

**Дмитрий Альберт
Елена Альберт**

**Macromedia
Flash
MX 2004**

Санкт-Петербург
«БХВ-Петербург»
2004

УДК 681.3.06
ББК 32.973.26-018.2
А56

Альберт Д. И., Альберт Е. Э.
A56 Самоучитель Macromedia Flash MX 2004. — СПб.: БХВ-Петербург,
2004. — 624 с.: ил.
ISBN 5-94157-415-0

Книга содержит всю информацию, необходимую для работы в среде новейшей версии популярной программы создания мультимедийных интернет-проектов Flash MX 2004. Подробно описаны элементы интерфейса, программный инструментарий и эффективные приемы его применения. Рассмотрены действенные принципы создания и использования графических объектов. Подробно освещены вопросы синтеза мультимедийных элементов: векторной и растровой графики, текста, видео и звука. Особое внимание уделено описанию технологии разработки анимации и различных подходов к ее реализации. Строгое и лаконичное описание объектно-ориентированного языка сценариев ActionScript, с разнообразными примерами его применения, вводит читателя в сферу мощных средств Flash-технологии. Многочисленные таблицы, иллюстрации и примеры позволят читателю систематизировать приобретенные знания и легко овладеть тонкостями профессиональной работы. Книга будет полезна как начинающему дизайнеру, так и опытному профессиональному, желающему освоить программу Flash MX 2004.

*Для широкого круга пользователей,
интересующихся вопросами компьютерного дизайна*

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Галина Смирнова</i>
Компьютерная верстка	<i>Наталья Смирновой</i>
Корректор	<i>Наталья Першакова</i>
Дизайн обложки	<i>Игоря Цырульникова</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 20.01.04.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 50,31.

Тираж 5000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

Содержание

Об авторах	1
Введение.....	3
ЧАСТЬ I. Основы. ВЕКТОРНЫЙ РЕДАКТОР.....	5
Глава 1. Что такое Flash?	7
Историческая справка	7
Преимущества Flash	8
Области применения Flash.....	9
Компоненты программы	10
Векторный редактор.....	10
Монтажная линейка Timeline	11
Программная среда ActionScript	11
Процесс создания Flash-фильма	11
Системные требования программы Flash MX 2004 и браузеры, поддерживающие проигрыватель Flash Player 7	14
Глава 2. Новые возможности Flash MX 2004.....	16
Приглашение к работе. Стартовая страница Start Page	16
Создание нового документа на основе шаблона	16
Усовершенствованная справочная система.....	17
Инструмент <i>Polystar</i>	17
Новые параметры работы инструмента <i>Brush</i>	17
Режим <i>Snap Align</i>	17
Поддержка форматов PDF и EPS	17
Усовершенствование функций импорта видеофайлов.....	18
Расширение возможностей работы с текстом.....	18
Функции <i>Find/Replace</i>	18
Встроенные эффекты монтажной линейки.....	19
Панель <i>History</i>	19
Запись макроса	19
Профили публикации	20
Автоматическая проверка наличия у пользователя проигрывателя Flash Player	20
ActionScript 2.0.....	20
Панель <i>Behaviors</i>	21

Глава 3. Интерфейс Flash MX 2004	22
Открытие и сохранение документа	22
Использование стартовой страницы Start Page.....	22
Сохранение документа.....	25
Обзор интерфейса	26
Главное меню.....	26
Стандартная панель.....	27
Рабочая область	28
Вспомогательная область.....	28
Полоса редактирования	28
Служебные панели	29
Панель инструментов.....	31
Инспектор свойств	33
Монтажная линейка	34
Контроллер	35
Библиотека.....	35
Палитра <i>History</i>	35
Инструменты масштабирования и перемещения по экрану	37
Изменение масштаба просмотра	37
Перемещение по экрану.....	39
Режимы отображения сцены.....	39
Настройки программы.....	40
Настройки горячих клавиш	49
Настройка глобальных параметров фильма	51
Использование учебных ресурсов Flash MX 2004	52
Глава 4. Инструментарий векторного редактора Flash MX 2004	55
Понятие о векторной графике.....	55
Элементы векторного объекта	57
Инструменты рисования и выделения.....	57
Блок управления цветом.....	57
Инструмент <i>Line</i>	59
Инструмент <i>Oval</i>	61
Инструмент <i>Rectangle</i>	61
Инструмент <i>Polystar</i>	62
Инструмент <i>Selection</i>	62
Инструмент <i>Lasso</i>	67
Инструмент <i>Pencil</i>	68
Инструмент <i>Brush</i>	69
Инструмент <i>Ink Bottle</i>	71
Инструмент <i>Paint Bucket</i>	72
Инструмент <i>Eyedropper</i>	73
Инструмент <i>Eraser</i>	73
Трансформация объектов	74
Инструмент <i>Free Transform</i>	74
Меню <i>Modify>Transform</i>	81

Палитра <i>Transform</i>	82
Палитра <i>Info</i>	83
Понятие о рабочем и наложенном уровнях	88
Рабочий уровень	88
Наложенный уровень	89
Выравнивание и распределение объектов	94
Палитра <i>Align</i>	94
Использование режима <i>Snap Align</i>	100
Использование режима <i>Snap to Objects</i>	101
Вспомогательные элементы: Rulers, Grid, Guides.....	102
Глава 5. Работа с контурами	105
Понятие о кривых.....	105
Инструменты <i>Pen</i> и <i>Subselection</i>	107
Инструмент <i>Pen</i>	107
Инструмент <i>Subselection</i>	111
Оптимизация кривых.....	112
Дополнительные возможности работы с векторными формами	115
Импорт векторных графических объектов и последовательностей в Flash.....	117
Импорт файлов Flash Movie.....	118
Импорт файлов FreeHand	119
Импорт файлов Adobe Illustrator.....	121
Глава 6. Работа с цветом	122
Понятие о цветовых моделях RGB и HSB	123
Цветовая модель RGB.....	123
Понятие о шестнадцатеричном представлении цвета (HEX).....	124
Цветовая модель HSB	125
Синтез сплошного цвета	126
Использование панели <i>Color Mixer</i>	126
Использование системной палитры <i>Color</i>	128
Работа с каталогом цветов <i>Color Swatches</i>	129
Синтез градиентной заливки	132
Создание градиента.....	133
Окрашивание градиентом.....	135
Редактирование градиента. Инструмент <i>Fill Transform</i>	136
Использование режима фиксации заливки <i>Lock Fill</i>	139
Глава 7. Работа с растровой графикой. Импорт видео	141
Понятие о растровой графике	141
Импорт растровой графики	142
Оптимизация растровых изображений	145
Работа с растровыми изображениями.....	147
Растровое изображение как объект наложенного уровня.....	148
Растровое изображение как объект рабочего уровня.....	149
Автотрассировка и ручная трассировка растровых изображений.....	157

Импорт и работа с видеофайлами.....	162
Внедрение видеофайлов в Flash-документ	163
Импорт связанных видеофайлов QuickTime movie	168
Работа с видеороликом как с объектом наложенного уровня	169
Глава 8. Работа с текстом	171
Текст в Flash	171
Создание и форматирование текста.....	172
Инструмент <i>Text</i> . Виды текстовых блоков	172
Импорт текста в Flash.....	174
Типы текста и настройка его параметров.....	174
Использование машинонезависимых шрифтов	186
Текст как объект наложенного уровня	187
Разбиение текста	190
Многоязыковая поддержка. Панель <i>Strings</i>	192
Использование общих шрифтовых ресурсов	199
ЧАСТЬ II. АНИМАЦИЯ.....	205
Глава 9. Монтажная линейка <i>Timeline</i>	208
Структура монтажной линейки	208
Работа со слоями	210
Типы слоев	215
Работа с кадрами	221
Виды кадров.....	222
Операции с кадрами	227
Метки, комментарии и указатели.....	234
Покадровая анимация.....	235
Калькирование. Множественное редактирование кадров	238
Глава 10. Символы Flash	240
Понятие символа.....	240
Типы символов	242
Создание и редактирование символов	244
Экземпляр символа и его свойства	254
Создание экземпляров	255
Трансформация экземпляров	255
Цветовые эффекты экземпляров	256
Свойства экземпляра типа <i>Graphic</i>	258
Свойства экземпляра типа <i>Movie Clip</i>	261
Свойства экземпляра типа <i>Button</i>	261
Изменение типа поведения экземпляра	262
Замена экземпляров	263
Импорт символов из других документов Flash.....	265

Глава 11. Работа с библиотекой Flash.....	267
Интерфейс библиотеки Flash.....	267
Операции с объектами библиотеки	269
Контекстное меню библиотеки	271
Стандартные библиотеки Flash.....	273
Работа с общими библиотеками (Shared Libraries).....	273
Глава 12. Основы классической анимации.....	277
Планирование фильма.....	277
Разработка образов персонажей	278
Понятие тайминга. Раскадровка.....	281
Основные принципы анимации	286
Вес.....	287
Форма	288
Отказное движение (замах)	289
Масштаб движения	290
Анимация походок	292
Анимационные эффекты	296
Огонь	296
Вода.....	297
Глава 13. Автоматическая анимация	301
Создание автоматической анимации	302
Анимация формы (морфинг)	302
Анимация движения.....	309
Разработка анимационных эффектов на основе автоматической анимации	319
Циклическое движение	319
Использование вложенных символов	322
Анимация растровой графики и текста	328
Анимация с применением маски.....	329
Встроенные эффекты монтажной линейки.....	337
Раздел <i>Assistants</i>	337
Раздел <i>Effects</i>	340
Раздел <i>Transform/Transition</i>	343
Глава 14. Работа со звуком	346
Основные параметры цифрового звука	346
Импорт звука в Flash	347
Типы синхронизации звука в Flash.....	349
Редактирование звука	352
Оптимизация звука	354
Использование звука в фильме	360
Экспорт звука	362
Глава 15. Работа над фильмом	363
Работа со сценами.....	363
Управление сценами	364

Использование макросов.....	365
Команды <i>Find/Replace</i>	369
Поиск и замена текстовой строки (<i>Text</i>)	371
Поиск и замена шрифта (<i>Font</i>)	372
Поиск и замена цвета (<i>Color</i>)	373
Поиск и замена экземпляра символа (<i>Symbol</i>), звука (<i>Sound</i>), видео (<i>Video</i>) или растрового изображения (<i>Bitmap</i>)	374
Использование Проводника <i>Movie Explorer</i>	374
Настройка отображения структуры документа	376
Редактирование элементов документа	378
Оптимизация фильма.....	380
ЧАСТЬ III. ТЕСТИРОВАНИЕ И ПУБЛИКАЦИЯ ПРОЕКТА	385
Глава 16. Тестирование фильма.....	388
Тестирование в рабочей среде	388
Работа в среде тестирования	390
Управление отображением и воспроизведением фильма	391
Эмуляция загрузки	392
Окно <i>Bandwidth Profiler</i>	393
Отчет о размерах элементов фильма	398
Глава 17. Публикация и экспорт документа	401
Настройка параметров публикации	401
Формат Flash (swf).....	404
Формат HTML (html/htm).....	407
Формат GIF Image (gif)	416
Формат JPEG Image (jpeg).....	420
Формат PNG Image (png)	421
Формат Windows Projector (exe).....	423
Формат Quick Time (mov)	423
Экспорт документа.....	426
Экспорт изображения	427
Экспорт фильма	428
Печать документа	430
Глава 18. Flash и HTML	433
Теги HTML, используемые для размещения Flash-ролика.....	433
Атрибуты и параметры тегов <i><object></i> и <i><embed></i>	436
Работа с шаблонами HTML	442
Разрешение вопросов политики безопасности	446
ЧАСТЬ IV. ИНТЕРАКТИВНОСТЬ	449
Глава 19. Введение в ActionScript	451
Представление о сценариях ActionScript	451
Использование панели <i>Actions</i> для создания сценариев	454

