры-клипы на основе имеющихся в библиотеке образцов, дублировать уже созданные на рабочем листе клипы и даже создавать абсолютно новые клипы, не основанные на имеющихся в библиотеке образцах. Давайте выясним, как все это реализовано.

Дублирование существующего клипа

Проще всего, конечно, создать новый клип, продублировав уже имеющийся на рабочем листе, вместе с его графикой, анимацией и сценариями. Для этого используется действие `duplicateMovieClip`. Формат его вызова таков:

```
duplicateMovieClip(<Дублируемый клип>, "<Имя создаваемого клипа>",
↳<Порядок перекрытия нового клипа>);
```

С первым параметром передается путь исходного клипа, который нужно продублировать. Со вторым параметром передается имя вновь создаваемого клипа; оно должно иметь строковый тип и быть уникальным. Последний, третий, параметр определяет порядок перекрытия нового клипа; он должен иметь числовой тип.

Однако при создании нового клипа способом дублирования нужно учесть четыре момента. Во-первых, содержимое переменных, объявленных в сценариях дублируемого клипа, а также его свойств не копируется в новый клип. Во-вторых, если дублируемый клип содержит анимацию, то в новом клипе она в любом случае начнет воспроизводиться с самого начала. В-третьих, созданный таким образом клип вкладывается в тот клип, в котором находится продублированный клип. В четвертых, созданные с помощью действия `duplicateMovieClip` клипы в порядке перекрытия всегда находятся "выше" созданных вручную.

Приведем пример сценария, использующего действие `duplicateMovieClip` и находящегося в клипе `car`:

```
duplicateMovieClip(wheel2, "wheel5", 3);
```

(Мы только что приделали к нашему автомобилю пятое, запасное, колесо!)

Вместо действия `duplicateMovieClip` можно использовать одноименный метод объекта `movieClip`. Вот формат его вызова:

```
<Дублируемый клип>.duplicateMovieClip("<Имя создаваемого клипа>",
↳<Порядок перекрытия нового клипа>);
```

Создание нового клипа на основе образца. Сценарные образцы

Чтобы создать совершенно новый клип, основанный на библиотечном образце-клипе, используется метод `attachMovie` объекта `movieClip`. Формат его вызова следующий:

```
<Клип>.attachMovie("<Имя сценарного образца>",
↳"<Имя создаваемого клипа>", <Порядок перекрытия нового клипа>);
```

Метод `attachMovie` вызывается для клипа, в который должен быть вложен вновь создаваемый клип. Второй и третий параметры нам уже знакомы, так что не будем останавливаться на них. А вот о первом параметре нужно поговорить подробнее.

Дело в том, что для успешного использования в методе `attachMovie` образец-клип нужно превратить в так называемый *сценарный* — доступный для вызова из сценария. Сейчас мы выясним, как это делается.

Первым делом нужно вывести на экран панель **Library**. Далее найдем в списке этой панели нужный образец и выберем в контекстном или дополни-

тельном меню пункт **Linkage**, после чего на экране появится давно знакомое нам диалоговое окно **Linkage Properties** (см. рис. 8.21).

Чтобы превратить образец в сценарный, нужно включить флажок **Export for ActionScript** и ввести идентификатор создаваемого образца в поле ввода **Identifier**. Все почти так же, как и в случае разделяемых образцов.

Далее нам нужно решить, куда будут помещены массивы данных этого сценарного образца. По умолчанию все они помещаются в первый кадр фильма и, соответственно, загружаются перед тем, как фильм начнет воспроизводиться. Если таких образцов слишком много или если они очень велики, пауза перед выводом первого кадра фильма может затянуться. Но мы можем заставить Flash поместить массив данных образца в тот кадр фильма, в котором они впервые встречаются. Для этого достаточно отключить флажок **Export in first frame**.

Однако здесь нас подстерегает другой сюрприз. Если мы не использовали где-либо в фильме сценарный образец, то Flash не поместит его в результирующий файл Shockwave/Flash, справедливо полагая, что он здесь лишний. Что мы там написали в сценариях, его (Flash) не касается. Так вот, для того чтобы Flash все же сохранил сценарный образец в готовом фильме, нам придется все-таки поместить его экземпляр на рабочий лист, задав для него либо очень маленькие размеры, либо максимальную прозрачность.

Вот, собственно, и все. Осталось нажать кнопку **OK** окна **Linkage Properties**. И можно использовать готовый сценарный образец.

Приведенный ниже сценарий добавит в наш автомобиль лишнюю фару, создав ее на основе сценарного образца `headlight`:

```
_root.car.attachMovie("headlight", "headlight3", 0);
```

Создание "пустого" клипа

Flash также позволяет поместить на рабочий лист "пустой" клип. Это выполняется с помощью метода `createEmptyMovieClip` объекта `movieClip`. Формат вызова метода такой:

```
<Клип>.createEmptyMovieClip("<Имя создаваемого клипа>",
↳<Порядок перекрытия создаваемого клипа>);
```

Метод `createEmptyMovieClip` вызывается для того клипа, в который нужно вложить создаваемый "пустой" клип. Оба его параметра нам уже знакомы.

Запишем сценарий с использованием этого метода:

```
_root.createEmptyMovieClip("tractor", 0);
```

Этот сценарий поместит на рабочий лист новый "пустой" клип `tractor` — в компанию к нашему многострадальному автомобилю. Конечно, никакого трактора там пока что нет, но — лиха беда начало...

Удаление клипа

Когда-нибудь нужда в созданных с помощью всех этих действий и методов клипах отпадет, и их придется удалить. Для удаления же клипов предназначено единственное действие `removeMovieClip`, записываемое в следующем формате:

```
removeMovieClip(<Удаляемый клип>);
```

Например, для того чтобы удалить пятое колесо, подойдет такой сценарий:

```
removeMovieClip(_root.car.wheel5);
```

Обойдемся без запаски — авось гаишник не остановит.

Для удаления клипа можно также воспользоваться методом `removeMovieClip` объекта `movieClip`. Формат его вызова очень прост:

```
<Удаляемый клип>.removeMovieClip();
```

> ***ВНИМАНИЕ!***
>
> С помощью действия или метода `removeMovieClip` можно удалить только те клипы, которые созданы из сценариев. Клипы, созданные в среде Flash, не могут быть удалены таким образом.

Изменение параметров встроенных клипов

Не сто́ит думать, что Flash может только создавать, дублировать и удалять вложенные клипы. Сейчас мы рассмотрим свойства и методы объекта `movieClip`, с помощью которых можно изменять различные параметры встроенных клипов: местоположение, размеры и пр.

Так, свойства `_x` и `_y` объекта `movieClip` предоставляют доступ соответственно к горизонтальной и вертикальной координатам клипа. Эти координаты отсчитываются в пикселах от точки отсчета клипа и задаются относительно внешнего клипа, т. е. того, в который текущий клип вложен.

Вот пример сценария, изменяющего координаты "фар" в нашем клипе `car`:

```
_root.car._x = 200;
_root.car.headlight1._y = _root.car.headlight2._y + 20;
```

Свойства `_width` и `_height` содержат соответственно ширину и высоту клипа.

Свойства `_xscale` и `_yscale` задают процент изменения размеров клипа соответственно по горизонтали и вертикали. Значения, бо́льшие 100, заставляют клип увеличить соответствующий размер, а меньшие 100 — уменьшить. Так, если задать для свойства `_xscale` значение 50, то клип сожмется по горизонтали вдвое.

Свойство _alpha позволяет получить или задать степень прозрачности клипа. Доступны любые целые значения от 0 (полная прозрачность) до 100 (полная непрозрачность).

Свойство _rotation задает угол поворота клипа в градусах. Например:

```
onClipEvent(enterFrame) {
  ++this._rotation;
  --this._alpha;
}
```

Этот сценарий заставляет вложенный клип в процессе воспроизведения фильма плавно поворачиваться и одновременно плавно исчезать. Фактически мы можем создавать настоящую анимацию только средствами ActionScript.

Свойство _visible позволяет сделать клип невидимым. Оно имеет логический тип: значение true делает клип видимым, а значение false — невидимым.

```
onClipEvent(mouseDown) {
  this._visible = ~this._visible;
}
```

Этот сценарий при каждом щелчке по клипу делает его то видимым, то невидимым.

А теперь познакомимся с методами объекта movieClip, с помощью которых можно изменять порядок перекрытия встроенных клипов.

Метод getDepth возвращает порядок перекрытия текущего клипа. Этот метод не принимает параметров.

Метод getNextHighestDepth возвращает ближайшее верхнее значение уровня перекрытия, еще ничем не занятое. Его можно использовать при создании новых клипов. Например, так:

```
var clip2 = clip1.duplicateMovieClip("clip2",
↳_root.getNextHighestDepth());
```

Это выражение создаст новый клип clip2, продублировав уже имеющийся на рабочем листе клип clip1, и поместит его "над" всеми клипами.

Метод getInstanceAtDepth возвращает ссылку на клип, имеющий заданный порядок перекрытия. Значение порядка перекрытия передается в качестве единственного параметра.

Метод swapDepths меняет значения порядка перекрытия двух клипов местами. Формат его вызова таков:

```
<Клип 1>.swapDepths("<Клип 2>"|<Порядок перекрытия>);
```

В качестве параметра этого метода может быть передан путь второго клипа или значение нужного порядка перекрытия. Причем путь второго клипа передается как строка.

Вот три примера использования этого метода:

```
_root.car.swapDepths("_root.tractor");
_root.car.swapDepths(10);
_root.car.swapDepths(_root.tractor.getDepth());
```

Получение служебной информации о клипе

Объект `movieClip` поддерживает набор особых свойств, доступных только для чтения. Эти свойства возвращают различную служебную информацию, зачастую не относящуюся к самому клипу. Ниже перечислены некоторые из них.

Свойство `_name` возвращает имя клипа, заданное с помощью панели **Properties**, в строковом виде.

Свойство `_target` возвращает полный путь клипа в строковом виде.

Свойства `_xmouse` и `_ymouse` возвращают соответственно горизонтальную и вертикальную координаты курсора мыши относительно точки отсчета клипа. Пользуясь этими свойствами, можно контролировать перемещения мыши. Например:

```
onClipEvent(enterFrame) {
  myCursor._x = _root._xmouse;
  myCursor._y = _root._ymouse;
}
```

Этот сценарий заставляет вложенный клип `myCursor` перемещаться вслед за курсором мыши. Фактически сам клип будет вести себя как курсор мыши.

Свойство `_currentframe` возвращает номер кадра клипа, на котором в данный момент стоит бегунок (фактически номер воспроизводимого в данный момент кадра).

Свойство `_totalframes` возвращает общее количество кадров, содержащихся во внутренней анимации клипа.

Свойство `_url` возвращает интернет-адрес, с которого был загружен клип или фильм. Иногда это тоже может пригодиться.

Свойство `_framesloaded` возвращает количество кадров клипа, данные которых проигрыватель Flash уже загрузил. А методы `getBytesLoaded` и `getBytesTotal` помогут узнать соответственно количество загруженных байтов клипа и полный размер клипа в байтах. Используя их, мы можем создавать индикаторы загрузки клипа или фильма.

Реализация drag'n'drop

Термином "*drag'n'drop*" ("тащи и бросай") обозначают целый набор операций, связанных с перетаскиванием каких-либо объектов операционной системы (файлов, папок, ярлыков), системных или прикладных программ. Так, во Flash с помощью перетаскивания изменяется местоположение фрагментов изображения и бегунка панели **Timeline**.

Мы также можем реализовать операции "drag'n'drop" в своих фильмах Flash. Сейчас мы выясним, как это сделать.

Первое, что нам понадобится, — это действие `startDrag`. Оно делает клип доступным для буксировки с помощью мыши. Формат вызова действия такой:

```
startDrag(<Клип>, [<За центр>, <X1>, <Y1>, <X2>, <Y2>]);
```

В качестве единственного обязательного параметра действию передается путь клипа, который нужно сделать перетаскиваемым. А вот об остальных пяти необязательных параметрах давайте поговорим подробнее.

Второй параметр (`за центр`) имеет логический тип. Он позволяет указать точку, к которой будет автоматически "приклеиваться" курсор мыши при перетаскивании. Это может быть точка, в которой пользователь щелкнул мышью (значение `false`), или центр клипа (значение `true`).

Остальные четыре параметра позволяют описать прямоугольную область, внутри которой можно будет перетаскивать клип и за пределы которой он не сможет выйти. Через параметры `X1` и `Y1` задаются горизонтальная и вертикальная координаты левого верхнего угла этой области, а за координаты правого нижнего угла "отвечают" параметры `X2` и `Y2`. Все координаты задаются относительно внешнего клипа.

Вот два примера применения действия `startDrag`:

```
startDrag(_root.car);
```

```
startDrag(_root.car, true, 100, 100, 500, 500);
```

Первое выражение просто делает клип перетаскиваемым. При этом пользователь сможет "ухватить" его мышью и перетащить на новое место. Второе же выражение, кроме этого, задает дополнительные параметры перетаскиваемого клипа.

Вместо действия `startDrag` можно использовать одноименный метод объекта `movieClip`:

```
<Клип>.startDrag([<За центр>, [<X1>, <Y1>, <X2>, <Y2>]]);
```

После вызова действия или метода `startDrag` клип останется перетаскиваемым на сколь угодно долгое время. Чтобы завершить операцию "drag'n'drop",

например, если пользователь отпустит кнопку мыши, нужно вызвать действие или метод `stopDrag`. Вот так (действие):

```
stopDrag();
```

или так (метод):

```
_root.car.stopDrag();
```

> ***НА ЗАМЕТКУ***
>
> Для завершения операции "drag'n'drop" также можно вызвать еще раз действие или метод `startDrag`, но уже для другого клипа. Только один клип в данный момент времени может быть перетаскиваемым.

Теперь давайте подумаем, как применить только что полученные сведения в деле написания реальных сценариев-обработчиков событий.

Для запуска и завершения операции "drag'n'drop" можно использовать обработчики событий `mouseDown` и `mouseUp` клипа — это очевидно. Давайте напишем эти обработчики и привяжем их к клипу, который нужно сделать перетаскиваемым.

```
onClipEvent(mouseDown) {
  this.startDrag(false);
}
onClipEvent(mouseUp) {
  this._x = _root._xmouse;
  this._y = _root._ymouse;
  this.stopDrag();
}
```

Обработчик события `mouseDown` запускает операцию перетаскивания клипа. А обработчик события `mouseUp` завершает эту операцию и одновременно помещает клип в ту точку, в которой пользователь отпустил кнопку мыши. Как видим, здесь нет ничего сложного.

Объект `movieClip` предоставляет доступное только для чтения свойство `_droptarget`, возвращающее ссылку на так называемую *"цель"* операции "drag'n'drop" — на клип, в который был "брошен" перетаскиваемый клип. Можно проверить результат, возвращаемый этим свойством, и, в зависимости от него, завершить или не завершить операцию перетаскивания.

Здесь имеет место небольшое затруднение. Дело в том, что свойство `_droptarget` возвращает результат в особом формате, применяемом в старых версиях Flash (это версии 1—4). Это так называемая "запись со слэшем", когда в качестве разделителя между именем экземпляра объекта и названием его свойства или метода используется косая черта (/), а не точка. Для того

чтобы преобразовать "запись со слэшем" в знакомую нам "запись с точкой", нужно воспользоваться встроенной во Flash функцией `eval`.

Давайте перепишем сценарии-обработчики событий `mouseDown` и `mouseUp` так, чтобы они не позволяли пользователю "бросить" перетаскиваемый клип в клип по имени `forbiddenArea`. Сценарии будут выглядеть так:

```
onClipEvent(mouseDown) {
  this.startDrag(false);
  oldX = this._x;
  oldY = this._y;
}
onClipEvent(mouseUp) {
  if (eval(this._droptarget) != _root.forbiddenArea) {
    this._x = _root._xmouse;
    this._y = _root._ymouse;
  } else {
    this._x = oldX;
    this._y = oldY;
  }
  this.stopDrag();
}
```

Теперь первый обработчик, помимо запуска операции "drag'n'drop", сохраняет изначальные координаты перетаскиваемого клипа в переменных `oldX` и `oldY`. Второй же обработчик, обнаружив, что пользователь перетащил клип в "запрещенное место", восстанавливает изначальное положение перетаскиваемого клипа. В обоих случаях операция "drag'n'drop" завершается.

Так же просто можно написать сценарий, который, наоборот, разрешит "бросать" перетаскиваемый элемент только в одно определенное место. Вообще, можно много чего придумать...

Создание фигурного курсора мыши

Очень многие приложения (не только созданные во Flash, но и обычные, работающие в системе Windows) "щеголяют" фигурными курсорами мыши. Давайте выясним, как создать такой курсор средствами Flash.

Предположим, что мы создали образец-клип и нарисовали в нем курсор желаемой формы. Далее мы поместили экземпляр созданного клипа на рабочий лист и назвали его `cursor`. Теперь, чтобы превратить этот клип в фи-

гурный курсор мыши, привяжем к нему такой обработчик события по имени `load`:

```
onClipEvent(load) {
  this.startDrag(true);
}
```

Обработчик `load` сразу же после загрузки клипа делает его перетаскиваемым. После этого клип `cursor` будет всегда перемещаться за курсором мыши и фактически сам станет курсором, чего мы и добивались.

Казалось бы, все замечательно. Одно плохо: "родной" курсор мыши, рисуемый самой системой Windows, портит всю картину. Его нужно спрятать. Для этого воспользуемся методом `hide` пока еще не изученного нами объекта `Mouse` (он будет рассмотрен далее в этой главе). Этот метод скрывает системный курсор мыши.

Перепишем обработчик события `load`. Он будет выглядеть так:

```
onClipEvent(load) {
  Mouse.hide();
  this.startDrag(true);
}
```

Вот теперь все работает, как надо.

Добавим к сказанному, что можно задать область, в которой будет действовать созданный нами фигурный курсор мыши. Для этого достаточно изменить второе выражение приведенного выше обработчика:

```
this.startDrag(true, 200, 100, 600, 400);
```

После этого наш фигурный курсор будет "заперт" в указанных границах.

Выявление совпадений и касаний

Часто бывает нужно знать, находится ли заданная точка внутри какого-либо элемента изображения или касаются ли друг друга два элемента. Эта задача — выявление совпадений и касаний — решается в разных случаях поразному. И, как правило, ее зачастую весьма непростое решение должно быть найдено программистом.

В нашем случае, однако, все намного проще. Разработчики Flash предусмотрели в объекте `movieClip` особый метод `hitTest`. С его помощью мы можем проверить, находится ли заданная точка внутри клипа или касаются ли друг друга два отдельных клипа.

Метод `hitTest` имеет два формата вызова. Первый из этих форматов позволяет выяснить совпадение точки и клипа:

```
<Клип>.hitTest(<X>, <Y>, <Проверять контур>);
```

Понятно, что первые два параметра определяют соответственно горизонтальную и вертикальную координаты нужной нам точки. Третий параметр имеет логический тип и позволяет указать, что считать границами клипа. Если этот параметр имеет значение `true`, Flash считает границами клипа контур содержащейся в нем графики. В случае значения `false` Flash "мысленно" вписывает клип в невидимый прямоугольник и считает границами клипа его контур.

Если заданная точка находится внутри клипа, метод `hitTest` возвращает значение `true`. В противном случае — значение `false`. Возвращаемое значение можно использовать в условном выражении или где-либо еще.

Давайте проверим, как работает этот метод. Создадим новый клип, представляющий собой сложную фигуру из множества кривых. Поместим его на рабочий лист и присвоим ему имя `hit`. После этого привяжем к этому клипу такой обработчик события `enterFrame`:

```
onClipEvent(enterFrame) {
  if (this.hitTest(_root._xmouse, _root._ymouse, true)) {
    this._alpha = 50;
  } else {
    this._alpha = 100;
  }
}
```

Этот сценарий-обработчик события проверяет, находится ли курсор мыши внутри контура нашего клипа. Если это так, клип делается полупрозрачным, в противном случае — полностью непрозрачным (видимым).

В нашем случае метод `hitTest` проверяет только "попадание" в сам клип, поскольку его третий параметр имеет значение `true`. Если же передать значение `false`, то метод станет проверять попадание в воображаемый прямоугольник, описанный вокруг нашего клипа. Это более грубая проверка совпадения точки и клипа.

Второй формат метода `hitTest` позволит нам проверить на совпадение два различных клипа, т. е. проверить, накладываются они друг на друга или перекрываются:

```
<Клип 1>.hitTest(<Клип 2>);
```

В этом случае метод `hitTest` проверяет "попадание" не фигуры в фигуру, а воображаемого прямоугольника, описанного вокруг первой фигуры, в воображаемый прямоугольник, описанный вокруг второй фигуры. Следовательно, метод ведет себя так, словно его в данный момент не существующий третий параметр имеет значение `false`.

Нам осталось проверить работу метода. Создадим новый образец-клип со сложной фигурой, поместим его экземпляр на рабочий лист, сделаем его поменьше и назовем `cursor`. К первому (и единственному) кадру фильма привяжем вот такой небольшой сценарий:

```
_root.cursor.startDrag(true);
```

Этот сценарий заставляет клип `cursor` двигаться за мышью, т. е. превращает его в фигурный курсор.

Теперь напишем сценарий-обработчик события `enterFrame` для клипа `hit`, так чтобы он проверял касание клипов `cursor` и `hit` одного другим:

```
onClipEvent(enterFrame) {
  if (this.hitTest(_root.cursor)) {
    this._alpha = 50;
  } else {
    this._alpha = 100;
  }
}
```

Работа с графикой

Хоть Flash и предоставляет мощные средства рисования графики и создания анимации, иногда бывает нужно кое-что дорисовать уже во время воспроизведения фильма. Например, если мы собираемся создать фильм, который в процессе воспроизведения будет выводить на рабочий лист график какой-либо функции, нам потребуются средства для рисования этого самого графика. Что в этом случае может предложить нам Flash?

А предлагает он целый набор методов и свойств объекта `movieClip`, служащих для рисования различных линий и заливок. Графика, нарисованная с их помощью, называется *созданной программно* или просто *программной* графикой.

В *главе 14* мы узнали о маскирующих слоях. Но Flash не останавливается на этом — он позволяет преобразовать в маску любой клип, за исключением, разумеется, внешней анимации.

Программное рисование графики

В этом разделе мы поговорим о том, как выполняется рисование графики из сценариев и какие методы объекта `movieClip` для этого используются.

Но сначала нужно уяснить два момента.

Во-первых, вся программно нарисованная графика выводится "ниже" нарисованной вручную, т. е. с помощью инструментов рисования Flash. Говорят, что программная графика имеет меньший номер в порядке перекрытия, чем нарисованная вручную. Поэтому перед написанием сценариев, рисующих графику, нужно позаботиться о том, чтобы созданная различными способами графика взаимно не перекрывалась. Рекомендуется выделить для программно рисуемой графики специальные "пустые" клипы, находящиеся на самых верхних слоях.

Во-вторых, как уже говорилось, рисование графики выполняется с помощью специальных методов объекта `movieClip`. Это значит, что программная графика будет нарисована в том клипе, методы которого мы будем вызывать.

Перейдем к делу.

Для рисования графики из сценариев используется так называемое *программное перо* (не путать с инструментом рисования "перо"!). С помощью особых методов объекта `movieClip` это перо можно устанавливать в определенные точки на рабочем листе и проводить им линии, прямые и кривые. Также с помощью программного пера задаются точки, где должны быть помещены заливки.

Для перемещения программного пера в нужную точку рабочего листа используется метод `moveTo`. Нужно, правда, иметь в виду, что этот метод только перемещает перо, но реально ничего не рисует. Вот формат вызова этого метода:

```
<Клип>.moveTo(<X>, <Y>);
```

Понятно, что в качестве параметров методу передаются горизонтальная и вертикальная координаты точки, в которую нужно переместить программное перо. Эти координаты отсчитываются относительно точки отсчета клипа.

Проще всего программно нарисовать прямую линию, использовав метод `lineTo`. Он проводит прямую от точки, в которой в данный момент находится программное перо, до точки, чьи координаты переданы в качестве параметров. Вот формат вызова этого метода:

```
<Клип>.lineTo(<X>, <Y>);
```

Как уже говорилось, параметры этого метода есть координаты конечной точки рисуемой прямой. Они отсчитываются от точки отсчета клипа либо от левого верхнего угла, если метод вызван для самого фильма. После рисования прямой линии программное перо останется в ее конечной точке; это позволяет без лишних перемещений нарисовать ломаную или какую-либо геометрическую фигуру.

Вот пример сценария, рисующего в клипе paintBox треугольник:

```
with (_root.paintBox) {
  moveTo(100, 200);
  lineTo(300, 200);
  lineTo(200, 100);
  lineTo(100, 200);
}
```

Рисование кривых выполняется чуть сложнее. Для этого используется метод curveTo, принимающий четыре параметра:

```
<Клип>.curveTo(<AX>, <AY>, <X>, <Y>);
```

Параметры X и Y определяют соответственно горизонтальную и вертикальную координаты конечной точки кривой (начальной же точкой является, опять же, точка, в которой в данный момент находится программное перо). А параметры AX и AY указывают горизонтальную и вертикальную координаты точки искривления. Все эти координаты отсчитываются, опять же, относительно точки отсчета клипа.

Вот пример сценария, рисующего нечто похожее на окружность:

```
with (_root.paintBox) {
  moveTo(300, 300);
  curveTo(400, 300, 400, 200);
  curveTo(400, 100, 300, 100);
  curveTo(200, 100, 200, 200);
  curveTo(200, 300, 300, 300);
}
```

Увы — более удобных способов программно нарисовать окружность Flash не предлагает. Приходится или создавать образец, рисовать в нем окружность и программно порождать от него экземпляры, или выходить из положения вышеописанным способом. Хотя, можно еще описать окружность в виде функции и нарисовать ее график.

Для рисования линии, неважно — прямой или кривой, будут использованы текущие параметры ее стиля: толщина, цвет и прозрачность. Задание текущего стиля выполняется с помощью метода lineStyle. Он вызывается вот так:

```
<Клип>.lineStyle([<Толщина линии>, [<Цвет линии>,
↳[<Прозрачность линии>]]]);
```

Толщина линии задается в пикселах, от 0 до 255. Если она не задана, то линия нарисована не будет. Цвет линии задается в формате 0xRRGGBB, где RR — доля красного цвета, GG — зеленого и BB — синего соответственно. Пример

задания цвета: `0x22FF44`. Прозрачность линии может принимать значения от 0 (полная прозрачность) до 100 (полная непрозрачность).

Важный момент: перед рисованием любой линии лучше с помощью метода `lineStyle` задать ее стиль. Flash сохранит его и сделает текущим; все последующие линии будут нарисованы с использованием текущего стиля.

Поэтому сценарий, рисующий треугольник, нам придется переписать:

```
with (_root.paintBox) {
  lineStyle(2, 0x000000);
  moveTo(100, 200);
  lineTo(300, 200);
  lineTo(200, 100);
  lineTo(100, 200);
}
```

Здесь мы с помощью метода `lineStyle` задали стиль линии: толщина в 2 пиксела и черный цвет. После этого треугольник уж точно будет нарисован.

А теперь давайте посмотрим на только что нарисованный треугольник и подумаем: чего же в нем не хватает? В *главе 3* мы выяснили, что, если нарисовать замкнутую фигуру, Flash автоматически создаст заливку; при этом не имеет значения, какой инструмент рисования использовался. Здесь же заливки нет. Почему?

Вероятно, разработчики Flash решили, что, раз уж мы пользуемся сценариями ActionScript, значит, мы достаточно квалифицированные программисты, и нам не нужна помощь. Поэтому нам придется самостоятельно создавать заливку для нарисованных программно фигур.

Для создания заливки служат два метода. Первый из них — `beginFill` — указывает Flash, что с этого момента нужно отслеживать все операции программного рисования и после создания замкнутой фигуры учесть на будущее, что ей нужна заливка. Второй — `endFill` — прекращает отслеживание операций рисования и рисует собственно заливку (если, конечно, ее можно нарисовать, т. е. если нарисованный нами контур полностью или частично замкнут).

Формат вызова метода `beginFill` такой:

```
<Клип>.beginFill([<Цвет заливки>, [<Прозрачность заливки>]]);
```

Цвет заливки задается в том же формате `0xRRGGBB`, а прозрачность — значением от 0 до 100. Если эти параметры не заданы, будут использованы текущие значения цвета и прозрачности заливки — те, что были указаны при предыдущем вызове метода `beginFill`. Поэтому при самом первом вызове этого метода нужно указать оба его параметра.

Метод `endFill` не принимает ни одного параметра:

```
<Клип>.endFill();
```

Теперь еще раз перепишем сценарий, чтобы он рисовал треугольник с заливкой:

```
with (_root.paintBox) {
  lineStyle(2, 0x000000);
  beginFill(0x222233, 100);
  moveTo(100, 200);
  lineTo(300, 200);
  lineTo(200, 100);
  lineTo(100, 200);
  endFill();
}
```

С помощью метода `beginFill` создается обычная, сплошная заливка. Если же нужно создать градиентную заливку, то придется воспользоваться методом `beginGradientFill` в паре с методом `endFill`. Полное описание этого метода приведено в интерактивной справке Flash.

Чтобы удалить все, что было нарисовано программно, нужно вызвать метод `clear`:

```
_root.paintBox.clear();
```

При этом будет также сброшен заданный с помощью метода `lineStyle` стиль линий. Графика же, нарисованная вручную, удалена не будет.

Пример сценария, рисующего график функции

А теперь давайте попрактикуемся и напишем сценарий, рисующий график функции $f(x) = x^2$. Это, скорее, курьез, чем реально необходимый пример, но он поможет нам глубже изучить методы рисования объекта `movieClip`.

Сначала создадим новый документ Flash. И привяжем к единственному кадру анимации следующий сценарий:

```
var dx = 200;
// Величина отступа от левого края рабочего листа
var x = 0;
var y = 0;
// Временные переменные
// Каждая точка графика будет представлять собой небольшой квадрат с
// черной заливкой
```

```
lineStyle(1, 0x000000);
// Стиль линий контура этого квадрата
for(x = -20; x < 21; x++) {
  // Используем цикл со счетчиком, чтобы вычислить значения функции
  // для аргументов в пределах от -20 до 20
  y = x * x;
  // Вычисляем значение функции
  beginFill(0x000000);
  // Начинаем рисование заливки квадрата
  moveTo(dx + x - 1, y - 1);
  lineTo(dx + x + 1, y - 1);
  lineTo(dx + x + 1, y + 1);
  lineTo(dx + x - 1, y + 1);
  lineTo(dx + x - 1, y - 1);
  // Рисуем квадрат
  endFill();
  // Заканчиваем рисование заливки
}
```

Теперь запустим на воспроизведение готовый фильм в отдельном окне. Работает!

Правда, график наш рисуется вверх ногами. Чтобы исправить этот огрех, перепишем наш сценарий. Теперь он будет выглядеть вот так (добавленный и измененный код выделен полужирным шрифтом):

```
var dx = 200;
dy = 400;
// Величина отступа от верхнего края рабочего листа
var x = 0;
var y = 0;
lineStyle(1, 0x000000);
for(x = -20; x < 21; x++) {
  y = x * x;
  beginFill(0x000000);
  moveTo(dx + x - 1, dy - y + 1);
  lineTo(dx + x + 1, dy - y + 1);
  lineTo(dx + x + 1, dy - y - 1);
```

```
  lineTo(dx + x - 1, dy - y - 1);
  lineTo(dx + x - 1, dy - y + 1);
  endFill();
}
```

После этих изменений наш график будет нарисован нормально.

Программное управление масками

А теперь выясним, что можно сделать с масками, используя методы объекта `movieClip`.

Чтобы маскировать вложенный клип другим клипом (который будет превращен в маскирующий), используется метод `setMask`:

```
<Маскируемый клип>.setMask(<Клип-маска>);
```

Самое интересное заключается в том, что оба клипа совсем не обязательно должны находиться в разных слоях (о слоях *см. главу 5*, о маскирующих слоях — *главу 14*). Кроме того, и маскирующий, и маскируемый клипы могут содержать сложную анимацию и даже двигаться. Ух, каких чудес можно натворить!

Давайте создадим новый документ Flash, поместим на рабочий лист клип, содержащий любое растровое изображение, а на него наложим клип с какой-либо простой геометрической фигурой. Назовем нижний клип `masked`, а верхний — `mask`. После этого привяжем к единственному кадру анимации такой вот сценарий:

```
masked.setMask(mask);
mask.startDrag(true);
```

Первое выражение этого сценария маскирует клип-растровое изображение `masked` клипом `mask`. Второе выражение заставляет клип `mask` (т. е. маску) перемещаться вслед за курсором мыши.

Теперь запустим на воспроизведение созданный фильм в отдельном окне и начнем перемещать курсор мыши с места на место. Забавно получилось, не правда ли?

Кстати, маски, созданные путем вызова метода `setMask`, имеют приоритет над масками, созданными вручную. Иными словами, если мы создадим маскирующий слой с клипом-маской `mask` и замаскируем им другой клип `masked`, то выражение:

```
mask.setMask(masked);
```

сделает клип `masked` маской, а клип `mask` — маскируемым.

Маскируемый клип в данный момент времени может иметь только одну маску. Так, если для клипа `masked` вызвать метод `setMask` еще раз, но пере-

дать ему в качестве параметра клип `mask2`, то он станет новой маской для `masked`, а предыдущая маска `mask` "уйдет в отставку".

Чтобы убрать маску совсем, достаточно вызвать метод `setMask` с параметром `null`:

```
masked.setMask(null);
```

Работа с кнопками

А теперь поговорим о кнопках и работе с ними с помощью сценариев. (Подробнее о кнопках *см. главу 18.*)

Все находящиеся на рабочем листе кнопки, для которых было задано имя, представляют собой экземпляры объекта `Button`. Их создает сам Flash; нам же остается только использовать их свойства и методы.

Оператор `typeof` для кнопки возвращает строку `"object"` — это значит, что кнопка относится к объектному типу данных.

Свойства и методы объекта *Button*

По своим свойствам и методам объект `Button` во многом похож на объект `movieClip`. Имея весьма схожие наборы свойств и методов, эти два объекта зачастую ведут себя одинаково. В частности, объект `Button` поддерживает свойства `_x`, `_y`, `_xmouse`, `_ymouse`, `_height`, `_width`, `_alpha` и `_rotation`, уже знакомые нам по объекту `movieClip`. Однако объект `Button` не поддерживает методы `play`, `stop`, `gotoAndPlay`, `gotoAndStop` и подобные им, так как назначение его все же иное, чем у объекта `movieClip`. Так что кнопку можно рассматривать как очень специализированный, "урезанный" вариант клипа.

Рассмотрим специфические свойства объекта `Button`, которые обязательно пригодятся нам в работе с его экземплярами. (Кстати, многие из этих свойств поддерживаются и объектом `movieClip`, но редко используются в клипах.)

Свойство `enabled` позволяет разрешить или запретить пользователю нажимать кнопку. Оно имеет логический тип: значение `true` разрешает доступ к кнопке, а значение `false` — запрещает.

Свойство `_visible` позволяет сделать кнопку видимой или невидимой. Оно имеет логический тип: значение `true` делает кнопку видимой, а значение `false` — невидимой. При этом невидимая кнопка также недоступна для нажатия.

Свойство `useHandCursor` задает, какую форму примет курсор мыши, если поместить его над кнопкой. Если задано значение `true`, то отображается курсор в виде "указующего перста" (как над гиперссылкой). Если же задано значение `false`, то курсор имеет вид обычной стрелки.

Известно, что обычные кнопки в окнах Windows-приложений можно нажимать, не только щелкая по ним мышью, но и нажимая клавишу <Space> основной клавиатуры. Для этого кнопка должна иметь *фокус ввода*, который даст ей возможность принимать нажатия клавиш клавиатуры. Кнопка, имеющая фокус ввода, выделяется точечным прямоугольником, окружающим ее надпись.

Фокус ввода можно перемещать с одной кнопки на другую, согласно особой последовательности, называемой *порядком обхода*. Каждая кнопка имеет свой уникальный *номер* в этом порядке. Для перехода на элемент управления с бóльшим номером в порядке обхода нужно нажать клавишу <Tab>, для перехода на элемент управления с меньшим номером — комбинацию клавиш <Shift>+<Tab>.

Для задания порядка обхода кнопок Flash служат два свойства объекта `Button`. Рассмотрим их.

Свойство `tabIndex` задает номер в порядке обхода для данной кнопки. Значением этого свойства может быть любое целое неотрицательное число.

Задать порядок обхода можно с помощью приведенного ниже сценария, привязанного к первому кадру анимации фильма:

```
btnForward.tabIndex = 0;

btnBackard.tabIndex = 1;

btnStop.tabIndex = 2;
```

Также для этого можно воспользоваться панелью **Accessibility**, подробно описанной в *главе 20*.

Если ни для одной кнопки в фильме Flash не задан номер в порядке обхода (т. е. значение свойства `tabIndex` равно `undefined`), Flash будет применять порядок обхода по умолчанию, и кнопки будут "обходиться" по нажатию клавиши <Tab> в том порядке, в котором они были созданы на рабочем листе. Также нужно иметь в виду, что значение `undefined` меньше любого числового, поэтому кнопки, для которых свойство `tabIndex` не было задано, будут в порядке обхода первыми.

Свойство `tabEnabled` позволяет вообще убрать кнопку из порядка обхода, после чего ей нельзя будет дать фокус ввода с помощью клавиатуры (хотя нажимать ее щелчком мыши будет можно). Это свойство имеет логический тип: значение `true` или `undefined` предоставляет возможность дать кнопке фокус ввода с помощью клавиатуры, а значение `false` такую возможность отменяет.

Обработка событий, возникающих в кнопках

Сценарий-обработчик события, возникающего в кнопке, как мы помним из *главы 18*, имеет такой формат записи:

```
on(<Событие>) {
  <Тело обработчика>
}
```

Например:

```
on(press) {
  _root.stop();
}
```

В данном примере обработчик выполняется в ответ на нажатие кнопки и останавливает анимацию основного фильма. Код этого обработчика должен быть привязан к самой кнопке.

Все события, поддерживаемые объектом `Button`, перечислены в табл. 19.11.

Таблица 19.11. *События, поддерживаемые объектом* `Button`

Событие	Описание
`press`	Возникает при щелчке на кнопке
`release`	Возникает при отпускании кнопки после повторного щелчка
`releaseOutside`	Возникает при отпускании кнопки, когда курсор мыши находится вне кнопки (подразумевается, что в момент, когда кнопка была нажата, курсор мыши находился над ней)
`rollOut`	Возникает при уводе курсора мыши с кнопки
`rollOver`	Возникает при помещении курсора мыши на кнопку
`dragOut`	Возникает при уводе курсора мыши с кнопки, если левая кнопка мыши нажата
`dragOver`	Возникает при помещении курсора мыши на кнопку, если левая кнопка мыши нажата
`keyPress("<Код>")`	Возникает при нажатии клавиши с заданным в качестве параметра кодом. При этом кнопка должна иметь фокус ввода

Создание и удаление кнопок из сценариев

Ранее в этой главе мы говорили, что имеется возможность создавать и удалять клипы из сценариев, пользуясь соответственно методами `attachMovie` и `removeMovieClip` объекта `movieClip` и аналогичными действиями. Такая

возможность реализована и для кнопок. Так, мы можем создать на рабочем листе новую кнопку в процессе воспроизведения фильма, привязав к кадру анимации такой сценарий:

```
_root.attachMovie("button", "button2", 1);
```

А затем мы можем удалить эту кнопку:

```
_root.button2.removeMovieClip();
```

Работа с текстовыми блоками

Текстовые блоки, описанные в *главе 7*, служат для отображения текста. Они бывают трех видов:

- статические текстовые блоки, служащие для отображения текста, который не изменяется в процессе воспроизведения фильма или работы Flash-приложения. Это могут быть надписи, заголовки, инструкции и пр.;
- поля ввода, служащие для ввода текста пользователем;
- динамические текстовые блоки, служащие для вывода текста, который изменяется в процессе воспроизведения фильма или работы Flash-приложения. Это может быть, например, текст, содержащий результаты работы приложения.

Flash поддерживает полноценную работу только с полями ввода и динамическими текстовыми блоками (как мы знаем из *главы 7*, они называются специальными текстовыми блоками). Именно средства для работы с ними мы сейчас рассмотрим. Средства для доступа к содержимому статических текстовых блоков весьма ограничены; все они описаны в интерактивной справке Flash.

Объект *TextField* и его свойства и методы

И поля ввода, и динамические текстовые блоки представляются одинаково — как экземпляры объекта `TextField`. Они также создаются самим Flash для каждого поля ввода или динамического текстового блока, помещенного на рабочий лист и имеющего имя.

Оператор `typeof` для специального текстового блока возвращает строку `"object"`, т. е. специальный текстовый блок имеет объектный тип данных.

Объект `TextField` имеет множество свойств и методов. Мы опишем только некоторые из них; полное описание этого объекта приведено в интерактивной справке Flash.

Прежде всего, объект `TextField` поддерживает свойства `_x`, `_y`, `_xmouse`, `_ymouse`, `_height`, `_width`, `_alpha` и `_rotation`, знакомые нам по объекту `movieClip`. Кроме того, он обладает свойствами `enabled`, `_visible`, `tabEnabled` и `tabIndex`, которые поддерживаются также объектом `Button`. Да-да, как и в случае кнопок, мы можем назначать для полей ввода номера в порядке обхода с помощью сценария вида:

```
txtName1.tabIndex = 0;
txtName2.tabIndex = 1;
txtEmail.tabIndex = 2;
btnOK.tabIndex = 3;
btnCancel.tabIndex = 4;
```

или панели **Accessibility** (*см. главу 20*).

Свойство `type` объекта `TextField` служит для задания вида данного экземпляра объекта `TextField`, т. е. определяет, будет это поле ввода или динамический текстовый блок. Это свойство имеет строковый тип: значение `"input"` превращает экземпляр объекта `TextField` в поле ввода, а значение `"dynamic"` — в динамический текстовый блок. Свойство доступно как для чтения, так и для записи, а это значит, что мы можем превращать поля ввода в динамические текстовые блоки и наоборот.

Свойство `text` позволяет получить или задать текст, находящийся в поле ввода или динамическом текстовом блоке. Оно имеет строковый тип.

Пример выражения со свойством `text`:

```
txtName.text = "Vasya Pupkin";
```

Нужно только помнить, что это свойство возвращает текст без тегов HTML, если таковые в нем присутствовали.

Свойство `variable` позволяет задать переменную, с которой будут связаны поле ввода или динамический текстовый блок. Свойство имеет строковый тип. Задав переменную, мы сможем получать и задавать содержимое поля ввода или текстового блока, просто обратившись к этой переменной:

```
var name:String = "";
txtName.variable = "name";
name = "Vasya Pupkin";
```

Свойство `html` позволяет разрешить или запретить вводить в поле ввода или динамический текстовый блок текст, отформатированный с помощью тегов HTML. Оно имеет логический тип: значение `true` разрешает вводить HTML-текст, а значение `false` — запрещает.

Свойство `htmlText` используется для получения или задания значения поля ввода или динамического текстового блока в том случае, если ввод HTML-текста разрешен.

```
txtName.html = true;
txtName.htmlText = "<B>Vasya</B> <I>Pupkin</I>";
```

Подробнее о поддержке тегов HTML в текстовых блоках рассказано в интерактивной справке Flash.

Свойство `maxChars` позволяет задать максимальное количество символов, которое может быть введено в поле ввода пользователем. (Количество символов, помещаемое в это поле ввода из сценариев, в любом случае не ограничено.) Это свойство принимает неотрицательные числовые значения; значение `null` отменяет ограничение на количество вводимых символов.

Доступное только для чтения свойство `length` возвращает количество символов, введенное в поле ввода.

Свойство `multiline` имеет логический тип и позволяет сделать поле ввода или динамический текстовый блок многострочным. Для этого достаточно присвоить этому свойству значение `true`. А присвоение значения `false` вновь делает поле ввода или динамический текстовый блок однострочным.

Свойство `password` позволяет превратить обычное поле ввода в поле ввода пароля. Для этого достаточно присвоить этому свойству логическое значение `true`. Присвоение значения `false` вновь делает поле ввода обычным.

Свойство `selectable` позволяет сделать поле ввода или динамический текстовый блок выделяемым; при этом пользователь сможет выделять текст, находящийся в данном поле или текстовом блоке, и копировать его в буфер обмена Windows. Это свойство имеет логический тип: значение `true` разрешает пользователю выделять текст, а значение `false` — запрещает.

Свойство `wordWrap` используется в том случае, если необходимо, чтобы текст в поле ввода или динамическом текстовом блоке переносился по строкам. Оно имеет логический тип: значение `true` устанавливает автоматический перенос строк текста, значение `false` его отменяет.

Помимо описанных, объект `TextField` поддерживает несколько свойств, задающих форматирование текста. Все эти свойства перечислены в табл. 19.12.

Таблица 19.12. *Свойства форматирования текста, поддерживаемые объектом* `TextField`

Свойство	Описание
`background`	При значении `true` поле ввода или динамический текстовый блок будет иметь непрозрачный фон. При значении `false` фона нет (или есть прозрачный фон)

Таблица 19.12 *(окончание)*

Свойство	Описание
backgroundColor	Цвет непрозрачного фона
border	При значении true поле ввода или динамический текстовый блок будет иметь рамку. При значении false рамки нет
borderColor	Цвет рамки
textColor	Цвет текста

Обработка событий, возникающих в текстовых блоках

Обработка событий, возникающих в текстовых блоках, выполняется не так, как в клипах и кнопках. Специфика заключается в том, что к текстовому блоку невозможно привязать сценарий. Поэтому нам придется пойти другим путем.

Давайте для примера создадим обработчик события, возникающего при изменении содержимого поля ввода. Для этого напишем функцию, содержащую код этого обработчика:

```
function txtNameChanged() {
  needToSave = true;
}
```

Пусть она просто присваивает логической переменной needToSave значение true. Впоследствии мы можем использовать это значение, чтобы проверить, изменил ли пользователь какие-то данные в поле ввода.