Встроенные функции управления воспроизведением	459
Функции воспроизведения и остановки.....	460
Функции выбора кадра	461
Функции безусловного перехода	462
Функция <i>trace()</i>	462
Программирование кнопок.....	463
Обработчики событий кнопок	463
Использование муви-клипов в качестве кнопок	466
Навигация внутри фильма при помощи монтажной линейки.....	468
Управление воспроизведением монтажной линейки	468
Организация структуры проекта.....	469
Создание сценариев	470
Навигация между фильмами при помощи гиперссылок	472
Организация структуры проекта.....	474
Создание сценариев	474
Представление об объектно-ориентированном программировании	475
Встроенные свойства клипов и кнопок.....	477
Программирование клипов	480
Обработчики событий клипов	481
Адресация клипов и кнопок	483
Абсолютная адресация	484
Относительная адресация	485
Предложение <i>with</i>	487
Управление автономным проигрывателем Projector	487
Поведения	489
Глава 20. Разработка сценариев	492
Переменные. Типы данных. Операторы	492
Создание переменных.....	492
Область видимости и срок жизни переменных	495
Понятие о типах данных и операциях с ними.....	498
Числовой тип данных	499
Строчный тип данных	502
Булев тип данных	506
Условные предложения	508
Предложения цикла	511
Предложение <i>while</i>	511
Предложение <i>for</i>	512
Применение вложенных циклов.....	513
Использование трехкадрового цикла	516
Создание пользовательских функций.....	517
Объявление функции	518
Создание литерала функции	520
Использование локальных переменных	521
Доступ к функции	521
Рекурсия	522

Массивы	523
Создание массива	523
Обращение к элементам и заполнение массива	524
Некоторые методы работы с массивами.....	526
Многомерные массивы	528
Глава 21. Применение сценариев	530
Использование методов обработчиков событий клипов и кнопок	530
Понятие об объектах <i>Listeners</i>	533
Использование динамического и пользовательского текста.....	535
Классы <i>TextField</i> и <i>TextFormat</i>	537
Основные свойства класса <i>TextField</i>	538
Динамическое создание и задание параметров текста.....	540
Обработчики событий объектов <i>TextField</i>	541
Динамическое форматирование текста. Класс <i>TextFormat</i>	542
Работа с клипами и кнопками.....	546
Перетаскивание объектов.....	546
Динамическое создание экземпляров	548
Класс <i>Math</i>	556
Свойства класса <i>Math</i>	557
Основные методы класса <i>Math</i>	557
Класс <i>Stage</i>	562
Класс <i>Mouse</i>	563
Методы класса <i>Mouse</i>	564
Класс <i>Key</i>	565
Методы класса <i>Key</i>	565
Свойства класса <i>Key</i>	567
Управление объектом при помощи клавиатуры	567
Контроль времени и даты	570
Загрузка внешних фильмов и изображений.....	573
Создание предзагрузчика.....	577
Создание слайд-шоу с динамической загрузкой изображений.....	579
Динамическое рисование при помощи ActionScript	583
Создание форм, состоящих из прямолинейных сегментов.....	583
Управление кривыми	587
Использование динамических масок	591
ЧАСТЬ V. ПРИЛОЖЕНИЯ	593
Приложение 1. Горячие клавиши и команды главного меню Flash MX 2004.....	595
Приложение 2. Интернет-ресурсы, посвященные Flash	607
Предметный указатель	608

Об авторах

Я хотел стать вторым Микеланджело, но обнаружил,
что не умею рисовать; хотел быть музыкантом, но обнаружил,
что не умею играть; пожелал стать певцом, но выяснилось,
что у меня нет голоса. Я не собирался писать,
это случилось само собой, помимо моего желания.

Бернард Шоу

Альберт Елена Эдуардовна закончила Ленинградский институт авиационного приборостроения. Второе высшее образование получила в Санкт-Петербургском государственном политехническом университете. Является автором курса "Векторная Графика и Анимация в Интернете. Macromedia Flash", который она преподавала на Факультете Переподготовки Специалистов Санкт-Петербургского государственного политехнического университета, начиная с 2000 года. Занимается живописью и компьютерным дизайном. Училась в студии изобразительного искусства "Контраст" под руководством Вячеслава Борисовича Шраги.



Альберт Дмитрий Июлевич закончил Санкт-Петербургский государственный политехнический университет. В течение года обучался в США (WA). Свыше двух лет преподаёт курс "Векторная Графика и Анимация в Интернете. Macromedia Flash", на Факультете переподготовки специалистов Санкт-Петербургского государственного политехнического университета. Занимается графикой и живописью. Учился в студии изобразительного искусства "Контраст" под руководством Вячеслава Борисовича Шраги. Увлекается музыкой, в свободное от написания компьютерной литературы время играет на ударных.



С любовью посвящаем эту книгу
нашим родителям и сыну Марку

Введение

*Docendo discimus.*¹

Жизнь — это движение. В основе движения лежит жизнь линии и формы. Художник подобен Пигмалиону, желающему вдохнуть жизнь в свою работу, наполнив ее движением.

Современный компьютерный дизайн динамичен, он требует использования адекватных средств для достижения своей цели. Дизайну необходимо движение. Книга, которую вы держите в руках, создавалась для того, чтобы помочь читателю освоить великолепный инструмент, дающий возможность воплотить любые творческие идеи и принести их другим людям при помощи Flash-технологии, позволяющей совместить графику, анимацию, звук, видео и интерактивность для создания ярких и запоминающихся мультимедиа-проектов.

Настоящая книга адресована как читателям, впервые знакомящимся с Flash-технологией, так и читателям, имеющим опыт работы с предыдущими версиями этой программы, но желающим систематизировать свои знания и освоить новые возможности, предлагаемые последней версией Macromedia Flash MX 2004. Книга будет полезна дизайнерам, работающим в области сетевых технологий и мультимедийных средств, художникам-аниматорам, а также всем интересующимся вопросами компьютерной графики и анимации.

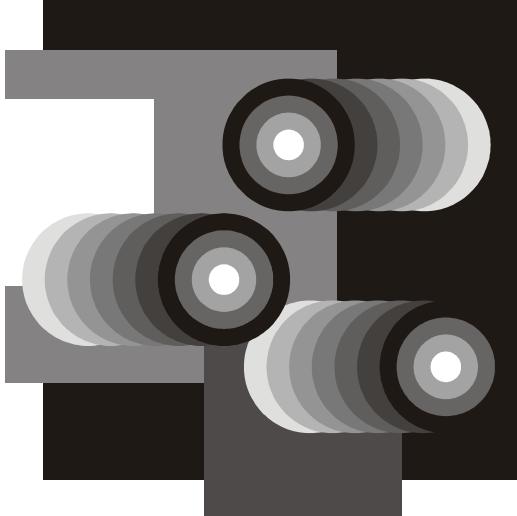
При написании книги авторы учли свой четырехлетний опыт преподавания Flash и направили все усилия на то, чтобы сделать повествование предельно ясным и полным. Книга содержит детальное изложение основных возможностей Flash MX 2004, описание приемов и методов эффективной работы с программой, пошаговые инструкции и практические примеры. Для облегчения восприятия и систематизации информации авторы снабдили текст многочисленными иллюстрациями, схемами и таблицами, демонстрирующими использование основных элементов программы и принципы работы с ней. Особое внимание было уделено вопросам, являющимся источниками ошибок у начинающих пользователей Flash.

¹Уча, мы сами учимся (лат.).

Авторы благодарят друг друга за взаимное воодушевление, которое не покидало их на протяжении всего процесса написания книги, а также своего полуторагодового сына Марка, который принимал во всем этом самое непосредственное участие.

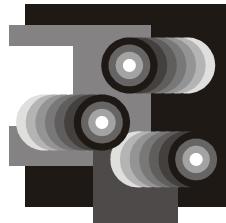
Авторы выражают благодарность художнику и своему учителю Вячеславу Борисовичу Шраге (www.shraga.da.ru) за открытие двери в мир Искусства.

Авторы будут также признательны отзывам, комментариям и пожеланиям, которые вы можете направлять по адресу: **book_flash@mail.ru**.



ЧАСТЬ I

**Основы.
ВЕКТОРНЫЙ РЕДАКТОР**



Глава 1

Что такое Flash?

Начинай взбираться вверх снизу.

Шумерская пословица

Историческая справка

История Flash началась в 1996 году с выхода программного пакета, который назывался Future Splash Animator. Новаторство этой разработки заключалось в использовании векторного формата представления графических данных в сети Интернет. Это являлось многообещающим и, как показало будущее, весьма перспективным подходом.

Первые версии программы представляли собой достаточно несовершенные инструменты, однако интерес web-дизайнеров и разработчиков к новой технологии постепенно возрастал до тех пор, пока в 1999 году не вышла четвертая версия Flash. С этого момента начался настоящий бум Flash-технологии, приведший к тому, что векторная интерактивная анимация стала незаменима при разработке мультимедийных интернет-проектов. Причиной этого явилось включение в программу множества новых инструментов и средств создания графики и анимации, и (что было настоящим прорывом) встроенного языка сценариев ActionScript, позволяющего снабдить проект достаточно широкой интерактивностью.

С выходом пятой версии Flash существенным изменениям подвергся интерфейс программы. Разработчики снабдили программу новыми инструментами (в частности инструментом Перо), которые существенно облегчили жизнь дизайнера. Язык ActionScript, обогатившись целым арсеналом новых возможностей (объектно-ориентированная модель, точечный синтаксис, встроенные математические функции, поддержка массивов, поддержка XML и др.), трансформировался в весьма развитой объектно-ориентированный язык, позволяющий разработчикам решать достаточно сложные задачи.

Весной 2002 года фирма Macromedia анонсировала следующую версию ставшего популярным пакета — Macromedia Flash MX. С выходом этого приложения дизайнеры получили более совершенный интерфейс (в частности, появился Инспектор свойств), усовершенствованный синтез цвета, но-

вые возможности работы с видеофайлами и др., но главным образом нововведения затронули язык сценариев ActionScript. В частности, были усовершенствованы событийная и объектная модели языка, появились возможности динамической работы с текстом, программного рисования, динамической загрузки изображений и звуков и др.

Новая версия Macromedia Flash MX 2004, выпущенная в свет в сентябре 2003 года, представляет собой чрезвычайно мощный и функциональный инструмент, позволяющий профессионально заниматься разработкой сложных мультимедийных интернет-проектов, отвечающих всем наиболее современным требованиям.

Преимущества Flash

Популярность Flash-технологии обусловлена целым комплексом характеристик самого приложения и создаваемой с его помощью конечной продукции.

- **Небольшой размер конечного файла** — является следствием использования векторного формата представления графической информации, эффективной технологии организации рабочего проекта, а также возможности компрессии (сжатия) конечного фильма. Размер файла является одним из наиболее критических параметров, определяющих эффективность того или иного интернет-проекта. Чем быстрее загружается из сети информация, тем больше шансов на то, что зритель дождется окончания загрузки, а не перейдет в нетерпении к другому более доступному ресурсу. Эффективность загрузки Flash-фильмов связана также с технологией *организации потока данных* (Data Streaming), в соответствии с которой в первую очередь загружаются те элементы, которые должны появиться вначале.
- **Качественное отображение графики и текста** — обусловлено применением технологии сглаживания (Antialiasing), а также новыми возможностями работы с текстом, которые предлагает Flash MX 2004. Сглаживание представляет собой процедуру, в результате которой краевые пиксели графических объектов смешиваются с пикселями фона, за счет чего эlimинируется эффект ступенчатости, характерный для растровой графики.
- **Независимость от масштабирования** — также является результатом использования векторной графики. Качество изображения не ухудшается от изменения масштаба просмотра. Для сравнения, если вы попробуете увеличивать растровое изображение, то на определенном этапе обнаружится эффект ступенчатости, станут видны отдельные элементы изображения. На практике независимость от масштаба позволяет простыми средствами реализовать весьма впечатляющие эффекты при работе над анимацией (например, наезд и панорамирование).
- **Мультимедийность** — включение в проект комплекса средств воздействия на зрителя. Сюда относятся визуальные образы (векторная и растровая

графика), анимация, видео, звук, интерактивность — возможность для зрителя взаимодействовать с проектом. Это позволяет добиться впечатляющих эффектов, привлечь к проекту широкую аудиторию.

- Устранение проблем совместимости между браузерами** — в отличие от HTML, Flash одинаково работает в различных браузерах. Во все современные браузеры, которые способны работать с графикой, включена поддержка Flash (Microsoft Internet Explorer 5.x, Netscape 4.7, Netscape 7, Mozilla 1.x, CompuServe 7, AOL 8, Opera 7.11).
- Широкая область применения** — применение Flash не ограничивается только сетью Интернет. Проект, реализованный во Flash, может распространяться на CD или иных носителях как исполняемый файл или видеофильм. Также при необходимости можно генерировать растревую последовательность или статичное изображение.
- Простота использования** — Flash MX 2004 обладает удобным интерфейсом. Технология использования его инструментальных средств достаточно прозрачна. Кроме того, во Flash MX 2004 имеется разветвленная справочная система, которая может периодически обновляться через Интернет. Все это позволяет начинающему пользователю быстро освоиться в рабочей среде программы.