Мы только что создали *функцию-обработчик события*. Ее следует поместить в сценарий, привязанный к первому кадру анимации основного фильма, чтобы Flash сразу же ее прочитал.

Да, но как собственно привязать ее к текстовому блоку? Очень просто. Объект TextField поддерживает набор свойств, соответствующих поддерживаемым им событиям. Мы просто присвоим написанную нами функцию нужному свойству, вот так:

```
_root.txtName.onChanged = txtNameChanged;
```

Да, такие "фокусы" считаются стандартными в языке ActionScript! А свойство onChanged, соответствующее событию, происходящему при изменении содержимого поля ввода, получит тип данных "функция".

НА ЗАМЕТКУ

Оператор `typeof` при передаче ему в качестве параметра функции или переменной, которой присвоена функция, вернет строку `"function"`.

Всего объект `TextField` поддерживает четыре события. Связанные с ними свойства, которым нужно присвоить соответствующую функцию-обработчик, перечислены в табл. 19.13.

Таблица 19.13. *Свойства, связанные с событиями, поддерживаемыми объектом* `TextField`

Свойство	Описание события
`onChanged`	Возникает после того, как пользователь изменит содержимое поля ввода
`onKillFocus`	Возникает после потери полем ввода или динамическим текстовым блоком фокуса ввода. В функцию-обработчик этого события передается единственный параметр — ссылка на элемент управления, принявший фокус
`onScroller`	Возникает при прокрутке содержимого поля ввода или динамического текстового блока
`onSetFocus`	Возникает после получения полем ввода или динамическим текстовым блоком фокуса ввода. В функцию-обработчик этого события передается единственный параметр — ссылка на элемент управления, имевший фокус ввода до этого момента

НА ЗАМЕТКУ

Свойства, соответствующие событиям, поддерживают также объекты `movieClip` и `Button`. Их описание можно найти в интерактивной справке Flash.

Использование перехватчиков событий

Мы уже знаем два способа обработки событий. Первый способ — это использование для написания сценариев-обработчиков событий специальных действий (`onClipEvent` и `on`). Второй способ — это присвоение соответствующему свойству функции, которая будет вызываться в ответ на это событие и, таким образом, станет его обработчиком. Первый способ компактнее, но работает не для всех объектов, второй же не столь компактен, но зато универсален (именно с его помощью обрабатываются события в текстовых блоках).

Однако Flash предоставляет нам еще один способ обработки нужного события — с использованием так называемого *перехватчика событий* (в термино-

логии Flash — "listener", "слушатель"). Это экземпляр объекта `Object`, содержащий особые свойства, соответствующие обрабатываемым им событиям. Этим свойствам присваиваются функции-обработчики соответствующих событий, после чего перехватчик особым образом привязывается к экземпляру объекта, события которого нужно обработать.

В случае текстового блока с помощью перехватчика могут быть обработаны события `onChanged` и `onScroller`. (Это значит, что перехватчик должен содержать свойства `onChanged` и `onScroller`.) Вот пример сценария, создающего такой перехватчик (предположим, что функции обработки нажатия клавиш `fOnChanged` и `fOnScroller` уже описаны):

```
myListener = new Object();
myListener.onChanged = fOnChanged;
myListener.onScroller = fOnScroller;
```

Изначально объект `Object` не содержит свойств `onChanged` и `onScroller`. Они будут созданы в экземпляре этого объекта `myListener`, который мы только что создали, и будут "принадлежать" только этому экземпляру.

Для привязки перехватчика к текстовому блоку используется метод `addListener` объекта `TextField`. Этот метод принимает единственный параметр — ссылку на перехватчик.

```
txtName.addListener(myListener);
```

Это выражение привязывает созданный нами перехватчик к полю ввода `txtName`.

Один и тот же перехватчик можно привязать сразу к нескольким текстовым блокам:

```
txtName2.addListener(myListener);
txtFamily.addListener(myListener);
```

Это значит, что события, происходящие в разных текстовых блоках, можно обрабатывать с помощью одного комплекта функций-обработчиков. Иногда это бывает полезно, особенно если обработчики, привязанные к разным текстовым блокам, делают одно и то же.

Чтобы "отвязать" от текстового блока ненужный более перехватчик, воспользуемся методом `removeListener`. Этот метод в качестве единственного параметра также принимает ссылку на перехватчик.

```
txtName.removeListener(myListener);
```

Теперь останется только совсем удалить ненужный перехватчик:

```
delete myListener;
```

Создание и удаление текстовых блоков из сценариев

Flash также позволяет создавать и удалять текстовые блоки с помощью особых сценариев. Сейчас мы выясним, как это делается.

Для создания поля ввода или динамического текстового блока используется метод `createTextField` объекта `movieClip`. Формат его вызова таков:

```
<Клип>.createTextField(<Имя>, <Порядок перекрытия>, <X>, <Y>,
↳<Ширина>, <Высота>);
```

Первый параметр задает имя создаваемого поля ввода или динамического текстового блока в строковом виде, а второй — его порядок перекрытия. Третий и четвертый параметры задают соответственно горизонтальную и вертикальную координаты верхнего левого угла создаваемого текстового блока относительно клипа, в котором он создается. Последние два параметра задают размеры — ширину и высоту — текстового блока. И координаты, и размеры задаются в пикселах.

Нужно помнить, что с помощью этого метода создается динамический текстовый блок, т. е. свойство `type` соответствующего экземпляра объекта `TextField` сразу же получит значение `"dynamic"`. Чтобы преобразовать этот экземпляр в поле ввода, нужно присвоить свойству `type` строку `"input"`. Например:

```
_root.createTextField("txtName", 0, 100, 100, 200, 50);
_root.txtName.type = "input";
_root.txtName.variable = "Name";
```

Этот сценарий создает поле ввода `txtName` и привязывает его к переменной `Name`.

Для удаления созданного с помощью метода `createTextField` текстового блока используется метод `removeTextField` объекта `TextField`:

```
_root.txtName.removeTextField();
```

> ***ВНИМАНИЕ!***
>
> С помощью действия или метода `removeTextField` можно удалить только те текстовые блоки, которые созданы из сценариев. Текстовые блоки, созданные вручную, не могут быть удалены таким образом.

Использование объекта *Selection*

Flash также предоставляет возможность управления текстовым курсором и выделением текста в текстовых блоках. Для этого предназначен объект

Selection, единственный экземпляр которого по имени Selection создается самим Flash.

Метод getFocus этого объекта возвращает в строковом виде имя переменной, привязанной к текстовому блоку, имеющему в данный момент фокус ввода. Если к текстовому блоку не привязана переменная или если фокус ввода имеет кнопка или клип, то возвращается путь (также в строковом виде). Если ни один текстовый блок, ни одна кнопка и ни один клип не имеют фокуса, возвращается значение null.

Для того чтобы дать какому-либо текстовому блоку (а также кнопке или клипу) фокус ввода, нужно вызвать метод setFocus. В качестве единственного параметра этому методу передается путь в строковом виде:

```
Selection.setFocus("_root.txtName");
Selection.setFocus("_root.btnOK");
```

Чтобы снять фокус ввода со всех элементов управления, достаточно передать методу setFocus в качестве параметра значение null.

С помощью объекта Selection мы можем получить позицию текстового курсора в поле ввода. Для этого достаточно вызвать не принимающий параметров метод getCaretIndex, который вернет нам номер символа, на котором стоит курсор. (Не стоит забывать, что нумерация символов начинается с нуля.) Если же ни одно поле ввода не имеет фокуса, возвращается значение −1.

Методы getBeginIndex и getEndIndex возвращают номера соответственно начального и конечного символа выделенного в текстовом блоке фрагмента текста. Если ничего не выделено, также возвращается значение −1.

Можно выделить нужный фрагмент текста, воспользовавшись методом setSelection. Первый параметр этого метода должен иметь значение, равное номеру первого символа выделяемого фрагмента, а второй параметр — значение, равное номеру последнего символа. Вот так:

```
Selection.setSelection(10, 30);
```

Чтобы просто поставить текстовый курсор в требуемую позицию, нужно передать методу setSelection номер нужного символа и в первом, и во втором параметре:

```
Selection.setSelection(10, 10);
```

Объект Selection поддерживает одно-единственное событие onSetFocus, возникающее при перемещении фокуса ввода. Обработка этого события возможна только с помощью перехватчика. Например:

```
myListener = new Object();
myListener.onSetFocus = fOnSetFocus;
Selection.addListener(myListener);
```

Это пример создания и привязки перехватчика к объекту `Selection`. Функция `fOnSetFocus` выполняет собственно обработку события `onSetFocus`; определение этой функции было опущено.

Обработка нажатий клавиш клавиатуры

Для обработки нажатия клавиш клавиатуры служит объект `Key`. Его единственный экземпляр, также носящий имя `Key`, создается самим Flash.

Свойства и методы объекта *Key*

Два важнейших метода, которые обязательно понадобятся нам для обработки нажатий клавиш, — это методы `getAscii` и `getCode`. Первый метод возвращает код символа ASCII, который присвоен нажатой клавише. Второй метод возвращает так называемый *виртуальный код* нажатой клавиши — низкоуровневый код, позволяющий узнать, какая клавиша на клавиатуре была нажата. Оба эти метода не принимают параметров.

Вот пример выражений вызова этих методов:

```
code1 = Key.getAscii();
code2 = Key.getCode();
```

Код символа ASCII зависит от текущей кодировки. Коды символов кодировки Windows 1251 приведены в интерактивной справке Flash; там же приведены и виртуальные коды клавиш.

Еще один полезный метод объекта `Key` — метод `isDown`. Он принимает единственный параметр — виртуальный код клавиши — и возвращает значение `true`, если эта клавиша нажата. Например:

```
if (Key.isDown(Key.ENTER)) { . . . }
```

Приведенное выше выражение проверяет, была ли нажата клавиша <Enter>.

Метод `isToggled` возвращает значение `true`, если в момент его вызова была включена одна из клавиш-переключателей: <NumLock> или <CapsLock>. Он принимает единственный параметр — числовое значение `20` (виртуальный код клавиши <CapsLock>) или `144` (виртуальный код клавиши <NumLock>).

Пример вызова метода `isToggled`:

```
if (Key.isToggled(20)) { s = s.toLowerCase; }
```

Кроме вышеописанных методов, объект `Key` поддерживает ряд свойств, доступных только для чтения и возвращающих значения виртуальных кодов

различных служебных клавиш клавиатуры. Все эти свойства приведены в табл. 19.14.

Таблица 19.14. *Свойства объекта* `Key`*, возвращающие коды служебных клавиш*

Свойство	Клавиша	Виртуальный код клавиши
BACKSPACE	<BackSpace>	8
CAPSLOCK	<CapsLock>	20
CONTROL	<Ctrl>	17
DELETEKEY	<Del>	46
DOWN	<Стрелка вниз>	40
END	<End>	35
ENTER	<Enter>	13
ESCAPE	<Esc>	27
HOME	<Home>	36
INSERT	<Ins>	45
LEFT	<Стрелка влево>	37
PGDN	<PgDn>	34
PGUP	<PgUp>	33
RIGHT	<Стрелка вправо>	39
SHIFT	<Shift>	16
SPACE	<Пробел>	32
TAB	<Tab>	9
UP	<Стрелка вверх>	38

Обработка нажатий клавиш в клипах

А теперь поговорим о том, как же собственно выполняется обработка нажатий клавиш клавиатуры. Выполнить ее можно двумя способами, которые мы и рассмотрим в этом разделе.

Обработка нажатий клавиш в клипах

Первый — более простой — способ обработки нажатий клавиш — заключается в привязке к клипу, который будет реагировать на нажатия, обработчи-

ка события keyDown, возникающего при нажатии клавиши. Мы рассмотрим создание этого обработчика на примере.

Создадим новый документ Flash. В нем создадим образец-клип, содержащий простую геометрическую фигуру, например, круг. Поместим экземпляр этого образца на рабочий лист и назовем его circle. Этот экземпляр мы заставим перемещаться по листу в ответ на нажатия клавиш-стрелок.

Теперь выделим наш экземпляр и привяжем к нему довольно сложный сценарий, содержащий обработчики сразу двух событий. Для удобства рассмотрим их отдельно друг от друга.

```
onClipEvent(load) {
  d = 1;
}
```

Обработчик события load помещает значение в переменную d сразу после загрузки клипа. Эта переменная задает величину перемещения нашего экземпляра при нажатии любой клавиши-стрелки. Можно задать и бóльшее число, чтобы заставить клип двигаться быстрее.

Второй фрагмент сценария выглядит более сложным:

```
onClipEvent(keyDown) {
  switch (Key.getCode()) {
    case Key.UP :
      _root.circle._y -= d;
      break;
    case Key.DOWN :
      _root.circle._y += d;
      break;
    case Key.LEFT :
      _root.circle._x -= d;
      break;
    case Key.RIGHT :
      _root.circle._x += d;
      break;
  }
}
```

Но, несмотря на кажущуюся сложность, этот обработчик события keyDown довольно-таки прост. Он сравнивает значение виртуального кода нажатой клавиши с кодами клавиш-стрелок и, если одна из них нажата, изменяет соответствующую координату клипа circle.

Теперь запустим созданный фильм на воспроизведение в отдельном окне. Он работает!

Конечно, наш пример очень прост. Тем не менее, руководствуясь им, можно создавать и несравнимо более сложные вещи. Например — как насчет игры "15", реализованной на Flash? Или всем известного "Сапера"?

Обработка нажатий клавиш с помощью перехватчика

Второй способ обработать нажатия клавиш — использовать перехватчик. Этот перехватчик должен иметь свойства `onKeyUp` и `onKeyDown`, которым должны быть присвоены функции, обрабатывающие соответствующие события. Событие `onKeyDown` возникает при нажатии клавиши, а событие `onKeyUp` — при ее отпускании.

Давайте перепишем созданный ранее сценарий так, чтобы он использовал обработчик. В данном случае он будет привязан не к клипу, а к первому кадру анимации фильма.

Рассмотрим этот сценарий по частям.

Сначала:

```
var d = 1;
```

Это уже знакомая нам переменная, задающая величину смещения клипа за один шаг.

А дальше:

```
function fOnKeyDown() {
  switch (Key.getCode()) {
    case Key.UP :
      _root.circle._y -= d;
      break;
    case Key.DOWN :
      _root.circle._y += d;
      break;
    case Key.LEFT :
      _root.circle._x -= d;
      break;
    case Key.RIGHT :
      _root.circle._x += d;
      break;
  }
}
```

А это объявление функции, обрабатывающей событие `onKeyDown`. Ее код практически совпадает с кодом обработчика события `keyDown` для клипа, написанного нами ранее.

И, наконец:

```
var myListener = new Object();
myListener.onKeyDown = fOnKeyDown;
Key.addListener(myListener);
```

Этот фрагмент кода создает и привязывает к объекту `Key` перехватчик. Собственно, это тоже нам знакомо.

Взаимодействие с мышью

Для взаимодействия с мышью служит объект `Mouse`. Единственный экземпляр этого объекта под именем `Mouse` создается самим Flash.

Прежде всего, этот объект предоставляет методы `hide` и `show`. Первый метод скрывает курсор мыши, а второй — вновь выводит его на экран. Ни один из них не принимает параметров. Эти методы используются, как правило, при создании фигурных курсоров мыши.

Объект `Mouse` поддерживает также три события:

- `событие onMouseDown` возникает при нажатии левой кнопки мыши;
- `событие onMouseUp` возникает при отпускании левой кнопки мыши;
- `событие onMouseMove` возникает при любом перемещении мыши;
- `событие onMouseWheel` возникает при прокрутке колесика мыши, если такое имеется.

Для обработки этих событий используются перехватчики. Все функции, обрабатывающие их, не принимают параметров, за исключением функции-обработчика события `onMouseWheel`. Она имеет такой формат:

```
function <Имя>([<Смещение>, <Клип, над которым находится курсор мыши>])
↳{ . . . }
```

Первый из необязательных параметров этой функции определяет смещение, на которое пользователь прокрутил колесико мыши. Оно передается в числовом виде в так называемых "тиках". Второй параметр передает ссылку на клип, над которым находился курсор мыши во время прокрутки ее колесика.

Для отслеживания координат мыши можно использовать свойства `_xmouse` и `_ymouse` объекта `movieClip`. Этот "нужный экземпляр" может быть как внешней анимацией (`_root`), так и каким-либо вложенным клипом. Эти же свойства поддерживаются и объектом `Button`.

Управление звуковым сопровождением фильма

Для управления звуковым сопровождением фильма служат экземпляры объекта `Sound`. (О работе со звуком во Flash *см. главу 17.*) Эти экземпляры создаются с помощью выражений, имеющих следующий формат:

```
<Переменная> = new Sound([<Клип, звуковым сопровождением которого нужно
↳управлять>]);
```

С единственным параметром этому конструктору передается ссылка на клип, звуковым сопровождением которого мы собираемся управлять. Если этот параметр не указан, создается экземпляр объекта, управляющий звуковым сопровождением всего фильма.

Вот два примера создания разных экземпляров объекта `Sound`:

```
carSound = new Sound(car);
globalSound = new Sound();
```

Экземпляр `carSound` позволяет управлять звуковым сопровождением клипа `car`, а экземпляр `globalSound` — всего фильма.

Объект `Sound` имеет множество свойств и методов. В этом разделе мы рассмотрим некоторые из них.

Метод `start` запускает воспроизведение звука. Он имеет следующий формат вызова:

```
<Экземпляр объекта Sound>.start([<Отметка>, <Количество повторов>]);
```

Этот метод может принимать два необязательных параметра. Первый параметр указывает, с какой отметки начнет воспроизводиться звук; отметка задается как количество секунд, которые Flash отсчитает с момента начала звука. Со вторым параметром передается количество повторов звука. Если ни один параметр не задан, звуковое сопровождение будет воспроизведено один раз с самого начала.

Например, выражение

```
carSound.start(10, 2);
```

запускает воспроизведение звука, привязанного к клипу `car`, с десятисекундной отметки и воспроизводит его два раза.

Метод `stop` немедленно останавливает воспроизведение звука. Формат его вызова таков:

```
<Экземпляр объекта Sound>.stop([<Идентификатор образца-звука>]);
```

В качестве единственного параметра этого метода может быть передано идентификатор сценарного образца-звука, воспроизведение экземпляра ко-

торого нужно остановить, в строковом виде. Если метод был вызван без параметра, останавливается воспроизведение всех звуков.

```
carSound.stop("engineSound");
```

Это выражение отключает воспроизведение звука `engineSound`, привязанного к клипу `car`.

Не принимающий параметров метод `getVolume` возвращает значение громкости звука. А метод `setVolume` позволяет задать новое значение громкости звука, переданное в качестве его единственного параметра. Доступны значения от 0 (звук не слышен) до 100 (полная громкость; значение по умолчанию).

```
carSound.setVolume(mySound.getVolume() + 10);
```

Это выражение увеличивает громкость работы автомобиля на десять единиц.

Пара методов `getPan` и `setPan` ведут себя так же и позволяют задать панорамирование звука в пространстве. Здесь доступны значения от −100 (звук в левом канале) до 100 (звук в правом канале); значение по умолчанию — 0 (звук "посередине").

```
carSound.setPan(50);
```

Объект `Sound` также поддерживает два свойства, доступные только для чтения. Свойство `position` возвращает текущую воспроизводимую позицию звука в миллисекундах. А свойство `duration` возвращает общую продолжительность звука, также в миллисекундах.

Использование таймеров

Во многих случаях нужно выполнять какое-либо действие через равные промежутки времени. Это может понадобиться, например, для создания анимации с помощью сценариев ActionScript или для обновления информации, взятой с удаленного сервера. Для таких задач во Flash предусмотрены *таймеры*. Поговорим о них.

Таймер создается с помощью действия `setInterval`. Оно имеет два формата вызова. Вот первый из них:

```
<Переменная> = setInterval(<Функция>, <Значение интервала>
↳[, <Список параметров функции, разделенных запятыми>]);
```

Здесь в качестве параметров передается функция, которая будет вызываться по истечении интервала времени, и само значение этого интервала в миллисекундах. Также в выражении вызова могут присутствовать необязательные параметры, задающие значения параметров вызываемой функции; их должно быть столько, сколько параметров принимает эта функция.

А вот второй формат вызова действия `setInterval`:

```
<Переменная> = setInterval(<Экземпляр объекта>, <Метод объекта>,
↳<Значение интервала>
↳[, <Список параметров метода, разделенных запятыми>]);
```

Он отличается тем, что вместо функции здесь передается экземпляр какого-либо объекта и его метод.

В любом случае действие `setInterval` возвращает особый *идентификатор таймера*, который в дальнейшем будет использован для его уничтожения.

Приведем примеры выражений, создающих таймеры:

```
timerID1 = setInterval(tick, 1000);
timerID2 = setInterval(car, moveBy, 100, d);
```

Первое выражение создает таймер, вызывающий функцию `tick` каждую секунду (1000 миллисекунд). Второе же создает таймер, вызывающий каждые 100 миллисекунд метод `moveBy` экземпляра объекта `car` и передающий ему параметр `d`. Разумеется, и функция `tick`, и метод `moveBy` должны быть объявлены прежде, чем будут использованы.

При использовании таймеров нужно иметь в виду следующее. Дело в том, что Flash отмеряет интервалы не очень точно. Он пытается привязать каждый интервал к частоте кадров фильма, если, конечно, это возможно. Если интервал меньше частоты кадров, то он обрабатывается с максимальной точностью, и код вызывается вовремя. Если же интервал больше частоты кадров, то код, связанный с таймером, исполняется только тогда, когда Flash воспроизводит очередной кадр фильма. Это сделано для того, чтобы лишний раз не перерисовывать экран, если код что-то изменит в изображении.

Если же нужно обновить экран прямо сейчас, следует вызвать не принимающее параметров действие `updateAfterEvent`. Оно часто используется при выполнении сценариев, изменяющих содержимое рабочего листа.

После того как нужда в таймере отпадет, его следует уничтожить, чтобы освободить системные ресурсы. Для этого используется действие `clearInterval`, которое имеет такой формат вызова:

```
clearInterval(<Идентификатор таймера>);
```

Например:

```
clearInterval(timerID2);
```

А теперь рассмотрим пример, показывающий, как можно использовать таймер для создания так называемого "ленивого" курсора мыши. Обычный курсор мыши нигде надолго не задерживается — он спешит за ней сразу после

малейшего ее перемещения. "Ленивый" же курсор отличается от обычного тем, что никогда не торопится за мышью. Давайте же сделаем такой курсор.

Прежде всего, создадим новый образец-клип и нарисуем в нем желаемый курсор. Постараемся нарисовать его так, чтобы точка отсчета пришлась как раз на "острие" этого курсора. Поместим экземпляр этого образца-клипа на рабочий лист и назовем его `cursor`.

Теперь напишем код сценария, который будет привязан к первому кадру фильма. Давайте рассмотрим его по частям.

```
function tick() {
  if (_root.cursor._x != _root._xmouse) {
    if (_root.cursor._x < _root._xmouse) {
      if (_root._xmouse - _root.cursor._x < d) {
        _root.cursor._x = _root._xmouse;
      } else {
        _root.cursor._x += d;
      }
    } else {
      if (_root.cursor._x - _root._xmouse < d) {
        _root.cursor._x = _root._xmouse;
      } else {
        _root.cursor._x -= d;
      }
    }
  }
  if (_root.cursor._y != _root._ymouse) {
    if (_root.cursor._y < _root._ymouse) {
      if (_root._ymouse - _root.cursor._y < d) {
        _root.cursor._y = _root._ymouse;
      } else {
        _root.cursor._y += d;
      }
    } else {
      if (_root.cursor._y - _root._ymouse < d) {
        _root.cursor._y = _root._ymouse;
      } else {
```

```
        _root.cursor._y -= d;
      }
    }
  }
  updateAfterEvent();
}
```

Мы объявили функцию, реализующую перемещение "ленивого" курсора. Она сравнивает координаты, горизонтальную и вертикальную, курсора и мыши и выполняет приращение координат курсора. Нужно отметить, что координаты мыши при этом отсчитываются относительно анимации основного фильма.

Продолжаем писать наш сценарий:

```
d = 25;
```

Это величина приращения, на которое будут изменяться координаты курсора.

И далее:

```
ctID = setInterval(tick, 50);
```

Здесь мы создаем таймер, вызывающий функцию `tick` каждые 50 миллисекунд.

И, наконец:

```
_root.onUnload = function() {
  clearInterval(ctID);
}
```

А здесь мы привязываем к событию `unload` клипа `cursor` обработчик, удаляющий созданный ранее таймер. Удалять за собой затребованные для своих нужд ресурсы, те же таймеры, — хороший стиль программирования.

Если уж зашла речь о таймерах, то нам может пригодиться функция `getTimer`. Она возвращает количество миллисекунд, прошедших с момента, когда фильм начал воспроизводиться.

Загрузка внешних файлов

Ранее в этой главе уже упоминалось о том, что язык ActionScript позволяет загрузить и поместить в фильм Flash внешний файл. Этим внешним файлом может быть другой фильм Shockwave/Flash, изображение в формате JPEG, видео Macromedia Flash Video или звук. Также имеется возможность открыть Web-обозреватель и загрузить в него какую-либо Web-страницу. Настала пора выяснить, как это делается.

Загрузка и выгрузка фильмов Shockwave/Flash и изображений в формате JPEG

Проще всего загрузить в основной фильм, воспроизводящийся в окне проигрывателя Flash, другой фильм Shockwave/Flash или изображение в формате JPEG из внешнего файла (*загружаемого* фильма). Такое часто делается, особенно при создании Web-сайтов, целиком основанных на Flash.

Вложение загружаемого фильма или изображения в основной фильм

Проще всего создать на рабочем листе "пустой" клип (например, с помощью метода `createEmptyMovieClip`, описанного ранее в этой главе) и заменить его загружаемым фильмом. При этом загружаемый фильм вкладывается в основной, становясь его составной частью.

Для загрузки фильма Shockwave/Flash или изображения JPEG из внешнего файла используется действие `loadMovie`, которое вызывается так:

```
loadMovie("<Интернет-адрес внешнего файла>",
↳<Путь заменяемого клипа>);
```

Примеры использования действия `loadMovie`:

```
loadMovie("/images/truckWheel.swf", _root.car.wheel2);
loadMovie(sURL, _root.car);
```

Также можно воспользоваться методом `loadMovie` объекта `movieClip`. Формат его вызова почти такой же, как у одноименного действия:

```
<Заменяемый клип>.loadMovie("<Интернет-адрес внешнего файла>");
```

Пример использования этого метода:

```
_root.car.wheel2.loadMovie("/images/truckWheel.swf");
```

Здесь мы загружаем фильм truckWheel.swf и заменяем им второе колесо нашего автомобиля.

А теперь предположим, что фильм truckWheel.swf из вышеприведенного примера содержит в себе сценарий

```
_root.stop();
```

останавливающий воспроизведение его анимации (вращение колеса). Если открыть этот фильм в проигрывателе Flash, то сценарий выполнится правильно. Но если загрузить фильм truckWheel.swf в другой (основной) фильм, как мы это только что сделали, этот сценарий остановит анимацию не колеса автомобиля, а основного фильма. Ведь модификатор `_root` всегда возвращает ссылку на внешнюю анимацию, и в нашем случае он

вернет ссылку на основной фильм, поскольку фильм truckWheel.swf стал его составной частью.

Что делать? Как нам заставить этот сценарий вложенного фильма выполняться корректно в любом случае? Очень просто: нужно привязать к первому кадру фильма truckWheel.swf вот такой совсем небольшой сценарий:

```
this._lockroot = true;
```

Здесь мы присвоили свойству `_lockroot` объекта `movieClip`, обозначающего сам фильм, значение `true`. После этого модификатор `_root` в любом сценарии, содержащемся в загружаемом фильме, всегда будет возвращать ссылку на внешнюю анимацию именно этого фильма — truckWheel.swf, даже если мы вложим его в другой фильм.

После того как нужда в загруженном из внешнего файла фильме или изображении пропадет, самое время его выгрузить. Зачем это нужно? Хотя бы затем, чтобы освободить ресурсы компьютера пользователя для других, более важных задач. Именно для этого язык ActionScript предоставляет действие `unloadMovie`, выгружающее из памяти и удаляющее с экрана ранее загруженный фильм. Формат вызова этого действия такой:

```
unloadMovie(<Путь выгружаемого клипа или изображения>);
```

Пример:

```
unloadMovie(_root.car.engine);
```

В этом выражении мы избавляемся от двигателя нашего автомобиля — зачем он нужен, раз автомобиль все равно не двигается с места.

А вот это выражение выгрузит сам основной фильм:

```
unloadMovie(_level0._root);
```

после чего проигрыватель Flash, оставшись не у дел, автоматически завершит свою работу.

Перекрывающиеся фильмы

Только что мы выяснили, как вложить загружаемый фильм в основной. Но Flash предоставляет также возможность наложить содержимое внешнего файла на содержимое основного фильма так, чтобы они перекрывали друг друга, полностью или частично. В таком случае эти фильмы не вкладываются друг в друга (подобно вложенным клипам), а остаются полностью независимыми. Такие фильмы называются *перекрывающимися*, а само это явление — *перекрытием фильмов*.

Зачем может понадобиться перекрытие фильмов? А, например, для создания Web-сайтов, основанных на Flash. В этом случае сначала загружается и выводится на экран "объединяющий" фильм, содержащий заголовок сайта, полосу навигации и какой-то изначальный текст. При щелчке на какой-либо

кнопке полосы навигации особый сценарий загружает фильм с соответствующим содержимым и выводит его на экран как перекрывающийся клип, перекрывая изначальное содержимое. Схема достаточно простая и наглядная; удивительно только, почему таких сайтов все еще довольно мало.

В *главе 4*, описывающей трансформации, давалось понятие порядка перекрытия фрагментов графики друг относительно друга (его еще называют z-координатой). В случае с перекрывающимися фильмами используется примерно такое же понятие, только называемое по-другому, а именно — *уровень фильма*. Отсчет уровней ведется, начиная с самого нижнего клипа (фильма) до самого верхнего. Фильмы с бóльшим номером уровня находятся "выше" фильмов с меньшим номером и перекрывают их.

На этом сходство с порядком перекрытия исчерпывается, и начинаются отличия. Их три.

Во-первых, самым "нижним" всегда является фильм, изначально воспроизводящийся в проигрывателе Flash, — он имеет уровень 0. Все остальные фильмы, которые будут загружены как перекрывающиеся, находятся "выше" него, т. е. имеют более высокий уровень — 1, 2 и т. д.

Во-вторых, для обозначения уровня фильма используются не числа, а особые модификаторы, в состав которых входят номера уровней. Они возвращают ссылку на внешнюю анимацию соответствующего фильма и имеют вид `_level<`*`Номер уровня`*`>`. Самый "нижний" фильм имеет модификатор `_level0`. Более "высокие" фильмы имеют модификаторы `_level1`, `_level2` и т. д.

Мы можем использовать эти модификаторы для доступа к перекрывающимся фильмам. Например:

```
_level0.gotoAndPlay(1);
_level1.car.wheel.stop();
```

> ***Внимание!***
>
> Перед тем как обратиться к перекрывающемуся фильму, находящемуся на каком-то уровне, нужно сначала его туда загрузить. Как это сделать, будет рассказано чуть позже.

В-третьих, мы можем загрузить какой-либо фильм на уровень, на котором уже имеется другой фильм. В этом случае загружаемый фильм заменит тот, что раньше был на этом уровне. Можно загрузить какой-либо фильм и на уровень 0 — в этом случае он заменит фильм, изначально загруженный в проигрыватель Flash.

Для загрузки перекрывающегося фильма используется действие `loadMovieNum`, аналогичное действию `loadMovie`. Формат его вызова таков:

```
loadMovieNum("<Интернет-адрес загружаемого файла>", <Уровень>);
```

Приведем два примера выражений, использующих действие `loadMovieNum`:

```
loadMovieNum("/videos/driving.swf", 0);
loadMovieNum("/images/ads/banner45.jpg", 2);
```

Первое выражение заменяет основной фильм фильмом `driving.swf`. Второе же выражение загружает рекламный баннер и помещает его над основным фильмом. (Непонятно, правда, зачем это нужно делать. Если только для испытания нервов зрителей.)

Для выгрузки перекрывающегося фильма используется действие `unloadMovieClip`. Вот формат его вызова:

```
unloadMovieNum(<Уровень выгружаемого клипа или изображения>);
```

Пример:

```
unloadMovieNum(_level2);
```

Это выражение убирает назойливо маячивший перед глазами рекламный баннер.

Загрузка звуков

Для загрузки звука из внешнего файла в формате MP3 (другие форматы не поддерживаются) следует воспользоваться методом `loadSound` объекта `Sound`. Вот формат его вызова:

```
<Звук>.loadSound(<Интернет-адрес или путь внешнего звукового файла>,
↳<Потоковый звук>);
```

Первый параметр задает интернет-адрес звукового файла или его путь на локальном диске компьютера в строковом виде. Второй параметр — логический — задает, будет ли загружаемый звук потоковым (значение `true`) или звуком-сигналом (значение `false`). (О потоковых звуках и сигналах *см. главу 17.*)

Вот пример сценария, загружающего звуковой файл с использованием метода `loadSound`:

```
newSound = new Sound();
newSound.loadSound("sounds/background_music.mp3", true);
newSound.play();
```

Объект `Sound` поддерживает событие `onLoad`, возникающее по окончании загрузки звука. Это событие можно обработать с помощью функции-обработчика, принимающей единственный параметр. Если звук был успешно загружен, этот параметр примет логическое значение `true`. Соответственно, если проигрыватель Flash почему-то не смог загрузить этот звук, обработчику передается значение `false`.

Вот пример функции-обработчика события `onLoad`:

```
function onNewSoundLoad(success) {
  if (success) {
    newSound.play();
  }
}
newSound.onLoad = onNewSoundLoad;
```

Файлы формата MP3, кроме собственно звука, могут хранить дополнительные данные — так называемые *теги MP3*. В этих тегах записываются, в частности, имя исполнителя, название альбома и песни, год выпуска, жанр и пр. Для доступа к этим тегам служит свойство `ID3` объекта `Sound`. Оно возвращает ссылку на экземпляр объекта `Object`, свойства которого и содержат информацию о различных тегах MP3. Эти свойства возвращают, в частности:

- свойство `TCOM` — имя исполнителя;
- свойство `TALB` — название альбома;
- свойство `TIT2` — название произведения;
- свойство `TCOM` — жанр произведения;
- свойство `TYER` — год выпуска альбома;
- свойство `TIME` — продолжительность произведения;
- свойство `TRCK` — номер произведения в альбоме.

Объект `Sound` поддерживает также методы `getBytesLoaded` и `getBytesTotal`, возвращающие соответственно размер загруженной части звукового файла в байтах и общий размер загружаемого звука в байтах.

Загрузка видео

Теперь рассмотрим процесс загрузки и воспроизведения внешних видеофайлов формата Macromedia Flash Video (другие форматы не поддерживаются). Подробно об этом формате было рассказано в *главе 16*.

Все находящиеся на рабочем листе экземпляры образцов-импортированных клипов, для которых было задано имя, представляют собой экземпляры объекта `Video`. Их создает сам Flash; нам же остается только использовать их свойства и методы.

Объект `Video` поддерживает метод `attachVideo`, выполняющий загрузку внешнего видеофайла и воспроизведение его в экземпляре образца-импортированного видео. Для загрузки и воспроизведения лучше всего ис-

пользовать "пустой" видеоклип, процесс создания которого мы сейчас рассмотрим.

Итак, чтобы загрузить и воспроизвести внешний видеоклип, нужно сначала создать на рабочем листе "пустой" экземпляр объекта `Video`. Для этого откроем панель **Library** и выберем в ее дополнительном меню пункт **New Video**. В списке панели **Library** появится новый пункт с именем вида `Embedded Video <Порядковый номер>`. Дважды щелкнем на этом имени, введем в появившееся поле ввода новое, более вразумительное имя и нажмем клавишу <Enter>. Поместим на рабочий лист экземпляр этого образца и дадим ему какое-нибудь имя, например, `vidExternal`. Все — "пустой" видеоклип создан.

Теперь привяжем к первому кадру анимации такой сценарий:

```
var nc = new NetConnection();
nc.connect(null);
var ns = new NetStream(nc);
vidExternal.attachVideo(ns);
ns.play("videos/introduction.flv");
```

Здесь мы используем для загрузки внешнего видеофайла экземпляры объектов `NetConnection` и `NetScream`. Подробное описание этих объектов можно найти в интерактивной справке Flash.

Загрузка Web-страниц

Напоследок мы рассмотрим используемый во Flash способ загрузки в окно Web-обозревателя какой-либо Web-страницы. Для этого предназначено действие `getURL`, формат вызова которого таков:

```
getURL("<Интернет-адрес Web-страницы>"[, "<Цель>"]);
```

С первым параметром этого действия передается сам интернет-адрес загружаемой Web-страницы. Через второй, необязательный, параметр задается цель гиперссылки. Параметр `цель` имеет строковый тип и может принимать следующие значения:

- `"_blank"` — Web-страница загрузится в отдельное окно Web-обозревателя;
- `"_parent"` — Web-страница загрузится в родительский набор фреймов;
- `"_self"` — Web-страница загрузится в текущий фрейм (это значение параметра по умолчанию);
- `"_top"` — Web-страница загрузится в текущее окно Web-обозревателя, заменив собой весь набор фреймов, если он есть.

Кроме того, можно передать со вторым параметром имя фрейма, в который должна быть загружена эта страница.

Вот пример сценария, привязанного к кнопке и загружающего Web-страницу сайта Macromedia в отдельное окно Web-обозревателя:

```
on(click) {
  getURL("http://www.macromedia.com", "_blank");
}
```

Ограничения на доступ к внешним файлам

Интернет — небезопасное место. Вирусы, "троянские программы", мошеннические сайты — всего этого добра там более чем достаточно. И неудивительно, что создатели программ, работающих с Интернетом, встраивают в них различные средства, призванные хоть как-то обеспечить безопасность пользователя и его данных. Пусть даже ценой каких-то ограничений.

Не стал исключением и наш любимый Flash. Всеобщая озабоченность безопасностью (вполне, кстати, оправданная) отразилась и на нем. Разработчики фирмы Macromedia предусмотрели для своего детища множество ограничений на работу с внешними файлами. Рассмотрим эти ограничения.

Нужно сказать сразу, что внешние файлы во Flash можно загружать с любого интернет-адреса. Они будут нормально отображаться в окне проигрывателя Flash. Но если мы попытаемся получить доступ к экземплярам объектов, функциям или переменным, находящимся во внешнем файле, Flash сразу даст нам по рукам. Смотреть, дескать, смотрите, но руками не троньте!

Какие же внешние файлы можно "трогать руками"? Какие ограничения Flash накладывает на доступ к внешним файлам? А вот какие...

Прежде всего, файлы, которые желают "общаться" друг с другом на уровне сценариев, должны быть загружены с одного и того же Web-сервера. А именно, если они были загружены с Web-сервера, имеющего IP-адрес 192.168.1.10, то они смогут обращаться к объектам, функциям и переменным друг друга. То же самое справедливо для файлов, загруженных с сервера **http://www.somesite.ru**. Однако если файлы были загружены с серверов **http://www.somesite.ru** и **http://somesite.ru**, то они не смогут "общаться" друг с другом на уровне сценариев.

Еще одно ограничение связано с протоколами *HTTP* [1] (HyperText Transport Protocol, транспортный протокол передачи гипертекста) и *HTTPS* [2] (HyperText Transport Protocol Secured, защищенный транспортный протокол

[1] По этому протоколу от Web-сервера Web-обозревателю передаются Web-страницы.
[2] Защищенная версия протокола HTTP.

передачи гипертекста). (*Протокол* — это набор правил, согласно которым программы обмениваются данными по сети.) Если основной фильм был загружен по протоколу HTTP, а внешний — по протоколу HTTPS (или наоборот), то они не смогут получить доступ друг к другу через сценарии.

В предыдущей версии Flash (MX) описанные ограничения были непреодолимыми. Во Flash MX 2004 появилась возможность их преодолеть. Сейчас мы выясним, как это делается.

Пусть мы загрузили основной фильм с сервера **http://www.somesite.ru**. И пусть нам понадобилось загрузить с сервера **http://somesite.ru** внешний файл car.swf, поместить его на рабочий лист, а впоследствии иметь к нему доступ через сценарии. Для этого нам нужно будет использовать метод `allowDomain` объекта `security`, находящегося в объекте `System`. Формат его вызова таков:

```
System.security.allowDomain("<Разрешенный интернет-адрес>");
```

В качестве единственного параметра этого метода передается интернет-адрес Web-сервера, с которого разрешена загрузка внешних файлов. После этого все файлы, загруженные с этого сервера, доступны из текущего фильма через сценарии.

В нашем случае нам нужно поместить в первый кадр анимации файла car.swf такой сценарий:

```
System.security.allowDomain("http://www.somesite.ru");
```

после чего все сценарии основного фильма, обращающиеся к файлу car.swf, будут работать.

Если основной фильм был загружен по протоколу HTTP, а внешний файл car.swf — по протоколу HTTPS, нам следует использовать метод `allowInsecureDomain` того же объекта. Формат вызова этого метода совпадает с форматом вызова метода `allowDomain`. А код необходимого сценария должен быть таким:

```
System.security.allowInsecureDomain("https://secured.somesite.ru");
_root.placeholder.loadMovie("https://secured.somesite.ru/car.swf");
```

Осталось заметить, что все вышесказанное относится только к файлам Shockwave/Flash. Для файлов Macromedia Flash Video и MP3 все это неактуально, так как они не содержат ни объектов, ни функций, не переменных, к которым можно получить доступ.

Работа с внешними данными

Загрузка и отображение на рабочем листе содержимого внешних файлов — это только часть возможностей, предлагаемых Flash для общения с внешним электронным миром. Другая, весьма обширная, часть позволяет получать и

обрабатывать данные, поставляемые так называемыми *серверными программами* — программами, работающими совместно с Web-сервером. Сейчас мы об этом поговорим.

Отправка данных серверной программе

Предположим, мы создали несколько полей ввода, собирающих данные от пользователя. Как теперь отправить эти данные серверной программе?

Прежде всего, их нужно поместить в переменные. Причем это должны быть переменные уровня клипа, т. е. не локальные и не глобальные. И еще: серверной программе отправляются сразу все данные, находящиеся во всех объявленных в клипе переменных. Так что нужно стараться не создавать лишних переменных уровня клипа.

Для посылки данных нам следует использовать уже изученные методы и действия, а именно:

- `getURL` — загрузка Web-страницы;
- `loadMovie` и `loadMovieNum` — загрузка внешнего клипа;
- `loadSound` — загрузка внешнего звука.

Поскольку серверная программа — это обычный файл, находящийся на диске компьютера, на котором работает Web-сервер (как и Web-страница), формат вызова всех этих действий и методов не изменится. Есть, правда, единственное исключение, о котором мы сейчас поговорим.

Все перечисленные действия и методы поддерживают еще один необязательный параметр. Он передается самым последним в списке параметров, имеет строковый тип и определяет *способ передачи данных*, описывающий то, как данные кодируются перед передачей и как затем передаются. Какой способ выбрать, зависит от серверной программы, с которой мы собираемся "общаться".

Первый способ передачи данных называется `GET`. При его использовании данные передаются Web-обозревателем Web-серверу как часть интернет-адреса. Чтобы передать данные этим способом, нам нужно будет передать с последним параметром описанным выше действиям и методам значение `GET`.

Например, пусть у нас имеется набор переменных, содержащих имя и пароль пользователя в строковом формате (они могут быть, скажем, введены этим самым пользователем в поля ввода):

```
name1 = Ivan
name2 = Ivanov
password = vanyusha
```

Тогда, чтобы отправить серверной программе program.exe, находящейся по адресу **http://www.site.ru/bin/**, эти данные методом GET, мы напишем такое выражение:

```
getURL("http://www.site.ru/bin/program.exe", "_blank", "GET");
```

Обратим внимание на последний параметр этого метода. В нем передается строка "GET" — знак того, что данные должны быть переданы методом GET.

В результате Flash отправит Web-серверу следующий адрес (сами данные в нем выделены полужирным шрифтом):

```
http://www.site.ru/bin/program.exe?name1=Ivan&&name2=Ivanov&
↳password=vanyusha
```

Как видим, пересылаемые способом GET данные объединяются в пары "переменная" = "значение", помещаются в самый конец интернет-адреса и отделяются от него вопросительным знаком. При этом одна пара от другой отделяется знаком "коммерческое и" (&). Все это достаточно просто и наглядно.

Получив этот запрос, Web-сервер первым делом отделит отправленные данные от собственно интернет-адреса серверной программы (в нашем случае это program.exe). Далее он запустит саму эту программу (если она еще не запущена) и передаст ей данные.

Второй способ носит название POST. Данные с его помощью передаются также парами "переменная" = "значение", но уже не в виде части интернет-адреса, а в виде так называемых дополнительных данных запроса. Опять же, Web-сервер выделит эти данные и передаст их серверной программе.

Чтобы реализовать способ POST, нам нужно будет передать соответствующему действию или методу с последним параметром строку "POST":

```
_root.loadMovie("http://www.site.ru/bin/getswf.exe", "POST");
```

Это выражение выполнит отправку данных способом POST. В результате его выполнения серверная программа вернет файл Shockwave/Flash, содержащий клип, который заменит в окне проигрывателя Flash основной фильм.

Все эти методы и действия хороши, пока нам не понадобится получить от серверной программы результат обработки переданных нами данных. В этом случае нам придется воспользоваться другими действиями и методами, которые будут описаны ниже.

Получение данных от серверной программы

Самый простой способ передать серверной программе данные и сразу же получить от нее результат их обработки — использовать действие loadVariables или одноименный метод объекта movieClip. Это действие

(метод) отправляет данные серверной программе и сразу же начинает ожидать получения результата. В принципе, с его помощью мы можем заставить приложение Flash загружать данные из текстового файла, находящегося на сервере или даже на диске нашего собственного компьютера.