Области применения Flash

Широчайшие возможности программы и в то же время сравнительная простота ее использования привели к тому, что Flash-технология стала поистине многофункциональной и нашла применение во множестве различных областей. С ее помощью осуществляется разработка разнообразных продуктов, в числе которых можно указать следующие:

- интерактивные ролики, заставки, сплэш-страницы с применением векторной анимации для сети Интернет;
- анимированные рекламные баннеры, обладающие интерактивностью;
- динамические web-страницы;
- динамические web-сайты, использующие комплексное представление информации (графика, анимация, звук), снабженные широчайшей интерактивностью;
- динамические элементы интерфейса web-сайта;
- игры и развлекательные порталы;
- программы дистанционного обучения, тесты, развивающие и образовательные программы;
- электронные on-line- и off-line-презентации;
- анимационные ролики (в том числе рекламные) для телевидения;

- мультимедийные интернет-открытки;
- портфолио;
- векторные рисунки.

Компоненты программы

Рабочую среду Flash MX 2004 условно можно разделить на следующие компоненты:

- векторный редактор;
- монтажная линейка (Timeline);
- программная среда ActionScript.

Векторный редактор

Векторный редактор Flash MX 2004 представляет собой инструментарий, позволяющий создавать векторные графические изображения в среде разработки и элементы интерфейса, обеспечивающие удобный доступ к различным инструментам. Среди инструментов векторного редактора можно выделить несколько групп, объединенных общим назначением:

- инструменты рисования** (**Oval** (Овал), **Rectangle** (Прямоугольник), **Line** (Линия), **Polystar** (Многоугольник), **Pencil** (Карандаш), **Brush** (Кисть), **Pen** (Перо)), с помощью которых непосредственно создается графическая информация;
- инструменты выделения** (**Selection** (Выделение), **Lasso** (Лассо), **Subselection** (Частичное выделение)), позволяющие организовать доступ к отдельным элементам или совокупностям графических объектов для их дальнейших преобразований;
- инструменты работы с цветом** (**Ink Bottle** (Чернильница), **Paint Bucket** (Ведро заливки), **Fill Transform** (Трансформация заливки), **Eyedropper** (Пипетка)), задающие цвет и различные параметры окрашивания графических объектов и их отдельных элементов;
- инструменты трансформации** (**Free Transform** (Свободное трансформирование)) — многофункциональный инструмент, позволяющий выполнять различные преобразования графических объектов;
- инструмент работы с текстом** (**Text** (Текст)), позволяющий снабдить проект информационным содержанием и задать параметры его применения;
- вспомогательные инструменты** (**Zoom** (Линза), **Hand** (Рука)), при помощи которых осуществляется изменение масштаба просмотра содержимого и перемещение по экрану.

Кроме перечисленных групп инструментов, векторный редактор включает в себя целый набор средств (панели, команды меню, диалоговые окна), обеспечивающих широкие возможности манипуляции графическими объектами.

Монтажная линейка Timeline

Монтажная линейка (Timeline) — это средство, позволяющее организовать изменение состояния графических объектов во времени, т. е. это инструмент, при помощи которого, собственно, и создается анимация. Монтажная линейка Flash включает в себя также палитру слоев, что позволяет решить множество вопросов, связанных с организацией структуры фильма (в частности, с использованием разных планов — переднего, среднего, заднего) и с одновременным выполнением нескольких анимационных процессов.

Программная среда ActionScript

Язык сценариев ActionScript (AS) — объектно-ориентированный язык программирования, встроенный во Flash, разработан на основе спецификации ECMA-262 (той же, что легла в основу языка клиентских сценариев JavaScript). Язык сценариев позволяет снабдить Flash-проект интерактивностью, заставить его взаимодействовать с пользователем. ActionScript используется для решения целого ряда задач: разработки навигации — перемещения между различными частями проекта, создания интересных динамических визуальных эффектов, обработки данных, введенных пользователем, создания интерактивной программной анимации и др.

Для удобства создания сценариев разработчики включили во Flash редактор ActionScript, который содержит множество удобных возможностей, таких как всплывающие подсказки, подсветка элементов языка, проверка синтаксиса и анализ ошибок, а также удобную и разветвленную справочную систему и др. Все это позволяет сделать процесс разработки сценария более комфорным.

Процесс создания Flash-фильма

В данной книге продукт, созданный в рабочей среде Flash, условимся называть *фильмом*. Таким образом, любой проект, будь то анимированный баннер, электронная презентация или интерактивная заставка, мы будем называть Flash-фильмом или роликом.

Технология разработки Flash-фильма включает в себя несколько этапов. Находясь в среде разработки, дизайнер создает так называемый *авторский файл* (документ). В ходе этого процесса разрабатываются визуальные элементы, цветовая схема проекта, а также создаются образы, которые находят воплощение при помощи векторного редактора программы. На данном этапе, если в этом есть необходимость, осуществляется импорт элементов из других

авторских файлов, а также импорт графики (векторной или растровой) из внешних редакторов, ее оптимизация и интеграция в рабочий проект.

Далее в дело включается аниматор. При помощи монтажной линейки он "оживляет" персонажи (или просто декоративные элементы). Формируется структура фильма. При необходимости осуществляется импорт видеофайлов или последовательностей, созданных во внешних приложениях.

Фильм снабжается фонограммой (звуковым сопровождением). Для этого необходимо импортировать звуковой файл (созданный заранее в одной из специально предназначенных для этого программ).

Наконец, фильм наполняется интерактивным содержанием. Диапазон применения языка сценариев может быть весьма разнообразен в зависимости от стоящих перед разработчиком задач. Дизайнер, хорошо владеющий рисунком, может создавать впечатляющие и вполне профессиональные проекты, используя лишь минимум средств, предлагаемых ActionScript. Очевидно, что в этом случае акцент должен быть сделан на разработке качественной художественной анимации, грамотного дизайнерского решения и выборе эффективного способа представления информации. Необходимость интенсивного использования программирования возникает, когда речь идет о разработке достаточно сложных схем интерактивного взаимодействия (игры, изощренные динамические элементы интерфейса) и обеспечении эффективного взаимодействия Flash-проекта с клиентским и/или серверным сценариями.

Документ Flash, называемый также авторским, или *исходным* файлом, сохраняется с расширением fla. Файл fla предназначен для работы в авторской среде Flash. Данный формат содержит всю графику, звук, информацию об организации внутренней структуры фильма и о применяемых разработчиком технологиях, а также сценарии ActionScript. Таким образом, располагая исходным файлом, можно разобраться в структуре и внутренней механике проекта. Исходный файл всегда имеет существенно больший объем, чем объем результирующего, конечного файла и никогда не покидает компьютера разработчика, если, конечно, нет особой договоренности с клиентом.

В качестве конечного продукта Flash MX 2004 может генерировать целый ряд файлов различных форматов.

В первую очередь, это формат SWF — что называется, "родной" для Flash. Файл обладает всей функциональностью, которой был снабжен исходный файл fla, и обладает существенно меньшим размером. Это достигается за счет реорганизации данных, осуществляющейся на этапе публикации (генерирования конечного файла). В ходе этой реорганизации упрощается структура фильма, неиспользованные элементы библиотеки отбрасываются, растровые изображения и звуки подвергаются компрессии, при этом также существует возможность дополнительно сжать и сам конечный файл. Файл SWF связывается с HTML-страницей при помощи специальных тегов, описывающих параметры его размещения и воспроизведения, после чего оба файла могут быть размещены в сети Интернет или интранет (intranet).

HTML-файл также может быть сгенерирован автоматически, причем все настройки могут быть заданы при помощи диалоговой формы, и, таким образом, разработчику не нужно заботится о создании соответствующего HTML-кода вручную.

В случае создания проекта, который должен распространяться как независимое приложение (например, электронная презентация или обучающая программа), удобно опубликовать исходный файл как исполняемый файл с расширением exe. Такой файл называется *проектором* (Projector) и включает в себя проигрыватель Flash Player, при помощи которого данный файл можно воспроизвести. Это удобная возможность в тех случаях, когда нет уверенности в том, что в файловой системе у конечного зрителя установлен проигрыватель Flash Player.

Если целью разработки является анимационный фильм, удобно сразу опубликовать его в видеоформате. Это можно сделать, используя видеоформат MOV (Quick Time Movie). Кроме того, можно экспорттировать фильм в один из наиболее широко используемых видеоформатов (AVI, FLV).

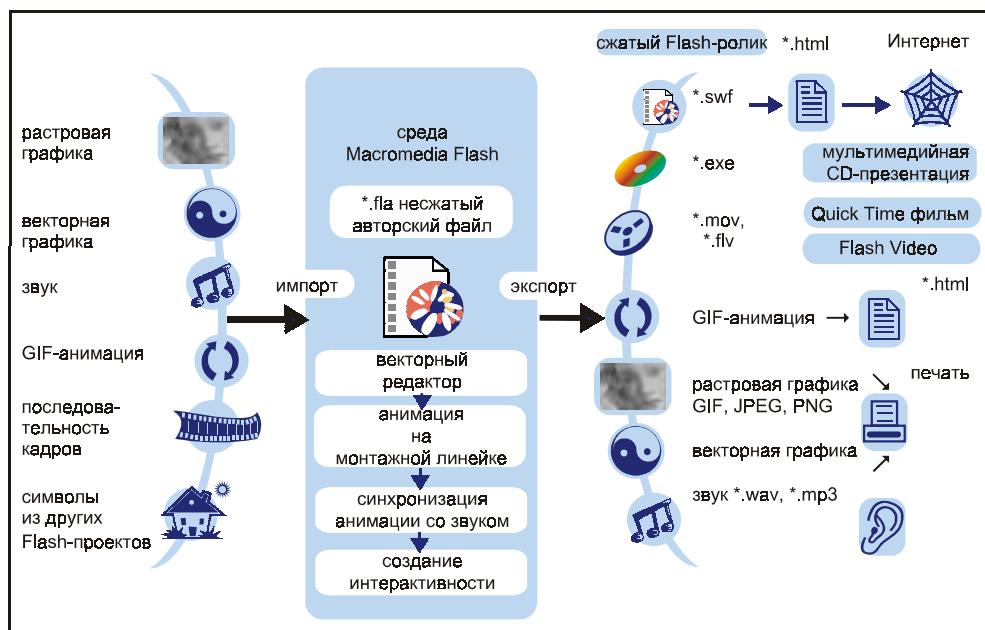


Рис. 1.1. Процесс создания Flash-проекта

Многие баннерные сети до сих пор работают с баннерами в формате GIF, в связи с чем может возникнуть необходимость публикации исходного доку-

мента в данный формат. Flash предлагает такую возможность. Также можно экспортить фильм как последовательность файлов (векторную или растровую) в один из наиболее распространенных форматов (EMF, VMF, EPS, JPG, BMP, PNG и др.).

Кроме указанных форматов, исходный файл можно опубликовать или экспортить, получив статичное изображение, сгенерированное на основе указанного кадра. При этом можно использовать один из наиболее широко применяемых форматов растровой или векторной графики (GIF, JPG, PNG, BMP, EPS, AI, WMF и др.).

Процесс создания Flash-фильма схематично представлен на рис. 1.1.

Системные требования программы Flash MX 2004 и браузеры, поддерживающие проигрыватель Flash Player 7

Для платформы **Windows** рекомендуются:

- процессор 600 МГц Intel Pentium III;
- операционная система Windows 98 SE, Windows 2000, Windows XP;
- 128 Мбайт оперативной памяти (желательно 256 Мбайт);
- 190 Мбайт свободного дискового пространства.

Для платформы **Macintosh** рекомендуются:

- процессор 500 МГц PowerPC G3;
- операционная система Mac OS 10.2.6;
- 128 Мбайт оперативной памяти (желательно 256 Мбайт);
- 190 Мбайт свободного дискового пространства.

Для воспроизведения Flash-фильмов на HTML-страницах проигрывателем Flash Player 7 в зависимости от операционной системы необходимо использовать следующие браузеры.

Браузеры для платформы **Windows 98**:

- Microsoft Internet Explorer 5.x;
- Netscape 4.7, Netscape 7.x;
- Mozilla 1.x;
- AOL 8;
- Opera 7.11.

Браузеры для платформы **Windows ME**:

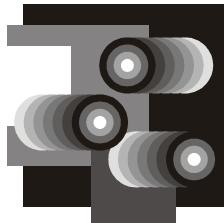
- Microsoft Internet Explorer 5.5;
- Netscape 4.7, Netscape 7.x;
- Mozilla 1.x;
- AOL 8;
- Opera 7.11.

Браузеры для платформы **Windows 2000**:

- Microsoft Internet Explorer 5.x;
- Netscape 4.7, Netscape 7.x;
- Mozilla 1.x;
- CompuServe 7;
- AOL 8;
- Opera 7.11.

Браузеры для платформы **Windows XP**:

- Microsoft Internet Explorer 6.0;
- Netscape 7.x;
- Mozilla 1.x;
- CompuServe 7;
- AOL 8;
- Opera 7.11.



Глава 2

Новые возможности Flash MX 2004

Знай я раньше то, что знаю теперь,
я бы теперь не знал этого.

Станислав Ежи Лец

С выходом новой версии программного пакета Flash в него был добавлен ряд нововведений, позволяющих повысить эффективность работы и расширить функциональные возможности. Дизайнер или разработчик, имеющий опыт работы с предыдущими версиями Flash, на первый взгляд, не заметит революционных отличий Flash MX 2004, однако при более длительном знакомстве обнаружит множество полезных новшеств, которые обеспечат лучший контроль над рабочим процессом и позволят полнее реализовать поставленные задачи.

В данной главе приведен краткий обзор новых возможностей Flash MX 2004, описание которых читатель сможет найти на страницах этой книги.

Приглашение к работе. Стартовая страница Start Page

При открытии Flash или закрытии исходного документа появляется стартовая страница, позволяющая выполнить набор действий, таких как: открытие недавно используемого файла, создание нового документа "с нуля" или выбор шаблона, на основе которого будет создаваться новый проект.

Создание нового документа на основе шаблона

Вместе с Flash MX 2004 поставляется целый набор шаблонов, которые могут служить основой при создании проекта. В комплект поставки включены

шаблоны для создания рекламных баннеров, презентаций, интерактивных тестов, слайд-шоу, фильмов для мобильных устройств (карманные ПК, сотовые телефоны) и др.

Усовершенствованная справочная система

Новая панель **Help** предоставляет удобные возможности вызова справки по требуемой теме. Кроме того, справочная информация может обновляться с сайта компании Macromedia.

Инструмент *Polystar*

Новый инструмент **Polystar** (Многоугольник) позволяет рисовать звезды и выпуклые многоугольники с произвольным числом сторон.

Новые параметры работы инструмента *Brush*

Инструмент **Brush** (Кисть) обогатился новым модификатором **Tilt** (Наклон), позволяющим активизировать чувствительность кисти к наклону при работе с графическим планшетом (дигитайзером). Еще одна новая возможность при работе с кистью заключается в появлении режима сглаживания (**Smoothing**), применение которого обеспечивает автоматическое сглаживание создаваемых векторных форм.

Режим *Snap Align*

Режим **Snap Align** (Притягивание с Выравниванием) используется при ручном позиционировании объектов и обеспечивает точный контроль их взаимного размещения. Притягивание с выравниванием позволяет вручную точно выравнивать объекты относительно их граней, центров, а также границ рабочей области.

Поддержка форматов PDF и EPS

В Flash MX 2004 включена поддержка векторной графики формата Adobe Illustrator вплоть до версии Adobe Illustrator 10. Таким образом, теперь нет необходимости в экспорте графики Adobe Illustrator в промежуточный фор-

мат SWF. Кроме того, теперь Flash поддерживает непосредственный импорт файлов в формате EPS и PDF.

Усовершенствование функций импорта видеофайлов

При импорте видеофайла можно выполнить его редактирование, "разрезав" исходный файл на несколько клипов, каждый из которых помещается в библиотеку документа как отдельный видеофайл. Кроме того, непосредственно при импорте можно осуществить ряд настроек, позволяющих изменить размер видеофильма, параметры кадрирования, выполнить простую тоновую коррекцию, а также разделить видео и звуковую дорожки, поместив их в библиотеку Flash-документа в виде самостоятельных объектов.

Расширение возможностей работы с текстом

В Flash MX 2004 значительно расширены возможности работы с текстом.

- Поддержка Unicode позволяет создавать проекты, поддерживающие любое количество языков.
- Возможность отключения сглаживания символов шрифта позволяет обеспечить лучшую читабельность при использовании текста маленького размера.
- Встроенные функции проверки написания могут быть полезны при создании текстового наполнения проекта.
- Новая панель **Strings** (Строки) предоставляет простой и удобный способ создания многоязыковых проектов. В рабочей среде создаются необходимые переводы, причем для каждого языка автоматически генерируется XML-документ, содержащий соответствующую текстовую информацию. Когда проект загружается на компьютере конечного пользователя, выполняется проверка языковых настроек системы, и Flash отображает текст на его родном языке, используя информацию "языковых" XML-файлов.

Функции *Find/Replace*

Программный пакет Flash MX 2004 содержит новые возможности поиска и замены элементов рабочей среды, облегчающие процесс редактирования документа. При помощи функций **Find** (Найти) и **Replace** (Заменить) можно

быстро найти и заменить текстовую строку, шрифт, цвет, экземпляр символа, звук, видео или растровое изображение, используемые в документе.

Поиск и замена объектов могут осуществляться как в пределах текущей сцены, так и во всем документе. Функция поиска также позволяет отредактировать найденный объект, организуя быстрый доступ к нему.

Встроенные эффекты монтажной линейки

Использование встроенных эффектов позволяет быстро реализовать простые типовые задачи, не прибегая к непосредственной работе с кадрами монтажной линейки. К числу поставляемых вместе с Flash эффектов относятся эффекты размытия (blur), перехода (transition), взрыва (explode) и т. п. Эффекты могут быть применены к любым графическим объектам рабочей среды, как рабочего, так и наложенного уровня. Реализация эффекта осуществляется при помощи появившегося в новой версии Flash языка JSFL (Flash JavaScript) путем автоматического создания новых слоев, символов и диапазонов анимации движения, в которых используются их экземпляры.

Панель **History**

Новый долгожданный элемент, который наконец-то включен в состав среды разработки Flash — это панель **History** (Истории), содержащая информацию обо всех выполненных пользователем действиях и отображающая их визуальное представление. Панель **History** также предоставляет возможность многократного выполнения однотипных операций, что позволяет существенно оптимизировать рабочий процесс.

Запись макроса

С выходом новой версии Flash разработчики получили в свое распоряжение новое мощное средство автоматизации рабочего процесса — язык Flash JavaScript. Панель **History** предоставляет возможность сохранения набора выполненных в рабочей среде действий (Commands) в виде сценария JSFL (Flash JavaScript), сохраняемого во внешнем файле, доступ к которому может быть получен из любого документа Flash. Файлы JSFL также могут быть созданы в любом текстовом редакторе. Используя средства языка Flash JavaScript, можно создавать сценарии, выполняющие роль макросов, и разрабатывать собственные инструменты.

Профили публикации

Новая версия Flash предоставляет возможность создания профиля публикации, сохраняемого во внешнем документе XML, который может быть сгенерирован автоматически. Профиль содержит все параметры публикации в используемые форматы, и, таким образом, можно осуществлять публикацию, используя различные параметры без необходимости их многократного изменения. Достаточно один раз заготовить набор профилей, которые в дальнейшем можно применять для публикации различных документов.

Автоматическая проверка наличия у пользователя проигрывателя Flash Player

Еще одной удобной возможностью новой версии Flash является наличие механизма проверки версии проигрывателя Flash Player у конечного пользователя. Flash автоматически генерирует SWF-файл, содержащий сценарий ActionScript, который выполняет эту проверку, и размещает его на начальной странице проекта. В случае если у пользователя установлена требуемая версия Flash Player, он будет автоматически переадресован к странице с основным содержимым, в противном случае будет выполнен переход к альтернативной странице, содержащей, например, HTML-версию проекта.

ActionScript 2.0

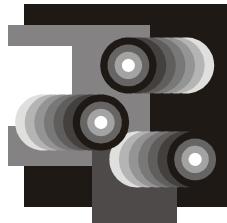
С выходом Flash MX 2004 была представлена новая версия языка сценариев ActionScript 2.0. Язык, используемый в предыдущих версиях Flash, отныне именуется ActionScript 1.0. К счастью для разработчиков, имеющих опыт работы с Flash, с переходом на новую версию ActionScript не претерпел революционных изменений. Однако он приобрел несколько важных отличий.

- Строгий контроль типов (Strict data typing) предполагает указание типа данных при объявлении переменной или задании параметров функции. Попытка присвоения переменной или передачи функции в качестве аргументов данных, тип которых отличен от типа, указанного при объявлении, вызовет ошибку, сообщение о которой будет выведено в среде тестирования. Строгий контроль типов данных обеспечивает соответствие стандарту ECMA-262 и позволяет избежать множества ошибок. Кроме того, применение строгого контроля типов данных при работе в редакторе ActionScript расширяет возможности использования всплывающих подсказок по коду.

- Наряду с контролем типа данных переменных и параметров функций в ActionScript 2.0 имеет место контроль типа возвращаемого функцией результата. При создании пользовательской функции можно явно указать тип возвращаемого ей результата или указать, что функция не возвращает значение.
- В ActionScript 2.0 появился формальный *синтаксис классов*. Классы сохраняются во внешних файлах, имеющих расширение as.

Панель *Behaviors*

В новой версии Flash пользователи уже не смогут найти режим **Normal**, используемый ранее при создании сценариев. Вместо него появилась новая панель **Behaviors**, которая позволяет назначить требуемому объекту сценарий, обеспечивающий выполнение стандартных процедур. *Поведения* представляют собой готовые фрагменты программного кода, которые могут быть использованы для создания базовой интерактивности, и не требуют дополнительного программирования.



Глава 3

Интерфейс Flash MX 2004

То, что поначалу удивляет, становится потом обычным.

Арабская пословица

Открытие и сохранение документа

В рабочей среде можно открыть файлы нескольких форматов. Во-первых, это документ Flash, имеющий расширение fla. Во-вторых, в рабочей среде можно открыть для просмотра Flash-ролик в формате SWF. Кроме того, существует возможность открытия более старых форматов, используемых ранними версиями Flash, — **FutureSplash Document** (SPA) и **SmartSketch Drawing** (SSK).

Для сохранения Flash-документа используется только формат FLA, правда при этом имеется возможность сохранения документов в виде, доступном для открытия в более старой версии — Flash MX.

Использование стартовой страницы Start Page

По окончании загрузки Flash MX 2004 на экран будет выведена стартовая страница (Start Page), предлагающая различные варианты открытия или создания нового документа Flash (рис. 3.1).

Стартовая страница позволяет выполнить одно из следующих действий.

- **Open a Recent Item** (Открыть недавно используемые файлы) — этот раздел содержит перечисление всех недавно используемых документов Flash. Щелчок на названии позволит быстро открыть документ в рабочей среде. Щелчок на команде **Open** позволит открыть любой документ Flash или SWF-ролик, выбрав его в файловой системе. Эквивалентом этих действий является использование главного меню **File**, содержащего команды **Open** (Открыть) и **Open Recent** (Открыть недавно используемые).