Нужно сразу сказать, что серверная программа должна отправить данные, полученные в результате обработки, в виде строки, содержащей набор пар `"переменная" = "значение"`, отделенных друг от друга знаком `&` (так же кодируются данные, передаваемые способом `GET`). Только тогда действие (метод) `loadVariables` сможет правильно их обработать.

Получив от серверной программы данные, действие (метод) `loadVariables` создает в указанном нами клипе соответствующие переменные, если они еще не созданы, и присваивает им принятые значения.

Формат вызова действия `loadVariables` таков:

```
loadVariables("Интернет-адрес серверной программы",
↳"Путь клипа, получающего данные"[, "GET|POST"]);
```

С первым параметром передается интернет-адрес серверной программы, которая должна получить данные и отправить результат, или текстового файла, содержащего их. Со вторым параметром передается путь клипа, который должен принять и обработать эти данные. Третий, необязательный, параметр используется для задания способа передачи отправляемых данных.

Примеры использования действия `loadVariables`:

```
loadVariables("http://site.ru/bin/program.pl", _root.dataHandler,
↳"POST");
loadVariables("data.txt", _root.dataHandler);
```

Оба эти выражения загружают полученные данные в клип `dataHandler`. Но только первое выражение получает их от серверной программы, передав ей способом `POST` значения переменных уровня клипа, а второе загружает их из текстового файла.

Метод `loadVariables` вызывается почти так же:

```
<Клип, получающий данные>.loadVariables("Интернет-адрес серверной
↳программы" [, "GET|POST"]);
```

Например:

```
_root.dataHandler.loadVariables("http://site.ru/bin/program.pl", "POST");
```

Для загрузки данных в перекрывающийся фильм используется действие `loadVariablesNum`. Вот формат его вызова

```
loadVariables("Интернет-адрес серверной программы",
↳"Уровень перекрытия фильма, принимающего данные"[, "GET|POST"]);
```

Комментарии, как говорится, излишни.

Вроде бы, все просто — выполняем действие или метод, получаем данные и обрабатываем их в своем приложении. Просто, да не совсем. Давайте рассмотрим небольшой сценарий, принимающий от серверной программы значение переменной `frameNumber`:

```
_root.loadVariables("http://site.ru/bin/program.exe");
_root.gotoAndStop(frameNumber);
```

Будет ли работать этот код? Может, будет, а может, и нет. И вот почему...

Когда мы даем приложению Flash команду получить данные от серверной программы или из текстового файла, мы не можем быть уверены в том, что эти данные будут получены сию же секунду. Более того, вообще нельзя быть уверенным, что Flash сможет получить их в течение какого-то строго определенного промежутка времени. Это может произойти как через долю секунды, так и через час (например, если произошел обрыв связи или если серверная программа перегружена другими запросами). И когда именно Flash получит результат, мы предугадать не можем.

Но если мы не можем предсказать время, в течение которого приложение получит внешние данные, то мы можем отследить сам момент их получения. Для этого достаточно обработать событие `data` объекта `movieClip`, возникающее после получения клипом всех (до самой последней переменной) данных.

С учетом сказанного перепишем приведенный выше сценарий:

```
_root.onData = function() {
  _root.gotoAndStop(frameNumber);
}
_root.loadVariables("http://site.ru/bin/program.exe");
```

Вот теперь он будет работать.

Использование объекта *LoadVars*

Существует другой способ получения данных от серверной программы — использование экземпляров объекта `LoadVars`. Этот объект предоставляет по сравнению с действиями `loadVariable` и `loadVariableNum` возможность несколько более строгого контроля над процессом обмена данными, хотя выполняет те же самые функции.

Прежде всего, нам нужно создать экземпляр объекта `LoadVars`:

```
myLoadVars = new LoadVars();
```

Затем мы просто создаем для него новые свойства, чьи имена совпадают с именами переменных, которые нужно передать серверной программе.

И разумеется, присваиваем этим свойствам нужные значения:

```
myLoadVars.name1 = "Ivan";
myLoadVars.name2 = "Ivanov";
myLoadVars.password = "vanyusha";
```

Чтобы отправить данные серверной программе, используется метод `send`. Формат его вызова таков:

```
<Экземпляр объекта LoadVars>.send("<Интернет-адрес серверной
↳программы>"[, "<Цель>", "GET|POST"]);
```

С первым параметром передается интернет-адрес серверной программы, которой посылаются эти данные. Со вторым параметром может быть передана цель, указывающая, куда будет выведен результат обработки этих данных (как правило, это Web-страница, сгенерированная серверным приложением). Третий необязательный параметр определяет способ отправки данных; если он не указан, выбирается способ `POST`.

Вот пример выражения, реализующего отправку данных с помощью экземпляра объекта `LoadVars`:

```
myLoadVars.send("http://site.ru/bin/program.exe", "_blank", "GET");
```

Для приема данных от серверной программы используется метод `load`. С единственным параметром этого метода передается интернет-адрес этой серверной программы или же путь к текстовому файлу, содержащему нужные данные. Что касается данных, то они, опять же, должны быть представлены в виде текстовой строки, состоящей из пар `"переменная" = "значение"`, отделенных друг от друга знаком `&`.

Вот пример использования метода `load`:

```
myLoadVars.load("http://site.ru/bin/program.exe");
```

Метод `sendAndLoad` одновременно отправляет серверной программе данные и получает от нее результат их обработки. Формат вызова этого метода:

```
<Экземпляр объекта LoadVars, отправляющий
↳данные>.sendAndLoad("<Интернет-адрес серверной программы>",
↳<Экземпляр объекта LoadVars, принимающий данные>[, "GET|POST"]);
```

Этот метод вызывается для того экземпляра объекта `LoadVars`, который должен отправить данные. С первым параметром передается интернет-адрес серверной программы, которая должна их получить, со вторым — ссылка на экземпляр объекта `LoadVars`, который получит результат. Третий параметр определяет способ отправки данных; если он не указан, выбирается способ `POST`.

Например:

```
mySendVars.sendAndLoad("http://site.ru/bin/program.exe", myLoadVars);
```

Для того чтобы узнать, получены ли данные, нужно обработать событие `load` объекта `LoadVars`, возникающее после получения всех данных. Функция-обработчик этого события должна принимать один логический параметр, обозначающий, успешно ли получены данные (значение `true`) или нет (`false`).

Вот пример функции-обработчика этого события:

```
myLoadVars.onLoad = function(isSucceed) {
  if (isSucceed) {
    _root.gotoAndStop(frameNumber);
  }
}
```

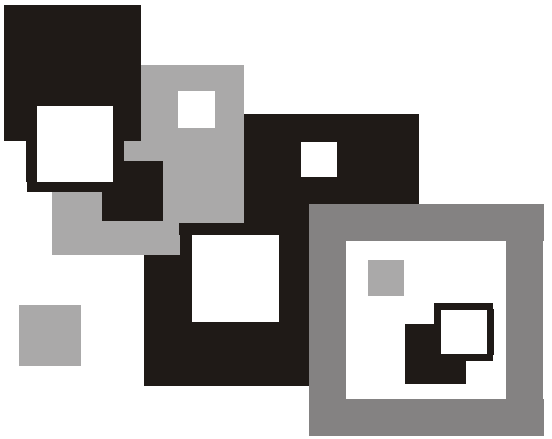
Свойство `loaded` объекта `LoadVars` возвращает значение `true`, если данные были успешно получены, и `false` в противном случае. Если операция приема данных не была запущена, возвращается значение `undefined`.

Объект `LoadVars` предоставляет также еще два метода, которые можно использовать для отображения процесса загрузки данных. Метод `getBytesLoaded` возвращает количество загруженных байт данных. А метод `getBytesTotal` возвращает общий объем загружаемых данных в байтах. Оба эти метода возвращают значение `undefined`, если операция загрузки данных не была запущена или еще реально не началась. Метод `getBytesTotal` также возвращает значение `undefined`, если Web-сервер, под управлением которого работает серверная программа, не сообщил размер передаваемых данных.

Что дальше?

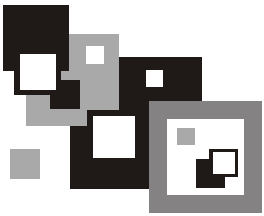
Вот и закончилась третья часть этой книги, посвященная Flash-программированию. Конечно, многое осталось нерассмотренным, но ведь эта книга посвящена графике и анимации, а для интересующихся программированием во Flash существуют другие книги и интерактивная справка, в которой описано абсолютно все.

Следующая глава будет посвящена оптимизации готовых изображений и фильмов перед их публикацией в формат Shockwave/Flash. Она совсем короткая.



ЧАСТЬ V

Публикация и экспорт графики. Прочие возможности Flash



Глава 20

Подготовка графики Flash к экспорту

Все хорошее когда-нибудь кончается... Не успели мы привыкнуть к нашему документу Flash, не успели как следует намучиться с ним, пытаясь заставить все работать, как надо, не успели насладиться сознанием своей власти и могущества, наконец добившись от него послушания, как его уже нужно отдавать заказчику. Как говорил персонаж одного старого мультфильма: "На самом интересном месте"...

У нас осталось совсем немного времени, чтобы проверить, все ли сделано правильно, и нет ли каких ошибок. Не будем торопиться, но и мешкать тоже не стоит. Нужно использовать эти минуты как можно плодотворнее. Как? А вот как...

Прежде всего, нам нужно будет оптимизировать наш фильм. Что такое *оптимизация*? Это довольно долгий и подчас трудоемкий процесс, состоящий в отыскивании скрытых ошибок и "узких мест" и устранении их. Оптимизировать фильм — это значит, задать правильные параметры экспорта импортированной растровой графики и звука, удалить ненужные образцы из библиотеки, ненужную графику на рабочем листе, проверить все созданные в фильме сценарии на предмет ошибок и исправить эти ошибки. Да, времени и труда оптимизация отнимает преизрядно, но после нее наш фильм будет выглядеть лучше, воспроизводиться плавнее, загружаться быстрее, а сценарии станут экономнее использовать память компьютера, не засоряя ее ненужными переменными.

Flash предоставляет мощные средства по выявлению "узких мест", препятствующих быстрой загрузке файла Shockwave/Flash и выводу его на экран. Воспользуемся ими для того, чтобы выяснить, какая часть фильма будет загружаться быстрее, а какая — медленнее. Располагая такими данными, мы сможем ускорить загрузку фильма, а значит, сделать его просмотр более плавным.

Второе, о чем нам следует позаботиться, — это *доступность* фильма (по-английски — *accessibility*). В терминологии Flash — это предоставление набора средств, с помощью которых мы можем сделать просмотр нашего фильма доступным для людей с различными физическими недостатками. В частности, мы можем сделать свой фильм доступным для пользователей с недостатками зрения. Специальное программное обеспечение через звуковую карту компьютера будет зачитывать им то, что находится на экране и в активном на данный момент поле ввода.

Начнем с оптимизации. Именно ее нужно провести в первую очередь.

Оптимизация фильма

Здесь мы рассмотрим средства, предлагаемые Flash для анализа процесса загрузки и воспроизведения фильма. Благодаря им мы сможем выявить "узкие места" нашего фильма. Также мы выясним, что можно сделать, дабы наши фильмы стали компактнее и быстрее.

Использование профилировщика загрузки для выявления "узких мест" фильма

Обычно проигрыватель Flash начинает воспроизведение фильма сразу же после того, как загрузит полностью несколько первых его кадров. В дальнейшем загрузка остальных кадров идет одновременно с воспроизведением. Благодаря такому подходу пользователю не нужно дожидаться, пока весь фильм будет загружен.

Однако если по каким-то причинам загрузка фильма будет приостановлена, проигрыватель Flash не сможет получить данные для дальнейшего воспроизведения этого фильма. Тогда в воспроизведении наступает пауза до тех пор, пока не будет получено достаточно данных для его продолжения.

Причин, по которым загрузка фильма может приостановиться, существует довольно много. Не последнее место в их ряду занимают проблемы с аппаратной частью компьютера и сетевым программным обеспечением. Например, при загрузке фильма из Интернета модем пользователя может внезапно оборвать соединение, или антивирус может не пропустить подозрительный с его точки зрения файл. В таком случае говорят о внешних причинах прерывания загрузки, с которыми мы ничего поделать не можем.

Однако мы способны устранить внутренние причины. Такими причинами могут быть: слишком большое растровое изображение, импортированное в фильм, большой звуковой файл, большой видеоклип и т. п. Понятно, что эти проблемы вызваны, в основном, слишком большими размерами импор-

тированных файлов, которые проигрывателю Flash нужно загрузить и вывести на экран. А на эти размеры мы вполне можем влиять.

В этом нам поможет, на начальном этапе, инструмент Flash, называемый *профилировщиком загрузки*. С его помощью мы без труда выясним, какой именно слишком объемный файл "тормозит" загрузку нашего фильма.

Чтобы подвергнуть профилировке какой-либо фильм, нужно открыть его в среде Flash и выбрать пункт **Test Movie** меню **Control** или нажать комбинацию клавиш <Ctrl>+<Enter> (т. е. запустить его на воспроизведение в отдельном окне). Чтобы протестировать только текущую сцену, нужно выбрать пункт **Test Scene** меню **Control** или нажать комбинацию клавиш <Ctrl>+<Alt>+<Enter>.

В том случае, если фильм уже экспортирован в формат Shockwave/Flash, надо просто открыть нужный файл Shockwave/Flash, воспользовавшись пунктом **Open** меню **File** или комбинацией клавиш <Ctrl>+<O>. Открытый таким образом фильм сразу начнет воспроизводиться в отдельном окне Flash.

Теперь откроем подменю **Download Settings** меню **View** окна просмотра фильма. В нем находится набор пунктов-переключателей, с помощью которых выбирается скорость загрузки фильма: **14.4 (1.2 KB/s)**, **28.8 (2.3 KB/s)**, **56K (4.7 KB/s)**, **DSL (32.6 KB/s)**, **T1 (131.2 KB/s)**, **User Setting 6 (2.3 K)**, **User Setting 7 (2.3 K)** и **User Setting 8 (2.3 K)**.

В случае если переключателя нужной нам скорости в наборе нет, мы сами сможем задать ее, выбрав пункт **Customize** в том же подменю **Download Settings** меню **View**. На экране появится диалоговое окно **Custom Download Settings** (рис. 20.1), в котором и задаются доступные в подменю **Download Settings** меню **View** значения скорости загрузки фильма.

Это окно содержит восемь пар полей ввода, соответствующих каждому из перечисленных выше пунктов-переключателей подменю **Download Settings**. В левом поле ввода каждой пары (**Menu text**) задается текст пункта подменю, а в правом (**Bit rate**) — соответствующее ему числовое значение скорости загрузки в байтах в секунду. После ввода данных нужно нажать кнопку **OK** для сохранения сделанных настроек или **Cancel** для отказа от них. А чтобы вернуться к значениям по умолчанию, следует нажать кнопку **Reset**.

Итак, мы выбрали скорость загрузки. Что теперь?

Прежде всего, Flash 8 предоставляет возможность быстро оценить скорость загрузки нашего фильма без вызова профилировщика загрузки. Для этого мы должны просто включить пункт-выключатель **Simulate Download** меню **View** или нажать комбинацию клавиш <Ctrl>+<Enter>. После этого засекаем время на секундомере и ждем... Когда Flash "загрузит" наш фильм, он выведет его в окно воспроизведения; если файл Shockwave/Flash, содержащий фильм, достаточно велик, ждать этого придется долго.

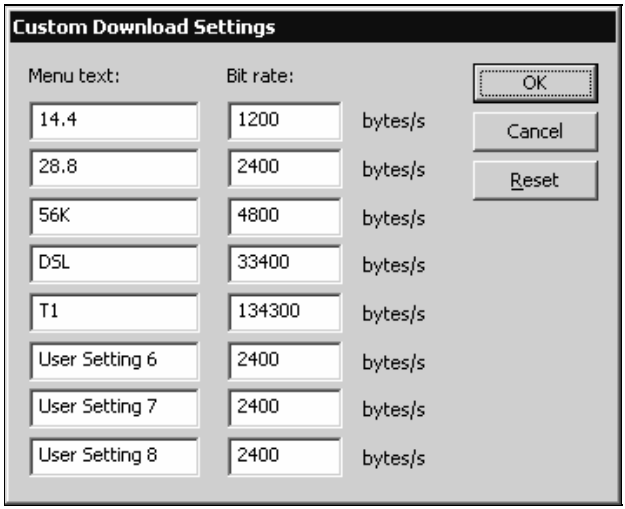


Рис. 20.1. Диалоговое окно **Custom Download Settings**

Если же нам нужно получить более подробные сведения о том, как будет загружаться фильм и где находятся его "узкие места", придется все-таки воспользоваться профилировщиком загрузки. Для вызова его на экран нужно включить пункт-выключатель **Bandwidth Profiler** меню **View** или нажать комбинацию клавиш <Ctrl>+<B>. После этого окно просмотра фильма будет разделено на две части: в нижней части будет воспроизводиться сам фильм, а в верхней появятся данные профилировщика загрузки.

С нижней частью окна просмотра все ясно. Давайте посмотрим на верхнюю часть (рис. 20.2). Видно, что она также разделена на две части: левую, с текстом, и правую, с диаграммой. Мы можем перемещать мышью разделяющую части толстую серую полосу и тем самым менять их размеры.

Текст, отображаемый в левой части, представляет различные сведения о фильме. А именно:

- в разделе **Movie**:
  - **Dim** — геометрические размеры "картинки" фильма в пикселах;
  - **Fr rate** — частоту кадров фильма;
  - **Size** — размер файла Shockwave/Flash, в котором хранится фильм, в килобайтах и байтах;
  - **Duration** — продолжительность фильма в кадрах и секундах;
  - **Preload** — величину задержки, необходимой для загрузки файла фильма перед началом его воспроизведения, в кадрах и секундах;

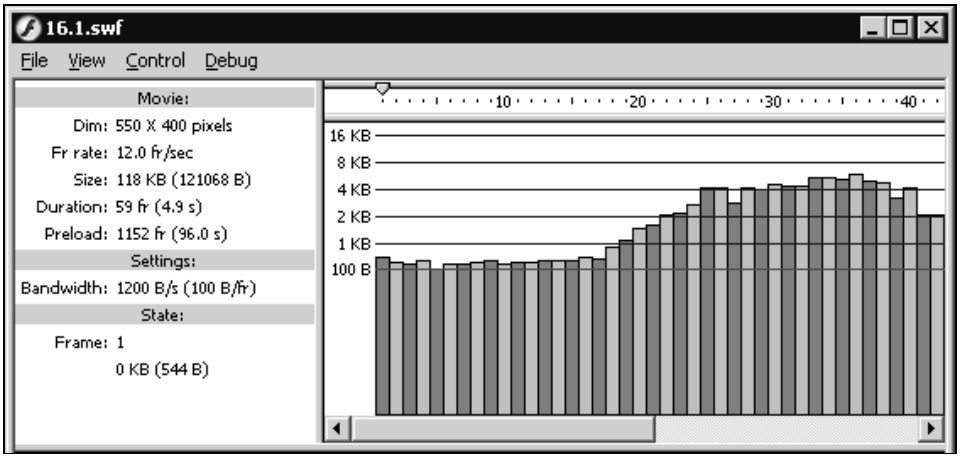


Рис. 20.2. Данные профилировщика загрузки в верхней части окна просмотра фильма Flash

- в разделе **Settings**: **Bandwidth** — скорость загрузки фильма, выбранную нами ранее в подменю **Download Settings** меню **View**;
- в разделе **State**: **Frame** — номер текущего кадра и (ниже) размер данных в килобайтах и байтах, необходимых для его отображения.

Если теперь включить пункт-выключатель **Simulate Download** меню **View**, запустив имитацию загрузки, то в разделе **State** появится строка **Loaded**, отображающая количество загруженных к данному моменту кадров и их размер в килобайтах и байтах.

Мы видим, что профилировщик загрузки обеспечивает доступ к довольно большому числу данных, которые могут помочь нам в работе по оптимизации фильма. Но самая ценная информация отображается с его помощью в правой верхней части окна воспроизведения фильма. Это *диаграмма загрузки* фильма.

Давайте посмотрим на нее (см. рис. 20.2). На горизонтальной оси этой диаграммы отложены номера кадров нашего фильма; в верхней части ее находится также временна́я шкала и бегунок, показывающий, какой кадр фильма в данный момент воспроизводится. На вертикальной же оси диаграммы отложены значения размера данных в байтах и килобайтах.

Смотрим дальше. На горизонтальной шкале диаграммы находятся разноцветные столбики, зачастую поставленные друг на друга (на рис. 20.3 это хорошо видно). Это значит, что при воспроизведении данного кадра проигрыватель Flash успеет загрузить несколько других кадров. А если на каком-то делении горизонтальной шкалы столбиков вообще нет, значит, проигрыватель Flash загрузил все данные, необходимые для их воспроизведения.

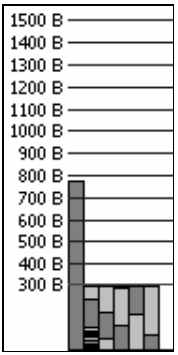


Рис. 20.3. Столбики диаграммы загрузки показывают размер каждого кадра и сколько кадров проигрыватель Flash загрузит при воспроизведении каждого кадра анимации

Одно из делений вертикальной оси выделено красным цветом. Оно показывает предельный объем, при превышении которого проигрыватель Flash не сможет загружать данные достаточно быстро и будет вынужден приостановить воспроизведение фильма до тех пор, пока остальные данные не будут загружены. Таким образом, мы легко сможем узнать, какой кадр "тормозит" наш фильм, и принять необходимые меры.

Если в меню **View** включить пункт-переключатель **Frame by Frame Graph** или нажать комбинацию клавиш <Ctrl>+<F>, вид диаграммы загрузки изменится. Серые столбики теперь будут показывать размер данных, относящихся к текущему кадру (рис. 20.4). Так мы сможем узнать, сколько данных нужно, чтобы воспроизвести тот или иной кадр.

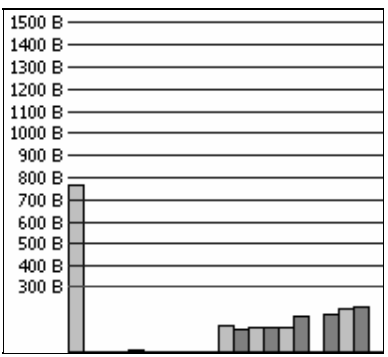


Рис. 20.4. Профилировщик загрузки показывает размер данных каждого кадра

Чтобы вернуться к предыдущему виду диаграммы загрузки (см. рис. 20.3), нужно включить пункт-переключатель **Streaming Graph** меню **View** или нажать комбинацию клавиш <Ctrl>+<G>.

Мы можем остановить воспроизведение фильма, нажав клавишу <Enter> или <Esc> или выбрав пункт **Stop** меню **Control**, и выделить нужный кадр, щелкнув мышью на соответствующей ему позиции временно́й шкалы профилировщика. Также можно щелкнуть на столбике, представляющем размер нужного кадра. После этого Flash выведет данные по выделенному нами кадру в левой части профилировщика загрузки.

Как оптимизировать фильм

А теперь настала пора поговорить собственно об оптимизации графики Flash. Этот разговор займет довольно много времени и будет вестись отдельно о векторной графике, отдельно об импортированной растровой графике и отдельно о звуке.

Оптимизация векторной графики

Нужно твердо запомнить одно простое правило: чем длиннее и сложнее фильм Flash, тем больше места он требует на диске, тем дольше он загружается и обрабатывается перед выводом на экран и тем медленнее выводится. Поэтому следует употребить все средства, чтобы он стал насколько возможно менее сложным и менее длинным. И делать это нужно еще до его публикации.

Прежде всего, для создания множества похожих элементов следует, по возможности, использовать образцы. Также, где только возможно, рекомендуется объединять фрагменты изображения в группы — это сильно облегчит работу с ними и позволит Flash удачнее их оптимизировать. Анимированные фрагменты лучше отделять от неподвижных, "разнося" их по разным слоям. И не стоит забывать о разделяемых образцах — они позволяют уменьшить размер файлов Shockwave/Flash.

Не следует использовать в изображении слишком много линий различных стилей (точечных, пунктирных и др.) — они занимают больше места в памяти, нежели сплошные линии. (Это относится как к обычным линиям, так и к контурам фигур.) Крайне желательно использовать лишь необходимый минимум мазков "кистью" — и они весьма охочи до памяти. Рекомендуется также выполнять оптимизацию кривых с помощью пункта **Optimize** подменю **Shape** меню **Modify** (*см. главу 3*); благодаря этому они станут проще, а значит, займут меньше памяти. Вообще, лучше не использовать без необходимости слишком сложные кривые.

При выборе цвета нужно стараться, чтобы он, по возможности, совпадал с одним из цветов безопасной палитры Web, а при необходимости — использовать минимум градиентных цветов — они требуют объема памяти примерно на 50 байтов больше, чем сплошные. И помните, что чрезмерное увлечение полупрозрачными фигурами может замедлить вывод изображения на экран.

При создании текстовых блоков желательно ограничить количество используемых шрифтов необходимым минимумом. Будет лучше, если мы ограничимся шрифтами-псевдонимами (*см. главу 7*). Если в фильме присутствуют поля ввода или динамические текстовые блоки, нужно ограничить набор символов, которые будут внедрены в результирующий файл Shockwave/Flash. (Как мы уже знаем, Flash при экспорте внедряет в результирующий файл информацию обо всех использованных в изображении шрифтах, что может сильно увеличить размер этого файла.)

Везде, где это возможно, лучше использовать трансформационную анимацию вместо покадровой. Для создания анимации лучше пользоваться образцами-клипами, а не анимированными графическими образцами. При создании трансформационной анимации имеет смысл задавать минимальные изменения — слишком большие изменения увеличивают размер фильма и замедляют его проигрывание. По возможности стоит отказаться от анимации импортированной растровой графики, а для создания звукового сопровождения использовать импортированный звук в формате MP3 — он занимает меньший объем памяти.

И вообще, самый лучший способ сделать фильм меньше — сделать его проще. Будьте проще — и пользователи вас поймут!

Задание параметров импортированной растровой графики

При экспорте фильма Flash, содержащего импортированную растровую графику, Flash выполняет ее сжатие. Мы можем управлять параметрами сжатия графики, а именно, степенью сжатия и выбором используемого алгоритма. Давайте выясним, как это делается.

Прежде всего, нам нужно вывести на экран панель **Library**. После этого отыщем в списке образцов тот, что соответствует нужному растровому изображению, выделим его и выберем в контекстном или дополнительном меню пункт **Properties**. На экране появится диалоговое окно **Bitmap Properties** (рис. 20.5).

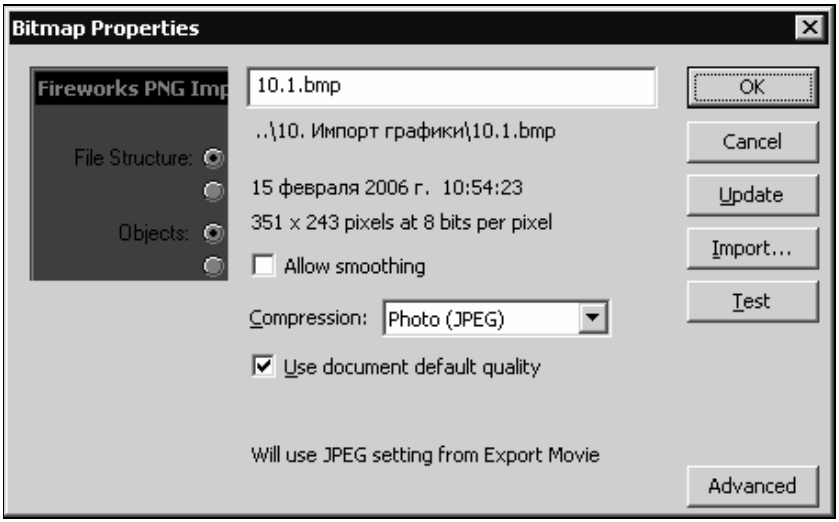


Рис. 20.5. Диалоговое окно **Bitmap Properties**

В поле ввода, находящемся в верхней части этого окна, задается имя образца. Ниже этого поля ввода отображены следующие данные (перечисляются в порядке сверху вниз):

- путь и имя изначального файла, из которого было импортировано это растровое изображение;
- дата и время создания образца;
- размеры изображения в пикселах и его цветность.

В левом верхнем углу диалогового окна **Bitmap Properties** находится небольшая область предварительного просмотра, в которой отображается само это растровое изображение. Мы можем перетаскивать содержимое этой области мышью, чтобы рассмотреть различные его части.

С помощью флажка **Allow smoothing** включается или отключается сглаживание границ изображения. Рекомендуется всегда его включать.

В раскрывающемся списке **Compression** задается алгоритм сжатия растрового изображения. В списке доступны два пункта, которые мы сейчас рассмотрим.

Выбор пункта **Photo (JPEG)** задает для изображения алгоритм сжатия JPEG. Как мы помним из *главы 1*, этот алгоритм применяется для сжатия изображений JPEG и реализует сжатие с потерями. Его стоит использовать для сложных полутоновых изображений с большим количеством цветов или оттенков серого, например, для фотографий или рисунков с градиентными заливками.

При выборе пункта **Photo (JPEG)** списка **Compression** ниже появится флажок **Use document default quality**. Будучи включенным, он задает для импортированного изображения качество изначального изображения, хранящегося в файле. Если этот флажок отключить, ниже его появится поле ввода **Quality**, где мы сможем ввести числовое значение от 1 до 100, задающее качество изображения. Максимальное качество определяется числом 100 единиц, а качество по умолчанию — числом 50.

Пункт **Lossless (PNG/GIF)** списка **Compression** задает для изображения алгоритм сжатия без потерь, который применяется для сжатия графики форматов GIF и PNG. Этот алгоритм лучше всего применять для простых штриховых изображений с небольшим количеством цветов, например, для схем или карандашных рисунков.

После задания нужных параметров сжатия надо щелкнуть на кнопке **Test** — и Flash обновит изображение в области предварительного просмотра в левом верхнем углу диалогового окна **Bitmap Properties**. А в самом низу окна появятся сведения о том, насколько компактнее получившееся новое изображение по сравнению с изначальным.

Задание нужных параметров сжатия завершается, как обычно, нажатием кнопки **OK**. А чтобы отказаться от них, нужно нажать кнопку **Cancel**.

Задание параметров звука

При экспорте фильма, содержащего импортированный звук, Flash перекодирует его с использованием одного из поддерживаемых им кодеков, после чего, если нужно, выполняет микширование нескольких звуков в один. Мы можем задать для любого импортированного звука кодек и параметры сжатия.

Итак, отыщем в списке образцов панели **Library** тот, что соответствует нужному нам звуку, выделим его и выберем в контекстном или дополнительном меню пункт **Export Settings**. На экране появится диалоговое окно **Sound Settings** (рис. 20.6). В этом окне находится раскрывающийся список **Compression**, с помощью которого выбирается алгоритм сжатия звука. Список содержит пять пунктов, которые мы подробно рассмотрим ниже.

Если в списке **Compression** выбран пункт **Default**, то к данному звуку будут применены параметры по умолчанию, заданные при публикации фильма. О том, как задать эти параметры, будет рассказано в *главе 21*.

Если выбран пункт **ADPCM**, то для сжатия звука будет применен алгоритм *ADPCM*, разработанный фирмой Microsoft в 90-х годах прошлого века и неплохо подходящий для сжатия коротких звуков. Ниже списка **Compression** при этом появятся дополнительные элементы управления, позволяющие задать параметры кодирования.

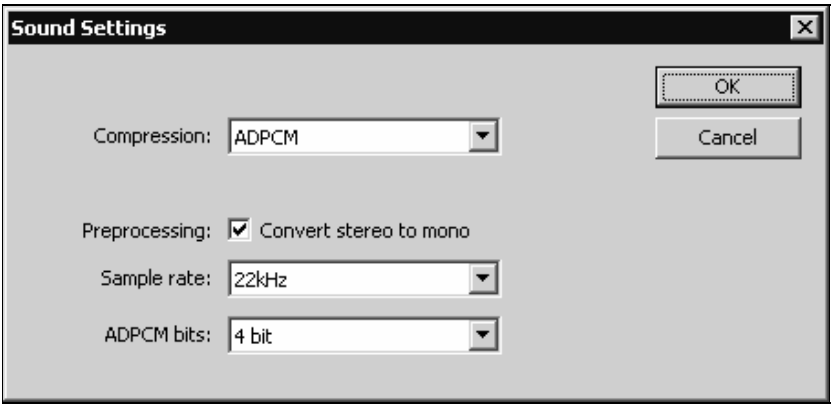


Рис. 20.6. Диалоговое окно **Sound Settings**

Флажок **Convert stereo to mono**, будучи включенным, предписывает Flash преобразовать при экспорте стереофонический звук в монофонический, благодаря чему массив звуковых данных уменьшится в два раза. Этот флажок включен по умолчанию.

Раскрывающийся список **Sample rate** позволяет задать частоту оцифровки звука. Он содержит четыре пункта:

- **5kHz** — 5 кГц (вполне разборчивая речь);
- **11kHz** — 11 кГц (минимальная допустимая частота оцифровки музыки; допускается для коротких фрагментов и в случаях, когда нужно любой ценой уменьшить размер файла Shockwave/Flash);
- **22kHz** — 22 кГц (наиболее часто используемая в фильмах Flash частота оцифровки звука, обеспечивающая приемлемое его качество при небольшом размере массива звуковых данных);
- **44kHz** — 44 кГц (звук CD-качества, но размер массива звуковых данных очень велик).

> ***ВНИМАНИЕ!***
>
> Flash может понизить частоту оцифровки импортированного звука, но не в состоянии ее повысить. Например, если в документ был импортирован звук, оцифрованный с частотой 22 кГц, то он может быть экспортирован с частотой 5, 11 или 22 кГц, но не 44 кГц.

Раскрывающийся список **ADPCM bits** позволяет косвенно задать ширину потока данных для данного звука. Этот список содержит четыре пункта: **2 bit**, **3 bit**, **4 bit** и **5 bit**, задающих ширину потока от минимальной до максимальной.

Если в списке **Compression** выбран пункт **MP3**, то звук будет сжат с помощью известного алгоритма сжатия MPEG 1 level 3, иначе говоря, MP3. Рекомендуется выбирать его, если импортированные звуки уже сжаты этим алгоритмом. С помощью появившихся в окне дополнительных элементов управления мы можем задать параметры сжатия.

Если выделенный нами образец-звук сжат алгоритмом MP3, то окно **Sound Settings** будет содержать включенный по умолчанию флажок **Use imported MP3 quality**. (Кроме того, в этом случае в списке **Compression** будет изначально выбран пункт **MP3**.) Он предписывает Flash экспортировать этот звук без изменения его параметров. Чтобы изменить параметры экспорта звука, нужно снять этот флажок.

Раскрывающийся список **Bit rate** задает ширину потока данных выделенного образца-звука. Он содержит двенадцать пунктов — от **8 kbps** до **160 kbps**, — соответствующих значениям 8-160 Кбит/с. Как говорилось в *главе 17*, чем меньше ширина потока данных, тем меньше места занимает этот звук, но и тем ниже его качество.

Если выбрать ширину потока данных 20 Кбит/с или выше, становится доступным флажок **Convert stereo to mono**. В противном случае этот флажок будет включен и недоступен для отключения. Это значит, что звуки низкого качества Flash при экспорте всегда преобразует в монофонические.

Раскрывающийся список **Quality** позволяет задать скорость сжатия звука и его качество. Он имеет три пункта: **Fast** (быстрое сжатие, но низкое качество), **Medium** (средняя скорость сжатия и среднее качество звука) и **Best** (низкая скорость сжатия и высокое качество звука).

Если в списке **Compression** выбран пункт **Raw**, то звук вообще не будет сжат при экспорте. Рекомендуется выбирать этот пункт только для кодирования коротких звуковых фрагментов. Здесь также возможно задать некоторые параметры экспортируемого звука.

Раскрывающийся список **Sample rate** позволит задать частоту оцифровки звука. Он содержит четыре пункта: **5kHz**, **11kHz**, **22kHz** и **44kHz**.

Флажок **Convert stereo to mono**, будучи включенным, предписывает Flash при экспорте преобразовать звук в монофонический. Этот флажок включен по умолчанию.

Если выбран пункт **Speech** списка **Compression**, звук будет сжат с помощью особого алгоритма, специально созданного для кодирования речи. Выбирать данный пункт стоит только для речи "персонажей" нашего фильма.

Параметров, управляющих сжатием звука, здесь немного. Фактически доступен только уже знакомый нам раскрывающийся список **Sample rate**, позволяющий задать частоту оцифровки звука. Вдобавок, будет присутствовать флажок **Convert stereo to mono**, но он будет включен и недоступен для отключения.

Задав нужные параметры экспортируемого звука, нужно нажать кнопку **OK**. Чтобы отказаться от задания новых параметров звука, следует нажать кнопку **Cancel**.

Кстати, все перечисленные выше элементы управления присутствуют и в уже знакомом нам диалоговом окне **Sound Properties** (см. рис. 17.4).

Обеспечение доступности фильма

Напоследок мы поговорим о том, как сделать наши фильмы доступными для людей с физическими недостатками. В первую очередь это будет касаться слепых и слабовидящих, для которых и операционная система Windows, и сам Flash предусматривают множество специальных средств. И одно из них — это технология "чтение с экрана".

Технология "чтение с экрана"

Увы, не всем дано радоваться ярким краскам наших изображений и фильмов Flash! Несчастные, потерявшие зрение, не смогут увидеть ничего, как бы мы не старались. Но, к счастью, невидящие люди смогут нас услышать. И в наших силах помочь им в этом.

Как? Сейчас узнаем. Но сначала познакомимся со средствами, благодаря которым слепые и слабовидящие получают возможность работать с компьютером.

Для этого используется особая технология с очень простым названием — *"чтение с экрана"*. Заключается она в следующем. Предположим, за спиной невидящего пользователя, работающего за компьютером, стоит помощник, который говорит ему о том, что находится в данный момент на экране и в активных элементах управлениях, например в полях ввода. А невидящий, пользуясь специальной клавиатурой, поддерживающей шрифт Брайля, перемещается по элементам управления и вводит в них нужные данные. Таким образом, он сможет ввести данные в регистрационную форму почтового сервера и написать письмо. Значит, он сможет общаться с друзьями, а это уже немало!

Роль такого помощника выполняет специальное программное обеспечение, встроенное в Windows и синтезирующее речь на основе считанного из окна или элемента управления текста. Проигрыватель Flash использует это программное обеспечение, чтобы произносить вслух текст, являющийся содержимым воспроизводимого фильма. А выводится звук через звуковую карту, которая ныне является стандартным компонентом компьютера.

ВНИМАНИЕ!

Технологию "чтение с экрана" поддерживает только ActiveX-версия проигрывателя Flash, предназначенная для Web-обозревателя Microsoft Internet Explorer и других программ, поддерживающих элементы ActiveX. Кроме того, не поддерживается доступность для безоконных режимов вывода фильмов Flash (подробнее о безоконных режимах *см. главу 21*).

Поддержка Flash технологии "чтение с экрана"

Как уже говорилось, Flash при создании доступных фильмов и приложений максимально идет навстречу разработчику. В частности, он автоматически обеспечивает доступность следующих фрагментов изображения:

- текстовых блоков;
- полей ввода;
- кнопок;
- компонентов (описание компонентов Flash см. в поставляемой с ним интерактивной справке);
- клипов;
- самого фильма (поскольку он тоже является клипом).

В случае текстовых блоков (обычных и динамических) Flash просто произносит содержащийся в них текст, например "Имя пользователя" или "Адрес". В случае других элементов все несколько сложнее.

Начнем с того, что поля ввода, кнопки и образцы-клипы, помещенные на рабочем листе, должны иметь уникальные имена, задаваемые с помощью специального поля ввода в панели **Properties**. Как мы уже знаем, эти имена необходимы для управления ими из сценариев ActionScript. Эти же имена называются программным обеспечением технологии "чтение с экрана" невидящему пользователю.

Поэтому, если мы хотим сделать свои фильмы доступными, мы всегда должны задавать их имена, по крайней мере, для полей ввода, кнопок и компонентов. Если же какой-либо элемент фильма, например, поле ввода, не имеет имени, программа чтения с экрана прочитает что-то вроде словосочетания "поле ввода". Не очень-то информативно, не так ли?

Конечно, имена нужно задавать не для всех элементов фильма, но только для тех, которые должны быть доступными. В частности, не стоит давать имена всем клипам на рабочем листе; достаточно дать их тем, которые используются для "общения" с пользователем.

Ну и, разумеется, нужно поместить рядом с полями ввода поясняющие надписи. Встретив комбинацию "надпись-поле ввода", программа чтения с экрана зачитает только текст надписи.

Дополнительные параметры доступности

Мы можем задать дополнительные параметры доступности того или иного фрагмента фильма и всего фильма, воспользовавшись особой панелью **Accessibility** (рис. 20.7). Чтобы вывести ее на экран, достаточно выбрать пункт **Accessibility** подменю **Other Panels** меню **Window** или нажать комбинацию клавиш <Alt>+<F2>.

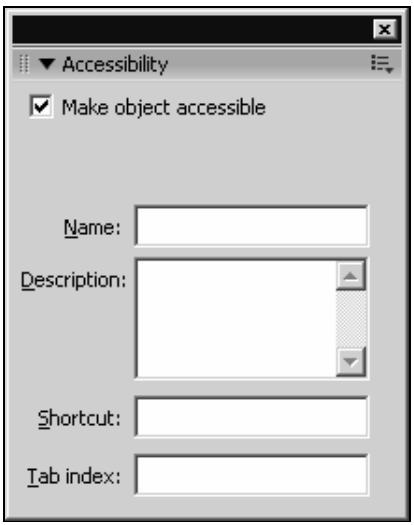


Рис. 20.7. Панель **Accessibility**

Итак, что же мы сможем сделать с помощью этой панели? А вот что...

Прежде всего, мы можем задать *альтернативное имя* для какого-либо фрагмента фильма или для всего фильма Flash. Заданное имя будет произнесено программой чтения с экрана вместо первоначального, того, которое мы задали в панели **Properties**. Для задания альтернативного имени достаточно выделить нужный фрагмент (для выделения самого фильма нужно щелкнуть в пустом пространстве рабочего листа) и ввести это имя в поле ввода **Name**.

Можно задать для какого-либо элемента или для всего фильма дополнительное описание, произносимое вместе с именем. Это описание вводится в область редактирования **Description**.

Для кнопок, полей ввода и компонентов мы можем задать *"горячие клавиши"*. Это особые комбинации клавиш, при нажатии которых данное поле ввода или компонент будет активизироваться, а кнопка — срабатывать. Текст, задающий "горячую клавишу", вводится в поле ввода **Shortcut**. Функциональные клавиши задаются их названиями: "Shift", "Ctrl" или "Alt", обычные клавиши — большими буквами латинского алфавита, а символ "плюс" между названиями клавиш обозначает, что эти клавиши должны нажиматься одновременно. Например, сочетание "Ctrl+H" обозначает, что для перехода к данному компоненту нужно одновременно нажать клавиши <Ctrl> и <H>.

В поле ввода **Tab index** задается номер в порядке обхода данного поля ввода, кнопки или компонента. Лучше его задать, чтобы незрячие пользователи могли выбирать нужные элементы управления с помощью клавиатуры. Да и зрячим пользователям это пойдет на пользу.

Мы можем скрыть какой-либо элемент от программы чтения с экрана, для чего достаточно отключить флажок **Make Object Accessible**. Точно так же можно отключить доступность для всего фильма.

Если какой-либо клип содержит вложенные клипы, мы можем скрыть их от программы чтения с экрана. Для этого достаточно выделить нужный клип на рабочем листе и отключить флажок **Make child objects accessible**.

Если поместить текстовый блок достаточно близко к компоненту, то программа чтения с экрана посчитает такой текстовый блок надписью этого компонента. Чтобы отключить эту особенность Flash, нужно щелкнуть мышью в свободном пространстве рабочего листа и снять флажок **Auto label**. Правда, после этого придется задать альтернативные имена и описания для всех элементов фильма.

Полезные советы по созданию доступных фильмов

В этом разделе будет приведен ряд дополнительных рекомендаций, которые помогут нам в создании доступных фильмов Flash.

Прежде всего, нужно помнить, что встроенное во Flash программное обеспечение, обеспечивающее чтение с экрана, читает только текст, помещенный в текстовые блоки. Программа "не понимает" графику. Поэтому, если мы превратим содержимое какого-либо текстового блока в графику, выбрав пункт **Break Apart** меню **Modify**, программа чтения с экрана прочитать его не сможет.

Далее, нужно сразу решить, будем ли мы давать текстовые описания для каждого элемента управления или дадим только общее описание для всего фильма. Если фильм не содержит никаких элементов управления, то лучше

всего выбрать первый вариант, в противном случае (например, если мы пишем сложное Flash-приложение) — второй.

Лучше всего не анимировать поля ввода, кнопки, компоненты и надписи к ним. Некоторые программы чтения с экрана некорректно обрабатывают анимированные элементы управления, что вряд ли понравится нашим пользователям.

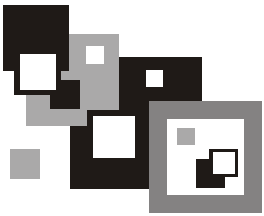
Если в фильме используется импортированный звук, нужно постараться сделать так, чтобы он не мешал пользователям слушать синтезированную программами чтения с экрана речь. Не следует использовать слишком много громких звуков, да еще и воспроизводящихся постоянно. Вообще, звук должен применяться умеренно и строго к месту.

Следует исключить в своем фильме ситуации, когда объемный текст отображается на экране очень короткое время, после чего пропадает без следа. Нужно помнить, что в этом случае программа чтения с экрана просто не успеет прочитать его пользователю.

И напоследок, не забудьте проверить, можно ли "путешествовать" по элементам управления нашего Flash-приложения, пользуясь только клавиатурой. Ведь клавиатура — единственный инструмент ввода информации в компьютер, который может использовать незрячий пользователь.