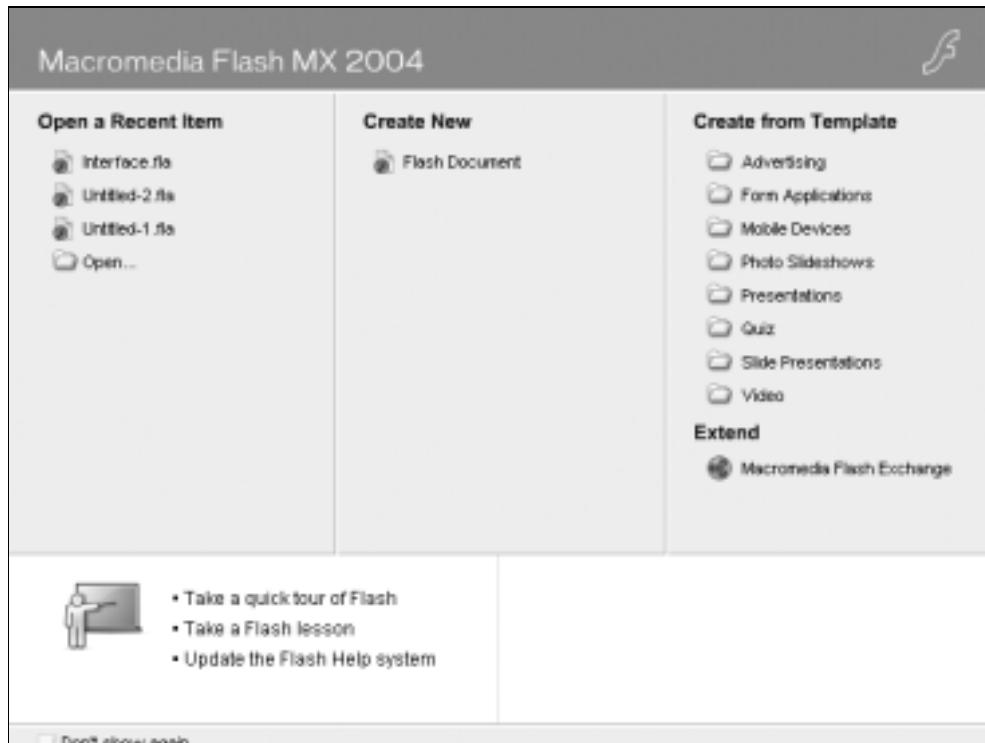


Рис. 3.1. Стартовая страница (Start Page)

- **Create New** (Создать новый) — щелчок на надписи **Flash Document** (Документ Flash) позволит создать новый документ Flash. При этом все параметры нового документа будут такими же, как у последнего созданного документа. Подробная информация о настройке параметров нового документа содержится в разд. "Настройка глобальных параметров фильма" настоящей главы. Эквивалентом этого действия является использование команды главного меню **File>New**.
- **Create from Template** (Создать на основе шаблона) — этот раздел позволяет использовать стандартные шаблоны в качестве отправной точки при создании документа. Здесь представлены группы тематически объединенных шаблонов, используемых для различных целей. Щелчок на названии группы приводит к открытию окна **New from Template**, содержащего перечисление всех групп шаблонов (раздел **Category**), названия доступных шаблонов для каждой категории (раздел **Templates**), предварительный просмотр шаблона (раздел **Preview**), краткое описание шаблона (раздел **Description**).

Вместе с Flash MX 2004 поставляются следующие шаблоны:

- **Advertising** (Реклама) — набор шаблонов, задающих различные стандартные размеры рабочей области для баннеров и всплывающих окон;
 - **Mobile Devices** (Мобильные устройства) — набор шаблонов, позволяющих создавать ролики для мобильных устройств (карманных ПК, сотовых телефонов). Шаблон содержит изображение мобильного устройства и задает размер рабочей области, исходя из размера дисплея и режима просмотра;
 - **Photo Slideshows** (Фотоальбом) — шаблон, позволяющий быстро создать слайд-шоу, представляющее собой последовательную смену изображений. Слайд-шоу снабжено элементом управления, позволяющим "листать" изображения, а также установить автоматический режим смены изображений;
 - **Presentations** (Презентации) — на основе этого шаблона можно быстро создать презентацию, состоящую из различных слайдов, переход между которыми выполняется при помощи монтажной линейки и кнопок навигации. Предлагается четыре стандартных варианта дизайна:
 - ◊ **Classic Presentation** (Классическая презентация);
 - ◊ **Retro Presentation** (Ретро-презентация);
 - ◊ **Sharp Presentation** (Строгая презентация);
 - ◊ **Tech Presentation** (Хай-тек-презентация);
 - **Quiz** (Викторины) — данный шаблон позволяет создать интерактивный тест, включающий различные процедуры опроса, автоматическую проверку результата и подсчет правильных ответов. Предлагается три стандартных варианта дизайна;
- **Extend** (Расширения) — данный раздел позволяет посетить сайт компании Macromedia и ознакомиться с разделом, посвященным расширениям Flash. Расширения представляют собой внешние модули (**Third Party Extensions**), созданные сторонними разработчиками, которые применяются для решения целого ряда задач, включая создание графиков и диаграмм, разработку анимационных текстовых эффектов, реализацию широких возможностей работы с растровой графикой и др.

Кроме указанных разделов, начальная страница также позволяет получить быстрый доступ к справочным материалам. Пункт **Take a quick tour of Flash** (Обзор возможностей Flash) позволяет перейти к разделу сайта компании Macromedia, содержащему справочные материалы. Пункт **Take a Flash lesson** (Урок Flash) обеспечивает локальный доступ к материалам, содержащимся в Справочной системе Flash MX 2004. При помощи пункта **Update the Flash Help system** (Обновить справочную систему) можно пополнить содержание справочной системы, загрузив с сайта компании Macromedia обновленную информацию.

Сохранение документа

В Flash MX 2004 существует несколько способов сохранения документа. Команды сохранения расположены в главном меню **File**.

- File>Save** (Сохранить) — сохранение файла в текущей директории с текущим именем. Выполнение этой команды приводит к перезаписи старой версии документа.
- File>Save and Compact** (Компактное сохранение) — при сохранении удаляется информация о всех произведенных отменах (**Undo**), очищается панель истории (**History**). В результате выполнения этой команды конечный объем файла определяется только теми элементами, которые входят в его состав. В противном случае на увеличение объема файла может влиять информация об элементах проекта, которые были удалены (например, растранных изображениях или звуковых файлах).
- File>Save As** (Сохранить как) — при помощи этой команды можно сохранить документ под новым именем или в другой директории. Для сохранения необходимо указать месторасположение и имя сохраняемого документа.
- File>Save as Template** (Сохранить как шаблон) — команда, позволяющая создать шаблон из текущего документа, на основе которого в дальнейшем можно будет создавать новые проекты. Файл шаблона также имеет расширение fla и является, по сути, обычным документом Flash. При сохранении необходимо указать имя (**Name**), категорию (**Category**) и краткое описание (**Description**) шаблона. Эта информация в дальнейшем будет представлена в соответствующем разделе стартовой страницы (Start Page). Файл шаблона при сохранении размещается в каталоге Windows. Путь к каталогу, содержащему пользовательский шаблон, может выглядеть следующим образом: C:/WINDOWS/Application Data/Macromedia/Flash MX 2004/en/Configuration/Templates/myNewTemplate.
- File>Save All** (Сохранить все) — при выполнении этой команды автоматически сохраняются все открытые в данный момент документы.
- File>Revert** (Вернуть) — команда, позволяющая вернуться к последней сохраненной версии документа. Все изменения, произведенные с момента последнего сохранения, будут утрачены.
- File>Send** (Отправить) — команда, дающая возможность отправить текущий документ в качестве вложения (**Attachment**) по электронной почте.
- File>Exit** (Выход) — закрытие Flash. Если в момент закрытия имеются не сохраненные документы, то будет выведено предупреждение об этом и предложена возможность сохранить изменения.

Сохранить документ можно также, щелкнув правой кнопкой мыши на его закладке, расположенной под главным меню рабочей среды (см. далее) и выбрав из появившегося контекстного меню любую из приведенных выше команд сохранения.

Примечание

Если документ не сохранен или содержит изменения, внесенные после сохранения, то в заголовке окна документа после его названия добавляется символ "**", например, myDoc.fla*.

Обзор интерфейса

Этот раздел содержит обзорное описание основных элементов интерфейса Flash MX 2004. Подробная информация представлена в соответствующих главах и разделах настоящей книги.

Внешний вид окна документа Flash MX 2004, содержащего основные элементы интерфейса, представлен на рис. 3.2. В качестве основных элементов интерфейса можно выделить следующие составляющие.

- Главное меню
- Стандартная панель
- Рабочая область
- Вспомогательная область
- Полоса редактирования
- Служебные панели
- Панель инструментов
- Инспектор свойств
- Монтажная линейка
- Контроллер
- Библиотека
- Панель **History**

Главное меню

Главное меню включает десять разделов, содержащих все основные команды Flash. Некоторые разделы содержат подменю, позволяющие выбрать необходимое действие.

Под главным меню расположены закладки (tabs), позволяющие быстро переключаться между открытыми документами Flash. Закладки видны только в том случае, если окно документа развернуто. При одновременной работе с несколькими документами можно быстро перейти к любому открытому документу, щелкнув на его закладке. Щелчок на закладке правой кнопкой мыши приводит к появлению контекстного меню, включающего набор команд, которые позволяют открыть новый документ или сохранить текущий.

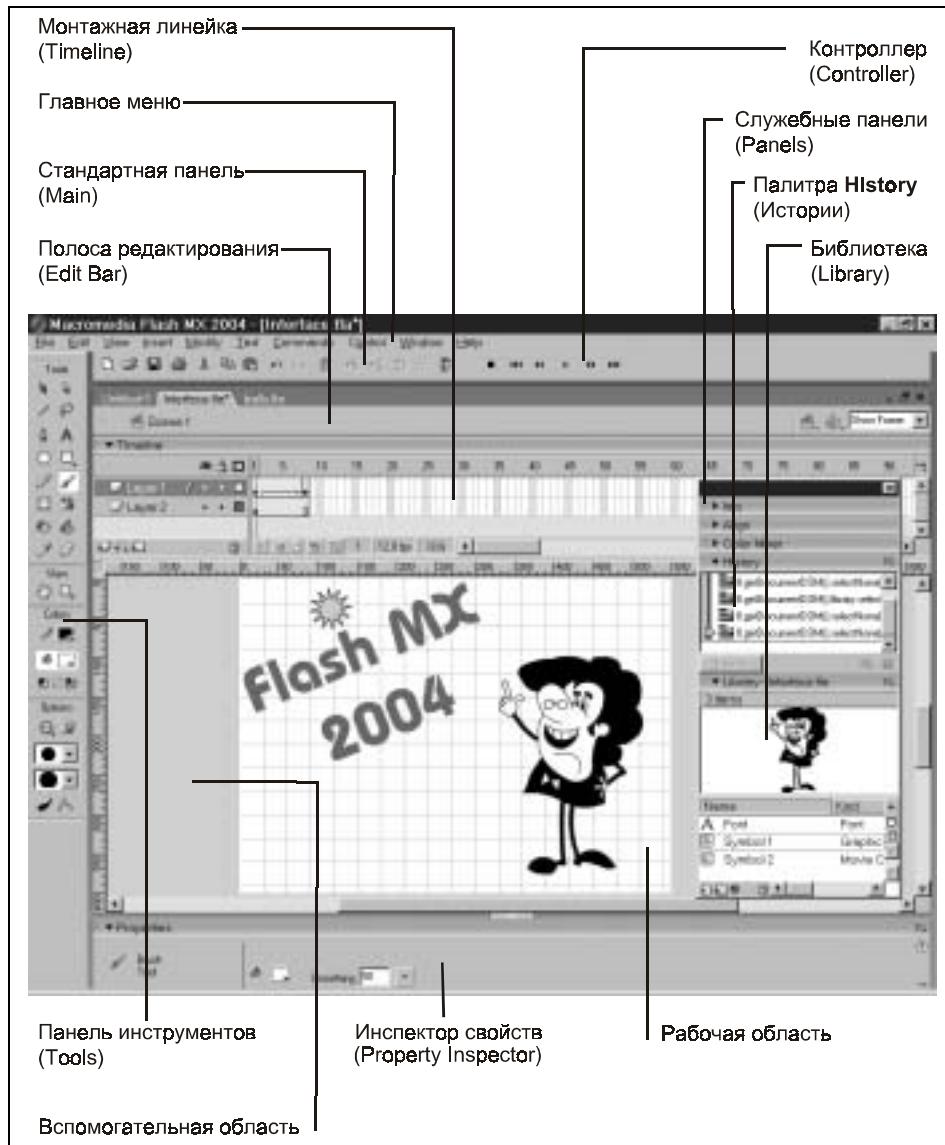


Рис. 3.2 Интерфейс Flash MX 2004

Стандартная панель

Чтобы включить отображение стандартной панели, нужно выполнить команду меню **Window>Toolbars>Main**. Стандартная панель содержит ряд кнопок, позволяющих выполнить типовые процедуры (создание, открытие, сохранение,

печатать документа, команды редактирования, отмены произведенных действий и некоторые операции трансформации графических объектов).

Рабочая область

Рабочая область (Work Area) по умолчанию представляет собой прямоугольник белого цвета. Все объекты, размещенные в пределах рабочей области, будут видны в конечном фильме. Размер и цвет рабочей области могут быть заданы произвольно (см. разд. "Настройка глобальных параметров фильма" данной главы).

Вспомогательная область

Серый фон вокруг рабочей области называется *вспомогательной областью*. Так же как и на рабочей области, на вспомогательной области можно размещать графические объекты и применять к ним все имеющиеся в распоряжении приемы работы. Однако объекты, находящиеся в пределах вспомогательной области, как правило, не видны в конечном фильме (это зависит от параметров публикации). Вспомогательная область используется для размещения элементов, которые не должны быть видны зрителю (например, объекты, содержащие сценарий), или для имитации эффекта манипуляции камерой (наезд, панорамирование и т. п.), когда в пределах рабочей области находится только часть целого изображения.

В совокупности рабочая область и вспомогательная область называются *сценой* (Stage).

Примечание

В Flash термин "сцена" имеет несколько значений. Причиной этого является то, что два понятия, используемые в официальной терминологии, — Stage и Scene — переводятся на русский язык совершенно одинаково — "сцена". Конечно, можно было бы найти соответствующий эквивалент, например, "подмостки", однако он несколько уводит от основного значения слова. Термин Stage используется, когда имеется в виду физическое размещение объектов в пределах рабочей и вспомогательной областей, а термин Scene применяется, когда речь идет о структурной организации всего проекта. В дальнейшем смысл термина "сцена" будет ясен из контекста.

Полоса редактирования

Полоса редактирования (Edit Bar) служит для ориентации в структуре рабочего проекта. На ней отображается информация о том, в какой среде происходит работа в данный момент (сцена, режим редактирования группы, среда редактирования символа). Кроме того, полоса редактирования содержит элементы управления, позволяющие перемещаться между структурными единицами проекта (сценами), получать доступ к редактированию символов и задавать произвольный масштаб просмотра.

Служебные панели

В рабочий среде Flash MX 2004 имеется целый набор рабочих панелей (Panels), служащих для разнообразных целей дизайна и разработки проектов. В данной книге наряду с термином *панель* будет использоваться эквивалентный термин *палитра*.

Вызов любой служебной панели осуществляется при помощи главного меню **Window**. Здесь все панели организованы в три группы в соответствии с их назначением.

- **Design Panels** — панели, при помощи которых выполняются задачи по созданию, трансформации и организации графики.
- **Development Panels** — панели, используемые при разработке интерактивности.
- **Other Panels** — остальные панели, позволяющие оптимизировать рабочий процесс создания проекта.

Для того чтобы открыть ту или иную панель, необходимо установить флајажок рядом с ее названием в главном меню **Window**. Снятие флајажка, соответственно, закрывает панель. Чтобы сразу закрыть все панели, можно использовать команду **Window>Hide Panels** или ее клавиатурный эквивалент <F4>. Повторное выполнение этой команды возвращает все панели на прежние места.

В связи с тем, что число панелей достаточно велико, и они занимают большую площадь экрана, держать открытыми их все одновременно представляется крайне нецелесообразным. Существует несколько возможностей, позволяющих оптимизировать схему размещения панелей.

Панели могут быть плавающими, парковаться и объединяться в группы.

Для того чтобы открепить панель, необходимо взяться за пиктограмму с изображением двух вертикальных полос в ее левом верхнем углу и перетащить панель в сторону. Закрыть плавающую панель можно, щелкнув на пиктограмме "крестик" в ее правом верхнем углу.

Чтобы припарковать панель к границе сцены, нужно взяться за ту же пиктограмму и подтащить панель к одной из границ. Появившаяся черная рамка укажет на то, что панель закреплена. Для включения запрета на прикрепление панелей можно установить флајажок **Disable panel docking** меню **Edit>Preferences**.

Для группировки панели необходимо просто припарковать ее к другой панели.

Стандартный вид панели представлен на рис. 3.3. Любая служебная панель может быть развернута и свернута. У развернутой панели черный треугольник обращен вправо. Для того чтобы свернуть панель, необходимо щелкнуть на полосе с заголовком в ее верхней части. В свернутом состоянии черный треугольник в левом верхнем углу панели обращен вниз. Повторный щелчок позволяет вновь развернуть панель. Два щелчка на верхней полосе окна

(полосе захвата) открепленной панели позволяет минимизировать панель. Для возврата в первоначальное состояние необходимо повторно дважды щелкнуть на полосе захвата.

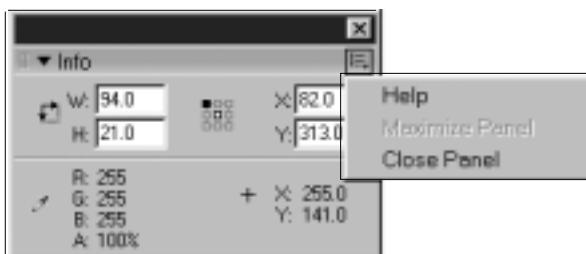


Рис. 3.3. Стандартный вид рабочей панели с ее контекстным меню

Любая служебная панель содержит собственное контекстное меню, содержащее как набор команд, специфических только для нее, так и набор стандартных команд для всех остальных панелей. Чтобы вызвать контекстное меню панели, нужно щелкнуть на пиктограмме в ее правом верхнем углу. Панель при этом должна быть развернута. Контекстное меню каждой панели содержит три стандартные команды **Help** (Вызов справки), **Maximize Panel** (Развернуть панель), **Close Panel** (Закрыть панель).

Для того чтобы развернуть окно сцены по ширине на весь экран, нужно либо закрыть все припаркованные справа панели, либо щелкнуть на пиктограмме стрелки, расположенной на середине правой грани сцены. Повторный щелчок вернет сцену в исходное положение. Развернуть окно сцены также можно при помощи команды главного меню **Window>Hide Panels** или ее клавиатурного эквивалента <F4>.

Для того чтобы каждый раз не тратить время на организацию панелей, можно сохранить схему их размещения при помощи команды **Window>Save Panel Layout**. В поле **Name** одноименного диалогового окна нужно ввести имя схемы. Таким образом можно создать несколько различных схем на все случаи жизни.

Команда **Window>Panel Sets** позволяет выбрать и применить одну из стандартных или пользовательских схем размещения.

◀▶ Примечание

Многие служебные панели и диалоговые окна Flash содержат кнопки, поля и другие элементы управления, названия которых не указаны непосредственно. В дальнейшем при описании подобных элементов в качестве их названий в этой книге будут использованы всплывающие подсказки, появляющиеся при подведении курсора мыши к соответствующему элементу.

Панель инструментов

Панель инструментов (Tools) — содержит инструментарий, предназначенный для создания графики и работы с ней.

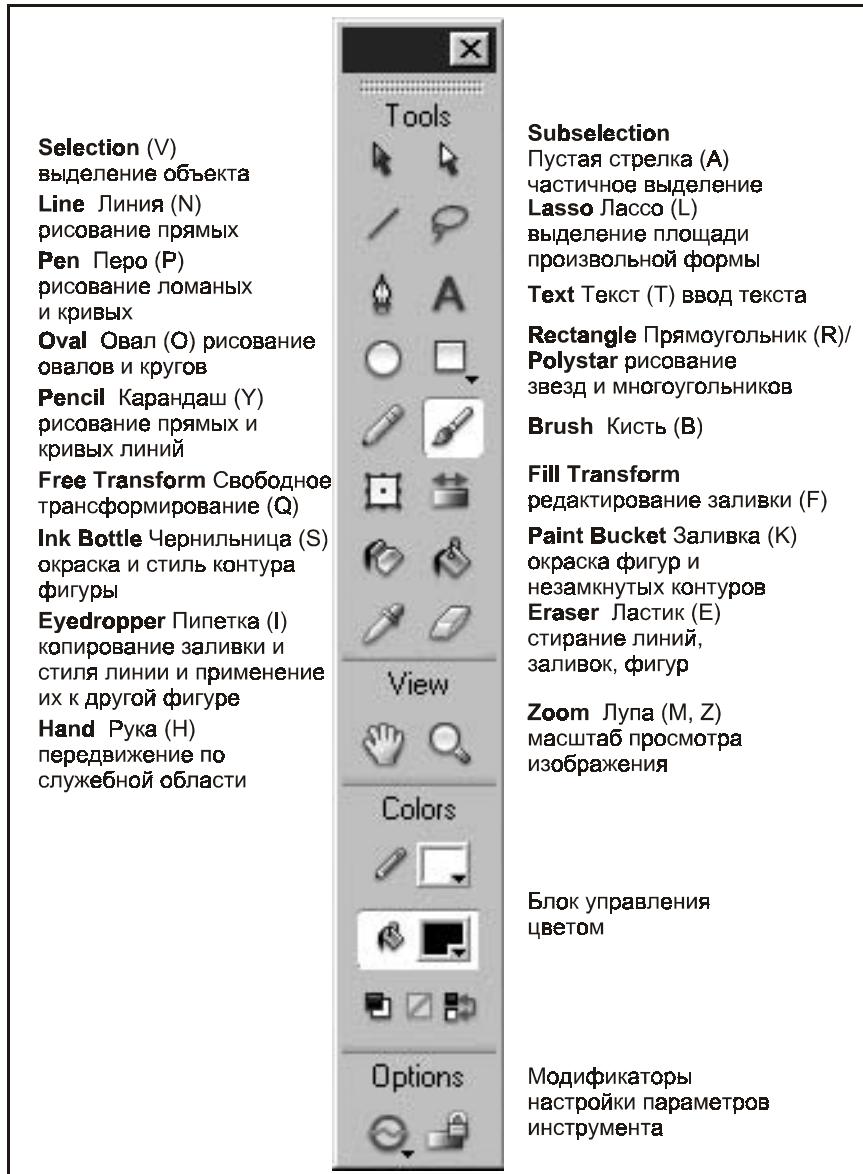


Рис. 3.4. Панель инструментов (Tools)

Панель инструментов состоит из четырех разделов (рис. 3.4).

- Tools** (Инструменты) — в этом разделе содержатся инструменты для создания графики и работы с ней.
- View** (Вид) — раздел, содержащий вспомогательные инструменты, используемые для изменения масштаба просмотра и перемещения по экрану.
- Colors** (Цвета) — блок управления цветом, при помощи которого могут быть заданы цвета элементов векторного объекта.
- Options** (Модификаторы) — в этом разделе содержатся модификаторы (**Modifiers**), позволяющие задать дополнительные параметры или режимы работы соответствующих инструментов.

Некоторые инструменты могут быть сгруппированы. При этом одна кнопка на панели инструментов позволяет получить доступ к нескольким вложенным инструментам. Если кнопка содержит вложенные инструменты, в ее нижней части имеется черный треугольник. Для того чтобы получить доступ к любому инструменту группы, необходимо щелкнуть на пиктограмме инструмента и удерживать кнопку мыши в нажатом состоянии до появления списка инструментов, содержащихся в группе. По умолчанию единственные сгруппированные инструменты — это **Rectangle** (Прямоугольник) и **Polystar** (Многоугольник).

Для того чтобы включить (или убрать с экрана) панель инструментов, нужно выполнить команду главного меню **Window>Tools** или воспользоваться ее клавиатурным эквивалентом <Ctrl>+<F2>.

Панель инструментов может быть плавающей или парковаться к левой или правой границе экрана. Чтобы открепить панель инструментов, нужно выполнить одно из следующих действий:

- перетащить панель, взявшись курсором мыши за ее верхнюю часть с изображением двух горизонтальных полос;
- два раза щелкнуть на верхней части панели инструментов;
- щелкнуть на верхней части панели инструментов, удерживая при этом клавишу <Ctrl>.

Плавающую панель инструментов можно перемещать в любую точку экрана. Для того чтобы закрепить панель, нужно выполнить одно из следующих действий:

- два раза щелкнуть на верхней части панели — это приведет к тому, что панель вернется в то положение, где она находилась до открепления;
- подвести панель к левой или правой границе экрана.

Чтобы убрать с экрана плавающую панель инструментов, достаточно щелкнуть на пиктограмме крестика в ее правом верхнем углу.

Команда главного меню **Edit>Customize Tools Panel** позволяет организовать порядок размещения инструментов в разделе **Tools** и **View** панели инструмен-

тов, сгруппировать их или удалить. В результате выполнения этой команды появляется одноименное окно, представленное на рис. 3.5.