Что дальше?

После того как мы подготовили наш фильм к публикации, можно приступать собственно к его публикации или экспорту. О том, как осуществляется публикация и экспорт фильма, будет рассказано в следующей главе.



Глава 21

Публикация и экспорт готовой графики Flash

Вот и закончен наш фильм. Добавлены последние штрихи, исправлены последние ошибки, выполнено последнее сохранение, создана последняя резервная копия. Что дальше?

А дальше — либо публикация фильма, либо его экспорт. Давайте выясним, что это за процессы, и в каких случаях они вызываются.

Начнем с публикации. *Публикацией* в терминологии Flash называется процесс сохранения готового фильма в одном из форматов, пригодных для его распространения. Как правило, фильмы Flash сохраняются в формате Shockwave/Flash, но возможно также сохранение в форматах QuickTime и "анимированный GIF". Разумеется, эти два формата поддерживают не все средства Flash: так, сценарии ActionScript ими не поддерживаются.

Помимо собственно файла фильма, Flash при публикации может создать Web-страницу, на которую будет помещен публикуемый фильм. Кроме необходимого кода HTML Flash помещает на эту страницу особый Web-сценарий, проверяющий наличие на компьютере клиента проигрывателя Flash и устанавливающий его в случае отсутствия. Так что нам останется только поместить готовый файл фильма и эту Web-страницу на Web-сервер — и можно открывать киносезон.

При публикации нашего фильма мы также можем заставить Flash создать так называемое *изображение-замену*. Это изображение будет выводиться на Web-странице в случае, если на компьютере клиента отсутствует проигрыватель Flash и установить его пользователь почему-то не может. По умолчанию оно содержит первый кадр нашего фильма и сохраняется в формате GIF, JPEG или PNG, т. е. в одном из общепринятых в Интернете графических форматов.

НА ЗАМЕТКУ

В качестве изображения-замены можно выбрать любой другой кадр фильма. Для этого достаточно дать ему имя `#Static`.

Экспорт отличается от публикации тем, что, осуществляя его, Flash просто создает файл с фильмом в одном из вышеперечисленных форматов. Полученный в результате экспорта файл можно как распространять среди потенциальных поклонников, так и использовать в других программах, например, как часть другого фильма Flash.

Теперь давайте поговорим о публикации и экспорте более подробно. И начнем с публикации.

Публикация

Прежде чем рассматривать сам процесс публикации, давайте решим, какой формат предпочтительнее для целей распространения нашего Flash-творчества. Будет лучше рассмотреть этот вопрос в самом начале.

Выбор формата публикации

Выбор формата публикации фильма и изображения-замены (если таковое есть) мы рассмотрим по отдельности.

Выбор формата публикации фильма

На самом деле, вопрос выбора формата для публикации фильма не так уж и сложен. Мы уже знаем, что существует всего три подходящих формата: Shockwave/Flash, QuickTime и "анимированный GIF". Вдобавок, каждый из этих форматов имеет свою, и достаточно узкую, область применения, так что ошибиться в выборе будет трудно.

Чаще всего фильмы, созданные во Flash, публикуются в формате Shockwave/Flash. И это понятно — только "родной" формат Flash поддерживает все его возможности: анимацию, импортированные графику и звук, сценарии, в конце концов. Ни формат QuickTime, ни формат "анимированный GIF" этим похвастаться не могут — они специализируются исключительно на видео.

Далее, из *главы 1* мы знаем, что файлы Shockwave/Flash бывают двух видов. Давайте повторим пройденное.

Файл первого вида — это собственно файл с расширением swf, содержащий только фильм Flash, и ничего более. Он компактен, а значит, может распро-

страняться через Интернет, но, к сожалению, для его просмотра требуется программа проигрывателя Flash. Впрочем, проигрыватель Flash сейчас установлен на абсолютном большинстве компьютеров, подключенных к Интернету, так что проблема отпадает.

Файл второго вида — это файл проектора, т. е. исполняемый файл, содержащий одновременно и фильм, и программу проигрывателя Flash. При запуске его на экране открывается окно проигрывателя, в котором и происходит воспроизведение фильма. Это довольно удобно, если фильм нужно распространять, скажем, на компакт-дисках, — в этом случае пользователю не придется устанавливать на свой компьютер никаких дополнительных программ. Однако файл проектора очень велик и поэтому мало пригоден для распространения через Интернет.

Flash в состоянии создать файл проектора в двух разных форматах. Это может быть обычный исполняемый файл Windows с расширением exe или исполняемый файл, предназначенный для компьютеров Apple Macintosh и имеющий расширение hqx.

Перейдем к формату QuickTime. Проигрыватель файлов этого формата стандартно поставляется в составе операционной системы компьютеров Apple Macintosh. Конечно, существует версия этого проигрывателя для Windows, но она установлена далеко не на всех компьютерах. Так что, если уж и публиковать фильм Flash в формате QuickTime, то ориентироваться стоит именно на поклонников Macintosh.

> ***Внимание!***
>
> Чтобы иметь возможность публиковать фильмы Flash в формате QuickTime, нужно установить на компьютер программное обеспечение Apple QuickTime соответствующей версии.

Фильм QuickTime, как и фильм Flash, может состоять из нескольких видеодорожек, воспроизводящихся одновременно. При публикации в формат QuickTime фильм преобразуется Flash в формат Shockwave/Flash (да-да, именно так!) и помещается в одну из таких видеодорожек. Но если фильм содержит внедренные клипы QuickTime, то каждый такой клип преобразуется в отдельную видеодорожку.

Но ни одна бочка меда не обходится без ложки дегтя. Возможности QuickTime по поддержке видео Shockwave/Flash находятся на уровне Flash 5. Так что при публикации нашего фильма мы должны будем выставить в параметрах публикации именно пятую версию проигрывателя Flash. Если же мы этого не сделаем, Flash нас об этом предупредит.

> ***Внимание!***
>
> Если фильм содержит привязанные клипы в формате QuickTime, то он может быть опубликован только в формате QuickTime.

Осталось упомянуть о формате "анимированный GIF". Здесь правило самое простое: в этом формате публикуются только совсем простые и короткие фильмы, не содержащие ни сценариев, ни сложной графики (содержащей, скажем, полупрозрачные фрагменты). Например, рекламные баннеры и элементы оформления Web-страниц.

Выбор формата публикации изображения-замены

В случае изображения-замены выбор также не очень велик. Это популярные форматы распространения графики в Интернете: GIF, JPEG и PNG.

Формат GIF, как уже говорилось в *главе 1*, прекрасно подходит для сохранения и распространения штриховых рисунков с небольшим количеством цветов — таких как схемы, карты, карандашные рисунки, элементы оформления Web-страниц и т. п. При сохранении графики в формате GIF выполняется сжатие массива данных по алгоритму, исключающему потери данных. Так что файлы в формате GIF получаются весьма компактными.

Формат PNG — неплохая альтернатива формату GIF. Он особенно хорош тем, что разработан группой независимых исследователей и поэтому бесплатен. Формат же GIF — увы! — как оказалось, стоит денег. Вдобавок PNG предоставляет заметно больше возможностей, чем старенький GIF.

Что касается формата JPEG, то главное его достоинство и одновременно недостаток — это применяемый в нем алгоритм сжатия, вызывающий потерю данных. Файлы формата JPEG очень компактны, но совершенно не подходят для хранения штриховых изображений. Этот формат хорошо использовать для распространения полноцветной полутоновой графики: картин, фотографий, графики с большим количеством градиентов и т. п.

Задание состава публикации

Итак, мы выбрали форматы, в которых будем распространять наш фильм и изображение-замену. Теперь можно приступать к собственно публикации.

Выберем в меню **File** пункт **Publish Settings** или нажмем комбинацию клавиш <Ctrl>+<Shift>+<F12>. На экране появится диалоговое окно **Publish Settings**. На вкладке **Formats** этого окна, показанной на рис. 21.1, и задается набор файлов, из которых будет состоять опубликованный фильм (*состав публикации*).

Чтобы включить тот или иной файл в состав публикации, достаточно включить соответствующий ему флажок из группы **Type** в левой части диалогового окна. Эти флажки имеют "говорящие" названия, так что ошибиться невозможно. Так, например, чтобы создать набор из файла Shockwave/Flash, Web-страницы и изображения-замены в формате GIF, нужно установить

флажки **Flash (.swf)**, **HTML (.html)** и **GIF Image (.gif)**. Все эти файлы будут сохранены Flash в той же папке, где хранится файл публикуемого документа Flash.

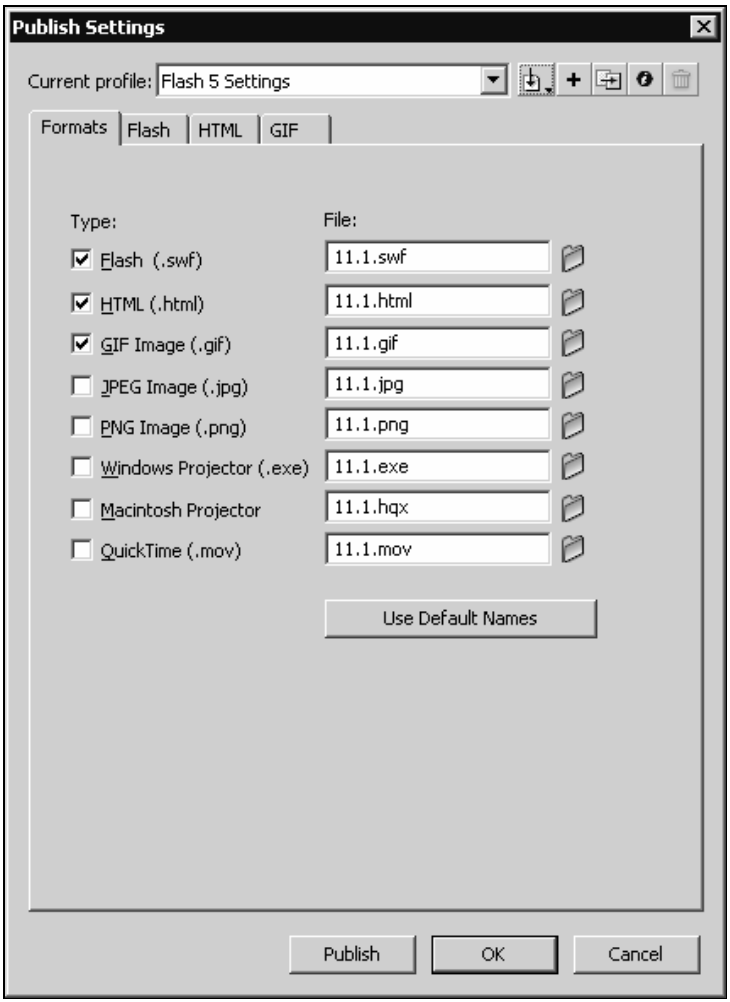


Рис. 21.1. Диалоговое окно **Publish Settings** (вкладка **Formats**)

В правой части диалогового окна **Publish Settings** находится группа полей ввода **File** — по одному полю на каждый флажок группы **Type** и, следовательно, на каждый формат. В них задаются имена соответствующих файлов, входящих в состав публикации. По умолчанию Flash создает их сам в соответствии с таким принципом: берется имя документа Flash и к нему добав-

ляется соответствующее формату расширение (в нашем случае — swf, html и gif). Эти имена можно изменить, либо введя их вручную, либо щелкнув на кнопке, находящейся правее соответствующего поля ввода и имеющей вид папки, и задав новое имя файла в появившемся на экране стандартном диалоговом окне сохранения файла Windows. Чтобы после этого вернуться к именам, созданным самим Flash, достаточно нажать кнопку **Use Default Names**.

После задания параметров публикации нужно нажать кнопку **OK** для их сохранения. Заданные нами параметры будут сохранены в самом файле документа Flash. Нажатие кнопки **Cancel** отменяет введенные параметры.

Да, но как же запустить сам процесс публикации? Очень просто!

Во-первых, если окно **Publish Settings** еще не закрыто, можно нажать кнопку **Publish**. После этого Flash запустит процесс публикации. Когда публикация будет закончена, окно **Publish Settings** останется на экране, так что мы можем, если понадобится, изменить параметры публикации и повторить ее.

Во-вторых, если окно **Publish Settings** уже закрыто, достаточно выбрать пункт **Publish** меню **File** или нажать комбинацию клавиш <Shift>+<F12>. При этом Flash выполнит публикацию с теми параметрами, которые мы задали ранее.

Параметры форматов публикации

Кроме вкладки **Formats**, диалоговое окно **Publish Settings** содержит и другие вкладки. Они имеют названия, совпадающие с названиями доступных во Flash форматов публикации файлов, и позволяют задать различные их параметры. Давайте рассмотрим эти вкладки.

Вкладка *Flash*

Если на вкладке **Formats** диалогового окна **Publish Settings** были включены флажки **Flash (.swf)**, **Windows Projector (.exe)** (т. е. проектор для Windows) или **Macintosh Projector** (т. е. проектор для Macintosh), становится доступной вкладка **Flash**, задающая параметры файла Shockwave/Flash. Ее содержимое показано на рис. 21.2.

С помощью раскрывающегося списка **Version** выбирается версия формата Shockwave/Flash создаваемого файла. Этот список содержит десять пунктов: **Flash Player 1**, **Flash Player 2**, **Flash Player 3**, **Flash Player 4**, **Flash Player 5**, **Flash Player 6** (т. е. Flash MX), **Flash Player 7** (Flash MX 2004), **Flash Player 8** (этот пункт выбран по умолчанию), **Flash Lite 1.0** и **Flash Lite 1.1** (это версии Flash для мобильных устройств). Названия этих пунктов говорят сами за себя.

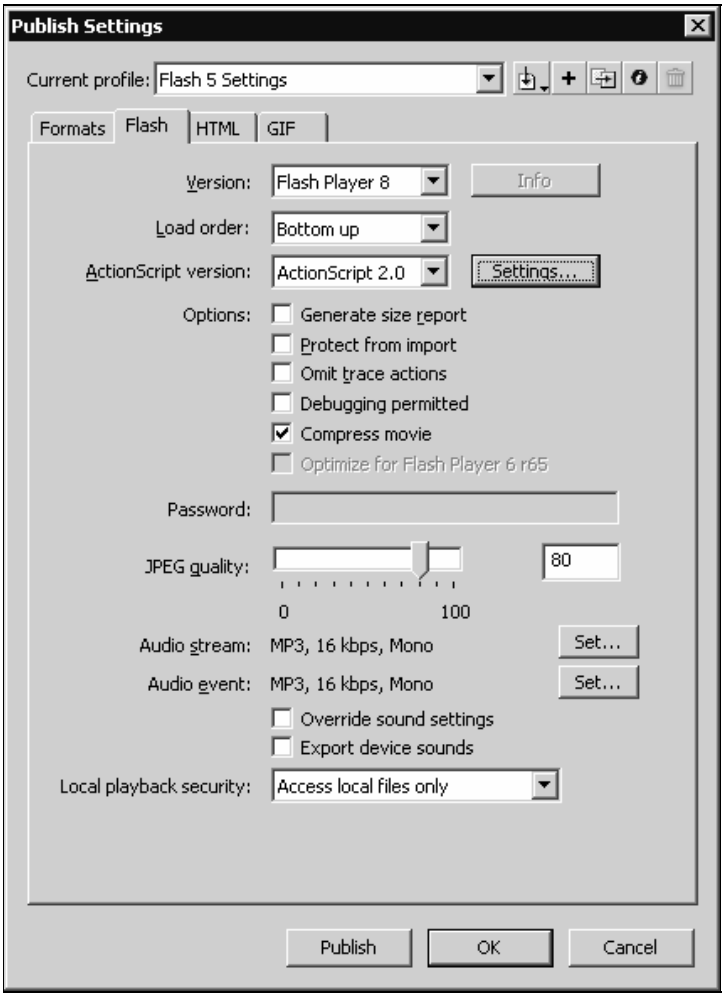


Рис. 21.2. Диалоговое окно **Publish Settings** (вкладка **Flash**)

Внимание!

Если вы создаете фильм в более новой версии Flash, а сохраняете в более старой, то вам придется исключить из него возможности, поддерживаемые новой версией.

Раскрывающийся список **Load order** задает порядок, в котором будут выводиться на экран слои изображения. Он содержит два пункта: **Bottom up** (слои грузятся в порядке снизу вверх; этот пункт выбран по умолчанию) и **Top down** (слои грузятся в порядке сверху вниз). Таким образом, мы можем

выбрать, какие части изображения будут отображены в первую очередь, какие — во вторую, и т. д.

Раскрывающийся список **ActionScript version** позволяет задать версию языка ActionScript, которая была использована для написания сценариев в документе. Здесь доступны два пункта: **ActionScript 1.0** и **ActionScript 2.0** (второй пункт выбран по умолчанию). В нашем случае не играет роли, какую версию ActionScript мы выберем, так что оставим выбранным пункт **ActionScript 2.0**.

С помощью флажка **Generate size report** включается или отключается создание при публикации текстового отчета. Такой отчет представляет собой текстовый файл с именем вида <Имя файла Shockwave/Flash> Report.txt и создается в той же папке, что и файл Shockwave/Flash. В нем приводятся данные о размерах каждого кадра фильма и кода всех сценариев в байтах; они могут быть полезными.

> ***НА ЗАМЕТКУ***
>
> Если был установлен флажок **Generate size report**, то содержимое созданного файла отчета будет после публикации выведено в панели **Output**.

Флажок **Protect from import**, будучи включен, запрещает импортировать наш фильм в другие документы Flash без нашего разрешения, а также делает доступным поле ввода **Password**, в которое мы сможем ввести пароль, разрешающий импорт фильма. После этого при попытке импорта нашего фильма Flash будет спрашивать пароль, и для того чтобы все-таки импортировать его, нужно будет этот пароль ввести. Если же оставить поле **Password** пустым, фильм вообще невозможно будет импортировать.

Флажок **Debugging permitted** включает или отключает возможность удаленной отладки. Если установить этот флажок, становится доступным поле ввода **Password**, в которое мы сможем ввести пароль, разрешающий удаленную отладку фильма. (Об отладке см. в интерактивной справке Flash.)

Флажок **Compress movie** включает или отключает сжатие полученного файла Shockwave/Flash. По умолчанию он включен, и выключать его не рекомендуется, так как сжатый файл получается намного меньшим по размеру, чем несжатый. Особенно эффективно сжимаются файлы, содержащие обильный текст или много сценариев. Нужно только иметь в виду, что сжиматься могут только файлы версии 6 (MX), 7 (MX 2004) и 8.

Если в списке **Version** выбран пункт **Flash Player 6** (MX), становится доступным флажок **Optimize for Flash Player 6 r65**. Этот флажок включает или отключает специальную оптимизацию под версию 6 r65 проигрывателя Flash.

Регулятор **JPEG quality** позволяет задать качество всех импортированных растровых изображений, для которых не были заданы индивидуальные параметры экспорта (об этом см. *главу 20*). Доступны для ввода значения в

диапазоне от 0 до 100. Чем выше качество, тем лучше выглядит изображение и тем больше размер файла Shockwave/Flash. В поле ввода, расположенном правее регулятора, можно ввести значение качества вручную.

При нажатии кнопки **Audio stream: Set** на экране появится диалоговое окно **Sound Settings** (см. рис. 20.6). С его помощью мы сможем задать параметры экспорта тех потоковых звуков, для которых это не было сделано индивидуально (об этом см. *главу 20*). Аналогично, нажав кнопку **Audio event: Set**, мы сможем в том же диалоговом окне задать параметры экспорта сигналов.

Если нам зачем-либо нужно, чтобы заданные нами в диалоговом окне **Publish Settings** параметры экспорта звуков перекрыли параметры, заданные для отдельных звуков, достаточно включить флажок **Override sound settings**. Это может понадобиться, например, если мы собираемся создать специальную "компактную" версию фильма, содержащую звуки низкого качества. Просто задаем низкое качество звука в окне **Publish Settings** и устанавливаем флажок **Override sound settings**.

Раскрывающийся список **Local playback security** позволяет задать разрешение на доступ к другим файлам Shockwave/Flash и JPEG, находящимся на дисках компьютера пользователя или в сети. В нем находятся два пункта: **Access local files only** (возможен доступ только к файлам на дисках компьютера пользователя) и **Access network only** (возможен доступ только к сетевым файлам). О работе с внешними файлами см. *главу 19*.

Вкладка *HTML*

Если на вкладке **Formats** диалогового окна **Publish Settings** был включен флажок **HTML (.html)**, становится доступной вкладка **HTML** (рис. 21.3).

С помощью раскрывающегося списка **Template** выбирается *шаблон HTML* — своего рода заготовка, на основе которой генерируется Web-страница. Таких шаблонов в списке довольно много:

- **Flash For Pocket PC 2003** — оптимизирован для карманных компьютеров, удовлетворяющих спецификации Pocket PC 2003;
- **Flash HTTPS** — оптимизирован для защищенных соединений, использующих протокол HTTPS;
- **Flash Only** — самый простой шаблон, содержащий только фильм Flash. Используется по умолчанию;
- **Flash with AICC Tracking**, **Flash with SCORM 1.2 Tracking** и **Flash with SCORM 2004 Tracking** — оптимизированы для использования с некоторыми расширениями Flash;
- **Flash with FSCommand** — включает код, необходимый для поддержки действия `FSCommand`, используемого для управления воспроизведением

фильма Flash с помощью сценариев JavaScript (описание этого действия можно найти в интерактивной справке Flash);

- **Flash with Named Anchors** — включает код, необходимый для поддержки якорей;
- **Image Map** — содержит карту-изображение HTML. Также требует, чтобы был создан файл формата GIF, JPEG или PNG;
- **QuickTime** — содержит фильм QuickTime.

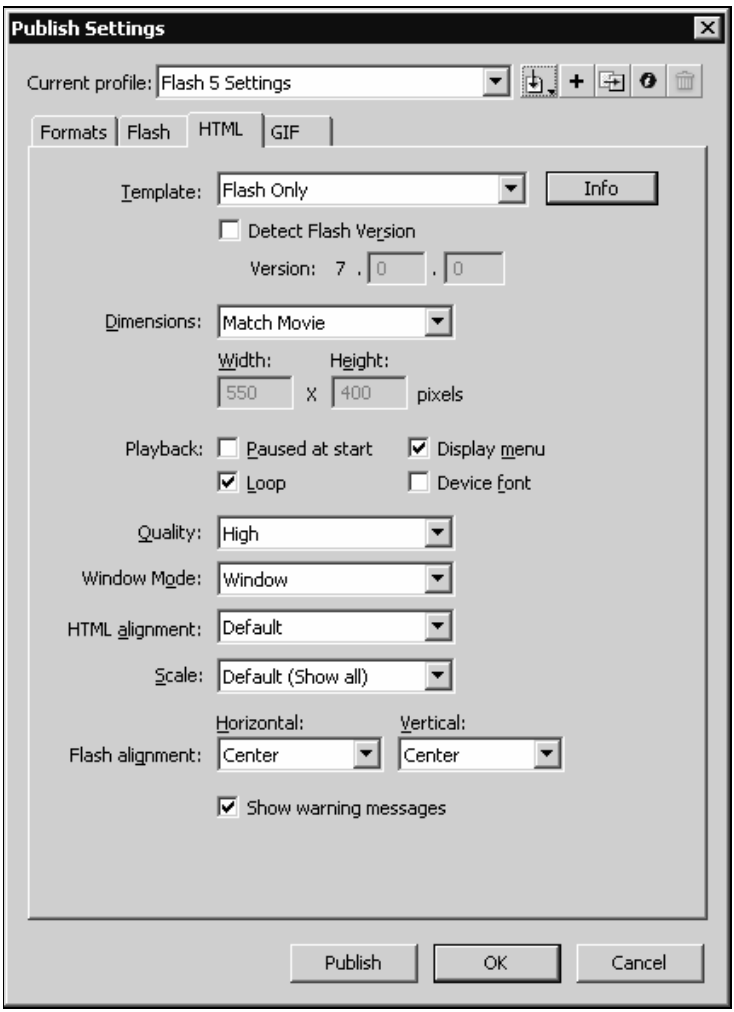


Рис. 21.3. Диалоговое окно **Publish Settings** (вкладка **HTML**)

Кнопка **Info** позволяет получить сведения по выбранному в списке **Template** шаблону. При нажатии ее на экране появится диалоговое окно **HTML Template Info** (рис. 21.4), содержащее эти самые сведения. Чтобы закрыть это окно, нужно нажать кнопку **OK**.

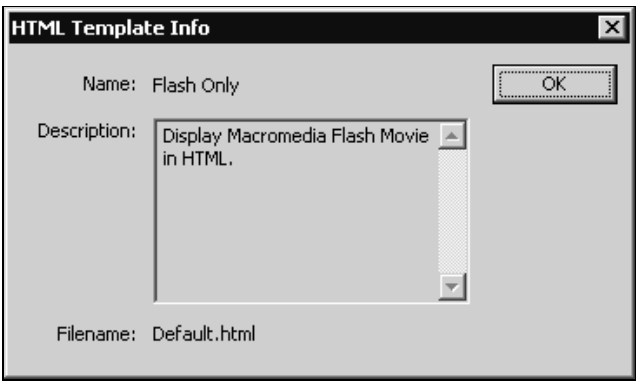


Рис. 21.4. Диалоговое окно **HTML Template Info**

Флажок **Detect Flash Version**, будучи включен, предписывает создать в коде Web-страницы особый сценарий JavaScript, проверяющий, установлен ли на компьютере пользователя проигрыватель Flash. Если проигрыватель установлен, выполняется переход на Web-страницу с опубликованным фильмом, в противном случае — на другую страницу, предлагающую загрузить и установить проигрыватель.

Если установить этот флажок, станет доступным набор из двух полей ввода **Version**. В этих полях ввода можно ввести соответственно меньший номер версии и номер ревизии проигрывателя Flash, который нужен для воспроизведения нашего фильма. Правда, нужда в таком точном задании версии проигрывателя Flash возникает довольно редко.

Раскрывающийся список **Dimensions** и поля ввода **Width** и **Height** позволят нам задать геометрические размеры фильма. Если в списке **Dimensions** выбран пункт **Match Movie**, то они будут равны размерам, заданным в параметрах публикуемого документа Flash. Если выбран пункт **Pixels**, то мы сами сможем задать размеры фильма с помощью полей ввода **Width** (ширина) и **Height** (длина) в пикселах. Если же выбран пункт **Percent**, то размеры фильма задаются в тех же полях ввода, как проценты относительно размеров окна Web-обозревателя.

Если включить флажок **Paused at start**, фильм не будет запущен на воспроизведение сразу же после загрузки. Вместо этого пользователю самому

придется запустить этот фильм, воспользовавшись контекстным меню проигрывателя или иным способом, например, предусмотренными в самом фильме кнопками управления. Если флажок **Paused at start** отключен (а он отключен по умолчанию), фильм начнет воспроизводиться сразу же после загрузки.

С помощью флажка **Loop** включается или отключается "зацикливание" фильма.

Флажок **Display menu** при включении разрешает появление контекстного меню проигрывателя Flash при щелчке на нем правой кнопкой мыши. С помощью такого меню пользователь может запускать и останавливать фильм, задавать качество его воспроизведения и некоторые другие параметры. Если этот флажок отключен, то в контекстном меню остается один-единственный пункт — **About Flash Player**, выводящий диалоговое окно со сведениями о программе проигрывателя. Запретить контекстное меню бывает нужно в тех случаях, когда оно может помешать, скажем, при создании Flash-приложений.

Флажок **Device font** разрешает или запрещает Flash подставлять вместо не установленных на компьютере клиента шрифтов наиболее подходящие из уже установленных. Нужно только помнить, что такая подстановка работает лишь для статичных текстовых блоков.

Раскрывающийся список **Quality** позволяет задать качество вывода графики Flash. Он содержит шесть пунктов:

- **Low** — проигрыватель пытается воспроизвести фильм как можно более плавно, не заботясь о качестве вывода и не используя сглаживание. Самое низкое качество;
- **Auto Low** — изначально качество графики низкое, сглаживание не используется. Далее, если компьютер окажется достаточно мощным, проигрыватель Flash повысит качество графики и включит сглаживание;
- **Auto High** — изначально качество графики высокое, используется сглаживание. Далее, если компьютер окажется недостаточно мощным, проигрыватель Flash понизит качество графики и отключит сглаживание;
- **Medium** — используется сглаживание для векторной, но не для импортированной растровой графики;
- **High** — проигрыватель Flash пытается вывести графику как можно более качественно, не заботясь о плавности воспроизведения анимации. Используется сглаживание векторной графики, а если нет анимации, то и растровой графики. Этот пункт выбран по умолчанию;
- **Best** — используется сглаживание и векторной, и растровой графики. Самое высокое качество, которое не уменьшается ни в каких случаях.

Раскрывающийся список **Window Mode** позволяет задать вид фильма на Web-странице. Этот список содержит три пункта:

- **Window** — фильм отображается на Web-странице как бы в собственном "окне". Содержимое Web-страницы, лежащее над и под этим "окном", не выводится на экран. Фон фильма совпадает с фоном Web-страницы. Самая высокая скорость воспроизведения. Этот пункт выбран по умолчанию;
- **Opaque Windowless** — фильм не будет отображаться в собственном "окне" (*безоконный режим*). Элементы, находящиеся на Web-странице над этим "окном", будут выведены на экран;
- **Transparent Windowless** — фильм не будет отображаться в собственном "окне". Элементы, находящиеся на Web-странице над и под этим "окном", будут видны. Если Web-страница слишком сложна, анимация может воспроизводиться рывками.

> ***ВНИМАНИЕ!***
>
> Безоконный режим воспроизведения фильмов Flash поддерживают только Internet Explorer версии 5.0 и более новые, Navigator версии 7 и более новые, Opera 6 и более новые и все версии Mozilla и Firefox.

Раскрывающийся список **HTML alignment** задает местоположение фильма на Web-странице. Список содержит пять пунктов:

- **Default** — фильм отображается в центре страницы. Если окно Web-обозревателя меньше самого фильма, он будет обрезан;
- **Left**, **Top**, **Bottom** и **Right** — фильм выравнивается соответственно по левой, верхней, нижней или правой границе Web-страницы. Опять же, если окно Web-обозревателя меньше фильма, он будет обрезан с соответствующей стороны.

С помощью раскрывающегося списка **Scale** мы можем задать параметры масштабирования изображения Flash, если оно не помещается в отведенные ему размеры. Этот список содержит четыре пункта:

- **Default (Show all)** — изображение будет показано целиком, для чего может быть применено масштабирование. Однако пропорции изображения искажены не будут, в результате чего вдоль его горизонтальных или вертикальных сторон могут появиться пустые места;
- **No border** — то же самое, что **Default (Show all)**, но пустые места появляться не будут — Flash обрежет изображение по горизонтали или вертикали, чтобы убрать их;
- **Exact fit** — будет показано все изображение, для чего может быть применено масштабирование. Но пропорции изображения при этом соблюдаться не будут;

- **No scale** — изображение ни в каком случае не будет масштабироваться, в результате чего может оказаться обрезанным, или вдоль его горизонтальных или вертикальных сторон могут появиться пустые места.

Набор из двух раскрывающихся списков **Flash alignment** позволяет задать выравнивание изображения внутри "окошка" проигрывателя Flash. В списке **Horizontal** задается выравнивание по горизонтали: **Left** (По левому краю), **Center** (По центру; значение по умолчанию) или **Right** (По правому краю). В списке **Vertical** задается выравнивание по вертикали: **Top** (По верхнему краю), **Center** (По центру — это значение по умолчанию) или **Bottom** (По нижнему краю).

При задании параметров публикации фильма Flash немудрено допустить ошибку. Например, возможна такая ситуация: для данного шаблона требуется изображение-замена, а мы не включили его в состав публикации. Но если установить флажок **Show warning messages**, то Flash будет предупреждать нас о таких ошибках. (Впрочем, этот флажок установлен по умолчанию.)

Вкладка *GIF*

Если на вкладке **Formats** диалогового окна **Publish Settings** был включен флажок **GIF (.gif)**, становится доступной вкладка **GIF** (рис. 21.5).

Группа **Dimensions** из полей ввода **Width** и **Height** служит для задания соответственно ширины и высоты результирующего изображения в пикселах. Если включен флажок **Match movie**, эти размеры совпадают с размерами рабочего листа фильма.

Если в группе **Playback** включен переключатель **Static**, Flash создаст статичное изображение в формате GIF, которое может быть использовано как изображение-замена. Если же включен переключатель **Animated**, Flash создаст "анимированный GIF"; в этом формате можно публиковать простые фильмы без интерактивности.

При включенном переключателе **Animated** становится доступной еще одна группа переключателей, задающая параметры "зацикливания" фильма. При включении переключателя **Loop continuously** фильм будет воспроизводиться непрерывно. Если же включен переключатель **Repeat times**, количество повторений фильма можно будет задать в соответствующем поле ввода.

> ***НА ЗАМЕТКУ***
>
> По умолчанию в формат "анимированный GIF" экспортируется весь фильм. Однако мы можем сохранить в этом формате только определенный фрагмент фильма. Для этого достаточно задать для начального и конечного кадров нужного фрагмента имена `#First` и `#Last` соответственно.

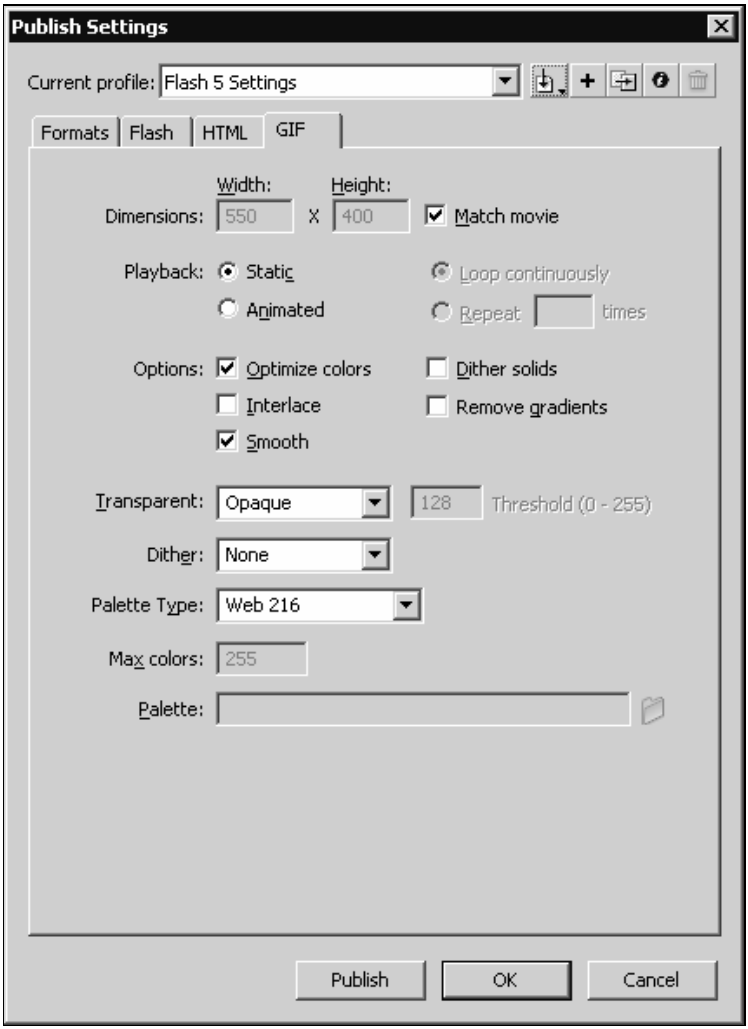


Рис. 21.5. Диалоговое окно **Publish Settings** (вкладка **GIF**)

Группа флажков **Options** управляет параметрами оптимизации изображения в формате GIF, которую проводит при его сохранении Flash. Рассмотрим их по очереди.

Флажок **Optimize colors** включает или отключает оптимизацию палитры изображения. Если он включен, Flash при сохранении удаляет неиспользуемые цвета. Это позволяет уменьшить размер результирующего файла на 1000—1500 байт, однако требует для обработки изображения больше оперативной памяти. Поскольку современные компьютеры имеют большой объем памя-

ти, а каналы доступа в Интернете все еще остаются низкоскоростными, лучше держать этот флажок включенным.

Флажок **Interlace**, будучи включен, вызывает создание изображения в формате GIF с чередованием строк. Такой файл отображается Web-обозревателем по частям, постепенно "проявляясь" в процессе загрузки. Если создаваемое изображение предназначено для публикации в Интернете, лучше включить этот флажок.

Флажок **Smooth** включает или отключает сглаживание контуров создаваемого изображения. Сглаженные изображения выглядят лучше, но в некоторых случаях сглаживание приводит к ухудшению качества; также увеличивается размер файла. Чтобы решить, стоит ли использовать в изображении сглаживание, лучше всего поэкспериментировать, опубликовав данное изображение дважды — с включенным и отключенным сглаживанием. После чего останется сравнить эти изображения и выбрать наилучшее.

Флажок **Dither solids** включает или отключает использование *составных цветов*. При создании изображения в формате GIF может возникнуть такая ситуация, когда для сохранения цвета какого-либо его фрагмента не хватит позиции в палитре. В этом случае Flash либо подберет цвет из уже существующих, что может вызвать искажение цветов, либо составит нужный цвет, смешав пикселы разных цветов. Такие цвета, составленные из пикселов других цветов, и называются составными.

Использование составных цветов передает цвета изображения более-менее точно, но вызывает увеличение размера файла. Чтобы выяснить, идет ли использование составных цветов на пользу нашему изображению, также придется поэкспериментировать.

Флажок **Remove gradients** включает или отключает удаление градиентов в результирующем изображении. Градиенты в изображении формата GIF, как правило, выглядят очень грубо, к тому же они увеличивают размер файла. Если установить флажок **Remove gradients**, Flash преобразует градиентные цвета в сплошные и закрасит их в ключевой цвет градиента. Опять же, на вопрос, включать или не включать этот флажок, ответит только эксперимент.

Раскрывающийся список **Transparent** позволяет сделать цвет фона рабочего листа прозрачным. Этот список имеет три пункта:

- **Opaque** — непрозрачный фон;
- **Transparent** — прозрачный фон;
- **Alpha** — фон регулируемой прозрачности. Степень прозрачности задается в поле ввода **Threshold**, расположенном правее списка **Transparent**; для ввода доступны величины от 0 (полная прозрачность) до 255 (полная непрозрачность).

Раскрывающийся список **Dither** позволяет выбрать способ создания составных цветов для градиентов. Он имеет три пункта:

- **None** — составные цвета вообще не используются, для отображения градиентов берутся подходящие цвета из палитры;
- **Ordered** — создаются составные цвета приемлемого качества, размер результирующего файла слегка увеличивается;
- **Diffusion** — создаются высококачественные составные цвета, размер результирующего файла увеличивается сильнее.

Пункт **Diffusion** работает только тогда, когда для файла задана безопасная палитра цветов Web.

Раскрывающийся список **Palette Type** служит для выбора цветовой палитры, которая будет создана для GIF-файла. В нем доступны четыре пункта.

Пункт **Web 216** включает использование безопасной палитры цветов Web. В этом случае результирующий файл имеет небольшой размер и быстро обрабатывается. Однако возможно искажение цветов графики за счет того, что цвета безопасной палитры могут не полностью совпадать с цветами, используемыми в изображении. Стóит выбирать этот пункт, если изображение предназначается для компьютеров, отображающих максимум 256 цветов.

Пункт **Adaptive** включает использование *адаптивной палитры*, содержащей только цвета, реально используемые в изображении. Если выбран этот пункт, становится доступно поле ввода **Max colors**, где задается максимальное количество доступных в палитре цветов.

При использовании адаптивной палитры не происходит никакого искажения цветов, однако размер файла и время обработки могут вырасти. Этот пункт стоит выбрать также, если изображение будет отображаться на компьютерах с современными мощными видеоподсистемами, одновременно выводящими миллионы цветов.

Пункт **Web Snap Adaptive** аналогичен пункту **Adaptive**, за тем исключением, что при любой возможности будут использоваться цвета из безопасной палитры Web. Благодаря этому размер файла уменьшится, но, возможно, появятся искажения цветов. Этот пункт — компромисс между **Web 216** и **Adaptive**.

Пункт **Custom** позволяет нам самим выбрать палитру цветов. После выбора этого пункта становится доступно поле ввода **Palette**, где мы сможем ввести имя нужного файла палитры. Также можно щелкнуть на расположенной правее поля ввода кнопке в виде папки и выбрать этот файл в появившемся на экране стандартном диалоговом окне открытия файла Windows. Поддерживается импорт палитр из файлов цветовых таблиц (расширение act) и различных графических файлов.

Вкладка *JPEG*

Если на вкладке **Formats** диалогового окна **Publish Settings** был включен флажок **JPEG (.jpg)**, становится доступной вкладка **JPEG** (рис. 21.6).

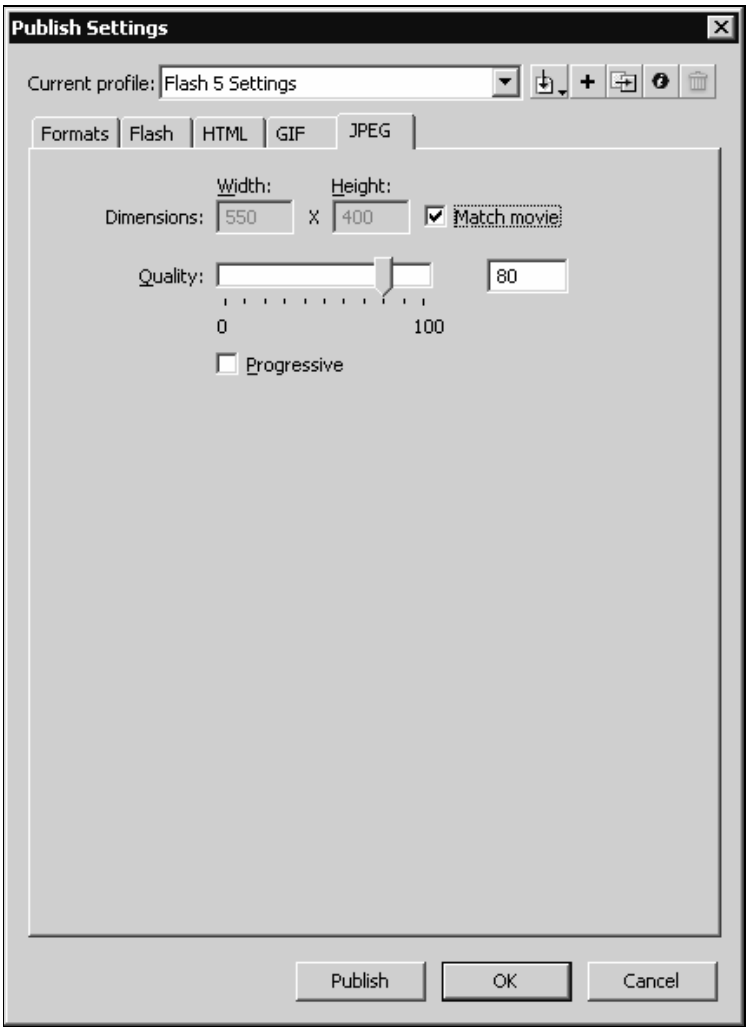


Рис. 21.6. Диалоговое окно **Publish Settings** (вкладка **JPEG**)

Группа **Dimensions** из полей ввода **Width** и **Height** служит для задания соответственно ширины и высоты результирующего изображения в пикселах.

Если включен флажок **Match movie**, размеры изображения совпадают с размерами рабочего листа.

Регулятор **Quality** позволяет задать уровень качества результирующего изображения в интервале значений от 0 до 100. Чем больше это значение, тем лучше выглядит изображение и тем больше размер его файла. А в поле ввода, расположенном правее регулятора, можно ввести значение уровня качества вручную.

Флажок **Progressive**, будучи установлен, вызывает создание изображения в формате JPEG с чередованием строк. Такое изображение отображается Web-обозревателем, постепенно "проявляясь" в процессе загрузки. Если создаваемое изображение предназначено для публикации в Интернете, лучше включить этот флажок.

Вкладка *PNG*

Если на вкладке **Formats** диалогового окна **Publish Settings** был включен флажок **PNG (.png)**, становится доступной вкладка **PNG** (рис. 21.7).

Группа **Dimensions** из полей ввода **Width** и **Height** служит для задания соответственно ширины и высоты результирующего изображения в пикселах. Если включен флажок **Match movie**, размеры изображения совпадают с размерами рабочего листа фильма.

Раскрывающийся список **Bit depth** позволяет задать количество бит, отводимых под информацию о цвете, иначе говоря, цветность изображения. В списке доступны для выбора три пункта:

- **8-bit** — восьмибитный цвет; 256 доступных цветов;
- **24-bit** — 24-битный цвет; 16,7 миллионов доступных цветов (цветность TrueColor);
- **24-bit with Alpha** — 24-битный цвет с каналом прозрачности (альфа-каналом). Этот пункт выбран по умолчанию.

Группа флажков **Options** управляет параметрами оптимизации изображения в формате PNG, которую проводит при его сохранении Flash. Давайте рассмотрим их по порядку.

Флажок **Optimize colors** включает или отключает оптимизацию палитры изображения. Если он установлен, Flash при сохранении удалит неиспользуемые цвета. Это позволяет уменьшить размер результирующего файла на 1000—1500 байт, однако требует для обработки изображения больше оперативной памяти. Поскольку современные компьютеры имеют большой объем памяти, а каналы доступа в Интернет все еще низкоскоростные, лучше держать этот флажок включенным.