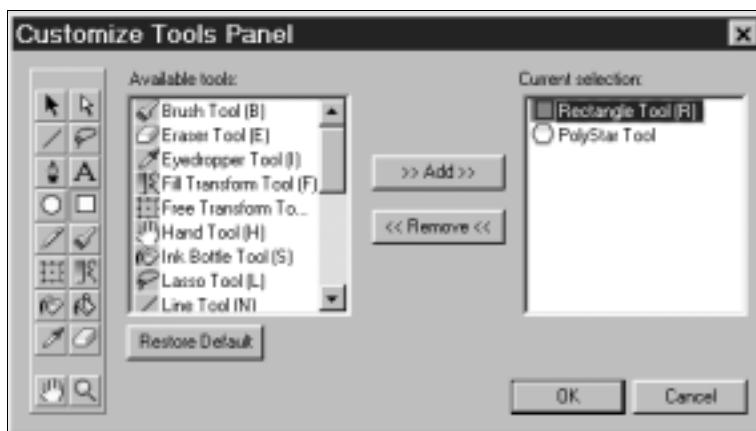


Рис. 3.5. Окно настройки панели инструментов **Customize Tools Panel**

Для того чтобы *объединить* несколько инструментов в группу, необходимо выделить пиктограмму инструмента, который будет содержать эту группу, в левой части окна (выделение отображается красной рамкой). После этого в поле **Available tools** (Доступные инструменты) нужно выбрать инструмент, который необходимо подгруппировать, и затем нажать кнопку **Add** (Добавить). Если необходимо добавить в группу еще несколько инструментов, нужно повторить описанную последовательность действий.

Для того чтобы *удалить* инструмент из панели инструментов, необходимо выделить его пиктограмму, после чего выделить его (или сгруппированный с ним инструмент) в поле **Current selection** (Текущее выделение) и нажать кнопку **Remove** (Удалить).

Кнопка **OK** позволяет применить сделанные настройки, кнопка **Restore Default** возвращает панели инструментов первоначальный вид.

Инспектор свойств

Инспектор свойств (Property inspector, панель **Properties**) упрощает процесс разработки проекта, обеспечивая быстрый доступ к различным параметрам настройки и атрибутам выделенного объекта, как на сцене, так и на монтажной линейке. При помощи Инспектора свойств можно задавать значения тех или иных параметров объекта или документа, не прибегая к использованию дополнительных меню или панелей, содержащих данные настройки (рис. 3.6).

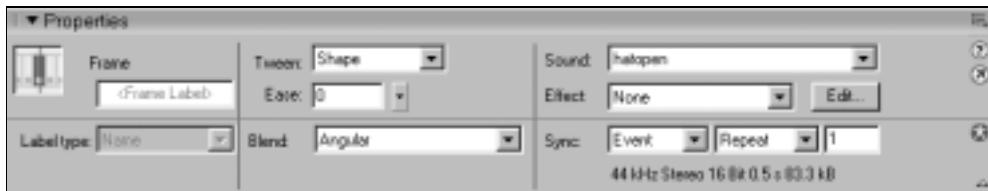


Рис. 3.6. Инспектор свойств (Property Inspector) при установке свойств кадра

Для управления открытием или закрытием Инспектора свойств используется команда главного меню **Window>Properties** или сочетание клавиш **<Ctrl>+<F3>**.

В левом верхнем углу Инспектора свойств указывается тип выделенного объекта. В зависимости от того, какой именно объект выделен в данный момент, Инспектор свойств содержит информацию и настройки текущего документа, контура, векторной формы, группы, растрового изображения, видеоролика, текста, кадра, экземпляра символа или звукового фрагмента. Если в области выделения одновременно находятся несколько разнородных объектов, то инспектор свойств указывает на смешанное выделение (**Mixed**).

Как и вспомогательные панели, Инспектор свойств может парковаться к краю окна или становиться плавающим; его также можно минимизировать или развернуть. Для выполнения этих операций используются те же приемы, что и для вспомогательных панелей.

Инспектор свойств может иметь более компактный вид или разворачиваться, открывая дополнительные параметры. Для того чтобы развернуть его, необходимо щелкнуть на белом треугольнике в правом нижнем углу; соответственно, эту же пиктограмму можно использовать для приведения панели **Properties** к более компактному виду.

Монтажная линейка

Монтажная линейка (Timeline) — это инструмент, при помощи которого создается анимация. Монтажная линейка включает в свой состав палитру слоев (Layers) и кадры (Frames), размещенные в пределах соответствующих слоев. Любой объект, созданный в рабочей среде, размещается в определенном кадре слоя монтажной линейки. При создании нового документа монтажная линейка практически пуста, т. е. она содержит один слой и один кадр.

Для включения (отключения) монтажной линейки используется команда главного меню **Window>Timeline** или сочетание клавиш **<Ctrl>+<T>**.

Монтажная линейка может быть плавающей или парковаться к границам сцены. Ее можно минимизировать или развернуть.

Контроллер

Контроллер (Controller) — панель, содержащая кнопки, при помощи которых осуществляется управление монтажной линейкой непосредственно в рабочей среде. Контроллер содержит стандартные команды воспроизведения (Playback).

Для включения (отключения) контроллера используется команда главного меню **Window>Toolbars>Controller**. Кроме того, для управления воспроизведением фильма можно использовать главное меню **Control**.

Контроллер может парковаться или становиться плавающим. Для того чтобы открепить его, нужно взяться курсором мыши за верхнюю грань над элементами управления или два раза щелкнуть на ней и оттащить в сторону. Вернуть панель на место можно, два раза щелкнув по ее заголовку.

Библиотека

Библиотека (Library) представляет собой хранилище символов (Symbols), а также всех импортированных объектов (растровой графики, видеороликов, звуков). Любой внешний объект, будучи импортирован во Flash, попадает в библиотеку. Библиотека позволяет организовать эти элементы, сгруппировав их при помощи папок. Используя возможности библиотеки, можно задать параметры оптимизации растровых изображений и звуков.

Открыть или закрыть окно библиотеки можно при помощи команды главного меню **Window>Library** или сочетания клавиш <Ctrl>+<L>.

Любой документ Flash содержит собственную библиотеку. Существует возможность импорта объектов из библиотеки другого документа в библиотеку текущего.

Flash MX 2004 также содержит стандартные библиотеки кнопок (Buttons), классов ActionScript (Classes) и обучающих роликов (Learning Interactions). **Window>Other Panels>Common Libraries**.

Палитра *History*

Палитра **History** (Истории) — это служебная панель, которая позволяет отслеживать все произведенные в ходе работы действия и при необходимости возвращаться на соответствующую стадию. Кроме того, при помощи панели истории можно сохранять последовательность действий в виде набора команд (**Commands**), которые могут быть применены в текущем или другом документе Flash наподобие макросов. Внешний вид палитры **History** представлен на рис. 3.7.



Рис. 3.7. Палитра **History**

Для открытия или закрытия панели служит команда **Window>Other Panels>History** или сочетание клавиш **<Ctrl>+<F10>**. Все произведенные действия указываются в порядке их выполнения сверху вниз. Действия, выполненные раньше, располагаются выше. Каждая операция снабжена пиктограммой и поясняющей надписью. Количество действий, которые отслеживаются в ходе работы, задается в окне **Undo levels** меню **Edit>Preferences** (см. разд. "Настстройки программы" данной главы).

Для того чтобы отменить действие или последовательность действий, необходимо переместить вертикальный слайдер, расположенный слева, на соответствующую строку. При этом все отменяемые действия будут отображаться на сером фоне. Для отказа от отмены действий нужно переместить слайдер в первоначальное положение. Если после отмены последовательности действий выполнить любую новую операцию, то восстановить отмены будет невозможно.

Для выполнения отмены или восстановления действия также можно использовать главное меню **Edit**. Команда **Edit>Undo** (**<Ctrl>+<Z>**) отменяет последнее действие, а команда **Edit>Redo** (**<Ctrl>+<Y>**) восстанавливает отмененное действие. Последовательное выполнение этих команд приводит к последовательной отмене (или восстановлению) действий.

Для очистки палитры **History** применяется команда ее контекстного меню **Clear History** (Очистить Историю). Выполнение этой команды приводит к удалению всей информации о произведенных действиях. После очистки палитры **History** отмена или возврат предыдущих действий становятся невозможными. О других возможностях палитры **History** будет сказано в разд. "Использование макросов" гл. 15.

Инструменты масштабирования и перемещения по экрану

Изменение масштаба просмотра

В Flash существует несколько различных способов изменения масштаба просмотра. Выбор того или иного способа диктуется контекстом ситуации и привычкой дизайнера.

- Инструмент **Zoom** (Лупа),  расположенный в разделе **View** (Вид) панели инструментов (Tools). Инструмент **Zoom** содержит два модификатора: **Enlarge** (Увеличить)  и **Reduce** (Уменьшить)  расположенные в разделе **Options** (Модификаторы). Во время работы с **Zoom** можно легко переключаться между модификаторами с помощью клавиши **<Alt>**. При помощи модификатора можно указать направление работы инструмента — увеличение или уменьшение масштаба. Существуют два способа применения инструмента **Zoom**. Щелчок инструментом на сцене приводит к изменению масштаба в два раза. Другой способ заключается в создании инструментом **Zoom** прямоугольной области. Для этого необходимо щелкнуть им на сцене и, не отпуская кнопки мыши, протянуть курсор. Масштаб просмотра будет увеличен таким образом, что выделенная область будет увеличена настолько, насколько позволит размер окна. Стоит обратить внимание на то, что даже при включенном модификаторе **Reduce** применение второго способа все равно приводит к увеличению масштаба. Двойной щелчок на пиктограмме инструмента **Zoom** устанавливает масштаб просмотра 100%.
- Поле отображения масштаба, расположенное справа вверху на полосе редактирования, позволяет выбрать одно из стандартных значений масштаба, либо ввести другое значение вручную. Масштаб может быть задан в диапазоне от 8% до 2000%.

Кроме стандартных числовых значений масштаба, поле масштаба содержит еще три следующих значения:

- **Fit in Window** (По размеру окна) — масштаб, при котором рабочая область точно умещается в текущем окне. Полосы прокрутки отсутствуют. В этом режиме нет возможности перемещения по экрану;
- **Show Frame** (Показать кадр) — масштаб, при котором рабочая область видна полностью; имеются полосы прокрутки, позволяющие увидеть содержимое вспомогательной области;
- **Show All** (Показать все) — масштаб, при котором в текущем окне отображаются *все* объекты, находящиеся в пределах рабочей и вспомогательной областей.

Таблица 3.1. Способы изменения масштаба просмотра

Стандартное меню (поле масштаба)	Инструмент Zoom (M, Z)	Команды меню View>Magnification	Двойные щелчки по инструментам
Выбрать один из масштабов в поле масштаба, находящемся справа над рабочей областью документа или ввести свое значение	Щелчок на модификаторе , затем щелкнуть в центре увеличивающейся области либо обвести нужный фрагмент в рамку. Изменение масштаба при щелчке в два раза	Щелчок на модификаторе , затем щелкнуть в центре уменьшающего фрагмента. Изменение масштаба при щелчке в два раза	Инструмент Zoom 100% отображение
Show Frame – показать весь кадр целиком	Для временного перехода к инструменту <Ctrl>+<Shift>+<Пробел>	Щелчок на модификаторе , затем щелкнуть в центре уменьшающего фрагмента. Изменение масштаба при щелчке в два раза	Инструмент Hand отображает всю рабочую область максимально возможно
Show All – показать только все содержимое кадра	Для временного перехода к инструменту <Ctrl>+<Пробел>	Show All – <Ctrl>+<3> Work Area – <Ctrl>+<Shift>+<W> показать только рабочую область	Zoom In <Ctrl>+<=> в 2 раза увеличить масштаб
Fit in Window – показать рабочую область по размеру окна	Переход между режимами увеличения и уменьшения при активной лупе, <Alt>	Zoom Out – <Ctrl>+<-> в 2 раза уменьшить масштаб	Диапазон значений масштаба от 8% до 2000%

Масштаб, эквивалентный выбору значения **Show Frame**, можно установить, два раза щелкнув на пиктограмме инструмента **Hand Tool** (Рука).

- Команды меню **View>Zoom In** (Увеличить масштаб) или **View>Zoom Out** (Уменьшить масштаб), соответственно увеличивают или уменьшают масштаб просмотра в два раза.
- Меню **View>Magnification** (Увеличение) содержит список, аналогичный списку поля масштаба, позволяющий выбрать одно из стандартных значений.

Информация по данной теме обобщена в табл. 3.1.