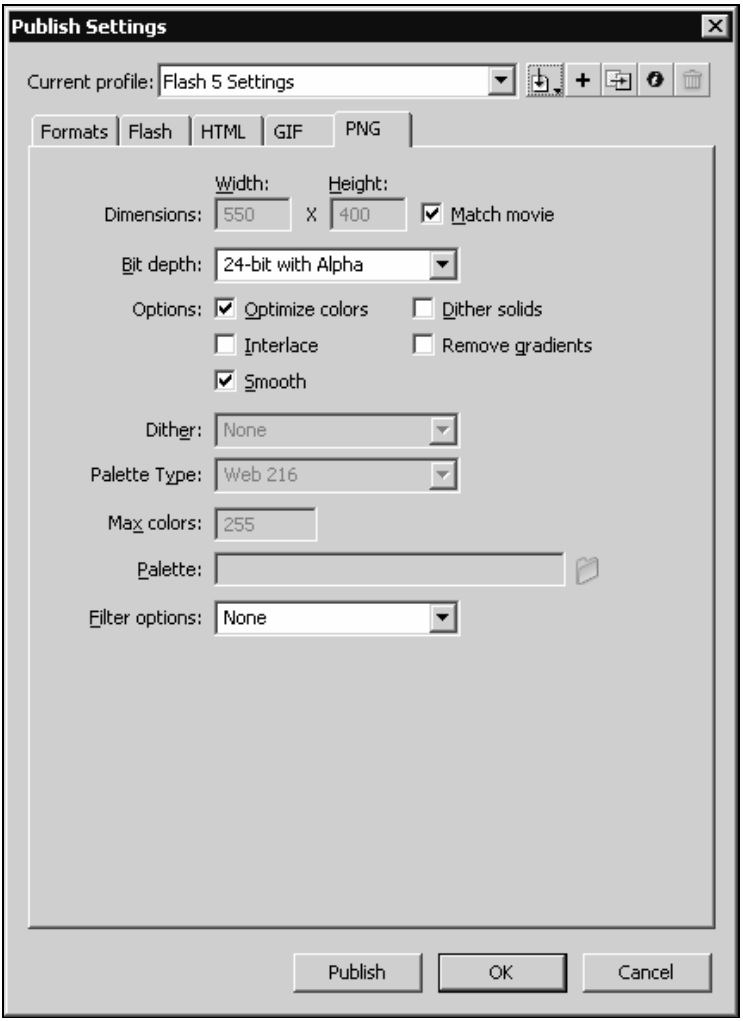


Рис. 21.7. Диалоговое окно **Publish Settings** (вкладка **PNG**)

Флажок **Interlace**, будучи включен, вызывает создание изображения в формате PNG с чередованием строк. Такое изображение отображается Web-обозревателем по частям, постепенно "проявляясь" в процессе загрузки. Если создаваемое изображение предназначено для публикации в Интернете, лучше включить этот флажок.

Флажок **Smooth** включает или отключает сглаживание контуров создаваемого изображения. Сглаженные изображения выглядят лучше, но в некоторых случаях при этом возможно ухудшение качества, а также увеличивается

размер файла. Чтобы решить, использовать в изображении сглаживание или нет, лучше всего поэкспериментировать, опубликовав данное изображение дважды — с включенным и отключенным сглаживанием. После чего останется сравнить эти изображения и выбрать наилучшее.

Флажок **Dither solids** включает или отключает использование составных цветов. При создании изображения PNG может возникнуть такая ситуация, когда для сохранения цвета какого-либо его фрагмента не хватит позиции в палитре. В этом случае Flash может либо подобрать цвет из уже существующих, что может вызвать искажение цветов, либо составить нужный цвет, смешав пикселы разных цветов. Такие цвета, составленные из пикселов других цветов, и называются составными.

Использование составных цветов передает цвета изображения более-менее точно, но вызывает увеличение размера файла. Чтобы выяснить, идет ли использование составных цветов на пользу нашему изображению, также придется поэкспериментировать.

Флажок **Remove gradients** включает или отключает удаление градиентов в результирующем изображении. Градиенты в изображении формата PNG, как правило, выглядят очень грубо, к тому же они увеличивают размер файла. Если установить флажок, Flash преобразует градиентные цвета в сплошные и закрасит их ключевым цветом градиента. Опять же, на вопрос, нужно ли включать этот флажок, ответит только эксперимент.

Если в раскрывающемся списке **Bit depth** был выбран пункт **8-bit**, становятся доступными элементы управления, расположенные ниже группы флажков **Options**. Давайте их рассмотрим.

Раскрывающийся список **Dither** позволяет выбрать способ создания составных цветов для градиентов. В списке три пункта:

- **None** — составные цвета вообще не используются, для отображения градиентов берутся подходящие цвета из палитры;
- **Ordered** — создаются составные цвета приемлемого качества, размер результирующего файла слегка увеличивается;
- **Diffusion** — создаются высококачественные составные цвета, размер результирующего файла увеличивается еще сильнее. Пункт работает только тогда, когда для файла задана безопасная палитра цветов Web.

Раскрывающийся список **Palette Type** служит для выбора палитры. В нем доступны следующие четыре пункта:

- **Web 216** — используется безопасная палитра цветов Web;
- **Adaptive** — используется адаптивная палитра цветов. Если выбран этот пункт списка, становится доступно поле ввода **Max colors**, в котором задается максимальное количество доступных в палитре цветов;

- **Web Snap Adaptive** — аналогичен пункту **Adaptive**, за тем исключением, что при случае будут использоваться цвета из безопасной палитры Web;
- **Custom** — позволяет нам самим выбрать палитру цветов. После выбора этого пункта становится доступно поле ввода **Palette**, в котором можно ввести имя нужного файла палитры. Поддерживается импорт из файлов цветовых таблиц (файлы с расширением act) и различных графических файлов.

С помощью раскрывающегося списка **Filter** можно задать способ дополнительной обработки массива данных изображения, позволяющий увеличить степень его сжатия. Здесь доступны шесть пунктов:

- **None** — дополнительная обработка не выполняется;
- **Sub** — в массив данных записываются значения разности между цветом текущего и предшествующего ему пикселов;
- **Up** — в массив данных записываются значения разности между цветом текущего и последующего пикселов;
- **Average** — среднее значение цветов двух соседних пикселов используется для определения цвета текущего пиксела;
- **Path** — среднее значение цветов трех соседних пикселов используется для определения цвета текущего пиксела;
- **Adaptive** — самый сложный способ обработки массива данных. В сочетании с адаптивной палитрой цветов позволяет сильно уменьшить размер PNG-файла. Рекомендуется к использованию, если изображение будет отображаться на компьютерах с современными мощными видеоподсистемами.

Вкладка *QuickTime*

Если на вкладке **Formats** диалогового окна **Publish Settings** был включен флажок **QuickTime (.mov)**, становится доступной вкладка **QuickTime** (рис. 21.8).

> ***ВНИМАНИЕ!***
>
> Чтобы получить возможность опубликовать фильм Flash в формате QuickTime, нужно задать на вкладке **Flash** диалогового окна **Publish Settings** (см. рис. 21.2) версию проигрывателя Flash от первой до пятой. Это выполняется с помощью раскрывающегося списка **Version** данной вкладки.

Группа **Dimensions** из полей ввода **Width** и **Height** служит для задания соответственно ширины и высоты результирующего изображения в пикселах. Если включен флажок **Match movie**, размеры изображения совпадают с размерами рабочего листа.

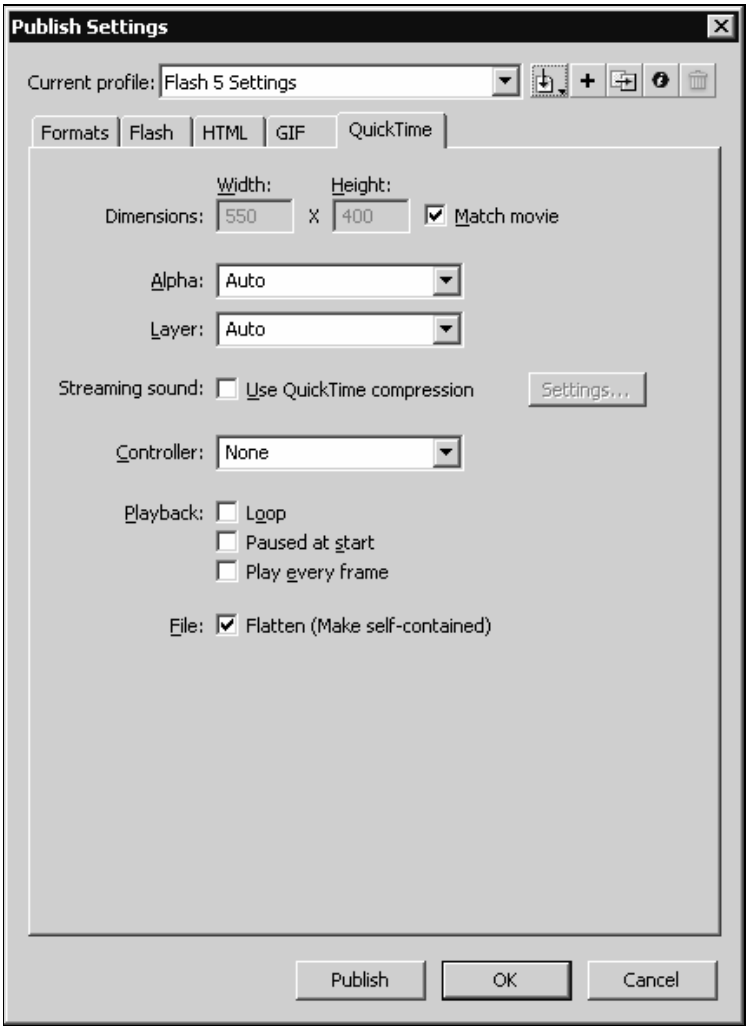


Рис. 21.8. Диалоговое окно **Publish Settings** (вкладка **QuickTime**)

С помощью раскрывающегося списка **Alpha** можно сделать фон слоя с фильмом Flash прозрачным. (Как мы уже знаем, файл в формате QuickTime может содержать несколько дорожек анимации, каждую — на своем слое.) Этот список содержит три пункта:

- **Alpha-transparent** — слой с фильмом Flash делается прозрачным, так что сквозь него будет видно содержимое остальных слоев;
- **Copy** — слой с фильмом Flash делается непрозрачным, и содержимое остальных слоев сквозь него видно не будет;

- **Auto** — если слои с графикой Flash находятся выше слоев с импортированными клипами формата QuickTime, то они делаются прозрачными, если ниже — непрозрачными. Этот пункт выбран по умолчанию.

Раскрывающий список **Layer** задает местоположение слоя, содержащего фильм Flash, относительно других слоев. Он содержит три пункта:

- **Top** — слой с фильмом Flash будет всегда помещаться наверху, выше всех остальных слоев;
- **Bottom** — слой с фильмом Flash будет всегда помещаться внизу, ниже всех остальных слоев;
- **Auto** — местоположение слоя с фильмом Flash в файле формата QuickTime соответствует его местоположению в документе Flash. Этот пункт выбран по умолчанию.

Флажок **Use QuickTime compression** включает или отключает сохранение звукового сопровождения фильма в формате QuickTime. Вероятно, его стоит всегда держать включенным, чтобы у пользователей не возникло проблем с воспроизведением звука.

Если флажок включен **Use QuickTime compression**, становится доступной кнопка **Settings**. При ее нажатии на экране появляется диалоговое окно **Sound Settings** (рис. 21.9).

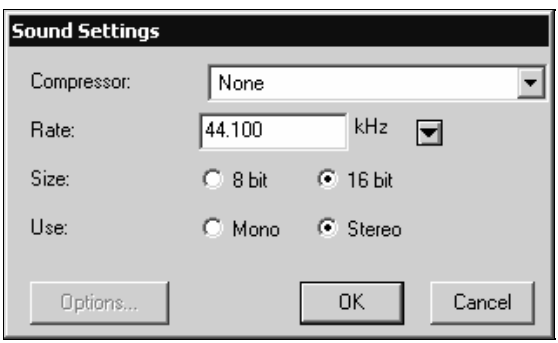


Рис. 21.9. Диалоговое окно **Sound Settings**

Раскрывающийся список **Compressor** служит для задания алгоритма сжатия. Формат QuickTime поддерживает множество алгоритмов сжатия звука; их описание можно найти в документации по этому формату.

При выборе некоторых пунктов в списке становится доступной кнопка **Options**. При нажатии этой кнопки на экране появится диалоговое окно задания параметров выбранного алгоритма (точнее, кодека) сжатия звука.

В поле ввода **Rate** вводится частота оцифровки звука. Также можно нажать расположенную правее этого поля ввода небольшую кнопку со стрелкой, направленной вниз, и выбрать одно из значений частоты оцифровки в появившемся на экране меню.

Группа переключателей **Size** устанавливает разрядность звука. Переключатель **8 bit** задает восьмиразрядный звук, а переключатель **16 bit** — шестнадцатиразрядный.

Группа переключателей **Use** задает количество каналов звука. Можно выбрать монофонический (переключатель **Mono**) и стереофонический (переключатель **Stereo**) звук.

После задания нужных параметров звука нужно нажать кнопку **OK** для их сохранения и кнопку **Cancel** для отказа от них.

Вернемся к окну **Publish Settings**. Раскрывающийся список **Controller** позволяет выбрать набор элементов управления воспроизведением, который будет использован при просмотре видеоклипа в формате QuickTime. Этот список содержит три пункта:

- **None** — элементы управления воспроизведением отсутствуют (значение по умолчанию);
- **Standard** — стандартный набор элементов управления;
- **QuickTime VR** — набор элементов управления, реализующий функции "виртуальной реальности".

С помощью группы флажков **Playback** можно управлять процессом воспроизведения фильма. В группе три флажка.

- Флажок **Loop** включает или отключает "зацикливание" фильма.
- Если установлен флажок **Paused at start**, фильм не будет запущен на воспроизведение сразу же после загрузки. Вместо этого пользователю самому придется запускать этот фильм. Если же флажок снят (а он снят по умолчанию), фильм начнет воспроизводиться сразу же после загрузки.
- Если установлен флажок **Play every frame**, проигрыватель QuickTime будет воспроизводить каждый кадр фильма, не пропуская их. Звуковое сопровождение в этом случае воспроизводиться не будет.

Если установлен единственный флажок **Flatten (Make self-contained)** группы **File**, Flash включит все импортированные файлы в один результирующий файл формата QuickTime. Если флажок снят, то в результирующий файл будут помещены только ссылки на соответствующие внешние файлы.

Предварительный просмотр публикуемой графики

Flash предоставляет возможность просмотреть подготовленный файл перед его публикацией. Для этого достаточно выбрать пункт **Publish Preview** меню **File**. На экране появится подменю **Publish Preview** с пунктами, соответствующими файлам из состава публикации. Если выбрать один из этих пунктов, Flash сохранит наше изображение в соответствующем формате и тут же откроет его в программе, зарегистрированной для данного типа файлов. (Для файлов Shockwave/Flash такой программой будет сам Flash, а для файлов форматов GIF, JPEG, PNG и Web-страниц — Web-обозреватель.) После закрытия этой программы Flash удалит созданный им файл.

Один из пунктов подменю **Publish Preview** меню **File** помечен как пункт по умолчанию и вызывается при нажатии клавиши <F12>. Если при публикации фильма создается Web-страница, то этому пункту будет соответствовать она, в противном случае — тот файл, что выбран в диалоговом окне **Publish Settings** (см. рис. 21.1) первым, если считать сверху вниз.

Создание карты-изображения

Карта-изображение — это особое графическое изображение, разбитое на независимые *области*, к каждой из которых привязана своя гиперссылка. Когда пользователь щелкает мышью внутри такой области, происходит переход по интернет-адресу, указанному в соответствующей гиперссылке.

Чтобы создать карту-изображение, нужно иметь:

- само графическое изображение;
- список, описывающий геометрическую форму и размеры каждой из составляющих карту-изображение областей, а также привязанные к ним гиперссылки. Этот список называется списком областей.

Для создания карт-изображений используется такой код HTML:

```
<MAP NAME="mymap">
  <AREA COORDS="130,116,214,182" HREF="http://www.macromedia.com/flash/">
  <AREA COORDS="180,244,252,300"
  ↳HREF="http://www.macromedia.com/dreamweaver/">
  <AREA COORDS="280,34,376,52"
  ↳HREF="http://www.macromedia.com/fireworks/">
</MAP>
<IMG SRC="mymap.gif" USEMAP="#mymap" WIDTH=550 HEIGHT=400 BORDER=0>
```

Описание списка областей находится внутри парного тега `<MAP>`. Графическое изображение задается, как обычно, тегом `<IMG>`, а для привязки к нему списка областей используется атрибут `USEMAP` этого тега. Описание списка областей в коде HTML должно всегда предшествовать тегу, задающему изображение.

Обычно карты-изображения создаются с помощью специальных программ, позволяющих нарисовать нужные области прямо на изображении. После этого остается задать для каждой области гиперссылку и вставить сформированный утилитой код HTML в Web-страницу. Вручную же создавать карты-изображения очень хлопотно.

Но если у нас есть пакет Macromedia Flash, никакие дополнительные программы нам не нужны.

Чтобы создать в среде Flash карту-изображение, сначала создадим новый документ. В единственном ключевом кадре анимации нарисуем нужное нам изображение и присвоим этому кадру имя `#Static`. (Если этого не сделать, Flash создаст изображение на основе текущего кадра.)

Нарисовав изображение, создадим еще один ключевой кадр. В этом кадре создадим набор кнопок, местоположение и форма которых должны в точности соответствовать нужным нам областям карты-изображения. К каждой из этих кнопок привяжем сценарий, обрабатывающий событие `press` или `release` и содержащий действие `getURL`. Например, сценарий может быть таким:

```
on (release) {
  getURL("http://www.macromedia.com");
}
```

Присвоим кадру, содержащему набор кнопок, имя `#Map`. (Если этого не сделать, Flash для создания областей использует последний кадр фильма.)

Теперь остается опубликовать Web-страницу с картой-изображением. Откроем диалоговое окно **Publish Settings** и включим в состав публикации файл HTML и файл нужного графического формата (GIF, JPEG или PNG), который и будет изображением. Нужно обязательно проверить, не содержат ли имена этих файлов русские буквы (что недопустимо).

Теперь осталось перейти на вкладку **HTML** и выбрать в раскрывающемся списке **Template** пункт **Image Map**. И после задания других параметров публикации можно нажать кнопку **Publish**. Все — Web-страница с картой-изображением готова!

Использование профилей публикации

Часто бывает нужно публиковать один и тот же фильм Flash с разными параметрами, например, с изображением-заменой или без него. В старых вер-

сиях Flash для того, чтобы сделать это, приходилось для каждой такой публикации вручную менять параметры. Flash MX 2004 и Flash 8 избавляют нас от этого труда, предоставляя возможность создания профилей публикации.

Профиль публикации — это набор параметров публикации, сохраненный в файле документа Flash под определенным именем. В нашем случае мы можем создать два профиля для каждого набора параметров и сохранить их под разными именами. Вообще же документ Flash может содержать сколько угодно профилей публикации.

При создании нового документа Flash создает в нем профиль по имени `Flash 5 Settings`. Если фильм Flash всегда будет публиковаться с одними и теми же параметрами, можно не создавать новых профилей, а использовать этот профиль.

Итак, чтобы создать новый профиль, нажмем кнопку со знаком "плюс", расположенную в верхней части окна **Publish Settings** (см. рис. 21.1). На экране появится диалоговое окно **Create New Profile** (рис. 21.10). В единственное поле ввода этого окна введем имя создаваемого профиля и нажмем кнопку **OK**. После этого новый профиль будет создан.

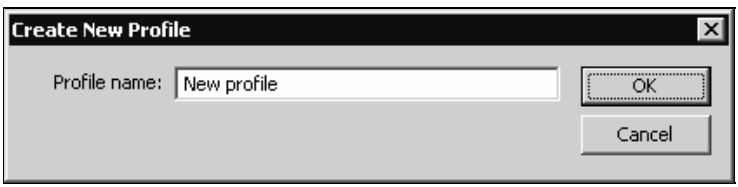


Рис. 21.10. Диалоговое окно **Create New Profile**

Вновь созданный профиль содержит значения параметров по умолчанию. Поэтому сразу же после создания профиля его следует "наполнить" необходимыми настройками. Для этого нужно:

1. Проверить, выбран ли созданный профиль в раскрывающемся списке **Current profile**, находящемся в верхней части диалогового окна **Publish Settings** (см. рис. 21.1).
2. Задать в окне **Publish Settings** параметры публикации, которые должны быть сохранены в этом профиле.
3. Нажать кнопку **OK** этого диалогового окна или переключиться на другой профиль (тот же самый `Flash 5 Settings`, созданный самим Flash при создании данного документа). Это нужно для того, чтобы сохранить заданные параметры в созданном нами профиле.

В раскрывающемся списке **Current profile** перечислены все созданные на данный момент профили. Чтобы переключиться на другой профиль, нужно просто выбрать в этом списке соответствующий ему пункт.

Для правки параметров какого-либо профиля мы сначала переключаемся на него. Далее изменяем в окне **Publish Settings** необходимые параметры и либо нажимаем кнопку **OK**, либо переключаемся на другой профиль.

Кнопка , расположенная справа от списка **Current profile** диалогового окна **Publish Settings** (третья в группе кнопок), предоставляет возможность продублировать выбранный в данный момент профиль, создав его точную копию. После ее нажатия на экране появится диалоговое окно **Duplicate Profile**, похожее на окно **Create New Profile**. В единственное поле ввода этого диалогового окна нужно будет ввести имя создаваемой копии выбранного профиля и нажать кнопку **OK**.

Для удаления ненужного профиля достаточно выбрать его в списке **Current profile** диалогового окна **Publish Settings** и нажать кнопку (последнюю в группе кнопок справа от списка). На экране появится окно-предупреждение, спрашивающее, действительно ли требуется удалить выбранный профиль; нажмем кнопку **OK** для его удаления и **Отмена** (**Cancel**) для отказа от этого.

ВНИМАНИЕ!

Если профиль публикации всего один, удалить его невозможно.

Если нужно изменить имя профиля, мы выберем его и нажмем кнопку (предпоследнюю в группе кнопок справа от списка **Current profile**). На экране появится диалоговое окно **Profile Properties**, похожее на окно **Create New Profile**. В единственное поле ввода этого окна введем новое имя профиля и нажмем кнопку **OK**. Таким способом можно, кстати, изменить имя профиля `Flash 5 Settings`.

Если нужно перенести какой-либо профиль в другой документ, используется следующая последовательность действий:

1. В раскрывающемся списке **Current profile** диалогового окна **Publish Settings** выбираем профиль, который нужно перенести.
2. Нажимаем кнопку и в появившемся на экране небольшом меню выбираем пункт **Export**.
3. В появившемся на экране стандартном диалоговом окне сохранения файла Windows вводим имя файла и нажимаем кнопку сохранения. После этого Flash выполнит экспорт профиля, в результате которого на диске будет создан *файл профиля публикации*, содержащий данные в формате XML и имеющий "говорящее" расширение xml.

4. Открываем документ Flash, в который нужно перенести этот профиль, и вызываем на экран диалоговое окно **Publish Settings**.
5. Вторично нажимаем кнопку и в ее меню выбираем пункт **Import**.
6. В появившемся на экране стандартном диалоговом окне открытия файла Windows выбираем нужный файл профиля и нажимаем кнопку открытия. После этого Flash импортирует профиль в активный документ.

Экспорт

Разобравшись с публикацией, перейдем к экспорту.

Как мы помним, экспорт фильма или изображения Flash отличается от его публикации тем, что файл Flash просто преобразуется в другой, выбранный нами формат. Получившийся файл можно распространять как через Интернет (только соответствующую Web-страницу нам придется делать самим), так и по электронной почте; можно также использовать его в дальнейшей работе, например, в качестве части другого изображения.

Форматы экспорта, поддерживаемые Flash

Все форматы экспорта графики, поддерживаемые Flash, приведены в табл. 21.1.

Таблица 21.1. *Форматы экспорта графики, поддерживаемые Flash*

Название формата	Расширение файлов
Adobe Illustrator	ai
Apple QuickTime	mov
AutoDesk AutoCAD	dxf
AVI	avi
Encapsulated PostScript	eps
GIF (обычный и анимированный)	gif
JPEG	jpg, jpe, jpeg
Macromedia Flash Video	flv
Macromedia Shockwave/Flash	swf
PNG	png

Таблица 21.1 *(окончание)*

Название формата	Расширение файлов
WAV (только звук)	wav
Метафайлы Windows	wmf
Растровые файлы Windows	bmp
Расширенные метафайлы Windows	emf

Из этой таблицы видно, что Flash может экспортировать графику как в векторные, так и в растровые форматы. В последнем случае вся векторная графика растрируется.

Интересная особенность Flash: он может экспортировать каждый кадр фильма в отдельный графический файл. При этом кадры могут сохраняться в файлах любого формата: статичный GIF, JPEG, PNG, BMP, Adobe Illustrator и др. Эти файлы получают имена вида <*Имя, заданное при экспорте*><*Номер кадра*>.<*Расширение, соответствующее формату экспорта*>. Такие наборы кадров могут пригодиться, например, для создания слайд-шоу.

Как экспортировать изображение или фильм

Проще всего выполнить экспорт фильма Flash. Для этого откроем нужный документ и выберем пункт **Export Movie** подменю **Export** меню **File** или нажмем комбинацию клавиш <Ctrl>+<Alt>+<Shift>+<S>. На экране появится стандартное диалоговое окно сохранения файла Windows. В раскрывающемся списке, расположенном в нижней части этого окна, выберем формат результирующего файла, зададим имя файла в поле ввода и нажмем кнопку сохранения.

Экспорт содержимого какого-либо кадра в виде статичного изображения выполняется чуть сложнее. Мы открываем документ, делаем нужный нам кадр текущим и в подменю **Export** меню **File** выбираем пункт **Export Image**. После этого нам останется в стандартном диалоговом окне сохранения файла Windows выбрать формат результирующего файла, задать его имя и нажать кнопку сохранения.

В некоторых случаях после нажатия кнопки сохранения Flash просто сохранит готовый файл на диске, не задавая нам лишних вопросов. Но в большинстве случаев на экране появится диалоговое окно, где нам будет предложено задать некоторые параметры сохраняемого файла. Файл будет создан

после нажатия кнопки **OK** этого окна. А кнопка **Cancel**, как всегда, отменяет операцию экспорта.

Параметры форматов экспорта

В этом разделе описываются параметры различных графических форматов, в которые Flash может экспортировать наше изображение, а также диалоговые окна, позволяющие настраивать эти параметры. Эти окна появляются после того, как мы зададим имя результирующего файла в окне сохранения.

Adobe Illustrator

Flash поддерживает создание файлов версий 88, 3, 5, 6 и 8—10 формата Adobe Illustrator. Параметры этого формата задаются в диалоговом окне **Export Adobe Illustrator** (рис. 21.11). Их не очень много.

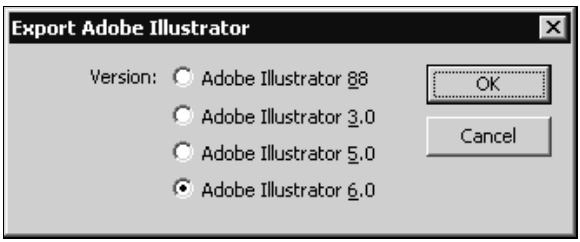


Рис. 21.11. Диалоговое окно **Export Adobe Illustrator**

Почти все это окно занимает группа переключателей **Version**. В ней четыре переключателя: **Adobe Illustrator 88**, **Adobe Illustrator 3.0**, **Adobe Illustrator 5.0** и **Adobe Illustrator 6.0**. Нет нужды объяснять их назначение. Однако очень странно, что среди них нет переключателя, задающего экспорт в формат **Adobe Illustrator 8.0** и более новых версий. Неужели это очередная ошибка в прекрасном пакете?

Apple QuickTime

Чтобы иметь возможность экспорта в этот формат, нам нужно будет установить на компьютер пакет QuickTime соответствующей версии.

Параметры этого формата задаются в диалоговом окне **Export QuickTime** (рис. 21.12). Все они были описаны ранее в этой главе, в параграфе, посвященном заданию параметров файла QuickTime при публикации.

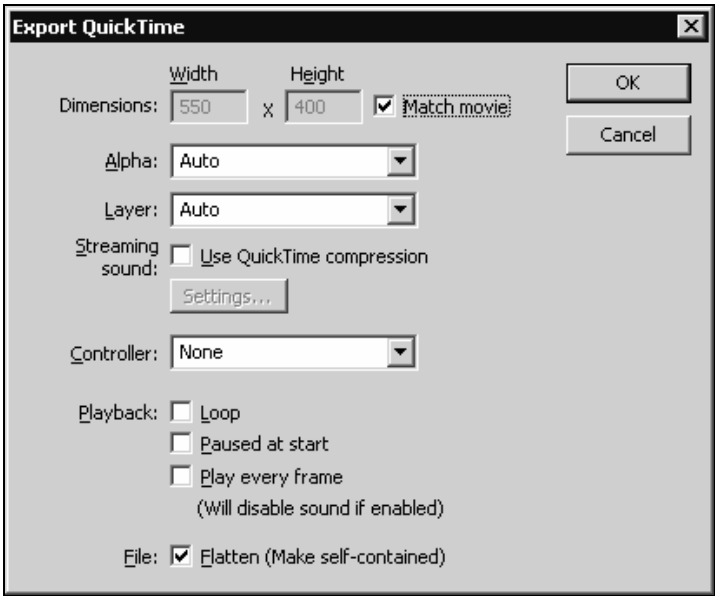


Рис. 21.12. Диалоговое окно **Export QuickTime**

AutoDesk AutoCAD

Flash позволяет экспортировать изображения в формате AutoCAD версии 10 или более поздней. Никаких дополнительных параметров при этом задавать не нужно.

AVI

Параметры этого формата задаются в диалоговом окне **Export Windows AVI** (рис. 21.13).

Группа **Dimensions** из полей ввода **Width** и **Height** служит для задания соответственно ширины и высоты результирующего фильма в пикселах. Если установить флажок **Maintain aspect ratio**, то при вводе данных в одно поле ввода Flash сам введет в другое поле ввода такое значение, чтобы пропорции изображения сохранились неизменными.

С помощью раскрывающегося списка **Video format** задается цветность фильма. Доступны четыре пункта:

- **8 bit color** — 256-цветное изображение;
- **16 bit color** — изображение с 16-битным цветом (HiColor). Этот пункт выбран по умолчанию;

- **24 bit color** — изображение с 24-битным цветом (TrueColor);
- **32 bit color w/ alpha** — изображение с 24-битным цветом и каналом прозрачности (альфа-канал).

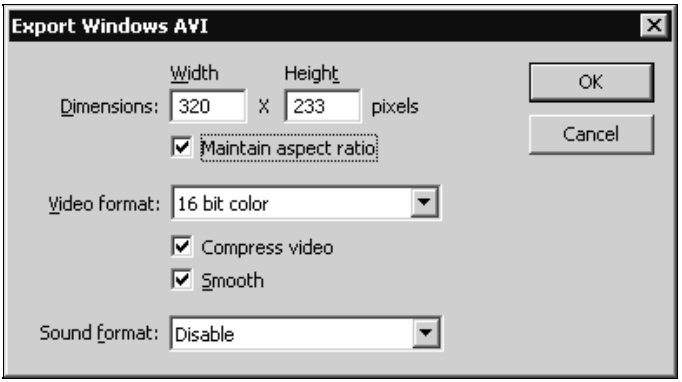


Рис. 21.13. Диалоговое окно **Export Windows AVI**

Если включен флажок **Compress video**, то после нажатия кнопки **OK** Flash выведет на экран стандартное диалоговое окно Windows — окно задания параметров сжатия видео. Это окно показано на рис. 21.14. С помощью раскрывающегося списка **Программа сжатия** выбирается кодек для сжатия видео, а с помощью расположенных ниже элементов управления задаются дополнительные параметры сжатия. После нажатия кнопки **OK** файл в формате AVI будет создан; для отказа от экспорта нужно нажать кнопку **Отмена**.

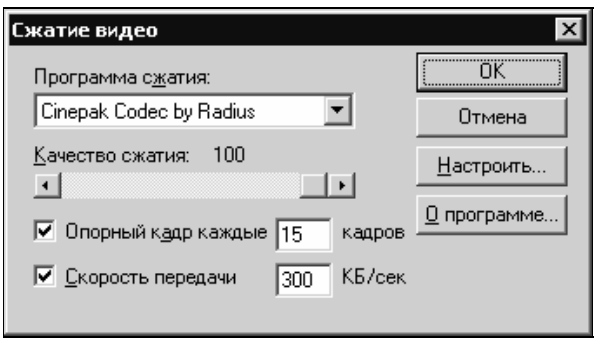


Рис. 21.14. Стандартное диалоговое окно Windows задания параметров сжатия видео

Вернемся к диалоговому окну **Export Windows AVI**.

Если флажок **Compress video** отключен, видео вообще не будет сжиматься, и результирующий AVI-файл может оказаться очень большим. Так что без сжатия следует сохранять только совсем короткие фильмы.

Флажок **Smooth** включает или отключает сглаживание контуров в фильме.

Раскрывающийся список **Sound format** позволяет задать наличие и качество звукового сопровождения фильма. Этот список содержит очень много пунктов, задающих параметры качества звука. Названия этих пунктов имеют вид ***<Частота оцифровки><Количество разрядов оцифровки><Моно или стерео>***. Например, пункт **22kHz 16 Bit Mono** задает шестнадцатиразрядный монофонический звук с частотой оцифровки 22 кГц. А пункт **Disable** позволяет вообще убрать звук из фильма.

GIF

Параметры статичного изображения GIF задаются в диалоговом окне **Export GIF** (рис. 21.15).

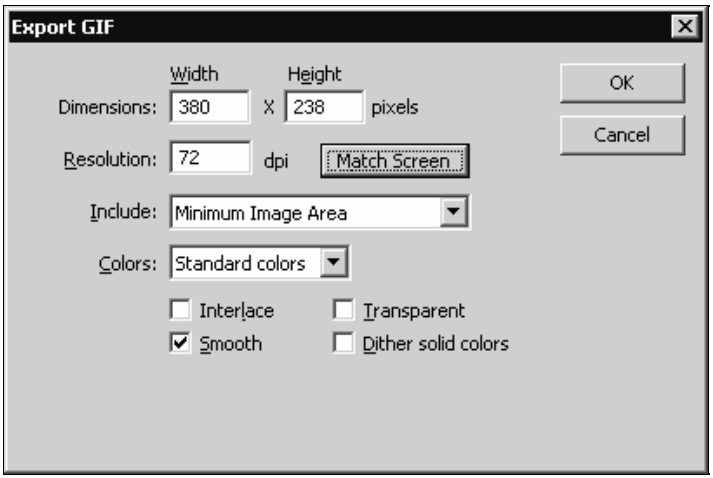


Рис. 21.15. Вид диалогового окна **Export GIF** при экспорте статичного изображения GIF

Группа **Dimensions** из полей ввода **Width** и **Height** служит для задания соответственно ширины и высоты результирующего изображения в пикселах. В поле ввода **Resolution** задается разрешающая способность изображения в точках (пикселах) на дюйм. Если нажать кнопку **Match Screen**, Flash устано-

вит эти параметры сам, основываясь на размерах рабочего листа и заданном разрешении экрана.

С помощью раскрывающегося списка **Include** задаются размеры свободного пространства, которое будет включено в состав изображения. Пункт **Minimum Image Area**, выбранный по умолчанию, предписывает Flash включить в изображение минимум пустого пространства. Если же выбрать пункт **Full Document Size**, то Flash включит в состав изображения все свободное пространство, имеющееся на рабочем листе.

> ***Внимание!***
>
> Раскрывающийся список **Include** присутствует в диалоговом окне **Export GIF** только в том случае, если выполняется экспорт статичного изображения (т. е. содержимого текущего кадра).

Раскрывающийся список **Colors** задает количество цветов в цветовой палитре изображения. Доступно восемь пунктов:

- **4**, **8**, **16**, **32**, **64**, **128**, **256 colors** — соответствующее количество цветов;
- **Standard colors** — безопасная палитра Web. Этот пункт выбран по умолчанию.

Группа флажков, расположенная ниже этого раскрывающегося списка, нам уже знакома. Флажок **Interlace**, будучи включен, вызывает создание изображения в формате GIF с чередованием строк. Флажок **Smooth** включает или отключает сглаживание контуров результирующего изображения. Включение флажка **Transparent** делает цвет фона рабочего листа прозрачным. А флажок **Dither solid colors** включает или отключает использование составных цветов.

"Анимированный GIF"

Параметры этого формата задаются в диалоговом окне **Export GIF** (рис. 21.16), имеющем несколько иной вид, чем тот, что был показан на рис. 21.15.

Бóльшая часть элементов управления этого окна была описана ранее, когда мы говорили об экспорте в обычный, статичный GIF. Здесь добавилось только поле ввода **Animation**, в котором вводится количество повторений фильма. Если нам нужно, чтобы фильм повторялся бесконечно (зациклился), достаточно ввести в это поле ноль.

> ***На заметку***
>
> По умолчанию в формат "анимированный GIF" экспортируется весь фильм. Однако мы можем сохранить в этом формате и один определенный фрагмент фильма. Для этого достаточно задать для начального и конечного кадров нужного фрагмента имена `#First` и `#Last` соответственно.

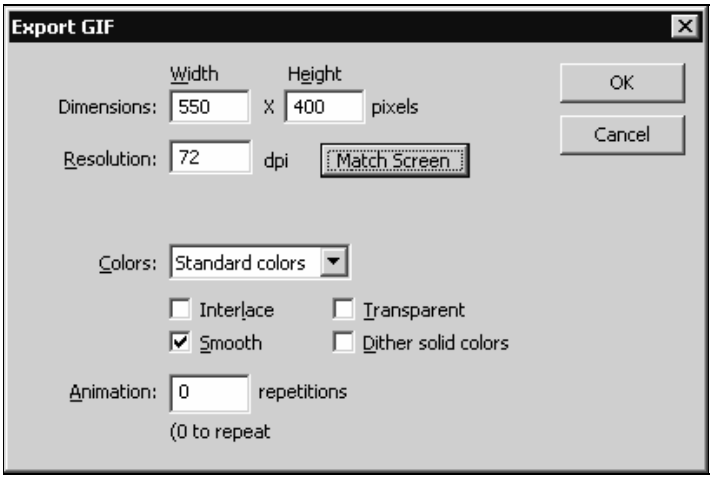


Рис. 21.16. Вид диалогового окна **Export GIF** при экспорте в формат "анимированный GIF"

JPEG

Параметры этого формата задаются в диалоговом окне **Export JPEG** (рис. 21.17).

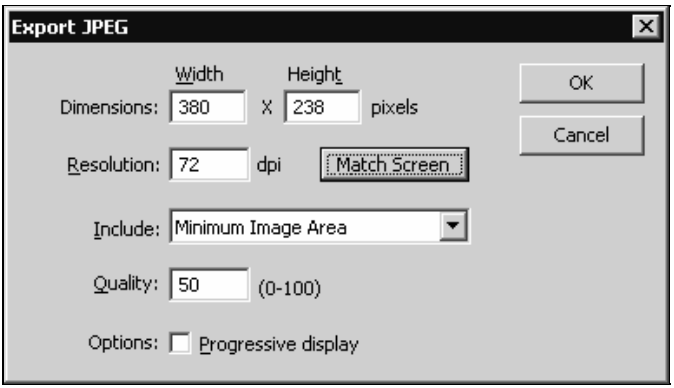


Рис. 21.17. Диалоговое окно **Export JPEG**

Группа **Dimensions** из полей ввода **Width** и **Height** служит для задания соответственно ширины и высоты результирующего изображения в пикселах. В поле ввода **Resolution** задается разрешающая способность изображения в

точках (пикселах) на дюйм. Если нажать кнопку **Match Screen**, Flash установит эти параметры сам, основываясь на размерах рабочего листа и заданном разрешении экрана.

С помощью раскрывающегося списка **Include** задаются размеры свободного пространства, которое будет включено в состав изображения. Пункт **Minimum Image Area**, выбранный по умолчанию, предписывает Flash включить в изображение минимум пустого пространства. Если же выбрать пункт **Full Document Size**, то Flash включит в состав изображения все свободное пространство, имеющееся на рабочем листе.

Поле ввода **Quality** позволяет задать уровень качества результирующего изображения в диапазоне значений от 0 до 100. Чем больше это значение, тем лучше выглядит изображение и тем больше получается его файл.

Флажок **Progressive display**, будучи включен, вызывает создание JPEG-файла с чередованием строк.

Macromedia Flash Video

Экспорт в формат Macromedia Flash Video был описан в *главе 16*.

Macromedia Shockwave/Flash

Параметры этого формата задаются в диалоговом окне **Export Flash Player** (рис. 21.18). Все они нам уже знакомы.

PNG

Параметры этого формата задаются в диалоговом окне **Export PNG** (рис. 21.19).

Группа **Dimensions** из полей ввода **Width** и **Height** служит для задания соответственно ширины и высоты результирующего изображения в пикселах. В поле ввода **Resolution** задается разрешающая способность изображения в точках (пикселах) на дюйм. Если нажать кнопку **Match Screen**, Flash установит эти параметры сам, основываясь на размерах рабочего листа и заданном разрешении экрана.

С помощью раскрывающегося списка **Include** задаются размеры свободного пространства, которое будет включено в состав изображения. Пункт **Minimum Image Area**, выбранный по умолчанию, предписывает Flash включить в изображение минимум пустого пространства. Если же выбрать пункт **Full Document Size**, Flash включит в состав изображения все свободное пространство, имеющееся на рабочем листе.

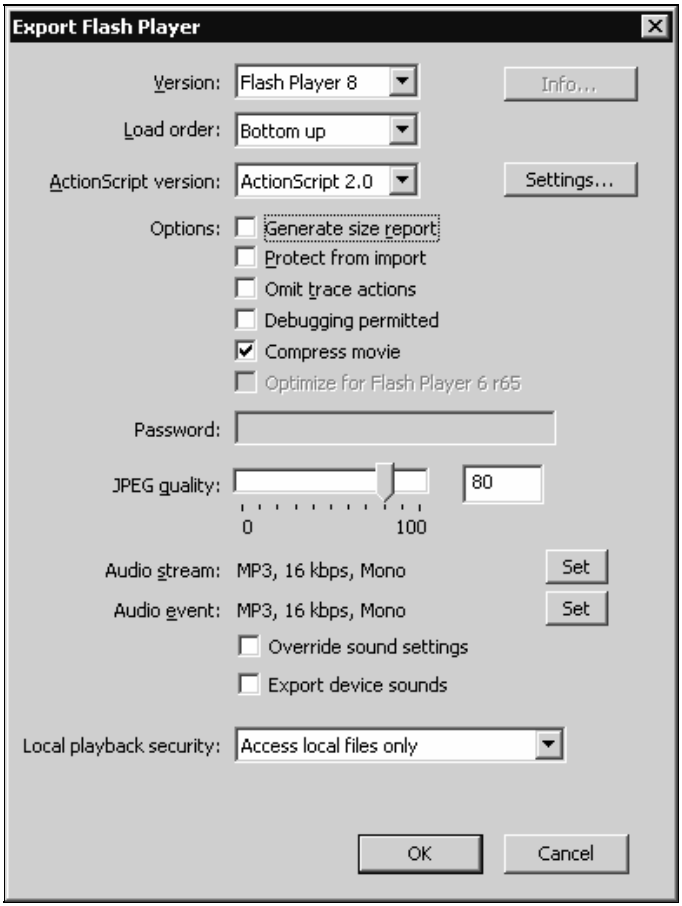


Рис. 21.18. Диалоговое окно **Export Flash Player**

Раскрывающийся список **Colors** служит для задания цветности изображения. В нем доступны три пункта:

- **8 bit** — 256-цветное изображение;
- **24 bit** — изображение с 24-битным цветом (TrueColor);
- **24 bit with alpha channel** — изображение с 24-битным цветом и каналом прозрачности (альфа-канал). Этот пункт выбран по умолчанию.

Раскрывающийся список **Filter** служит для задания способа дополнительной обработки массива данных изображения, позволяющего увеличить степень его сжатия.

Здесь доступны шесть пунктов:

- **None** — дополнительная обработка не выполняется;
- **Sub** — в массив данных записываются значения разности между цветом текущего и предшествующего ему пикселов;
- **Up** — в массив данных записываются значения разности между цветом текущего и последующего пикселов;
- **Average** — среднее значение цветов двух соседних пикселов используется для определения цвета текущего пиксела;
- **Path** — среднее значение цветов трех соседних пикселов используется для определения цвета текущего пиксела;
- **Adaptive** — самый сложный способ обработки массива данных. В сочетании с адаптивной палитрой цветов позволяет сильно уменьшить размер файла PNG.

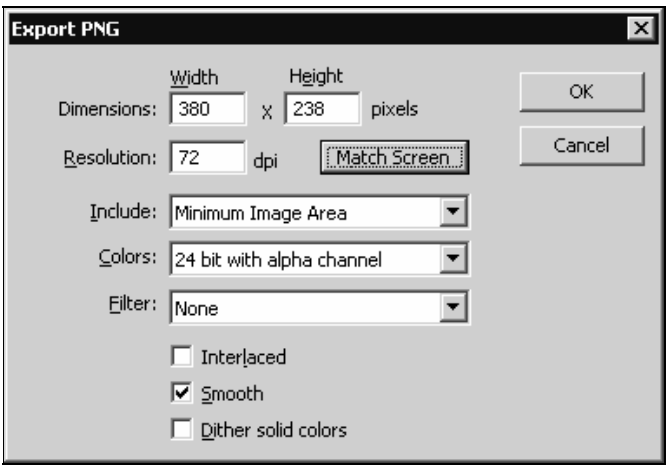


Рис. 21.19. Диалоговое окно **Export PNG**

Флажок **Interlace**, будучи включен, вызывает создание изображения с чередованием строк. Флажок **Smooth** включает или отключает сглаживание контуров результирующего изображения. А флажок **Dither solid colors** включает или отключает использование составных цветов.

WAV

Параметры этого формата задаются в диалоговом окне **Export Windows WAV** (рис. 21.20).

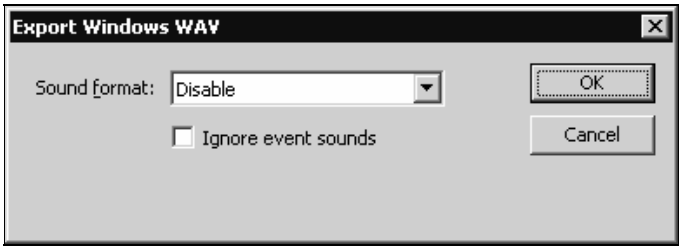


Рис. 21.20. Диалоговое окно **Export Windows WAV**

Раскрывающийся список **Sound format** позволяет указать наличие экспортируемого звука и задать его параметры. Этот список содержит очень много пунктов, названия которых имеют вид *<Частота оцифровки><Количество разрядов оцифровки><Моно или стерео>*. Например, пункт **22kHz 16 Bit Mono** задает шестнадцатиразрядный монофонический звук с частотой оцифровки 22 кГц.

Очень странно, но в этом списке также предусмотрен пункт **Disable**. Если его выбрать, Flash создаст "пустой" звуковой файл. Зачем это было сделано, непонятно, ведь если звук не нужен, его можно вообще не экспортировать.

Если установить флажок **Ignore event sounds**, Flash не включит в звуковое сопровождение фильма сигналы. Тогда получившийся в результате экспорта аудиофайл будет содержать только потоковые звуки.

Растровые файлы Windows (формат BMP)

Параметры этого формата задаются в диалоговом окне **Export Bitmap** (рис. 21.21).

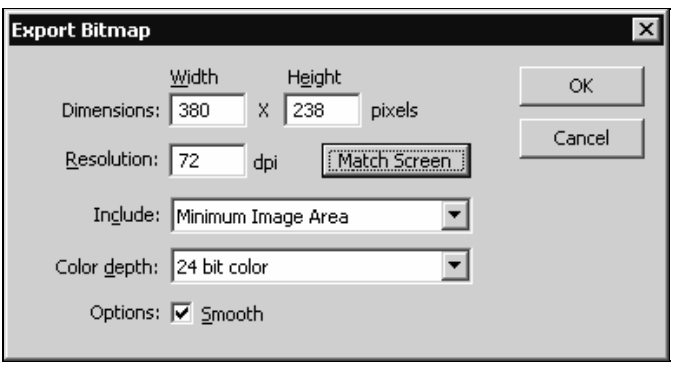


Рис. 21.21. Диалоговое окно **Export Bitmap**

Группа **Dimensions** из полей ввода **Width** и **Height** служит для задания соответственно ширины и высоты результирующего изображения в пикселах. В поле ввода **Resolution** задается разрешающая способность изображения в точках (пикселах) на дюйм. Если нажать кнопку **Match Screen**, Flash установит эти параметры сам, основываясь на размерах рабочего листа и заданном разрешении экрана.

Раскрывающийся список **Include** служит для задания размеров свободного пространства, которое будет включено в состав изображения. Пункт **Minimum Image Area**, выбранный по умолчанию, предписывает Flash включить в изображение минимум пустого пространства. Если же выбрать пункт **Full Document Size**, то Flash включит в состав изображения все свободное пространство, имеющееся на рабочем листе.

Раскрывающийся список **Color depth** задает цветность изображения. Здесь доступны четыре пункта:

- **8 bit grayscale** — черно-белое изображение с 256 градациями серого цвета;
- **8 bit color** — 256-цветное изображение;
- **24 bit color** — изображение с 24-битным цветом (TrueColor). Этот пункт выбран по умолчанию;
- **32 bit color w/ alpha** — изображение с 24-битным цветом и каналом прозрачности (альфа-канал).

Флажок **Smooth** включает или отключает сглаживание контуров результирующего изображения.

Как графика Flash помещается на Web-страницы

А теперь давайте вернемся к теме публикации графики Flash на Web-страницах. И поговорим о том, каким же образом две совершенно разные технологии приходят к взаимному согласию.

Понятие о внедренных элементах

В *главе 1* говорилось, что изначально Всемирная Паутина WWW была исключительно текстовой. А это значит, что язык HTML, на котором пишутся Web-страницы, имел средства для вывода только текстовой информации. Ни таблиц, ни рисунков, ни тем более звуков и фильмов тогдашние Web-странички не содержали. Только текст, текст и еще раз текст.

Когда Интернет "пошел в народ", разработчикам языка HTML срочно потребовалось снабдить его поддержкой графики. Они решили проблему, придумав тег <IMG>, в атрибуте SRC которого указывался интернет-адрес файла, содержащего изображение. Вот так выглядит соответствующий код:

```
<IMG SRC="logo.jpg">
```

Прочитав этот тег, Web-обозреватель получает от Web-сервера файл logo.jpg и выводит его содержимое на экран.

Однако изображение, которое помещаем мы на Web-страницу с помощью тега <IMG>, выглядит явно чужеродным по отношению к изначально "текстовому" HTML. Такие элементы-"чужаки", хранящиеся в отдельных от самой Web-страницы файлах, называются *внедренными элементами*.

Точно такими же внедренными элементами являются и фильмы в формате AVI, и звуки в формате WAV, и графика Flash, помещаемые на Web-страницах. Только в их случае используются другие теги: <OBJECT> и <EMBED>. Сейчас мы их рассмотрим.

Теги <OBJECT> и <EMBED>

Собственно, оба тега — и <OBJECT>, и <EMBED> — выполняют одну и ту же задачу. Они задают местоположение внедренного элемента на Web-странице и интернет-адрес файла, содержащего данные, необходимые для отображения этого элемента. Различие их состоит лишь в том, что тег <EMBED> объявлен устаревшим и не рекомендован к использованию, а вместо него рекомендуется использовать тег <OBJECT>. Но давайте обо всем по порядку...

Тег <OBJECT>

Парный тег <OBJECT>, в общем случае, служит для помещения на Web-страницу компонента ActiveX. В том числе с его помощью можно поместить на Web-страницу встраиваемый проигрыватель Flash, оформленный в виде компонента ActiveX, и загрузить в него нужный файл Shockwave/Flash. (Как мы помним из *главы 1*, встраиваемый проигрыватель Flash работает совместно с Web-обозревателем.) Также с помощью этого тега можно поместить на Web-страницу проигрыватели Windows Media и QuickTime и загрузить в них фильмы в форматах AVI и QuickTime, звуки в форматах WAV и MIDI и пр.

Вот пример использования тега <OBJECT> для помещения на Web-страницу проигрывателя Flash, в котором должен воспроизводиться фильм movie.swf:

```
<OBJECT CLASSID="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" WIDTH="100"
↳HEIGHT="100" CODEBASE="http://fpdownload.macromedia.com/pub/shockwave/
↳cabs/flash/swflash.cab#version=8,0,0,0">
```

```
  <PARAM NAME="MOVIE" VALUE="movie.swf">
  <PARAM NAME="PLAY" VALUE="true">
  <PARAM NAME="LOOP" VALUE="true">
  <PARAM NAME="QUALITY" VALUE="high">
</OBJECT>
```

Видно, что внутри тега `<OBJECT>` помещается набор тегов `<PARAM>`, с помощью которых задаются различные параметры самого проигрывателя Flash, оформленного в виде компонента ActiveX. Каждому такому параметру соответствует свой тег `<PARAM>`; имя параметра задается атрибутом `NAME`, а значение — атрибутом `VALUE`.

Давайте посмотрим на приведенный выше HTML-код. Здесь мы поместили на Web-страницу проигрыватель Flash, задали файл Shockwave/Flash (параметр `"MOVIE"`), заставили его воспроизвестись сразу же после загрузки (параметр `"PLAY"`) и зациклили (параметр `"LOOP"`). Напоследок мы задали качества вывода графики (параметр `"QUALITY"`). Как видите, все параметры имеют "говорящие" имена, так что сразу ясно, какой параметр за что "отвечает".

Тег `<OBJECT>` содержит довольно много атрибутов. С их помощью задаются, в частности, размеры внедренного компонента ActiveX на Web-странице (атрибуты `WIDTH` и `HEIGHT`) и интернет-адрес его дистрибутива (атрибут `CODEBASE`). Таким образом, если нужный компонент на клиентском компьютере не установлен, Web-обозреватель сможет самостоятельно загрузить и установить его.

Тег `<OBJECT>` поддерживается Web-обозревателями Microsoft Internet Explorer, начиная с версии 3.0, и Netscape Navigator версии не старше 4.0. Также он поддерживается всеми программами, появившимися в последнее время: Opera, Mozilla, Firefox и Apple Safari.

Тег `<OBJECT>` стандартизирован и рекомендован к использованию комитетом W^3C. Предполагается, что со временем он заменит собой все остальные теги, предназначенные для помещения на Web-страницы внедренных элементов, в том числе и тег `<IMG>`.

Тег *<EMBED>*

Парный тег `<EMBED>`, в общем случае, служит для помещения на Web-страницу модуля расширения Web-обозревателя. В том числе с его помощью можно поместить на Web-страницу встраиваемый проигрыватель Flash, оформленный в виде модуля расширения, и загрузить в него нужный файл Shockwave/Flash. Вот пример тега `<EMBED>`, который это делает:

```
<EMBED SRC="movie.swf" WIDTH="100" HEIGHT="100" PLAY="true" LOOP="true"
↳QUALITY="high"
```

```
↳PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer">
</EMBED>
```

В отличие от тега <OBJECT>, здесь все необходимые параметры задаются в атрибутах самого тега <EMBED>. В них указываются размеры внедренного модуля на Web-странице (атрибуты WIDTH и HEIGHT), имя файла Shockwave/Flash (атрибут SRC), путь к дистрибутиву модуля расширения (атрибут PLUGINSPAGE) и качество воспроизведения (атрибут QUALITY). Никакие дополнительные теги в этом случае не нужны.

Тег <EMBED> начал поддерживаться Web-обозревателями Microsoft Internet Explorer версии 3.0 и Netscape Navigator 2.0. Хотя он в настоящее время не рекомендован к использованию комитетом W^3C, но поддерживается до сих пор. Однако во всех вновь создаваемых Web-страницах рекомендуется использовать тег <OBJECT>.

Совместное использование тегов *<OBJECT>* и *<EMBED>*

На практике для решения проблем совместимости разных программ Web-обозревателей теги <OBJECT> и <EMBED> используются совместно. Например, вот так:

```
<OBJECT CLASSID="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" WIDTH="100"
↳HEIGHT="100" CODEBASE="http://fpdownload.macromedia.com/pub/shockwave/
↳cabs/flash/swflash.cab#version=8,0,0,0">
  <PARAM NAME="MOVIE" VALUE="movie.swf">
  <PARAM NAME="PLAY" VALUE="true">
  <PARAM NAME="LOOP" VALUE="true">
  <PARAM NAME="QUALITY" VALUE="high">
  <EMBED SRC="movie.swf" WIDTH="100" HEIGHT="100" PLAY="true" LOOP="true"
  ↳QUALITY="high"
  ↳PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer">
  </EMBED>
</OBJECT>
```

Видно, что в этом случае тег <EMBED> вложен внутрь тега <OBJECT> вместе с набором тегов <PARAM>. Что же происходит, если клиент пытается просмотреть Web-страницу, содержащую такой код? Давайте рассмотрим эту ситуацию для разных Web-обозревателей.

Если клиент использует современный Web-обозреватель, а именно, Microsoft Internet Explorer 3.0 или Netscape Navigator 4.0 (или более новые их версии), происходит следующее. Web-обозреватель считывает тег <OBJECT>, затем — теги <PARAM>, загружает компонент ActiveX, файл фильма и выводит

его на экран. Тег <EMBED>, вложенный внутрь тега <OBJECT>, в этом случае игнорируется.

Если же клиент использует более старые версии программ Web-обозревателей, все происходит по-другому. Так как теги <OBJECT> и <PARAM> неизвестны этим программам, они их игнорируют — это стандартное поведение Web-обозревателя, встретившего незнакомый тег. В этом случае обрабатывается тег <EMBED>.

> ***НА ЗАМЕТКУ***
>
> Возможен, правда, и третий вариант. Клиент может использовать Microsoft Internet Explorer 2.0 — первую версию Web-обозревателя фирмы Microsoft, не поддерживающую ни тег <OBJECT>, ни тег <EMBED>. Но вряд ли сейчас кто-то еще пользуется этой курьезной программой, так что ее не стоит принимать в расчет.

Параметры встраиваемого проигрывателя Flash

Встраиваемый проигрыватель Flash поддерживает довольно много параметров. В случае тега <OBJECT> одни параметры задаются как одноименный атрибут этого тега, а другие — с помощью тега <PARAM>. В случае тега <EMBED> все параметры задаются через его атрибуты.

> ***ВНИМАНИЕ!***
>
> Значения всех описанных ниже параметров задаются в строковом виде.

Параметр ALLOWSCRIPTACCESS разрешает или запрещает фильму Flash взаимодействовать с Web-страницей с помощью действия FSCommand (его описание можно найти в интерактивной справке Flash). Может принимать три значения:

- "always" — взаимодействие разрешено в любом случае;
- "never" — взаимодействие запрещено в любом случае;
- "samedomain" — взаимодействие разрешено только в том случае, если и Web-страница, и фильм Flash загружены с одного и того же Web-сервера. Это значение по умолчанию.

В случае тега <OBJECT> значение этого параметра задается с помощью тега <PARAM>, в случае тега <EMBED> — в одноименном атрибуте.

Параметр ALIGN задает способ выравнивания "окна" проигрывателя Flash на Web-странице. Может принимать одно из пяти значений:

- "Default" — выравнивание по центру Web-страницы;
- "L" — выравнивание по левой границе Web-страницы;

- "T" — выравнивание по верхней границе Web-страницы;
- "R" — выравнивание по правой границе Web-страницы;
- "B" — выравнивание по нижней границе Web-страницы.

Если окно Web-обозревателя меньше, чем размеры изображения Shockwave/Flash, оно будет обрезано.

Поддерживается тегами <OBJECT> и <EMBED>. Задается как одноименный атрибут этих тегов.

Необязательный параметр BASE задает базовый интернет-адрес, используемый как точка для отсчета всех относительных интернет-адресов, присутствующих в гиперссылках. Поддерживается тегами <OBJECT> и <EMBED>. Задается как одноименный атрибут этих тегов.

Необязательный параметр BGCOLOR определяет цвет фона фильма в формате 0xRRGGBB. Перекрывает значение цвета фона, заданное в среде Flash. Если не указан, используется изначальный цвет фона фильма Flash.

В случае тега <OBJECT> значение этого параметра задается с помощью тега <PARAM>, в случае тега <EMBED> — в одноименном атрибуте.

Параметр CLASSID определяет идентификатор компонента ActiveX. В случае проигрывателя Flash должен иметь значение:

```
"clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
```

Поддерживается только тегом <OBJECT> и задается через его одноименный атрибут.

Параметр CODEBASE определяет интернет-адрес дистрибутива компонента ActiveX. В случае проигрывателя Flash 8 должен иметь значение:

```
"http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#
↳version=8,0,0,0"
```

Поддерживается только тегом <OBJECT> и задается через его одноименный атрибут.

Необязательный параметр DEVICEFONTS разрешает или запрещает проигрывателю Flash использовать для отображения текста шрифты-псевдонимы вместо шрифтов, не установленных на клиентском компьютере. (О шрифтах-псевдонимах было рассказано в *главе 7*.) Может принимать одно из двух значений:

- "true" — подстановка разрешена;
- "false" — подстановка запрещена (значение по умолчанию).

Нужно только иметь в виду, что такая подстановка работает лишь в статичных текстовых блоках.

В случае тега <OBJECT> значение этого параметра задается с помощью тега <PARAM>, в случае тега <EMBED> — в одноименном атрибуте.

Параметр HEIGHT определяет высоту "окна" проигрывателя Flash в пикселах или процентах от высоты окна Web-обозревателя. Формат использования:

HEIGHT="<*Высота в пикселах*>|<*Процент от высоты окна*>%"

Поддерживается тегами <OBJECT> и <EMBED>. Задается как одноименный атрибут этих тегов.

Необязательный параметр ID задает имя данного компонента ActiveX, в нашем случае — проигрывателя Flash. Через это имя можно впоследствии получить доступ к проигрывателю из Web-сценариев.

Поддерживается только тегом <OBJECT> и задается с помощью одноименного атрибута. В случае тега <EMBED> нужно использовать параметр NAME, который будет описан ниже.

Необязательный параметр LOOP позволяет зациклить фильм, для чего ему нужно присвоить значение "true". Если же ему присвоить значение "false", воспроизведение фильма остановится после его завершения. Значение по умолчанию — "true".

В случае тега <OBJECT> значение этого параметра задается с помощью тега <PARAM>, в случае тега <EMBED> — в одноименном атрибуте.

Необязательный параметр MENU разрешает или запрещает проигрывателю Flash выводить контекстное меню при щелчке правой кнопкой мыши. Может принимать одно из двух значений:

- "true" — при щелчке правой кнопкой мыши выводится полноразмерное контекстное меню;
- "false" — контекстное меню будет содержать только пункт **About Flash Player**.

Значение по умолчанию — "true".

В случае тега <OBJECT> значение этого параметра задается с помощью тега <PARAM>, в случае тега <EMBED> — в одноименном атрибуте.

Параметр MOVIE задает интернет-адрес файла Shockwave/Flash, который должен быть загружен в проигрыватель Flash. Поддерживается только тегом <OBJECT> и задается с помощью тега <PARAM>. В случае тега <EMBED> нужно использовать параметр SRC, описанный далее в этой главе.

Необязательный параметр NAME определяет имя модуля расширения, в нашем случае — проигрывателя Flash. Через это имя можно впоследствии получить доступ к проигрывателю из Web-сценариев.

Поддерживается только тегом <EMBED> и задается с помощью одноименного атрибута. В случае тега <OBJECT> нужно использовать параметр ID, описанный ранее.

Необязательный параметр PLAY управляет воспроизведением фильма после загрузки. Если он равен "true", то фильм сразу же после загрузки будет запущен на воспроизведение. Если же он равен "false", то фильм не будет воспроизведен после загрузки, и пользователю придется сделать это самостоятельно. Значение по умолчанию — "true".

В случае тега <OBJECT> значение этого параметра задается с помощью тега <PARAM>, в случае тега <EMBED> — в одноименном атрибуте.

Параметр PLUGINSPAGE задает интернет-адрес Web-страницы, содержащей ссылку на дистрибутив модуля расширения. В случае проигрывателя Flash 8 должен иметь значение:

"http://www.macromedia.com/go/getflashplayer"

Поддерживается только тегом <EMBED> и задается через одноименный атрибут этого тега.

Необязательный параметр QUALITY устанавливает качество отображения графики Flash. Может принимать одно из шести значений:

- "low" — скорость воспроизведения имеет приоритет перед качеством графики, сглаживание не используется. Самое низкое качество;
- "autolow" — изначально качество графики низкое, сглаживание не используется. Далее, если компьютер окажется достаточно мощным, проигрыватель Flash повысит качество графики и включит сглаживание;
- "autohigh" — изначально качество графики высокое, используется сглаживание. Далее, если компьютер окажется недостаточно мощным, проигрыватель Flash понизит качество графики и отключит сглаживание;
- "medium" — используется сглаживание для векторной, но не для растровой графики. Среднее качество;
- "high" — качество графики имеет приоритет над скоростью воспроизведения; используется сглаживание векторной, а если нет анимации, то и растровой графики;
- "best" — используется сглаживание и векторной, и растровой графики. Самое высокое качество, которое не уменьшается ни в каких случаях.

Значение по умолчанию — "high".

В случае тега <OBJECT> значение этого параметра задается с помощью тега <PARAM>, в случае тега <EMBED> — в одноименном атрибуте.

Необязательный параметр SALIGN задает выравнивание фильма в "окне" проигрывателя Flash. Может принимать одно из восьми значений:

- "L" — выравнивание по левому краю;
- "T" — выравнивание по верхнему краю;
- "R" — выравнивание по правому краю;
- "B" — выравнивание по нижнему краю;
- "TL" — выравнивание по верхнему и левому краю;
- "TR" — выравнивание по верхнему и правому краю;
- "BR" — выравнивание по нижнему и правому краю;
- "BL" — выравнивание по нижнему и левому краю.

Если значение этого параметра не задано, фильм располагается в центре "окна" проигрывателя Flash.

В случае тега <OBJECT> значение этого параметра задается с помощью тега <PARAM>, в случае тега <EMBED> — в одноименном атрибуте.

Необязательный параметр SCALE определяет параметры масштабирования фильма. Может принимать одно из трех значений:

- "showall" — изображение будет выведено на экран целиком, для чего может быть применено масштабирование. Однако пропорции изображения искажены не будут, и в результате этого вдоль горизонтальных или вертикальных сторон его могут появиться пустые области;
- "noborder" — то же самое, что "showall", но пустые области появляться не будут — Flash обрежет изображение по горизонтали или вертикали, чтобы их избежать;
- "exactfit" — изображение будет выведено на экран целиком, для чего может быть применено масштабирование, в результате которого могут исказиться размеры изображения.

Значение по умолчанию — "showall".

Этот параметр имеет смысл использовать только тогда, когда значения параметров HEIGHT и WIDTH заданы в процентах от размера окна Web-обозревателя.

В случае тега <OBJECT> значение этого параметра задается с помощью тега <PARAM>, в случае тега <EMBED> — в одноименном атрибуте.

Необязательный параметр SEAMLESSTABBING управляет порядком обхода элементов управления, если таковые присутствуют в фильме Flash. Если его значение равно "true", пользователь имеет возможность "выйти" за пределы фильма Flash, пользуясь клавишами <Tab> и <Shift>+<Tab>. Значение

"false" запрещает "выход" за пределы фильма. Значение по умолчанию — "true". Поддерживается только тегом <OBJECT> и задается с помощью тега <PARAM>.

Параметр SRC задает интернет-адрес файла Shockwave/Flash, который должен быть загружен в проигрыватель Flash. Поддерживается только тегом <EMBED> и задается через одноименный атрибут этого тега. В случае тега <OBJECT> нужно использовать описанный ранее параметр MOVIE.

Необязательный параметр SWLIVECONNECT определяет, должен ли Web-обозреватель загружать исполняющую подсистему JavaScript. Если его значение равно "false", подсистема JavaScript не загружается, когда фильм Flash запускается на воспроизведение. Если оно равно "true", подсистема JavaScript загружается. Значение по умолчанию — "false".

Этот параметр можно использовать в целях экономии памяти клиентского компьютера, выгрузив подсистему JavaScript. Однако если в сценариях фильма используется действие FSCommand (его описание приведено в интерактивной справке Flash), следует установить значение этого параметра, равным "true".

Поддерживается только тегом <EMBED> и задается через одноименный атрибут этого тега.

Параметр WIDTH задает ширину "окна" проигрывателя Flash в пикселах или процентах от ширины окна Web-обозревателя. Формат использования:

WIDTH="*<Ширина в пикселах>*|*<Процент от ширины окна>*%"

Поддерживается тегами <OBJECT> и <EMBED>. Задается как одноименный атрибут этих тегов.

Необязательный параметр WMODE задает вид "окна" проигрывателя на Web-странице. Может принимать одно из трех значений:

- "Window" — фильм отображается на Web-странице в собственном "окошке". Самая высокая скорость воспроизведения;
- "Opaque" — фильм не будет отображаться в собственном "окошке". Элементы, находящиеся на Web-странице "ниже" этого фильма, не будут видны, но элементы, находящиеся "выше" его, видны будут;
- "Transparent" — фильм не будет отображаться в собственном "окошке". Элементы, находящиеся на Web-странице "выше" и "ниже" этого фильма, будут видны. Качество вывода анимации при этом может ухудшиться.

Значение по умолчанию — "window".

Поддерживается только тегом <OBJECT> и задается с помощью тега <PARAM>.

Создание и правка шаблонов HTML

Последнее, что мы рассмотрим в этой главе, — это создание и правка шаблонов HTML, используемых для формирования Web-страниц при публикации фильмов Flash. Вообще говоря, нам должно хватить на первое время тех шаблонов, которые поставляются в составе Flash. Но если мы начнем создавать Web-сайты с использованием графики Flash, то рано или поздно должны столкнуться с тем, что стандартные шаблоны нам не подходят.

Шаблоны, на основе которых создаются Web-страницы, — это обычные файлы HTML. Добраться до них не очень сложно. Если мы работаем в Windows 95/98/ME, то откроем папку en\First Run\HTML, расположенную в папке, где установлен Flash. Если же мы работаем в Windows NT/2000/XP/2003, то откроем папку Local Settings\Application Data\Macromedia\Flash 8\en\Configuration\HTML, расположенную в папке нашего пользовательского профиля.

От обычных Web-страниц шаблоны HTML отличаются тем, что их HTML-код содержит так называемые *макросы*. Это особые команды, представляющие собой набор символов, начинающийся со знака доллара (`$`). При формировании Web-страницы Flash заменяет эти макросы реальным кодом HTML, который создает сам на основе заданных нами параметров публикации.

Давайте рассмотрим несколько примеров таких макросов, реально присутствующих в стандартных шаблонах.

В начале HTML-кода каждого шаблона HTML обязательно должны находиться два макроса, задающие соответственно его имя и текстовое описание. Это имя и описание появляются в диалоговом окне **HTML Template Info** (см. рис. 21.4). Имя шаблона задает макрос `$TT`, а описание — пара макросов `$DS` и `$DF`. Вот как это выглядит (макросы выделены полужирным шрифтом):

```
$TTFlash Only
$DS
Display Macromedia Flash Movie in HTML.
$DF
```

Приведенный выше код задает имя и описание шаблона **Flash Only**, используемого для создания простых Web-страниц с внедренным фильмом Flash. Хорошо видно, что имя шаблона следует сразу за макросом `$TT`, а описание находится между макросами `$DS` и `$DF`.

А вот еще один пример кода, содержащего макросы:

```
<EMBED $PE WIDTH="$WI" HEIGHT="$HE" NAME="$TI" ALIGN="$HA"
↳PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer">
</EMBED>
```

Этот код задает тег <EMBED>, задающий параметры модуля расширения. Значения ширины и высоты задаются соответственно макросами $WI и $HE, имя — макросом $TI, а выравнивание — макросом $HA. Однако значение атрибута PLUGINSPAGE задано не макросом, а его реальным значением. Дело в том, что это значение для всех фильмов Flash одно и то же, поэтому его лучше всего задать "жестко".

А что делает макрос $PE? Задает все остальные атрибуты тега <EMBED>, которые должны иметь значения, отличные от значений по умолчанию. Вот такой макрос-универсал.

А вот пример кода, задающего тег <OBJECT> и, стало быть, компонент ActiveX:

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
↳CODEBASE="http://fpdownload.macromedia.com/pub/shockwave/
↳cabs/flash/swflash.cab#version=8,0,0,0" WIDTH="$WI" HEIGHT="$HE"
↳ID="$TI"
↳ALIGN="$HA">
  $PO
</OBJECT>
```

Здесь используются все те же макросы за одним исключением — вместо макроса $PE задание параметров проигрывателя Flash выполняет макрос-универсал $PO.

Все поддерживаемые Flash макросы перечислены в табл. 21.2.

Таблица 21.2. *Макросы, поддерживаемые Flash*

Параметр	Макрос
Имя шаблона	$TT
Описание шаблона (начало)	$DS
Описание шаблона (конец)	$DE
Имя, задаваемое с помощью параметров ID и NAME	$TI
Ширина (параметр WIDTH)	$WI
Высота (параметр HEIGHT)	$HE
Интернет-адрес файла Shockwave/Flash, в котором хранится фильм (параметры MOVIE и SRC)	$MO
Местоположение "окна" проигрывателя на Web-странице (параметр ALIGN)	$HA

Таблица 21.2 *(продолжение)*

Параметр	Макрос
Зацикливание фильма (параметр LOOP)	$LO
Параметры, задаваемые тегами <PARAM>, чьи значения отличаются от значений по умолчанию	$PO
Атрибуты тега <EMBED>, чьи значения отличаются от значений по умолчанию	$PE
Должен ли фильм быть запущен на воспроизведение сразу же после загрузки (параметр PLAY)	$PL
Качество вывода графики (параметр QUALITY)	$QU
Масштабирование фильма в "окне" проигрывателя Flash (параметр SCALE)	$SC
Выравнивание фильма в "окне" проигрывателя Flash (параметр SALIGN)	$SA
Вид "окна" проигрывателя Flash на Web-странице (параметр WMODE)	$WM
Подстановка шрифтов-псевдонимов (параметр DEVICEFONTS)	$DE
Цвет фона фильма (параметр BGCOLOR)	$BG
Весь текст, содержащийся в фильме	$MT
Все гиперссылки, содержащиеся в фильме	$MU
Ширина карты-изображения	$IW
Высота карты-изображения	$IH
Интернет-адрес карты-изображения	$IS
Имя карты-изображения (значение атрибута NAME тега <MAP>)	$IU
Список областей карты-изображения (тег <MAP>)	$IM
Ширина "картинки" фильма QuickTime, замещающего фильм Shockwave/Flash	$QW
Высота "картинки" фильма QuickTime, замещающего фильм Shockwave/Flash	$QH
Интернет-адрес файла фильма QuickTime, замещающего фильм Shockwave/Flash	$QN
Ширина изображения-замены GIF	$GW
Высота изображения-замены GIF	$GH
Интернет-адрес файла изображения-замены GIF	$GN

Таблица 21.2 *(окончание)*

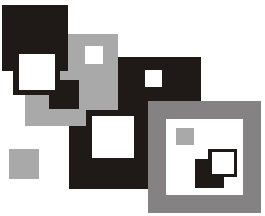
Параметр	Макрос
Ширина изображения-замены JPEG	$JW
Высота изображения-замены JPEG	$JH
Интернет-адрес файла изображения-замены JPEG	$JN
Ширина изображения-замены PNG	$PW
Высота изображения-замены PNG	$PH
Интернет-адрес файла изображения-замены PNG	$PN

Чтобы получше изучить разные макросы, проще всего открыть любой поставляемый с Flash шаблон и, держа перед глазами табл. 21.2, просмотреть его HTML-код. Можно также попробовать изменить код шаблона и посмотреть, что из этого получится.

Что дальше?

Вот мы и опубликовали наш так долго создававшийся фильм Flash. Теперь можно ожидать появления толп восторженных поклонников...

А если серьезно, то разговор о создании графики, фильмов и приложений Flash практически закончен. Нам осталось только рассмотреть некоторые дополнительные возможности, предлагаемые Flash для облегчения нашей работы, и средства его настройки. Этим и займемся мы в двух последних главах книги.



Глава 22

Дополнительные возможности Flash

Тема этой главы не относится напрямую к созданию графики Flash. Она касается дополнительных возможностей, предоставляемых этой программой, которые мы можем в дальнейшем использовать в работе, а можем и не использовать. В любом случае знать о них полезно.

Итак, что же это за дополнительные возможности, которым посвящена целая глава?

Во-первых, это панель **Movie Explorer**. Она представляет в виде иерархического списка все кадры анимации, образцы и экземпляры открытого во Flash документа. Она может помочь быстро найти экземпляр какого-либо образца или сценарий, хотя вряд ли ее стоит держать постоянно открытой.

Во-вторых, это панель поиска и замены. Она позволяет найти не только какой-либо текст, но и цвет, шрифт, образец, звук, импортированное видео или растровое изображение. Пожалуй, рассказ о панели поиска и замены будет самым долгим.

В-третьих, это развитые средства печати создаваемой во Flash графики. Они позволяют выполнять печать как из самой среды Flash, так и из проигрывателя Flash; причем в последнем случае можно указывать, какие части фильма будут распечатаны, а какие нет.

А теперь рассмотрим перечисленные возможности Flash подробнее. И начнем с панели **Movie Explorer**.

Панель *Movie Explorer*

В процессе изучения Flash мы создали много сложных изображений и фильмов. Они содержали и векторную, и импортированную растровую

графику, и текстовые блоки, и образцы, и экземпляры этих образцов, и импортированное видео, и звуки, и сценарии. Как разобраться во всем этом богатстве?

Нам поможет панель **Movie Explorer**. Она представляет в виде большого иерархического списка все элементы нашего фильма и позволяет быстро найти то, что нам нужно. Этакое "бюро находок" от фирмы Macromedia.

Чтобы вывести эту панель на экран, нужно включить пункт-выключатель **Movie Explorer** меню **Window** или нажать комбинацию клавиш <Alt>+<F3>. Сама панель **Movie Explorer** показана на рис. 22.1.

Рис. 22.1. Панель **Movie Explorer**

Бóльшую часть панели **Movie Explorer**, как уже говорилось, занимает обширный иерархический список всех составных частей активного документа Flash. Сцены фильма представлены в виде "деревьев" этого списка, а внутри их, возможно, присутствуют слои и кадры, отображаемые в виде "ветвей" списка, еще "глубже" находятся "ветви"-экземпляры различных образцов и специальные текстовые блоки и, наконец, пункты-сценарии и специальные текстовые блоки.

На взгляд автора, эта структура выглядит несколько запутанной, и разобраться в действительно сложном фильме довольно трудно. К счастью, мы

имеем возможность вывести в список панели **Movie Explorer** только те фрагменты фильма, которые нам нужны, и найти нужный экземпляр, введя его имя. Как это сделать, будет рассмотрено чуть позже.

Мы можем разворачивать и сворачивать "ветви" этого списка, как папки в окне Проводника Windows. Для этого также можно использовать пункты **Expand Branch** и **Collapse Branch** контекстного и дополнительного меню панели; первый пункт разворачивает выбранную в списке "ветвь", а второй — сворачивает. А пункт **Collapse Others** сворачивает все "ветви", кроме выбранной в списке.

Мы можем выбирать любые пункты иерархического списка, просто щелкая на них мышью. Также имеется возможность выбирать одновременно несколько пунктов одним из описанных в *главе 8* способов (т. е. с помощью клавиш <Shift> и <Ctrl>).

Как только мы выберем какой-либо пункт списка, Flash выделит на рабочем листе соответствующий ему фрагмент фильма. Если выбранный фрагмент почему-либо не виден на рабочем листе (например, он находится в другом кадре), можно выбрать пункт **Goto Location** контекстного или дополнительного меню, чтобы его увидеть. А если дважды щелкнуть на пункте списка, соответствующего сценарию, Flash откроет этот сценарий в панели **Actions**.

Как только мы добавляем или удаляем что-то на рабочем листе, содержимое списка панели **Movie Explorer** изменяется, отражая эти изменения. Так что Flash постоянно ведет учет своих "активов".

По умолчанию в панели **Movie Explorer** отображаются только сцены, экземпляры и привязанные к ним сценарии. Но мы можем просмотреть и все имеющиеся в библиотеке образцы. Для этого достаточно включить пункт-выключатель **Show Symbol Definitions** в контекстном или дополнительном меню панели. После этого в списке появится "дерево" **Symbol Definitions**, представляющее все содержимое библиотеки. А пункт **Show Movie Elements** контекстного или дополнительного меню позволяет включить или отключить вывод в списке сцен, кадров и экземпляров — иногда это может пригодиться.

> ***Внимание!***
>
> К сожалению, в списке панели **Movie Explorer** почему-то не отображаются образцы, созданные из импортированных файлов, и образцы-шрифты. И это притом, что экземпляры импортированных образцов там все-таки отображаются.

Теперь давайте обратимся к "дереву" **Symbol Definitions**. Мы можем выбрать в нем любой образец и выбрать пункт **Find in Library** контекстного или дополнительного меню. После этого Flash откроет панель **Library** и выделит данный образец в ее списке. А выбор пункта **Select Symbol Instances** заставит Flash выбрать в списке панели **Movie Explorer** и на рабочем листе все экземпляры этого образца.

Если же нам, наоборот, нужно найти образец, на основе которого создан какой-либо экземпляр, следует сначала выбрать в списке панели этот экземпляр, а потом выбрать пункт **Goto Symbol Definition** контекстного или дополнительного меню. В результате Flash выберет в списке панели **Movie Explorer** искомый образец, и нам останется только выбрать пункт **Find in Library**, чтобы Flash показал его в панели **Library**.

По умолчанию в панели **Movie Explorer** отображается только содержимое текущей сцены. Чтобы просмотреть содержимое всех сцен, нужно включить пункт-выключатель **Show All Scenes** контекстного или дополнительного меню.

Если наш фильм настолько сложен, что список панели **Movie Explorer** оказывается слишком большим, чтобы в нем разобраться, мы можем воспользоваться встроенными функциями поиска. Если известно имя искомого экземпляра образца, то мы введем его в поле ввода **Find**, расположенное над списком (см. рис. 22.1). Сразу же после этого Flash выведет в списке только те экземпляры, чьи имена содержат введенную последовательность символов. К сожалению, образцы при этом в любом случае показаны не будут. Чтобы вернуться к полному списку, нужно удалить все символы из поля ввода **Find**.

В верхней части панели расположена группа **Show** из шести кнопок-выключателей, позволяющая более точно выбрать фрагменты фильма, которые будут выводиться в списке. Рассмотрим эти кнопки в порядке слева направо:

- вывод текстовых блоков;
- вывод образцов и их экземпляров;
- вывод сценариев;
- вывод экземпляров импортированных образцов (растровых изображений, фильмов и звуков);
- вывод слоев и кадров.

Самая правая из расположенных в верхней части панели **Movie Explorer** кнопок при нажатии выводит на экран диалоговое окно **Movie Explorer Settings** (рис. 22.2), где мы сможем более точно задать параметры списка этой панели.

Группа флажков **Show** позволяет задать объекты, которые нужно показывать в списке панели **Movie Explorer**. В ней десять флажков:

- **Text** — текстовые блоки;
- **Buttons** — кнопки (т. е. экземпляры образцов-кнопок);
- **Movie clips** — клипы (т. е. экземпляры образцов-клипов);
- **Video** — экземпляры образцов-импортированных клипов;

- **ActionScripts** — сценарии;
- **Bitmaps** — экземпляры образцов-растровых изображений;
- **Graphics** — экземпляры графических образцов;
- **Sounds** — экземпляры образцов-звуков;
- **Layers** — слои;
- **Frames** — кадры.

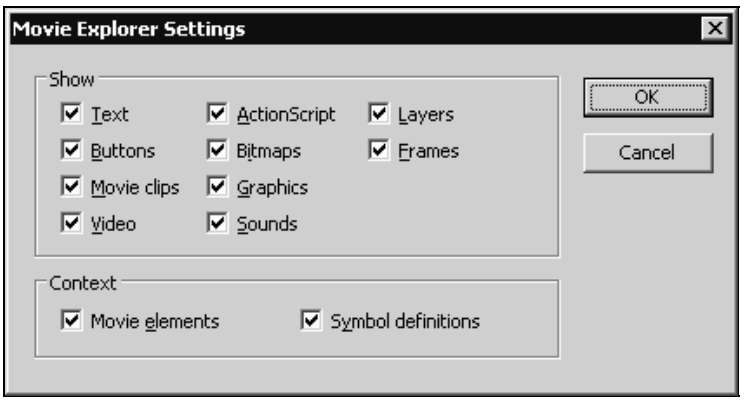


Рис. 22.2. Диалоговое окно **Movie Explorer Settings**

Группа флажков **Context** позволяет включить или отключить показ в списке всех образцов, созданных в самом Flash (флажок **Symbol definitions**), и всего, что не является образцами, — экземпляров, текстовых блоков, кадров, слоев, сценариев (флажок **Movie elements**).

Задав нужные параметры отображения списка, нужно нажать кнопку **OK**, чтобы немедленно их применить. Нажатие кнопки **Cancel** позволяет отменить новые параметры.

Остальные пункты, доступные в контекстном и дополнительном меню панели **Movie Explorer**, перечислены ниже.

Пункт **Rename** доступен для:

- образцов (будет изменено имя образца);
- экземпляров образцов (будет изменено имя образца, на основе которого был создан этот экземпляр);
- текстовых блоков (будет изменено содержимое текстового блока);
- слоев (будет изменено имя слоя);
- сцен (будет изменено имя сцены).

После выбора этого пункта вместо имени элемента в списке появится небольшое поле ввода, в котором будет подставлено старое имя или содержимое данного фрагмента фильма. После ввода нового имени или содержимого нужно нажать клавишу <Enter> для его сохранения или <Esc> для отмены.

Пункты **Edit in Place** и **Edit in New Window** позволяют исправить выбранный образец: первый — в режиме правки "на месте", второй — в отдельном окне Flash. Чтобы изменить выбранный образец "на месте", также можно дважды щелкнуть на соответствующем пункте списка.

Пункты **Cut**, **Copy**, **Paste** и **Clear** нам уже знакомы. Они предназначены для работы с буфером обмена Windows: вырезания, копирования и вставки соответствующего фрагмента фильма. Последний же пункт — **Clear** — позволяет удалить выделенный в списке фрагмент фильма.

Пункт **Copy All Text to Clipboard** выполняет очень странную операцию — он копирует в буфер обмена текстовое представление списка панели **Movie Explorer**. В полученном таким образом тексте с помощью отступов разного размера имитируется вложенность пунктов списка. Зачем и кому это может понадобиться, непонятно. Аналогично, пункт **Print** позволяет распечатать этот список.

Поиск и замена фрагментов фильма

Очень часто бывает нужно найти в изображении или фильме Flash какой-либо текст и, возможно, заменить его на другой. Специально для этого Flash, как и многие другие программы, работающие с документами, предлагает богатые возможности по поиску и замене. Причем текстом они не ограничиваются — с помощью Flash можно искать и заменять цвета, образцы, шрифты и многое другое.

Панель *Find and Replace*

Задание параметров поиска выполняется в специальной панели, которая так и называется — **Find and Replace** (рис. 22.3). Чтобы вызвать ее на экран, нужно включить пункт-выключатель **Find and Replace** меню **Edit** или нажать комбинацию клавиш <Ctrl>+<F>.

В раскрывающемся списке **Search in** задается, где будет выполняться поиск. Выбор пункта **Current Document** списка задает поиск во всем активном документе, а выбор пункта **Current Scene** — только в текущей сцене.

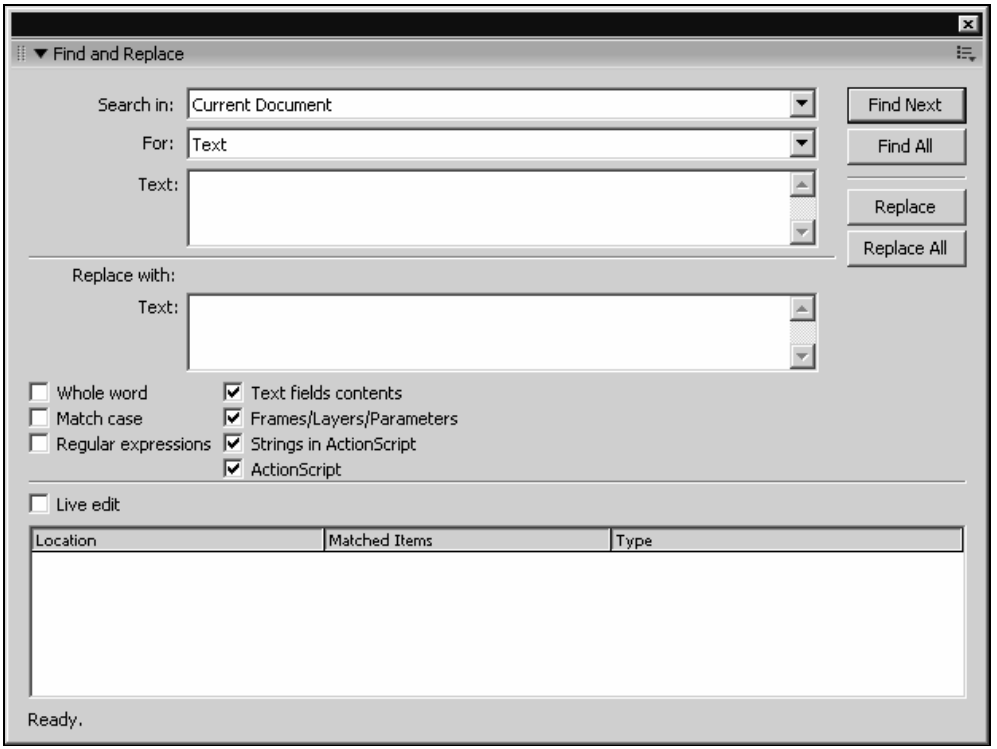


Рис. 22.3. Панель **Find and Replace** (включен поиск текста)

В раскрывающемся списке **For** нужно выбрать, что мы, собственно, собираемся искать. Давайте рассмотрим все пункты этого списка подробно.

Если в списке **For** выбран пункт **Text**, будет выполнен поиск текста, и панель **Find and Replace** примет вид, показанный на рис. 22.3. Искомая подстрока вводится в область редактирования **Text**, расположенную в верхней группе элементов управления, а заменяемая — в область редактирования **Text** нижней группы.

Флажки, расположенные в нижней половине панели, позволяют задать дополнительные параметры поиска.

Включение флажка **Whole word** задает поиск только целых слов, ограниченных с обеих сторон пробелами, кавычками, знаками препинания или чем-то подобным. Если же он отключен, будет выполняться поиск также и частей слов.

Флажок **Match case**, будучи включен, заставляет Flash учитывать при поиске регистр символов. Так, если мы включим этот флажок и наберем в области редактирования **Text** верхней группы подстроку "Flash", будут найдены вхо-

ждения только этой подстроки, но никак не "flash", "FLASH" и "fLaSh". Если же флажок **Match case** отключен, регистр символов при поиске не учитывается; в данном случае будут найдены все приведенные выше подстроки.

Флажок **Regular expressions** включает или отключает использование при поиске и замене *регулярных выражений*. Это особые команды для сложного поиска, которые мы рассмотрим позже.

Флажок **Text fields contents** включает или отключает поиск введенной подстроки в содержимом текстовых блоков.

Флажок **Frames/Layers/Parameters** включает или отключает поиск введенной подстроки в именах кадров и слоев и параметрах компонентов (компоненты и их параметры во всех подробностях описаны в интерактивной справке Flash).

Флажок **Strings in ActionScript** включает или отключает поиск введенной подстроки в строковых значениях, присутствующих в сценариях. А флажок **ActionScript** включает или отключает поиск подстроки в самих сценариях.

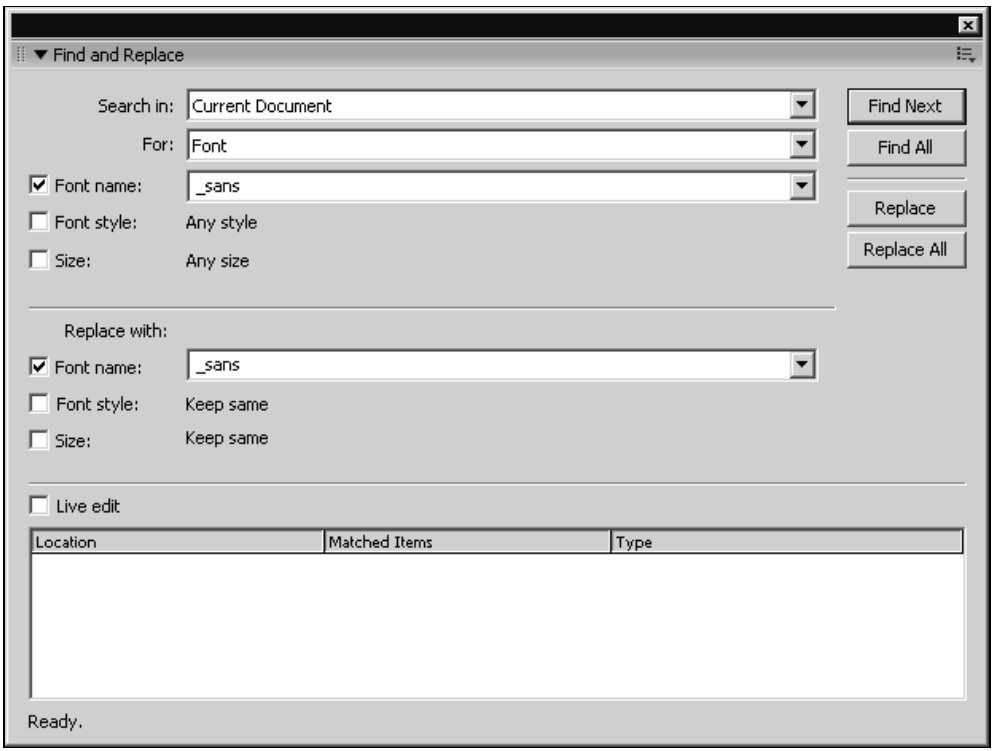


Рис. 22.4. Панель **Find and Replace** (включен поиск шрифта)

Если в списке **For** выбран пункт **Font**, будет выполнен поиск шрифта. А панель **Find and Replace** примет вид, показанный на рис. 22.4.

Искомый шрифт выбирается в раскрывающемся списке **Font name** верхней группы элементов управления панели **Find and Replace**. Этот список доступен при включенном флажке, расположенном левее его. Если же этот флажок отключить, шрифт при поиске учитываться не будет.

Если включен второй сверху флажок верхней группы, правее его появится раскрывающийся список **Font style**, с помощью которого можно выбрать искомый стиль шрифта. Пункт **Plain** этого списка задает поиск обычного шрифта, пункт **Italic** — курсива, **Bold** — полужирного, а **BoldItalic** — полужирного курсива. Если же флажок списка отключен, при поиске стиль шрифта учитываться не будет.

Если включен третий сверху флажок верхней группы, справа от него появятся два поля ввода с регуляторами. В поле ввода **Min** задается минимальный размер искомого шрифта, а в поле ввода **Max** — максимальный. Если же флажок отключен, при поиске размер шрифта учитываться не будет.

Заменяющий шрифт выбирается в раскрывающемся списке **Font name** нижней группы элементов панели **Find and Replace**. Этот список доступен только при включенном флажке, расположенном слева от него. Если же данный флажок отключен, Flash не будет заменять найденный шрифт.

Если включен второй сверху флажок нижней группы, правее его появится раскрывающийся список **Font style**, с помощью которого можно выбрать стиль заменяющего шрифта. Пункт **Plain** этого списка задает обычный шрифт, пункт **Italic** — курсив, **Bold** — полужирный, а **BoldItalic** — полужирный курсив. Если же флажок отключен, стиль шрифта при замене останется неизменным.

Если включен третий сверху флажок нижней группы, правее его появится поле ввода с регулятором, в котором задается размер заменяющего шрифта. Если же флажок отключен, размер шрифта останется неизменным.

Если в списке **For** выбран пункт **Color**, будет выполнен поиск цвета. Панель **Find and Replace** в этом случае примет вид, показанный на рис. 22.5.

Искомый цвет задается с помощью верхнего селектора цвета **Color**. А нижний селектор цвета **Color** задает заменяющий цвет.

Расположенные в нижней части панели флажки позволяют задать, где Flash будет искать заданный нами цвет. Всего этих флажков три:

- **Fills** — поиск будет выполнен среди цветов заливок;
- **Strokes** — поиск будет выполнен среди цветов линий контуров;
- **Text** — поиск будет выполнен среди цветов текста.

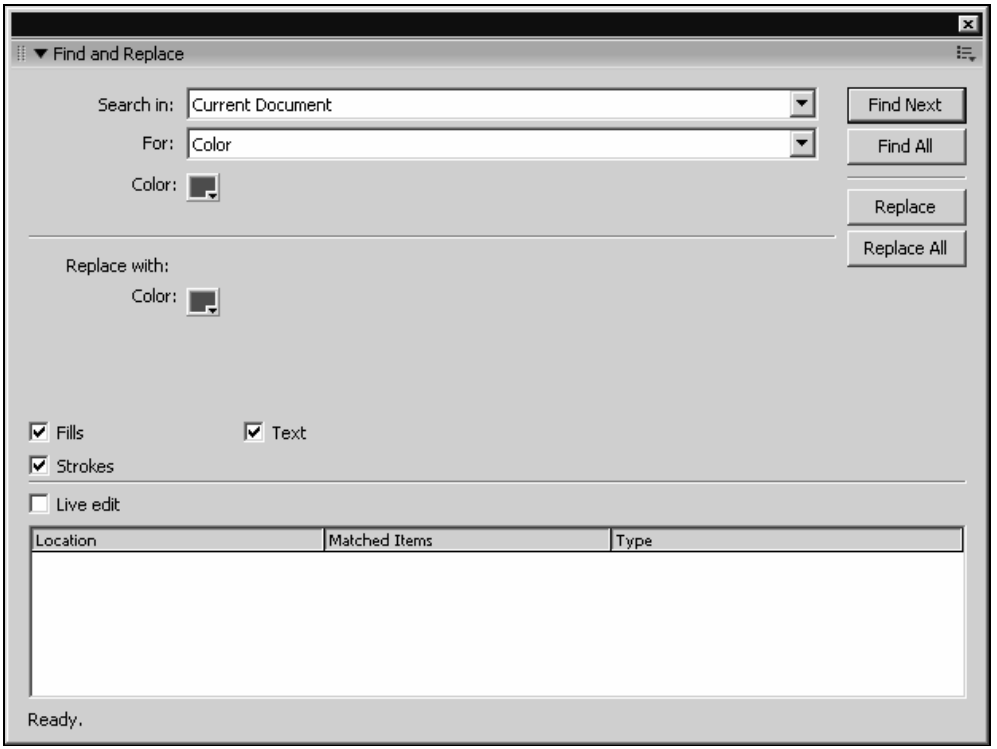


Рис. 22.5. Панель **Find and Replace** (включен поиск цвета)

Если в списке **For** выбран пункт **Symbol**, будет выполнен поиск экземпляров, основанных на заданном образце. Панель **Find and Replace** в этом случае примет вид, показанный на рис. 22.6.

Здесь все очень просто. В верхнем раскрывающемся списке **Name** выбирается искомый образец, а в нижнем — заменяющий. То есть мы можем выполнить операцию смены образца.

Аналогично последнему рассмотренному случаю будет выглядеть панель **Find and Replace** и в случае выбора в списке **For** пунктов **Sound**, **Video** и **Bitmap**. В этом случае будет выполнен поиск соответственно звука, импортированного видео и импортированного растрового изображения.

После того как заданы все параметры поиска (и замены), следует выполнить этот поиск (замену). Давайте посмотрим, как это сделать, на примере поиска (и замены) подстроки текста.

Чтобы выполнить поиск первого вхождения введенного текста, нужно нажать кнопку **Find Next**. Если Flash нашел введенный нами текст, в списке, находящемся в нижней части панели **Find and Replace**, появится строка,

описывающая, что и где найдено. Если дважды щелкнуть на этой строке, Flash выделит текстовый блок, в котором встретился этот текст, и мы сможем его исправить. Чтобы выполнить поиск следующего вхождения этой же подстроки, нужно снова нажать кнопку **Find Next**; также можно выбрать пункт **Find Next** меню **Edit** или нажать клавишу <F3>. Если же Flash не найдет заданной нами подстроки, он выведет окно-предупреждение с текстом "No items found".

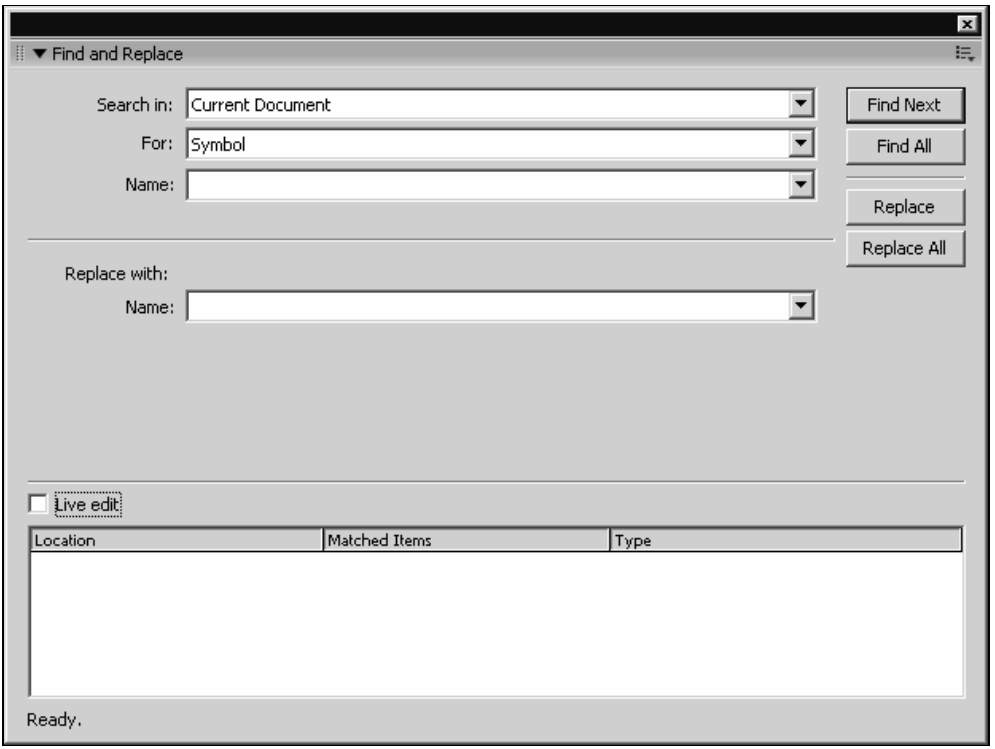


Рис. 22.6. Панель **Find and Replace** (включен поиск экземпляров, основанных на заданном образце)

Если нажать кнопку **Find All**, Flash выполнит поиск сразу всех вхождений введенной подстроки, и в списке в нижней части панели появятся строки, соответствующие найденным вхождениям.

Чтобы выполнить замену искомой подстроки, нужно нажать кнопку **Replace**. Как только Flash найдет эту подстроку (т. е. при первом обнаруженном ее вхождении), он тотчас ее заменит и выведет в списке в нижней части панели строку, описывающую найденное. Для замены следующей подстроки

нужно опять нажать кнопку **Replace** и т. д. А кнопка **Replace All** заставляет Flash найти и заменить все присутствующие в документе подстроки.

Точно так же выполняется поиск и замена шрифтов, цветов и образцов.

> ***НА ЗАМЕТКУ***
>
> В нижней части панели **Find and Replace** присутствует также флажок **Live edit**. В документации написано, что он, будучи включен, позволяет пользователю править найденные подстроки (а также цвета, шрифты и образцы). Однако, как показывает опыт, "живая правка" возможна и при отключенном флажке **Live edit**, т. е. состояние флажка ни на что не влияет.

Регулярные выражения и их использование

А теперь настало время познакомиться с регулярными выражениями.

Предположим, что перед нами стоит задача отыскать в текстовых блоках адрес электронной почты, оканчивающийся на ".com". Как это сделать? Может быть, поискать значок "собака" (@)? Но ведь он может встретиться и в обычном тексте, не являющемся почтовым адресом. Искать подстроку ".com"? Но тогда Flash вывалит в таблицу результатов поиска панели **Find and Replace** все найденные им интернет-адреса. Что же делать?..

Нам поможет только регулярное выражение. А именно, такое:

```
\b.+@.+\.com
```

На первый взгляд, это бессмысленный набор символов, в котором с трудом угадывается нечто знакомое. Чтобы разобраться в нем, напишем таблицу, содержащую описание символов регулярных выражений поиска всех оканчивающихся на ".com" почтовых адресов (табл. 22.1).

Таблица 22.1. *Символы, используемые в регулярном выражении поиска почтовых адресов, оканчивающихся на ".com"*

Символы	Описание
`\b`	Начало слова
`.`	Любой символ
`+`	Предыдущий символ обязательно должен встретиться один или более раз
`@`	Знак "собака"
`.+`	Опять любой символ, который обязательно должен встретиться один или более раз

Таблица 22.1 *(окончание)*

Символы	Описание
`\.`	Обычная точка. Ее предваряет обратная косая черта, так как точка — служебный символ (*литерал*)
`com`	Окончание почтового адреса

Теперь, когда нам все стало ясно, можно опробовать приведенное выше регулярное выражение. Выберем в раскрывающемся списке **For** панели **Find and Replace** пункт **Text**, введем наше регулярное выражение в область редактирования **Text** и не забудем включить флажок **Regular expressions**. Ура, работает!

Обобщим сведения, почерпнутые из приведенного примера.

В регулярных выражениях для обозначения различных фрагментов текста используются специальные символы — литералы. Если же нужно найти какой-либо символ, совпадающий с литералом, мы должны будем предварить его символом обратной косой черты (`\`). Например, чтобы найти точку, мы используем последовательность символов `\.`.

С помощью регулярных выражений мы можем выполнять поиск самых разнообразных слов и словосочетаний. Например, регулярное выражение

```
(multi|hyper)media
```

совпадает со словами "multimedia" и "hypermedia", но не совпадает со словом "media". Литерал "вертикальная черта" (`|`) задает поиск либо первой, либо второй подстроки (в нашем случае — либо "multi", либо "hyper"), а скобки здесь использованы для того, чтобы отделить друг от друга две части выражения. Если бы мы их не поставили, получилось бы выражение `multi|hypermedia`, совпадающее со словами "multi" и "hypermedia", а нам это не нужно.

А вот в выражении

```
[\.!\?]$
```

в квадратных скобках перечислены символы, один из которых должен встретиться в тексте. Это следующие символы: точка (предварена символом `\`, так как является литералом), восклицательный знак и вопросительный знак (предварен обратной косой чертой, так как он тоже литерал). Последний литерал — символ `$` — обозначает конец строки. В результате мы получим регулярное выражение, которое ищет один из перечисленных знаков препинания, стоящий в конце строки.

Мы убедились, что при поиске подстрок регулярные выражения очень помогают. Но помогут ли они при замене подстрок? Разумеется!

Предположим, что нам нужно найти адреса электронной почты и в порыве патриотизма заменить в них окончание "com" на "ru". Для поиска в этом случае мы используем уже знакомое и работающее регулярное выражение

```
/b(.+@.+\.)com
```

Стоп! Зачем мы взяли часть выражения в скобки? Ведь скобки используются для группировки, т. е. для отделения одной части регулярного выражения от других его частей. Здесь же ничего отделять не нужно — все и так работает.

Дело в том, что Flash, помимо всего прочего, помещает подстроку, совпадающую с заключенной в скобки частью регулярного выражения, в особую переменную, откуда ее потом можно будет извлечь. Таких переменных (иногда говорят — ячеек) предусмотрено девять, и обратиться к их содержимому можно по их номерам. В частности, извлечь содержимое первой ячейки в нашем случае можно так:

```
$1
```

Тогда заменить окончание почтового адреса следует с помощью такого регулярного выражения:

```
$1ru
```

Как видим, здесь все совсем просто. Мы берем подстроку, найденную первым регулярным выражением, и добавляем к ней окончание "ru".

Можно проверить два приведенных регулярных выражения. Первое нужно набрать в области редактирования **Text**, верней группы панели **Find and Replace**, а второе — в области редактирования **Text** нижней группы. И, конечно же, не забываем включить флажок **Regular expressions**.

Напоследок осталось привести полный список всех литералов регулярных выражений. Они перечислены в табл. 22.2.

Таблица 22.2. *Литералы регулярных выражений, поддерживаемые Flash*

Литерал	Описание
^	Начало строки
$	Конец строки
*	Предыдущий символ должен встретиться один или больше раз или не встретиться вообще
+	Предыдущий символ обязательно должен встретиться один или более раз
?	Предыдущий символ должен встретиться один раз или не встретиться вообще
.	Любой символ, за исключением символа новой строки

Таблица 22.2 *(окончание)*

Литерал	Описание
x|y	Должен встретиться символ x или символ y
{n}	Предыдущий символ должен встретиться точно n раз
{m,n}	Предыдущий символ должен встретиться от n до m раз
[abc]	Должен встретиться один из символов, перечисленных в квадратных скобках. Можно задавать диапазоны символов; например, [a-d] заменяет [abcd]
[^abc]	Должен встретиться любой символ, кроме перечисленных в квадратных скобках. Можно задавать диапазоны символов; например, вместо [abcd] задать [a-d]
\b	Граница слова (пробел или возврат каретки)
\B	Не граница слова
\d	Любая цифра. Эквивалентен [0-9]
\B	Любой символ, кроме цифры. Эквивалентен [^0-9]
\f	Прогон листа
\n	Перевод строки
\r	Возврат каретки
\s	Любой пробельный символ (пробел, табуляция, прогон страницы или перевод строки)
\S	Любой символ, кроме пробельного
\t	Табуляция
\w	Любой алфавитно-цифровой символ или подчеркивание. Эквивалентен [a-zA-Z0-9_]
\W	Любой символ, кроме алфавитно-цифрового и символа подчеркивания. Эквивалентен [^a-zA-Z0-9_]

Печать

Последнее, что мы рассмотрим в этой главе, — это процесс печати фильмов Flash. Мы узнаем, как распечатать документ, открытый в среде Flash, и как позволить пользователю печатать только избранные кадры уже готового фильма, сохраненного в формате Shockwave/Flash. И начнем мы с рассмотрения процесса печати в среде Flash.

Печать документов Flash

Процесс печати документов, организован во Flash несколько по-иному, нежели в других программах. Сейчас мы рассмотрим эти отличия. И заодно перечислим все операции, связанные с печатью документов во Flash.

Собственно печать документа Flash осуществляется обычным для Windows-приложения способом. Мы выбираем пункт **Print** меню **File** или нажимаем комбинацию клавиш <Ctrl>+<P>. После этого на экране появляется стандартное диалоговое окно печати Windows, в котором мы можем выбрать принтер и задать количество печатаемых копий.

Как и многие программы, в которых предусмотрена печать документов, Flash предоставляет возможность задать параметры настройки печати, а именно: размер бумаги, отступы, масштаб и центрирование изображения. Чтобы получить доступ к этим настройкам, выберем пункт **Page Setup** меню **File**. На экране появится диалоговое окно **Page Setup** (рис. 22.7).

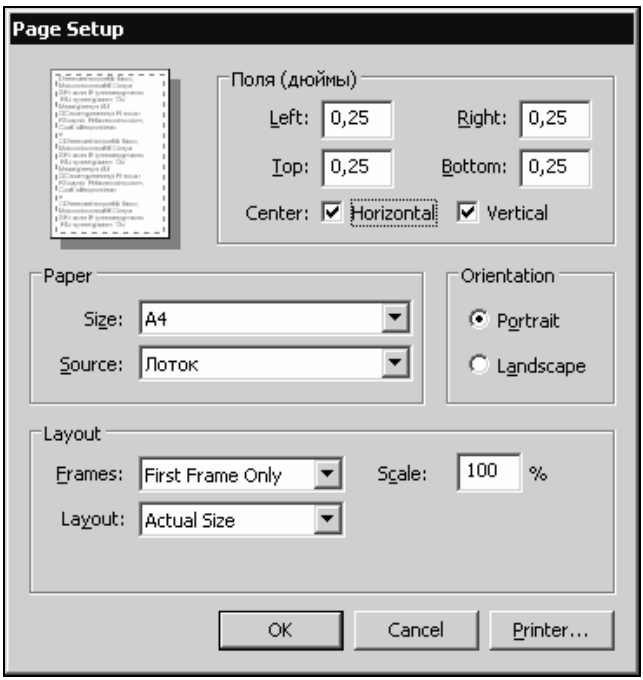


Рис. 22.7. Диалоговое окно **Page Setup**

В правом верхнем углу этого окна находится группа элементов управления, с помощью которых задаются отступы от краев бумаги и центрирование пе-

чатаемого изображения. Сами отступы задаются в полях ввода **Left** (отступ слева), **Top** (сверху), **Right** (справа) и **Bottom** (внизу) в дюймах. Центрирование включается и выключается с помощью флажков **Horizontal** (по горизонтали) и **Vertical** (по вертикали) группы **Center**.

В группе **Paper** находятся элементы управления, служащие для задания размера бумаги и способа ее подачи. Размер бумаги выбирается в раскрывающемся списке **Size**, а способ подачи — в списке **Source**.

Группа переключателей **Orientation** служит для выбора ориентации листа бумаги, на котором будет печататься изображение. Переключатель **Portrait** включает портретную (вертикальную) ориентацию, а переключатель **Landscape** — ландшафтную (горизонтальную).

В группе **Layout** находятся элементы управления, служащие для задания расположения печатаемого изображения на листе. Рассмотрим их подробнее.

С помощью пунктов **First Frame Only** и **All frames** раскрывающегося списка **Frames** можно указать, какой кадр анимации будет напечатан. Первый пункт вызывает печать только первого кадра анимации, второй — всех кадров.

С помощью раскрывающегося списка **Layout** задается расположение кадров фильма на листе бумаги. Этот список имеет пять пунктов:

- **Actual Size** — на одну страницу выводится один кадр фильма;
- **Fit On One Page** — на одну страницу выводится один кадр фильма, причем выполняется автоматическое масштабирование для того, чтобы кадр заполнил страницу целиком;
- **Storyboard — Boxes** — на одну страницу рядами выводятся несколько кадров фильма, причем каждый кадр помещается в рамке;
- **Storyboard — Grid** — на одну страницу рядами выводятся несколько кадров фильма, причем они отделяются друг от друга линиями;
- **Storyboard — Blank** — на одну страницу рядами выводятся несколько кадров фильма без рамок и разделяющих линий.

Если выбран пункт **Actual Size** списка **Layout**, становится доступным поле ввода **Scale**. В этом поле задается масштаб изображения (кадра), выводимого на страницу, в процентах. Это может помочь в случае, если кадры нашего фильма очень мелкие или, наоборот, очень крупные.

Если в списке **Layout** выбран пункт **Storyboard — Boxes**, **Storyboard — Grid** или **Storyboard — Blank**, в окне **Page Setup** появятся флажок **Label frames** и поля ввода **Frames** и **Frame margin**. Если включить флажок **Label frames**, то под каждым кадром на печатаемой странице Flash будет подставлять его имя или номер. В поле ввода **Frames** вводится количество кадров, располагающихся в одном горизонтальном ряду; косвенно этот параметр влияет на количество кадров, помещаемых на одну страницу, и, соответственно, на мас-

штаб кадров. А в поле ввода **Frame margin** указывается расстояние между отдельными кадрами, располагаемыми на странице; это расстояние задается в текущей единице измерения.

Создание фильмов, предназначенных для печати

Помимо возможности распечатать фильм на бумаге из самой среды Flash, существует возможность сделать это и из проигрывателя Flash. Пользователь открывает в проигрывателе Flash файл с фильмом Shockwave/Flash или Web-страницу, на которую помещен этот фильм, выбирает пункт **Print** в контекстном меню, задает в появившемся на экране стандартном диалоговом окне печати Windows необходимые параметры и нажимает кнопку **OK**.

Изначально проигрыватель печатает все кадры открытого в нем фильма. Однако мы можем разрешить печатать только избранные его кадры; остальные же кадры при этом автоматически станут непечатаемыми. Иногда это бывает полезно. (Например, если мы создаем каталог товаров в виде фильма Shockwave/Flash, то можем предусмотреть печать только одного-единственного кадра — формы заказа товара.) Посмотрим, как это делается.

Чтобы сделать какой-либо кадр печатаемым, его нужно сначала сделать текущим, выделив его на временно́й шкале. После этого переключаемся в панель **Properties** и даем этому кадру имя `#p`. Вот и все.

Повторяем эту операцию для всех кадров, которые требуется сделать печатаемыми. Остальные кадры фильма автоматически станут непечатаемыми и выводиться на принтер не будут.

По умолчанию границы печатаемой области кадра совпадают с границами рабочего листа, заданными при создании этого фильма. Однако мы можем задать другую *область печати* — например, для того, чтобы распечатать одну только форму заказа товара (упоминавшуюся выше), исключив логотип фирмы и кнопки навигации. Делается это следующим способом.

Создадим в фильме еще один кадр. Нарисуем в нем прямоугольник, размеры которого совпадут с желаемой областью печати. Дадим кадру имя `#b`. При этом нужно помнить, что кадр, задающий размеры области печати, должен быть единственным для одного фильма.

Любая графика, не входящая в область печати, напечатана не будет. Этим способом, кстати, можно воспользоваться в целях скрытия информации, не предназначенной для копирования.

Кстати, мы можем и вовсе запретить печать фильма. Сделать это можно двумя различными способами.

Во-первых, можно воспользоваться все той же панелью **Properties**. Выделим на временно́й шкале самый первый кадр нашего фильма и дадим ему имя `!#p`. Если пользователь при просмотре такого фильма откроет контекстное меню проигрывателя Flash, пункт **Print** в нем будет недоступен.

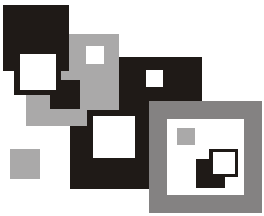
Во-вторых, можно вообще убрать контекстное меню при публикации фильма. Для этого достаточно отключить флажок **Display menu** на вкладке **HTML** диалогового окна **Publish Settings** (см. рис. 21.3). Вот только после этого пользователь вообще не сможет проделать с нашим фильмом никаких действий: ни остановить его воспроизведение, ни запустить его снова, ни задать качество вывода графики.

Что дальше?

Вот мы и познакомились со всеми дополнительными возможностями Flash, которые могут пригодиться в работе. Во всяком случае можно не сомневаться, что панель **Find and Replace** нам точно пригодится.

А в завершение мы рассмотрим возможность настройки среды программы Flash под привычки пользователя. Последняя в этой книге глава посвящена именно этой, весьма важной теме.

Глава 23



Настройка Flash

Разработчики практически каждого программного продукта стремятся сделать свое детище популярным, распространить его между как можно большим числом пользователей. Именно поэтому любой более или менее серьезный программный продукт предоставляет возможности настройки своей среды под привычки и потребности конкретного пользователя. А иначе нельзя — в противном случае продукт просто не сможет занять сколько-нибудь широкую нишу на рынке программного обеспечения; им будет пользоваться только кучка фанатичных приверженцев.

И наш любимый программный продукт Macromedia Flash 8 предлагает пользователю весьма богатый спектр возможностей настройки своей среды "под пользователя". Эти возможности позволят нам приспособить Flash под свои вкусы и привычки и сделать его более дружественным к нам.

Общие настройки

Все пользовательские настройки программы Flash выполняются в диалоговом окне **Preferences** (рис. 23.1). Чтобы вызвать его на экран, достаточно выбрать пункт **Preferences** меню **Edit** или нажать комбинацию клавиш <Ctrl>+<U>.

В левой части этого окна находится список **Category**, отображающий категории параметров Flash. Рассмотрим по очереди те из них, что могут нам пригодиться.

Категория *General*

Категория **General** диалогового окна **Preferences** (см. рис. 23.1) позволяет задать основные параметры программы.

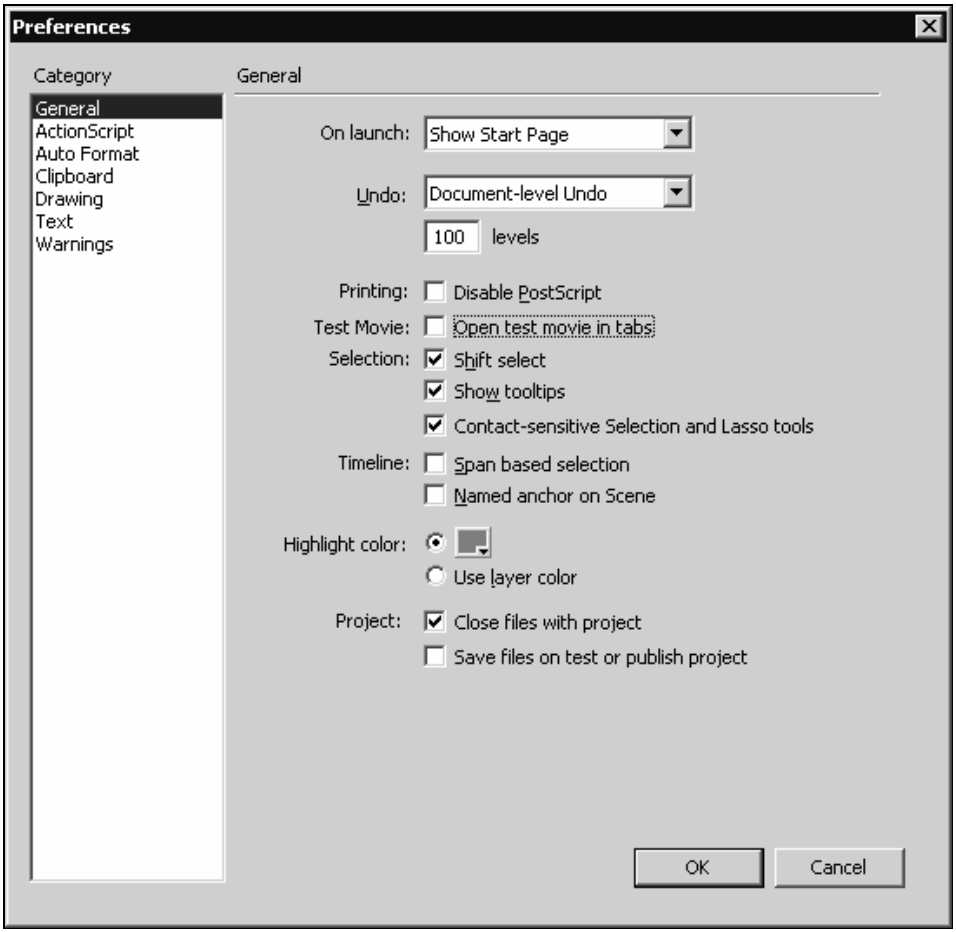


Рис. 23.1. Диалоговое окно **Preferences** (категория **General**)

Раскрывающийся список **On launch** задает поведение Flash сразу после запуска. Он содержит четыре пункта:

- **No document** — ничего не происходит;
- **New document** — создается новый документ;
- **Last documents open** — открываются все документы, которые были открыты при завершении работы программы;
- **Show Start Page** — отображается стартовая страница. Этот пункт выбран по умолчанию.

Раскрывающийся список **Undo** задает режим работы отката. Если выбран пункт **Document-level Undo**, то Flash будет сохранять список всех выполнен-

ных в документе операций, которые потом можно будет отменить, выполнив, собственно, операцию отката. Этот пункт, кстати, выбран по умолчанию. Пункт **Object-level Undo** при выборе заставляет Flash создавать отдельный список выполненных операций для каждого фрагмента изображения; это значит, что мы в этом случае можем выполнить откат для выделенного фрагмента, не затрагивая другие.

В поле ввода **levels**, расположенном под списком **Undo**, задается количество выполненных пользователем операций, сведения о которых Flash хранит в памяти. В большинстве случаев лучше оставить значение по умолчанию (100), но, вообще, туда можно ввести любое значение в интервале от 2 до 9999. Ввод меньшего значения позволит сэкономить оперативную память компьютера (которой всегда не хватает), но в этом случае есть риск столкнуться с ситуацией, когда мы не сможем выполнить откат операции, совершенной достаточно давно. Если же ввести бóльшее значение, то мы будем чувствовать себя свободнее при различных экспериментах с Flash, так как всегда сможем выполнить откат, но созданный Flash список операций отнимет много памяти.

Включение флажка **Disable PostScript** запрещает PostScript-вывод при печати на принтере, поддерживающем PostScript. Это может замедлить печать, так что данный флажок стóит устанавливать только при возникновении проблем с печатью документа.

Если включить флажок **Open test movie in tabs**, Flash при воспроизведении фильма в отдельном окне выведет этот самый фильм не в отдельном окне, а на особой вкладке самого окна документа. По умолчанию этот флажок отключен, и Flash выводит фильм в отдельном окне просмотра фильма.

Если включен флажок **Shift select** (а он включен по умолчанию), то для множественного выделения нам придется удерживать нажатой клавишу <Shift>. Если же флажок отключен, нам не нужно удерживать клавишу <Shift> — Flash будет выделять фрагменты при последовательных щелчках на них мышью.

Включать или не включать этот флажок — на взгляд автора, дело вкуса. Однако нужно помнить, что множественное выделение с помощью клавиши <Shift> — негласный стандарт в Windows-приложениях, и лучше сразу к нему привыкнуть.

Флажок **Show tooltips** включает или отключает вывод всплывающих подсказок при наведении курсора мыши на кнопки и другие элементы управления панелей и инструментариев. Автор считает, что его лучше держать включенным (он включен по умолчанию), по крайней мере — начинающим пользователям.

Флажок **Contact-sensitive Selection and Lasso tools** при включении заставляет Flash при выделении перетаскиванием инструментами "стрелка" и "лассо"

целиком выделять фрагменты графики, даже если мышь их только коснется. Этот флажок включен по умолчанию. Если его отключить, для выделения нужных фрагментов нам придется выделить их целиком.

Включение флажка **Span based selection** заставляет Flash при щелчке на растянутом кадре выделять весь этот кадр. Если этот флажок отключен (а он отключен по умолчанию), то при щелчке будет выделен только тот подкадр, на котором мы щелкнули мышью.

Если включен флажок **Named anchor on Scene**, Flash будет делать первый ключевой кадр каждой сцены якорем. По умолчанию этот флажок отключен.

С помощью группы переключателей **Highlight color** можно задать цвет контуров, которые будут отображаться при включенном режиме наложения контуров. Если включен переключатель **Use this color**, мы сможем задать этот цвет с помощью расположенной правее этого переключателя кнопки-селектора цвета. Если же включен переключатель **Use layer color**, Flash будет использовать цвет, заданный в диалоговом окне **Layer Properties** (см. рис. 5.2).

Категория *ActionScript*

Категория **ActionScript** диалогового окна **Preferences** позволяет настроить параметры панели **Actions**. Ее содержимое показано на рис. 23.2.

Флажок **Automatic indentation** включает или отключает создание автоматических отступов для строк кода. При этом размер отступа задается в символах в поле ввода **Tab size**. По умолчанию флажок **Automatic indentation** включен, а величина отступа равна четырем символам.

Флажок **Code hints** включает или отключает выдачу подсказок по коду. Задержка перед появлением подсказки устанавливается с помощью регулятора **Delay**. По умолчанию флажок **Code hints** включен, а величина задержки равна нулю (нет задержки).

С помощью двух раскрывающихся списков, расположенных в группе **Text**, задаются параметры шрифта, которым выводятся тексты сценариев ActionScript. В левом списке задается шрифт текста, а в правом — его размер. Для отображения текста программы рекомендуется выбирать моноширинные шрифты, например, Courier New или Lucida Console, — набранный ими сценарий будет выглядеть лучше.

При работе со сценариями, содержащими символы разных языков, нужно будет включить флажок **Use dynamic font mapping**. В противном случае его лучше отключить, чтобы Flash работал быстрее.

Флажок **Syntax coloring** позволяет включить или отключить цветовую подсветку различных участков кода сценария. Это очень полезно для начинающих

пользователей, поэтому данный флажок изначально включен. А шесть селекторов цвета, расположенных ниже этого флажка, позволяют задать цвет:

- **Foreground** — всего текста программы, не относящегося к тому, что перечислено ниже;
- **Keywords** — ключевых слов ActionScript;
- **Identifiers** — имен переменных, функций и объектов;
- **Background** — фона области редактирования кода в панели **Actions**;
- **Comments** — комментариев;
- **Strings** — строковых констант.

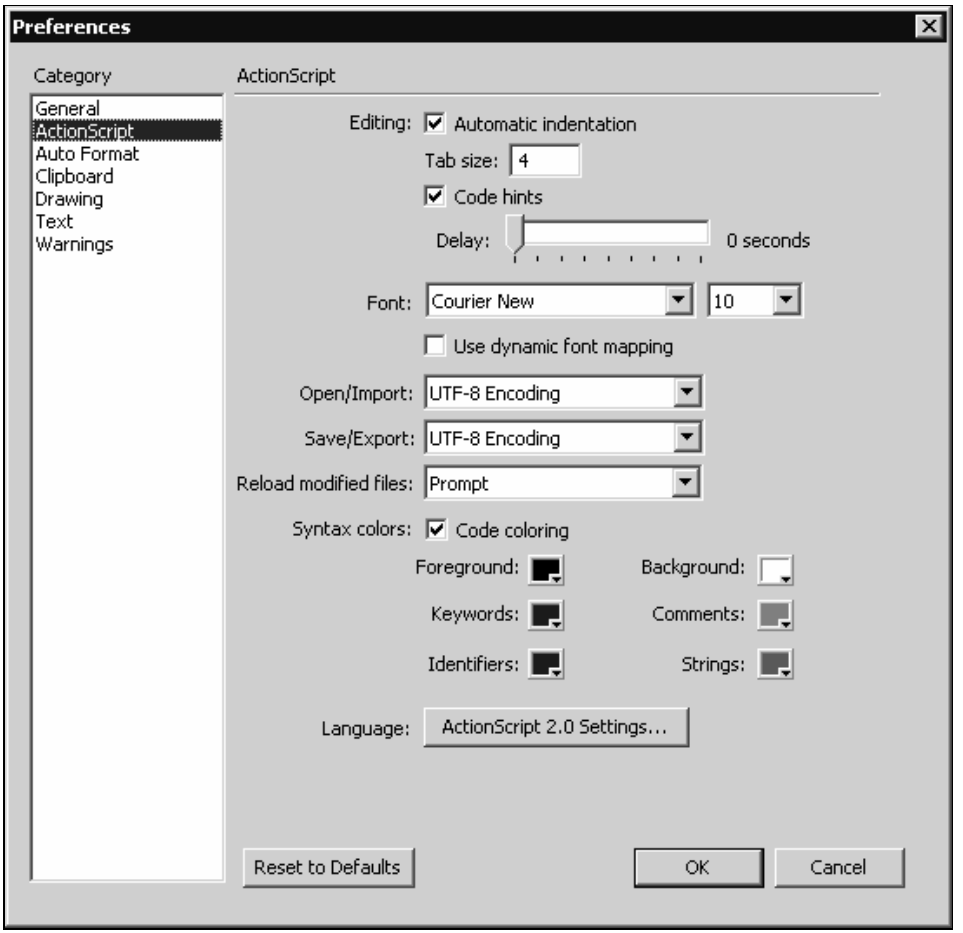


Рис. 23.2. Диалоговое окно **Preferences** (категория **ActionScript**)

Нажатие кнопки **Reset to Defaults** приводит к установке настроек панели **Actions** по умолчанию.

Категория *Clipboard*

Категория **Clipboard** диалогового окна **Preferences** позволяет настроить параметры операций, связанных с буфером обмена Windows. Ее содержимое показано на рис. 23.3.

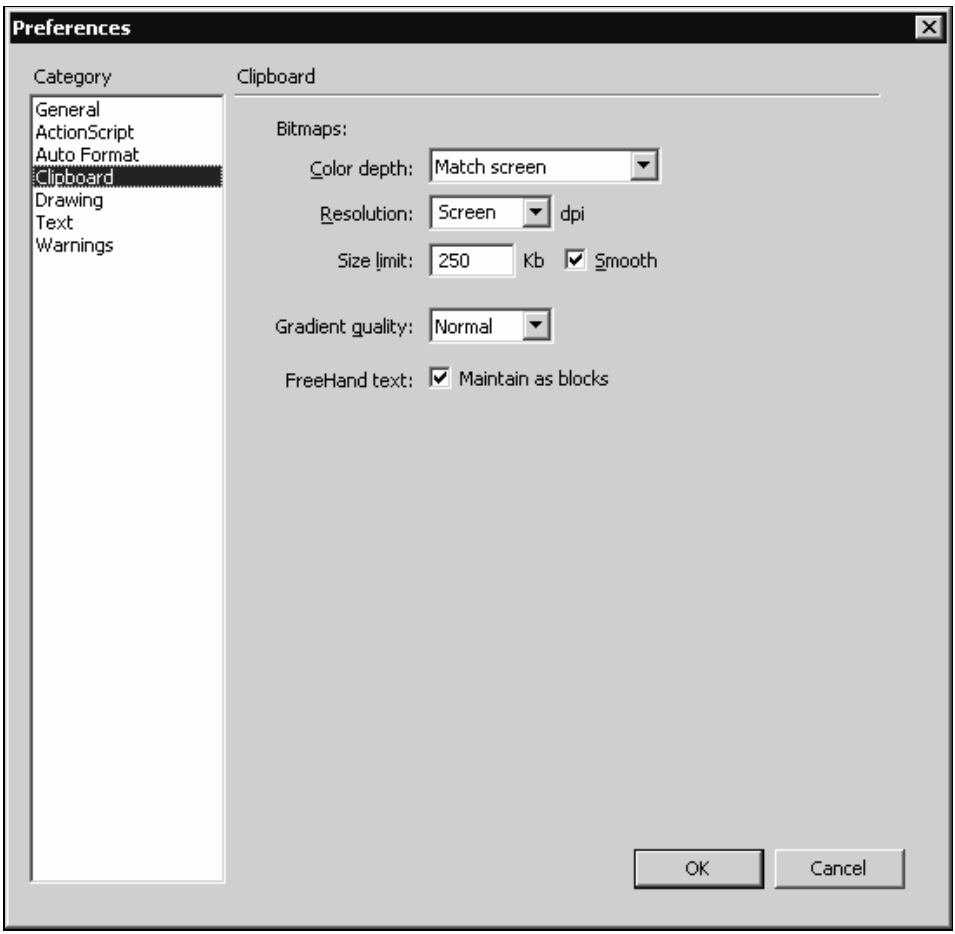


Рис. 23.3. Диалоговое окно **Preferences** (категория **Clipboard**)

Группа элементов управления **Bitmaps** позволяет задать параметры изображения, копируемого в буфер обмена, в растровом формате.

Раскрывающийся список **Color depth** служит для выбора цветности копируемого в буфер обмена изображения. В нем доступны семь пунктов:

- **None** — черно-белое изображение;
- **Match screen** — цветность такая же, как у текущего видеорежима (этот пункт выбран по умолчанию);
- **4 bit color** — четырехбитный цвет, 16 доступных цветов;
- **8 bit color** — восьмибитный цвет, 256 доступных цветов;
- **16 bit color** — 16-битный цвет, 65 536 доступных цветов (цветность HiColor);
- **24 bit color** — 24-битный цвет, 16 777 216 доступных цветов (цветность TrueColor);
- **32 bit color w/ alpha** — 24-битный цвет с каналом прозрачности (альфа-каналом).

Раскрывающийся список **Resolution** служит для задания разрешения растрового изображения. Выбранный по умолчанию пункт **Screen** задает то же самое разрешение, что и у экрана монитора. Остальные три пункта позволяют задать разрешение в 72, 150 и 300 точек (пикселов) на дюйм. Также можно ввести нужное значение разрешения вручную прямо в этот список.

В поле ввода **Size limit** вводится максимальный размер памяти, отводимой под растровое изображение, в килобайтах. Чем больше этот размер, тем бóльшее по размерам графическое изображение может быть помещено в буфер обмена. Доступны для ввода значения от 20 до 5000 килобайт.

Флажок **Smooth** включает или отключает сглаживание контуров растрового изображения. Рекомендуется оставлять его включенным, однако следует помнить, что сглаживание может в некоторых случаях ухудшить качество изображения, особенно если оно имеет множество мелких деталей.

Группа элементов управления **Gradient quality** позволяет задать параметры градиентных цветов, содержащихся в копируемом векторном изображении. В нем присутствуют четыре пункта: **None** (градиентные цвета отсутствуют), **Fast** (низкое качество), **Normal** (значение по умолчанию) и **Best** (самое высокое качество). Чем выше качество градиентных цветов, тем больше памяти они отнимают и тем дольше выводятся на экран.

С помощью единственного элемента управления группы **FreeHand text** — флажка **Maintain text as blocks** — можно включить или отключить копирование текста в виде текстовых блоков. Если изображение, содержащее текстовые блоки, вставить потом в документ Macromedia Freehand, то содержащийся в блоках текст можно будет редактировать. По умолчанию этот флажок включен, и отключать его не рекомендуется.

Категория *Drawing*

Категория **Drawing** диалогового окна **Preferences** служит для настройки параметров рисования и правки графики на рабочем листе. Содержимое этой категории показано на рис. 23.4.

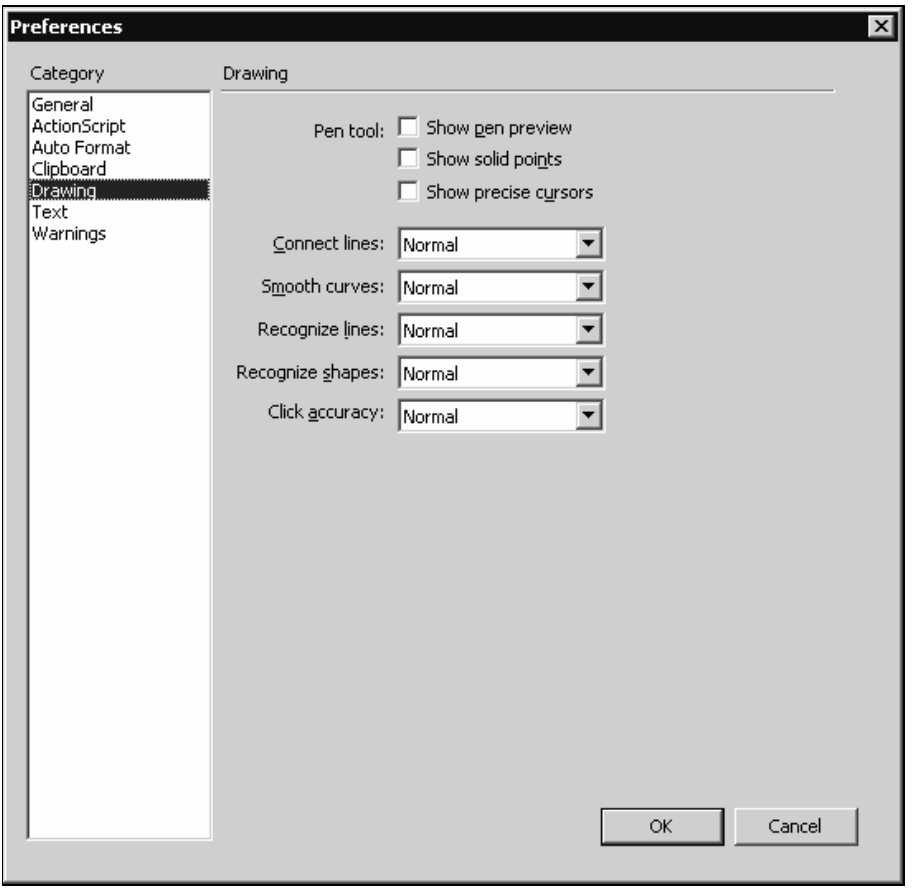


Рис. 23.4. Диалоговое окно **Preferences** (категория **Drawing**)

Группа флажков **Pen tool** позволяет задать параметры инструмента "перо". Давайте их рассмотрим.

Флажок **Show pen preview** включает показ линий, рисуемых с помощью инструмента "перо", т. е. при рисовании Flash будет показывать "резиновую" линию, тянущуюся от начальной точки до курсора мыши. Если же его отключить, то "резиновая" линия не показывается. Вывод "резиновых" линий может очень помочь при рисовании, но отнимает довольно много процес-

сорного времени, поэтому на очень медленных компьютерах данный флажок лучше отключить (по умолчанию он, кстати, и отключен).

Флажок **Show solid points** задает режим отображения угловых точек и точек искривления. Если он включен, невыделенные точки показываются в виде сплошных кружков и прямоугольников, а выделенные — в виде полых. Если же он отключен (а он отключен по умолчанию), то невыделенные точки, наоборот, показываются в виде полых кружков и прямоугольников, а выделенные — в виде сплошных. Включить или отключить этот флажок — на взгляд автора, дело вкуса и привычки.

Флажок **Show precise cursors** изменяет вид курсора мыши при выбранном инструменте "перо". Если он отключен (а он отключен по умолчанию), курсор мыши имеет вид чертежного рейсфедера. Если же этот флажок включить, курсор мыши примет вид небольшого прицела, что, по мнению разработчиков Flash, должно повысить точность его позиционирования. Включите этот флажок, если хотите "бить без промаха".

Раскрывающийся список **Connect lines** позволяет указать, как близко от уже нарисованной линии должна находиться рисуемая, чтобы Flash соединил их. В этом списке три пункта:

- **Must be close** — рисуемая линия должна быть рядом с нарисованной;
- **Normal** — обычное поведение; выбран по умолчанию;
- **Can be distant** — линии могут находиться достаточно далеко друг от друга.

Раскрывающийся список **Smooth curves** позволяет задать параметры сглаживания кривых, рисуемых с помощью инструмента "карандаш" при включенных режимах **Straighten** или **Smooth**. В нем четыре пункта: **Off** (нет сглаживания), **Rough** ("грубые" кривые), **Normal** (обычное поведение; выбран по умолчанию) и **Smooth** (гладкие кривые).

Раскрывающийся список **Recognize lines** позволяет указать, насколько линии, рисуемые с помощью инструмента "карандаш" при включенных режимах **Straighten** или **Smooth**, должны быть близки к прямым для того, чтобы Flash сделал их прямыми. В списке четыре пункта: **Off** (распознавание отключено), **Strict** (линии должны быть строго прямыми с минимальными отклонениями), **Normal** (обычное поведение; выбран по умолчанию) и **Tolerant** (отклонения от "прямизны" могут быть достаточно велики).

Раскрывающийся список **Recognize shapes** позволяет указать, насколько правильными должны быть фигуры, рисуемые с помощью инструмента "карандаш" при включенных режимах **Straighten** или **Smooth**, для того, чтобы Flash распознал их и преобразовал в классические геометрические фигуры. В списке четыре пункта: **Off** (распознавание отключено), **Strict** (фигуры должны быть правильными с минимальными отклонениями), **Normal** (обычное поведение; пункт выбран по умолчанию) и **Tolerant** (отклонения могут быть достаточно велики).

Раскрывающийся список **Click accuracy** определяет, насколько близко курсор мыши (при выбранном инструменте "стрелка") должен находиться от фрагмента изображения для того, чтобы Flash при щелчке выделил этот фрагмент. В списке три пункта: **Strict** (курсор мыши должен находиться точно над графическим фрагментом), **Normal** (обычное поведение; выбран по умолчанию) и **Tolerant** (отклонение может быть достаточно велико).

Категория *Text*

Категория **Text** диалогового окна **Preferences** (рис. 23.5) служит для настройки текста и текстовых блоков.

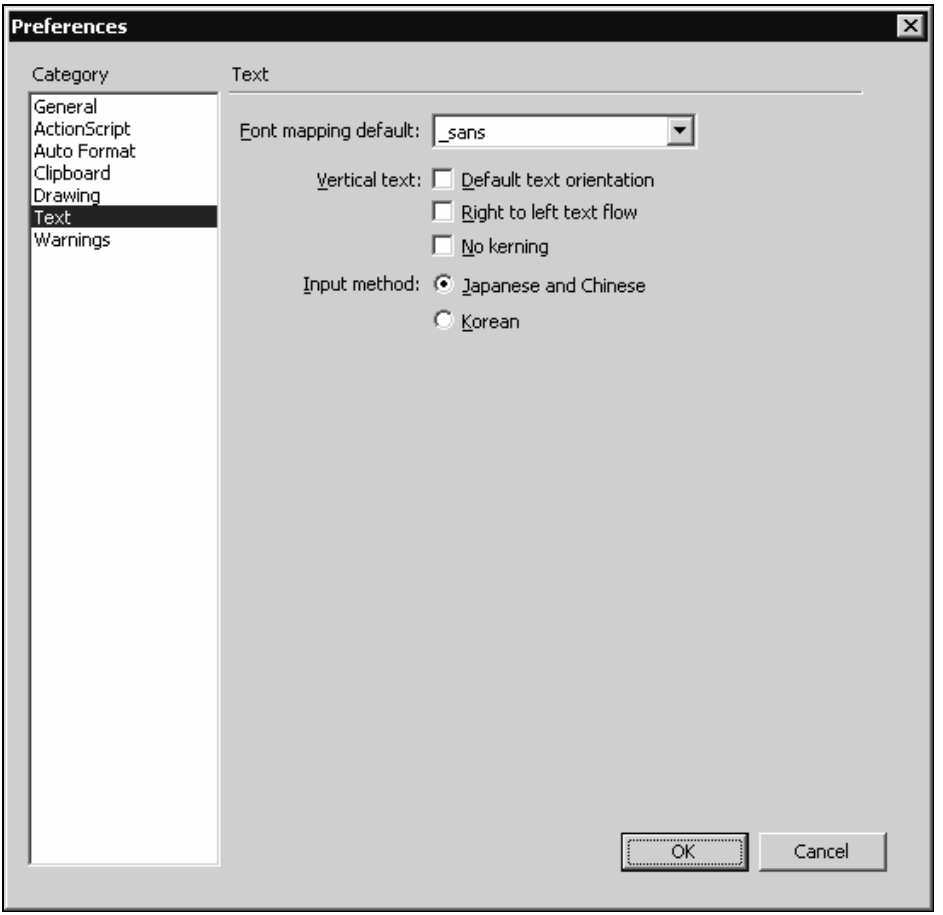


Рис. 23.5. Диалоговое окно **Preferences** (категория **Text**)

Здесь нам будет полезен только раскрывающийся список **Font mapping default**, задающий шрифт по умолчанию, используемый для подстановки отсутствующих шрифтов. Остальные же элементы управления задают параметры текста, набранного на восточных языках, и нам на первых порах вряд ли пригодятся.

Категория *Warnings*

Категория **Warnings** диалогового окна **Preferences** позволяет включить или отключить выдачу различных предупреждений. Содержимое этой категории показано на рис. 23.6.

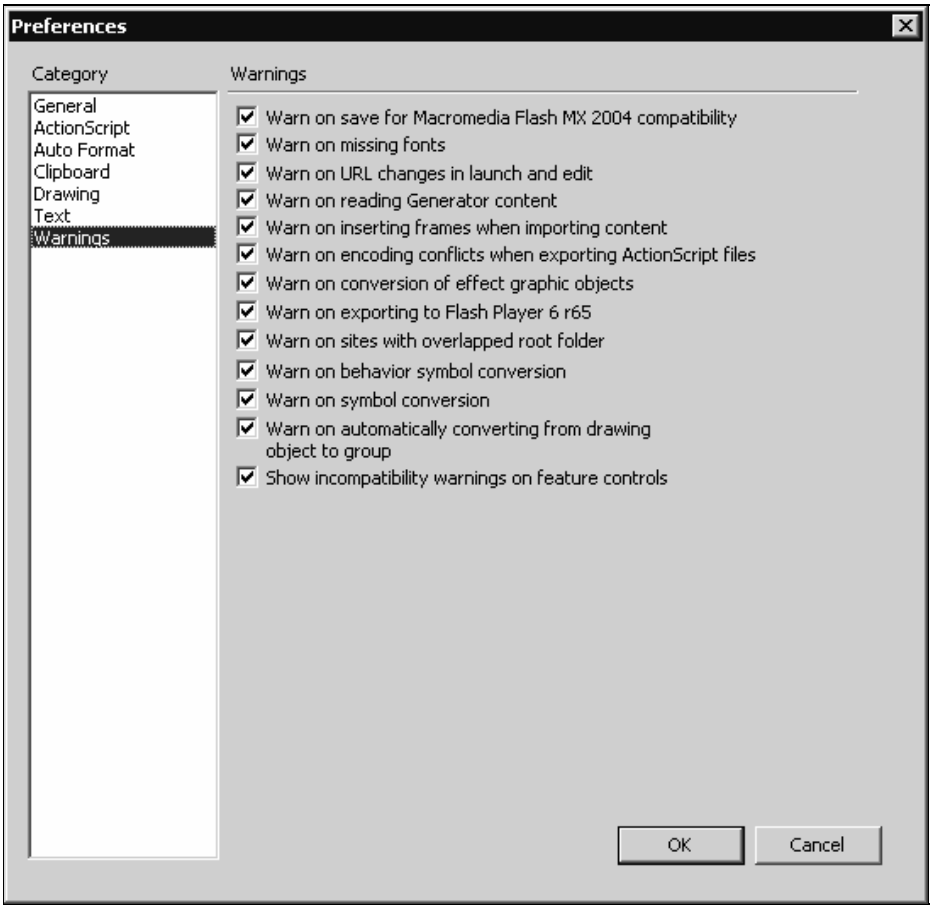


Рис. 23.6. Диалоговое окно **Preferences** (категория **Warnings**)

Здесь расположена большая группа флажков, назначение некоторых из которых описано в табл. 23.1. Заметим, что все флажки этой вкладки по умолчанию включены.

Таблица 23.1. *Некоторые флажки категории* ***Warnings*** *диалогового окна* ***Preferences***

Название флажка	Назначение
Warn on save for Macromedia Flash MX 2004 compatibility	Включает или отключает выдачу предупреждения, выдаваемого при сохранении в формате Flash MX 2004 документа, который не может быть правильно сохранен в этом формате
Warn on missing fonts	Включает или отключает выдачу предупреждения, выдаваемого при открытии документа, содержащего текст, написанный шрифтами, не установленными на компьютере
Warn on URL changes in launch and edit	Включает или отключает выдачу предупреждения при изменении интернет-адреса документа
Warn on reading Generator content	Если установлен, Flash помечает каждый образец, созданный в программе Macromedia Generator, красным крестиком
Warn on inserting frames when importing content	Включает или отключает выдачу предупреждения, выдаваемого при импорте аудио- и видеофайлов и добавлении при этом в фильм дополнительных кадров
Warn on conversion of effect graphic objects	Включает или выключает выдачу предупреждения при правке образца, к которому был применен эффект или преобразование (об эффектах и преобразованиях было рассказано в *главе 15*)
Warn on exporting to Flash Player 6 r65	Включает или отключает выдачу предупреждения при сохранении документа в формат Shockwave/Flash версии 6 r65
Warn on behavior symbol conversion	Включает или отключает выдачу предупреждения при смене типа образца, к которому было применено поведение (о поведениях было рассказано в *главе 18*)
Warn on symbol conversion	Включает или отключает выдачу предупреждения при смене типа образца
Warn on automatically converting from drawing object to group	Включает или отключает выдачу предупреждения при группировке фрагмента изображения, нарисованного в режиме объектного рисования
Show incompatibility warnings on feature control	Включает или отключает выдачу предупреждения, если выбранная в диалоговом окне **Publish Settings** версия проигрывателя Flash не поддерживает какие-либо элементы нашего фильма

Осталось сказать, что, закончив установку параметров Flash, нужно нажать кнопку **OK**. Чтобы отказаться от заданных параметров Flash, нужно нажать кнопку **Cancel**.

Настройка комбинаций клавиш

Комбинации клавиш, используемые во Flash для быстрого выполнения различных операций без вызова меню, настраиваются в диалоговом окне **Keyboard Shortcuts** (рис. 23.7). Чтобы вызвать его на экран, нужно выбрать пункт **Keyboard Shortcuts** меню **Edit**.

Рис. 23.7. Диалоговое окно **Keyboard Shortcuts**

Нужно сказать сразу, что Macromedia Flash 8 поставляется с шестью предопределенными наборами комбинаций клавиш. Каждый из этих наборов объединяет комбинации клавиш, применяемые в шести популярных графических программах (включая и сам Flash). Выбирается нужный набор с помощью раскрывающегося списка **Current**, в котором доступны шесть пунктов:

- **Macromedia Standard** — стандартный набор комбинаций клавиш, используемый по умолчанию во Flash 8;
- **Fireworks 4** — комбинации клавиш Macromedia Fireworks 4;
- **Flash 5** — комбинации клавиш, используемые в Macromedia Flash 5;
- **Freehand 10** — комбинации клавиш Macromedia Freehand 10;
- **Illustrator 10** — комбинации клавиш Adobe Illustrator 10;
- **Photoshop 6** — комбинации клавиш Adobe PhotoShop 6.

Если нам не подходит ни один из этих предопределенных наборов, мы можем создать свой на базе одного из них. Для этого выберем в списке **Current** набор, наиболее близкий к нашим потребностям, и нажмем кнопку , расположенную правее этого списка. На экране появится диалоговое окно **Duplicate** (рис. 23.8). Введем в единственное поле ввода **Duplicate name** этого окна имя создаваемого набора и нажмем кнопку **OK**. Нажатие кнопки **Cancel** отменяет создание нового набора.

Рис. 23.8. Диалоговое окно **Duplicate**

Сразу после создания нового набора его имя появится в раскрывающемся списке **Current**. Теперь мы можем изменять комбинации клавиш, входящие в этот набор, как нам заблагорассудится.

Прежде всего, нужно задать с помощью раскрывающегося списка **Commands** группу комбинаций клавиш, которую требуется изменить. Этот список содержит семь пунктов:

- **Actions Panel Commands** — операции, доступные в панели **Actions**.
- **Drawing Menu Commands** — пункты главного меню, отображаемые в обычном режиме;

- **Mobile Menu Commands** — пункты меню окна просмотра фильма, предназначенного для мобильных устройств;
- **Script Edit Commands** — пункты меню, доступные при работе с окном кода (окно кода и файлы сценариев ActionScript описаны в интерактивной справке Flash);
- **Test Movie Menu Commands** — пункты меню окна просмотра фильма;
- **Timeline Commands** — операции, доступные в панели **Timeline**;
- **Tools Panel** — инструменты, доступные в главном инструментарии;
- **Workspace Accessibility Commands** — операции, с помощью которых выполняется переход с одного элемента управления на другой;

Если мы выберем один из пунктов списка **Commands**, соответствующий набор пунктов меню, инструментов или операций появится в иерархическом списке, расположенном ниже (см. рис. 23.7). Если были выбраны пункты **Drawing Menu Commands**, **Test Movie Menu Commands** или **Script Edit Commands**, то отображается иерархический список, "ветвями" которого являются меню и подменю, а пунктами — соответствующие пункты этих меню и подменю. Если же был выбран какой-либо из остальных пунктов, отображается обычный список, содержащий названия инструментов или операций.

Кроме названия собственно пункта меню, инструмента или операции, каждый пункт этого списка содержит также обозначения всех клавиатурных комбинаций, привязанных к нему (к ней). Так что мы сразу можем увидеть, подходит нам эта комбинация клавиш или нет.

Если мы выберем любой из пунктов иерархического списка, ниже его появится краткое описание пункта меню, инструмента или операции. А еще ниже, в списке **Shortcuts**, будут отображены все привязанные к нему (к ней) комбинации клавиш.

Чтобы изменить какую-либо комбинацию клавиш, выберем ее в списке **Shortcuts**. Выбранная комбинация клавиш появится в поле ввода **Press key**, расположенного под списком **Shortcuts**. Щелкнем мышью по содержимому этого поля и нажмем нужную комбинацию клавиш. После этого нажмем кнопку **Change**, и новая комбинация клавиш заменит старую в списке **Shortcuts**.

Чтобы добавить новую комбинацию клавиш к уже существующим, нажмем кнопку со знаком "плюс", расположенную над списком **Shortcuts**. В поле ввода **Press key** появится надпись `<empty>`. Так же, как и при изменении комбинации клавиш, выделим содержимое поля ввода **Press key** и нажмем нужную комбинацию клавиш. После этого, опять же, нажмем кнопку **Change**, и новая комбинация клавиш добавится в список **Shortcuts**.

Чтобы удалить ненужную или ошибочно введенную комбинацию клавиш, выберем ее в списке **Shortcuts** и нажмем кнопку со знаком "минус", расположенную над списком **Shortcuts**. Выбранная клавиатурная комбинация тотчас исчезнет из этого списка.

Мы также можем переименовать созданный нами набор комбинаций клавиш. Для этого выберем его в списке **Current** и нажмем кнопку , расположенную правее этого списка. На экране появится диалоговое окно **Rename**, аналогичное окну **Duplicate** (см. рис. 23.8). Введем в единственное поле ввода **New name** новое имя набора и нажмем кнопку **OK**. Если нажать кнопку **Cancel**, Flash оставит набору старое имя.

Разумеется, Flash предоставляет также возможность удалить ненужный или ошибочно созданный набор комбинаций клавиш. Для этого нажмем кнопку , расположенную правее списка **Current**. На экране появится диалоговое окно **Delete Set** (рис. 23.9). Выберем в списке, занимающем бóльшую часть этого окна, нужный набор и нажмем кнопку **Delete**; если нажать кнопку **Cancel**, ни один набор не будет удален.

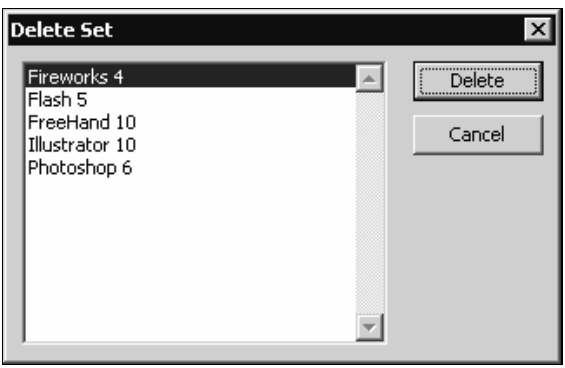


Рис. 23.9. Диалоговое окно **Delete Set**

Внимание!

Набор **Macromedia Standard**, поставляемый вместе с Flash, невозможно ни переименовать, ни удалить.

Еще одна возможность, появившаяся только во Flash 8, — экспорт набора комбинаций клавиш в виде Web-страницы. Такая Web-страница будет содержать набор таблиц, каждая из которых соответствует одной из групп комбинаций клавиш и содержит все входящие в эту группу комбинации клавиш.

Чтобы выполнить экспорт набора, его нужно будет выбрать в списке **Current** и нажать кнопку . На экране появится стандартное диалоговое окно сохранения файла Windows; вводим в него имя Web-страницы и нажимаем кнопку сохранения.

Создав новые наборы комбинаций клавиш или внеся изменения в старые, нужно нажать кнопку **OK** для их сохранения. Чтобы отказаться от изменений и вернуть предыдущие настройки, достаточно нажать кнопку **Cancel**.

Что дальше?

А дальше — ничего! Только несколько слов на прощание...

Заключение

Все в жизни рано или поздно кончается. Кончается зима, кончается терпение, кончается йогурт в пакете, даже сериал "Санта-Барбара", и тот закончился. Когда-нибудь закончится существование самой нашей Вселенной (правда, это будет нескоро). Что уж говорить о книгах, даже очень толстых...

Итак, автору больше нечего рассказать о создании графики и анимации в пакете Macromedia Flash 8.

Но все ли он рассказал? Нет — конечно, не все.

Очень многое осталось "за кадром". В книге ничего не рассказывалось о сложном программировании во Flash: использовании компонентов и написании Flash-приложений. Также ничего не было сказано о *расширениях* Flash — дополнительных модулях, подключаемых к основной среде и выполняющих какие-либо специальные задачи. Совершенно обойденной осталась тема автоматизации повторяющихся действий — а Flash предоставляет для этого весьма богатые средства.

Увы — пакет Flash 8 настолько сложен и объемен, что ему впору посвятить несколько книг, и немаленьких. Ибо невозможно объять необъятное, тем более в одной, пусть даже и немаленькой книге.

К счастью, вместе с Flash 8 поставляется весьма подробная интерактивная справка, в которой описано буквально все необходимое для работы с Flash. А если там чего-то недостает, то дополнительную информацию можно найти на сайте фирмы Macromedia и других сайтах, посвященных Flash. Благо, их сейчас много в Интернете...

Кстати, если уже речь зашла о сайтах, то вот список некоторых из них (табл. 31). Конечно, он не полон — слишком уж подвижная среда этот Интернет, слишком часто появляются там новые сайты (и, чего греха таить, пропадают старые).

Таблица 31. *Список интернет-ресурсов по теме книги*

Интернет-адрес ресурса	Описание
http://www.macromedia.com/	"Домашний" сайт фирмы Macromedia, разработчика Flash. Посвящен как самому Flash, так и другим продуктам этой фирмы (Dreamweaver, Fireworks, Director и др.)

Таблица 31 *(окончание)*

Интернет-адрес ресурса	Описание
http://msdn.microsoft.com/ie/	"Домашний" сайт Internet Explorer. Исчерпывающее описание Web-обозревателя фирмы Microsoft, особенностей поддержки им HTML, CSS, JavaScript и многого другого. К несчастью, читать эту документацию можно только с сайта
http://www.w3c.org/	"Домашний" сайт комитета W^3C. Все интернет-стандарты в удобном для чтения виде
http://www.actionscripts.org/	Большой "флэшерский" сайт. Уроки программирования на ActionScript, готовые исходные тексты сценариев и пр.
http://flashkit.com/	Еще большой один "флэшерский" сайт
http://www.computerarts.co.uk/flash/	Сайт английских "флэшеров". Обещается много чего...
http://subscribe.ru/, **http://www.maillist.ru/**	Русские серверы почтовых рассылок. Поищите на них рассылки, посвященные Flash, и сходной тематики
fido7.ru.flash	Группа новостей FIDO, посвященная Flash

Что ж, на этом — все. До свидания — на просторах Всемирной Паутины!

Владимир Дронов.
E-mail: **vlad@vgi.volsu.ru**
Web-сайт: **http://vgi.volsu.ru:8000/~Vladimir.Dronov/**

Предметный указатель

A

Accessibility 525
Actions 393, 404, 605
ActionScript 36, 391
ADPCM 373, 520
Advanced Effect 215
Align 130
AVI 31

B

Behaviors 409
Bitmap Properties 518
Blur 338
BMP 21

C

CGA 14
Character Embedding 200
CMYK 265
Color Mixer 160
Color Swatches 180
Convert to Symbol 210
Copy to Grid 334
Create New Profile 555
Custom Anti-Alias 194
Custom Download Settings 513
Custom Ease In / Ease Out 306

D

Deflate 23
Delete Preset 256
Delete Set 617
DIB 21
Distributed Duplicate 335
DivX 27
Document Properties 58
Drag'n'drop 461
  цель 462
Drop Shadow 338
Duplicate 615
Duplicate Profile 556
Duplicate Symbol 221
DVD-Video 11

E

Edit Envelope 384
Edit Snapping 107
Expand 339
Expand Fill 125
Explode 341
Export Adobe Illustrator 559
Export Bitmap 568
Export Flash Player 565
Export GIF 562, 563
Export JPEG 564
Export PNG 565
Export QuickTime 559
Export Windows AVI 560
Export Windows WAV 567

F

Filters 243
Find 74
Find and Replace 406, 407, 588
Fireworks 262
Fireworks PNG Import Settings 262
Font Mapping 204
Font Symbol Properties 239
Format Options 190
Freehand 263
FreeHand Import 263

G

GIF 21, 531, 541
  анимированный 22, 33, 531
GIF89A 22
Go to Line 408
Grid 70
Guides 68

H

Help 72, 360
HiColor 14
HSB 161
HTML 12, 34, 536
HTML Template Info 538
HTTP 500
HTTPS 500, 536

I

Import Options 266
Import Video 353, 354, 356, 357, 359, 365, 367
Info 134
Insert Target Path 451
Intel Indeo 27

J

Java 41
JavaScript 35
JPEG 23, 531, 545
  прогрессивный 23

K

KAR 375
Keyboard Shortcuts 614

L

Layer Properties 151
Library 211, 222, 229
Linkage Properties 235
LZW 22

M

Macromedia Flash Video 349, 370, 498
Magic Wand Settings 272
Manage Workspace Layouts 54
Metro 42
MIDI 375
Missing Font Warning 203
Movie Explorer 584
Movie Explorer Settings 586
MP3 27, 370, 374, 379, 497, 518
MPEG 32
  I 27
  I level 3 27, 370, 374
  II 27
  IV 27

N

New Document 56
New Folder 225

O

OGG Vorbis 374
On2 VP6 349
Optimize Curves 123
Output 406

P

Page Setup 598
PCM 373
PDF 266
PNG 22, 262, 531, 546
Preferences 602
Profile Properties 556
Properties 85, 110, 301, 380
Publish Settings 531

Q

QuickTime 11, 31, 530, 549

R

RealAudio 374
RealMedia 33
Rectangle Settings 83
Rename 617
Rename Panel Group 54
Rename Workspace Layout 54
Resolve Library Conflict 227
RGB 14, 161
RLE 21

S

Save Preset As 255
Save Workspace Layout 54
Scale and Rotate 142
Scene 294
Select Source Symbol 231
Shockwave/Flash 37, 529, 533
Soften Fill Edges 126
Sorenson Spark 348
Sound Properties 523
Sound Settings 520, 551
Stroke Style 175
Swap Bitmap 269
Swap Embedded Video 361
Swap Symbol 217
Symbol Properties 224, 231, 234

T

Timeline 148, 279
Tool Settings 85
Trace Bitmap 270
Transform 140, 145, 342
Transition 344
TrueColor 14, 15
  с каналом прозрачности 15
TrueType 19, 191
Type 1 191
Typeof 424

U

Unicode 415
Update Library Items 232

V

Video for Windows 11
Video Properties 363, 370
VideoCD 11

W

W3C 12
WAV 378
Web 12
Web-браузер 12
Web-дизайнер 34
Web-обозреватель 12
Web-страница 12, 499
Web-сценарий 35
WMA 374
WMV 32
World Wide Web 12
WWW 12
WWWC 12

А

Альфа-канал 15
Анимация:
  внешняя 319
  внутренняя 318
  зависимая 320, 360
  классическая 25
  многослойная 324
  многоуровневая 318
  независимая 320
  оборванная 302, 309
  покадровая 25, 277, 518
  составная 318
  трансформационная 28, 298, 518
Апплет Java 41

Б

Базовая линия 189
Бегунок 279, 358, 366
Безоконный режим 540
Библиотека 208, 229
  общего использования 241
  открытие 231
  разделяемая 233
Битрейт 349
Блок 428
Блокировка 146
Буфер обмена 108

В

Векторизация 19, 270
Видео 349
Видеоадаптер 10
  алфавитно-цифровой 10
  графический 10
Виртуальный код 484
Внедрение 352, 354, 363
Внедрение шрифтов 191, 193, 200
Внедренный элемент 570
Возврат 128
Воспроизведение 283, 295, 296, 380, 397, 399, 402, 452
  в отдельном окне 284, 296, 604
Вращение 141, 171, 216
Временная шкала 279, 358, 366, 384
Вставка 108, 261, 287, 588
  в исходную позицию 109
  с преобразованием формата 109
Выделение 63, 97, 100, 103, 113, 149, 154, 186, 193, 224, 272, 280, 283, 286, 294
  двойным щелчком 99
  множественное 99, 114, 149, 154, 604
  перетаскиванием мыши 99, 114, 604
  скрытие 100
  сложное 114
Выравнивание 129, 130, 194
  абзаца 189
  по пикселям 178
Выражение 393, 412
  блочное 428
  выбора 430
  математическое 413
  сложное 428
  условное 429
Вырезание 108, 287, 588

Г

Гиперссылка 196, 553
  цель 196
Горячая клавиша 526
Горячая область 401
Градиент 164
График 307, 384
Графика 569
  векторная 17
  гибридная 20
  пиксельная 13
  полноцветная 14, 15
  программная 466
  растровая 13, 518
  с палитрой 14, 15
  созданная программно 466
  статичная 12
  фотореалистичная 14
Графическое рисование 104, 137
Громкость 381, 383

Группа панелей 46
заголовок 46
закрытие 48, 53
переименование 53
перемещение 48
плавающая 46
развертывание 48
размеры 48, 53
ручка 46
сжатие 48
Группа фрагментов 103
Группировка 103

Д

Дата и время 447
Движение 299
параметры 303
создание 300
удаление 301
управление 306
Действие 393, 412
Декодер 28
Декремент 420
Деформация 143, 216
Диаграмма:
загрузки 515
Диалоговое окно:
выбора цвета 158
задания параметров сжатия видео 561
сведений о результатах оптимизации 124
специальной вставки 109
Диапазон наложения 290
размер 290
Док 48
скрытие 49
Документ:
Flash 39
возврат к последнему сохранению 71
заголовок 58
закрытие 72
описание 58
открытие 71, 603
параметры 58
пересохранение 71
размеры 58
создание 55, 56, 603
сохранение 71
Дорожка 279
Дорожка звука 380
Доступность 512, 523
Дублирование 105

Е

Единица измерения 60

З

Заголовок:
файла 14
Заливка 79, 91, 93
Замена 407, 588
Зацикливание 283, 285
Звук 376, 403, 489, 497, 518, 520
параметры 383
Звуковой эффект 381
Зеркальное отражение 135

И

Идентификатор:
разделяемого образца 234
таймера 491
Изображение:
замена 528, 531
фоновое 325
Изображение-заливка 167, 269
Импорт 259, 261
видео 352, 353, 363, 364, 371
звука 379
Имя:
альтернативное 525
кадра 304
образца 208, 210, 224, 227
папки 225
переменной 200, 413, 417, 419
поля ввода 198
рабочего окружения 54
слоя 149, 150, 151
сцены 295
функции 436

Инвертирование 288
Индекс 439
  цвета 14
Инкремент 420
Инструмент 50
  белая стрелка 112
  ведро с краской 91, 174
  карандаш 86, 610
  кисть 92, 174
  лассо 116, 272, 604
  ластик 117
  линия 81
  лупа 64
  многоугольник 84
  перо 88, 609
  пипетка 179
  прямоугольник 82
  рука 65
  стрелка 97, 604, 611
  текст 184
  трансформатор 139, 143
  трансформатор градиента 169
  чернильница 179
  эллипс 82
Инструментарий 50
  главный 50, 53
  документа 51, 53, 62
  перемещение 51
  проигрывателя 53
  ручка 51
  стандартный 51, 53
Интернет 11
Интерфейс:
  многодокументный 46
  однодокументный 46
Искажение формы 142, 216
История 75

K

Кадр 24, 279, 280
  длительность 287
  дублирование 287
  ключевой 28, 286, 369
  перемещение 286
  печатаемый 600
  примечание 304
  промежуточный 28, 301
  просмотр на месте 292
  растянутый 282, 605
  создание 281, 286
  текущий 279
  удаление 287
Канал 377, 384
  прозрачности 15
Карта-изображение 553
  область 553
Касание 464
Касательная 90, 114, 308
Качество 65, 245, 285
Кегль 188
Кернинг 189, 194
Клип 209, 242, 256, 326, 392, 449
  дублирование 455
  создание 455
  удаление 458
Ключевое слово 400, 412
Кнопка 208, 242, 392, 400, 455, 473
  выключатель 82
  отключения цвета 159
  переключатель 64
  с меню 82
Код:
  символа 415
Кодек 28
Кодер 28
Кодировка 415
Количество каналов 377
Командное кодирование 374
Командно-сэмплерное кодирование 375
Комментарий 428
Компонент ActiveX 41, 570
Константа 413, 416
Контур 79, 110, 112
Конфликт имен 227, 230
Координатная линейка 66
Координатная сетка 69
  параметры 70
Копирование 108, 287, 588, 608
Красная строка 190
Кривая Безье 88
Кэширование 326

Л

Линейка 165
Линия:
  свободной формы 86
  стиль 174, 178, 517
Литерал 595
Локальный хост 235

М

Макрос 579
Маркер:
  вращения 141, 171, 173
  диапазона 290
  изменения радиуса 173
  изменения размера 140, 170, 172, 185, 198
  искажения формы 142
  конечный 283
  конца звука 385
  конца клипа 358, 366
  морфинга 312
  начала звука 385
  начала клипа 358, 366
  огибающей 143, 385
  сдвига 141, 171
  смещения 170, 172
Маска 330
Массив 13, 439, 448
  вложенный 440
  размер 439
  элемент 439
Масштаб 63, 285
Масштабирование 16
Меню:
  главное 45
  дополнительное временной шкалы 291
  панели, дополнительное 48, 53
Метод 441
Микширование 382, 383
Модификатор 51, 419
  волшебная палочка 272
  вращение и сдвиг 141
  изменение размера 139
  искажение формы 142
  кран 121
  объектное рисование 104
  огибающая 143
  полигон 116
  приклеивание 81, 84, 86, 106
  размер кисти 93
  размер разрыва 92
  режим закраски 94
  режим стирания 119
  сглаживание 122
  сглаживание линий 87
  скругленные углы 83
  спрямление 123
  увеличение 64
  уменьшение 64
  фиксация заливки 174
  форма кисти 93
  форма ластика 118
Модуль расширения 41, 571
Мозаика 171
Морфинг 299
  параметры 312
  создание 310
  удаление 311
Мультимедиа 11, 372

Н

Набор фильтров 254
  выбор 255
  создание 254
  удаление 256
Набор фреймов 197
Наложение 256, 288
  контуров 289, 605
Направляющая 66, 107
  блокировка 67, 68
  параметры 68
  перемещение 67
  скрытие 68
  удаление 67, 69
Начертание 188

О

Область видимости 418
Область выделения 116

Область инструментария 51
вспомогательных инструментов 51
задания цвета 51, 80, 159
модификаторов 51
основных инструментов 51
Область печати 600
Обновление:
справки 75
Обработчик события 392, 399, 404, 453
заголовок 399
тело 399
Образец 207, 218, 517
анимированный 318
графический 208, 268, 518
дублирование 221
заглушка 238
звук 209, 379, 386
импорт 236
импортированный 209, 222, 236, 268
импортированный клип 209, 360, 362
источник 231
клип 209, 318, 320, 518
кнопка 208, 400
копирование 229
обновляемый 231
переименование 224
перемещение 225, 226
подтип 209
правка 218, 269
разделяемый 222, 233, 240
растровое изображение 209, 268
создание 209, 227, 234, 401
сценарный 456
тип 208, 210, 216, 224
удаление 224
шрифт 209, 239
Объединение 136
Объект 441
внешний 443
внутренний 443
встроенный 445
Объектное рисование 104, 135, 136
Огибающая 385
Окно:
активное 52
главное 45
документа 46, 52, 61, 148, 279
просмотра фильма 285, 513, 514, 604
Окружность выделения 172
Операнд 413
Оператор 413
арифметический 420
бинарный 420
комментария 428
логические 423
объединения строк 421
объявления переменной 418
присваивания 413
присваивания простого 421
присваивания сложного 421
сравнения 422
строгого сравнения 423
унарный 420
условный 430
Описание 525
Оптимизация 122, 511
многопроходная 124
однопроходная 124
палитры 542
Откат 128, 603
Отступ 406, 605
Оттенок 214
Оцифровка 26
Ошибка:
синтаксическая 406

П

Палитра 14, 157, 180
адаптивная 544
безопасная 182
загрузка 182
очистка 182
по умолчанию 181, 182
сохранение 181
Панель 46
активная 46, 48
закрытие 53
перемещение 53
скрытие 53

Панорамирование 381, 383
Папка 153, 225
  блокировка 154
  видимость 154
  переименование 154, 227
  перемещение 154, 227
  создание 153, 154, 225, 226
  схематичный режим 154
  удаление 154, 227
Параметр 393, 436
  фактический 437
  формальный 436
Перекрытие фильмов 495
Переменная 199, 202, 413, 417, 477
  временная 417
  глобальная 419
  локальная 419, 437
  объявление 418
  уровня клипа 418
Перемещение 105, 133
Перехватчик событий 480
Печать:
  документа Flash 598, 604
  панели Movie Explorer 588
  статьи 74
  сценариев 408
  фильма 600
Пиксел 13
Пиксельная сетка 106
Поведение 408
  категория 409
  правка 410
  создание 409
  удаление 410
Подкадр 282, 286, 605
Подклип 365
  переименование 367
  создание 366
  удаление 367
Подсказка:
  по коду 405, 605
  список 405
Подстановка шрифта 204, 612
Поиск 586, 588
  в коде сценария 406
  статей 74
Поле ввода 197, 476, 518
  многострочное 199
  однострочное 199
  пароля 199, 478
  с круговым регулятором 245
  с регулятором 162
Порядок обхода 474
  номер 474, 526
Порядок перекрытия 136, 150, 213
Потоковый звук 382, 383
Правка образца:
  в отдельном окне 220
  на месте 219
  обычный режим 220, 225
Предварительный просмотр 553
Преобразование 333
  Transform 342
  Transition 344
  правка 346
  удаление 346
Привязка 353, 355, 364
Привязка сценария 392
  к кадру 393
  к клипу 398
  к кнопке 404
Приклеивание 68, 69, 105
  выравнивающее 107
  к пикселам 106
Примитив 17, 62
Приоритет оператора 414, 426
Программа:
  серверная 502
Программное перо 467
Проектор 39, 530
Прозрачность 161, 214
Проигрыватель Flash 38
  автономный 39
  встраиваемый 38, 42, 570, 571, 573
Протокол 501
Профилировщик загрузки 513
Профиль 357
  выбор 556
  дублирование 556
  импорт 557
  правка 556
  публикации 555
  создание 555
  удаление 556
  экспорт 556
Прямая оцифровка 373

Прямоугольник выделения 99, 169, 185, 212
Публикация 39, 528, 529, 533
  состав 531
Пульт 284
Пункт:
  переключатель 65
Пункт меню:
  выключатель 53
Путь 443, 450
  абсолютный 450
  относительный 451

P

Рабочая область 62
Рабочее окружение 54
  выбор 54
  переименование 54
  создание 54
  удаление 54
Рабочий лист 62, 132, 300
Разбиение 103, 195, 218, 271
Размер изображения 14, 350
Размеры 133, 139, 141, 170, 216
Разрешение 16
Разрядность 377
Распределение 129, 131
  по слоям 155
Расширение 619
Регулярное выражение 590, 594
Регулятор 162, 246
Рекурсия 438
  бесконечная 438

C

Сброс трансформаций 145, 216
Свойство 441
Сглаживание 65, 87, 123, 178, 194, 608
Сдвиг 141, 171, 216
Селектор цвета 59, 157
Сериф 192
Сжатие 17, 26, 373, 518, 520
  без потерь 23
  с потерями 23, 26
Сигнал 382, 383
Символ:
  специальный 415
Синхронизация 382
Скобки 427
Слияние 102
Слой 148, 279
  блокировка 151
  видимость 150, 151
  маскируемый 330
  маскирующий 330, 472
  направляющая 327
  обычный 152
  переименование 150
  перемещение 150, 153, 154
  создание 149
  специальный 324
  схематичный режим 151
  тип 152
  удаление 150
Смена образца 217, 269, 310, 361, 592
Смеситель цветов 160
Смещение 170
Событие 392, 410, 453, 475, 479
Совпадение 464
Способ передачи данных 502
  GET 502
  POST 503
Справка 72
  по панели 54
  статья 72
Ссылка 440, 442
Стартовая страница 45, 603
Строка 414
Сцена 294
  дублирование 295
  переименование 295
  перемещение 295
  создание 294
  текущая 294, 296
  удаление 295
Сценарий 391, 412
Счетчик цикла 432
Сэмпл 375

Т

Таймер 490
Тег:
  HTML 199
  MP3 379, 498
  PNG 262
Текстовый блок 183, 242, 476, 482, 518
  вертикальный 195
  динамический 202, 476, 518
  закрытый 185, 187
  открытый 185, 186
  прокручиваемый 202
  с фиксированной высотой 185, 187
  с фиксированной шириной 186, 187
  создание 184, 482
  специальный 197, 476
  статический 202, 476
  удаление 482
Тип данных 414
  null 416
  undefined 416
  клип 449
  логический 416, 447
  объект 440, 443, 473, 476
  преобразование 424
  совместимость 424
  строковый 414, 445
  функция 479
  числовой 416, 446
Точка:
  искривления 110, 114, 308
  конечная 110
  отсчета 134, 210, 212, 216, 220
  приклеивания 106
  угловая 111, 113, 114
  фиксации 141
Траектория 327
Трансформация 129, 145, 195, 216
  свободная 144
  цвета 169
Трекер 376
Трекерный модуль 376
Трекинг 188

У

Удаление 105, 288, 588
  частичное 117
Указатель цвета 165
  добавление 166
Уровень:
  анимации 318
  фильма 496
Условие 429

Ф

Файл профиля публикации 556
Фильм:
  загружаемый 494
  перекрывающийся 495
Фильтр 242
  Adjust Color 253
  Bevel 249
  Blur 246
  Drop Shadow 244
  Glow 247
  Gradient Bevel 252
  Gradient Glow 250
  включение 254
  отключение 254
  перемещение 254
  создание 243
  удаление 254
Фокус ввода 474, 483
Фон 325
Формат:
  видео 9, 31, 351
  графики 9, 21, 260, 557
  звука 378
Форматирование 187
  абзаца 189
  символов 187
Фрагментация 100
Фрейм 197
Функция 435
  вызов 437
  глобальная 437
  обработчик события 479
  объявление 436
  тело 436
  уровня клипа 437

Ц

Цвет 80, 253, 518
вид 162
градиентный 164, 518
градиентный линейный 164, 171
градиентный многоцветный 166
градиентный радиальный 164, 172
графический 167, 169, 269, 272
добавление 162, 181
дублирование 181
зависимый 173
заливки 80, 110
индексированный 14
ключевой 164
код 158, 161
контура 80, 110
независимый 173
прозрачный 15
простой 163
режим задания 161
сложный 163
создание 159, 164, 167
составной 543
сплошной 162, 163, 518
удаление 181
фиксированный 173
фона 59
шрифта 188
Цветность 14, 608
Цветовая таблица 181
Цветовой набор 181
Цветовой эффект 213
Цикл 432
перезапуск 435
прерывание 435
просмотра 444
с постусловием 433
с предусловием 434
со счетчиком 432
тело 432

Ч

Частота кадров 24, 60, 350
Частота оцифровки 376
Чередование строк 22
Число 416
Чтение с экрана 523

Ш

Шаблон 56
HTML 536, 579
Ширина потока данных 349, 376
Шкала кадров 279
Шрифт 188, 191, 193, 518
псевдоним 192, 518
размер 188

Э

Экземпляр 208, 213
анимированный 318
создание 212, 360
тип 216
Экземпляр объекта 442
создание 442
удаление 443
Экспорт 39, 529, 558
Эффект 333
Blur 337
Copy to Grid 334
Distributed Duplicate 335
Drop Shadow 338
Expand 339
Explode 341
правка 346
создание 334
удаление 346

Я

Якорь 305, 537, 605
Яркость 214