Перемещение по экрану

Перемещаться по сцене можно при помощи инструмента **Hand** (Рука) , расположенного в разделе **View** панели инструментов. Активизировать инструмент **Hand** при работе с любым другим инструментом можно при помощи клавиши <Backspace> (Пробел).

Для перемещения по сцене также служат вертикальная и горизонтальная полосы прокрутки, расположенные снизу и справа от рабочей области.

Режимы отображения сцены

Для того чтобы ускорить работу процессора в ходе создания графики, можно использовать различные режимы отображения содержимого сцены. Выбор того или иного режима просмотра никак не влияет на то, как будут выглядеть объекты в конечном фильме. Для выбора соответствующего режима применяется команда главного меню **View>Preview Mode**. Здесь содержатся следующие режимы:

- **Outlines** (Контуры) — в этом режиме все векторные формы и обводки отображаются в виде тонких линий. Использование данного режима позволяет увеличить быстродействие. Использование данного режима бывает также весьма удобно (часто при увеличенном масштабе просмотра) при поиске мест разрывов контуров и при "очистке" границ форм от лишних сегментов;
- **Fast** (Черновой) — в этом режиме сглаживание отключено, однако объекты отображаются в цвете со всеми атрибутами. Режим позволяет ускорить обработку отображения графики машиной;
- **Antialias** (Сглаживание) — сглаживание применяется к векторным формам, контурам и растровым изображениям. Границы форм выглядят четко, без зубчатых краев. Графика в данном режиме отрисовывается несколько дольше, чем в режиме **Fast**;

- **Antialias Text** (Сглаживание текста) — применяется к тексту, в результате чего границы букв выглядят четко и аккуратно. Этот режим дает лучший результат при работе с достаточно крупным размером шрифта (кеглем). При использовании большого количества текста применение данного режима может замедлить работу компьютера;
- **Full** (Полное сглаживание) — сглаживание применяется ко всему содержимому сцены. Использование этого режима может существенно снизить скорость обработки графики, особенно при работе с визуально насыщенной сценой.

Информация по данной теме обобщена в табл. 3.2.

Таблица 3.2. Виды отображения содержимого рабочей области

Режим	Команда меню	Клавиатурный эквивалент	Область действия режима
Просмотр в контурах без заливок	View>Preview Mode>Outlines	<Ctrl>+<Alt>+<Shift>+<O>	Для всех кадров, слоев, сцен в фильме
Быстрый просмотр, отключение сглаживания	View>Preview Mode>Fast	<Ctrl>+<Alt>+<Shift>+<F>	
Сглаживание всей графики без текста	View>Preview Mode>Antialias	<Ctrl>+<Alt>+<Shift>+<A>	
Сглаживание всей графики и текста	View>Preview Mode>Antialias Text	<Ctrl>+<Alt>+<Shift>+<T>	
Полное сглаживание всех объектов	View>Preview Mode>Full		
Отключение отображения выделения	View>Hide Edges	<Ctrl>+<H>	Для выделенных объектов
Отображение содержимого слоя в контурах	Modify>Timeline>Layer Properties		Для всех кадров слоя
Видимый / Скрытый слой			

Настройки программы

Все настройки рабочей среды Flash осуществляются при помощи команды главного меню **Edit>Preferences** (Настройки).

Окно **Preferences** содержит пять вкладок, в которых задаются различные настройки элементов рабочей среды Flash.

□ Вкладка **General** (Общие) содержит следующие параметры (рис. 3.8).

- **Undo levels** (Количество отмен) — задает количество отмен действий. Допускаются значения от 2 до 9999. С увеличением значения этого параметра скорость работы может снижаться, поскольку хранение информации о предыдущих действиях требует существенного объема оперативной памяти.

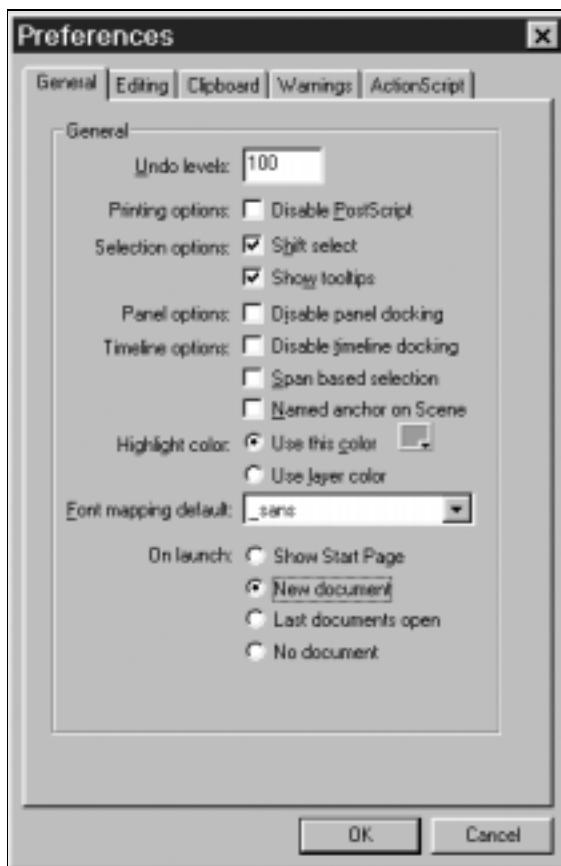


Рис. 3.8. Вкладка **General**
окна настройки рабочей среды **Preferences**

- **Printing options** (Параметры печати) — данный раздел дает возможность отключить режим печати PostScript (**Disable PostScript**). При пе-

чати на Postscript-принтере это может привести к замедлению скорости печати. Используйте эту опцию, если при печати на PostScript-принтере возникают проблемы.

- **Selection options** (Параметры выделения).
 - ◊ **Shift select** — если данный режим активен, то при выделении объектов использование клавиши <Shift> добавляет новые объекты к уже выделенным. При снятии этого флагка использование клавиши <Shift> приводит к тому, что щелчок на новом объекте выделяет его при этом снимая выделение с остальных объектов.
 - ◊ **Show tooltips** — включает всплывающие подсказки, появляющиеся при наведении курсора на различные элементы интерфейса.
- **Panel Options** (Параметры панелей) — запрет закрепления панелей (**Disable panel docking**). При включении данного режима все панели становятся плавающими.
- **Timeline options** (Параметры монтажной линейки).
 - ◊ **Disable timeline docking** — запрет закрепления монтажной линейки. Монтажная линейка располагается в своем собственном окне.
 - ◊ **Span based selection** — выделение диапазонов кадров. Активизация этого режима изменяет способ выделения кадров монтажной линейки (см. гл. 9).
 - ◊ **Named anchor on Scene** — данный режим автоматически помещает якорь (**Named anchor**) в первый кадр каждой сцены (см. гл. 9).
- **Highlight color** (Цвет выделения).
 - ◊ **Use this color** — выбор цвета, которым будут отображаться рамки выделенных объектов наложенного уровня вне зависимости от цвета слоя, на котором они находятся.
 - ◊ **Use layer color** — для выделения объектов наложенного уровня будет использован цвет слоя (см. гл. 9).
- **Font mapping default** (Соответствие шрифтов по умолчанию) — раскрывающийся список, позволяющий выбрать шрифт, используемый по умолчанию для замены при открытии документа, содержащего отсутствующие в системе шрифты.
- **On launch** (При запуске) — выбор варианта открытия документа при запуске Flash MX 2004.
 - ◊ **Show Start Page** — показать стартовую страницу.
 - ◊ **New document** — создать новый документ.

- ◊ **Last document open** — открыть последний сохраненный документ.
- ◊ **No document** — не создавать и не открывать документ.
- Вкладка **Editing** (Редактирование) содержит следующие параметры (рис. 3.9).



Рис. 3.9. Вкладка **Editing**
окна настройки рабочей среды **Preferences**

- **Pen tool** (Перо) — настройки инструмента **Pen**.
 - ◊ **Show pen preview** — показывать новый сегмент кривой еще до размещения конечной опорной точки. Эта опция позволяет сделать процесс работы с пером более контролируемым и наглядным.
 - ◊ **Show solid points** — отображать выделенные опорные точки контура пустыми кружками. Остальные точки контура отображаются заполненными кружками.
 - ◊ **Show precise Cursors** — отображать курсор в виде перекрестия, это позволяет более точно размещать опорные точки.

- **Vertical text** (Текст по вертикали).
 - ◊ **Default text orientation** — вертикальная ориентация текста по умолчанию. Вертикальный текст используется в некоторых азиатских языках.
 - ◊ **Right to left text flow** — направление текста по умолчанию — справа налево.
 - ◊ **No kerning** — отключить автоматический кернинг для вертикального текста.
- **Connect lines** (Соединение линий) — параметр, указывающий, насколько близко друг от друга должны размещаться концы сегментов контуров для того, чтобы произошло притягивание. Этот параметр также отвечает за распознавание горизонтальных и вертикальных линий. Он определяет, насколько близка должна быть линия к строгой горизонтали или вертикал, для того чтобы стать таковой. Данный параметр имеет силу при условии, что режим притягивания (**View**>**Snapping**>**Snap to Objects**) активен.
 - ◊ **Must be close** — должны быть близко.
 - ◊ **Normal** — нормально.
 - ◊ **Can be distant** — могут быть отдалены.
- **Smooth curves** (Сглаживание кривых) — параметр, задающий степень сглаживания контуров, созданных инструментом **Pencil** (Карандаш) в режиме автоматического распознавания форм (**Smooth/Straighten**).
 - ◊ **Off** — сглаживание отключено.
 - ◊ **Rough** — грубое сглаживание, контур сглаживается в незначительной степени.
 - ◊ **Normal** — нормальное.
 - ◊ **Smooth** — высокая степень сглаживания.
- **Recognize lines** (Распознавание линий) — параметр, указывающий, насколько прямой должна быть линия, нарисованная инструментом **Pencil** в режиме спрямления (**Straighten**), для того чтобы Flash сделал ее идеально прямой.
 - ◊ **Off** — распознавание прямых отключено.
 - ◊ **Strict** — строго, линия должна быть близка к прямой.
 - ◊ **Normal** — нормально.
 - ◊ **Tolerant** — линия может достаточно сильно отклоняться от правильной прямой.

- **Recognize shapes** (Распознавание форм) — параметр, определяющий, насколько точно форма контура, созданного инструментом **Pencil** в режиме спрямления (**Straighten**), должна соответствовать правильной геометрической форме, для того чтобы произошло автоматическое распознавание.
 - ◊ **Off** — автоматическое распознавание отключено.
 - ◊ **Strict** — строго, форма контура должна быть близка к правильной геометрической форме.
 - ◊ **Normal** — нормально.
 - ◊ **Tolerant** — форма контура может достаточно сильно отклоняться от правильной геометрической формы.
 - **Click accuracy** (Точность щелчка) — параметр, определяющий, насколько близко к объекту должен быть помещен курсор, для того чтобы произвести с ним соответствующее действие.
 - ◊ **Strict** — близко.
 - ◊ **Normal** — нормально.
 - ◊ **Tolerant** — достаточно отдален.
 - **Input Languages Settings** — языковые настройки для японского (китайского) или корейского языков.
- Вкладка **Clipboard** (Буфер обмена) позволяет задать параметры растровых изображений, скопированных в буфер обмена из рабочей среды Flash. При копировании происходит преобразование векторного изображения в точечное (растрирование). В буфере обмена изображение хранится в формате Windows Metafile, который содержит информацию одновременно и в растровом, и в векторном представлении. Вкладка содержит следующие установки (рис. 3.10).
- **Color depth** — глубина цвета, задается при помощи раскрывающегося списка.
 - **Resolution** — разрешение задается в точках на дюйм — dpi (dots per inch). Предлагаются значения **Screen** — экранное разрешение, **72 dpi**, **150 dpi**, **300 dpi**.
 - **Size limit** — максимальный размер оперативной памяти в килобайтах, отведенный для растрового изображения, помещаемого в буфер обмена.
 - **Smooth** — сглаживание растровых изображений, помещаемых в буфер обмена.

- **Gradients** — качество передачи градиентов, помещаемых в буфер. Чем выше качество передачи, тем больше времени может занять процесс копирования. Этот параметр имеет значение только в случае импорта изображения через буфер обмена во внешнее приложение.
 - ◊ **None** — не передавать.
 - ◊ **Fast** — плохое.
 - ◊ **Normal** — нормальное.
 - ◊ **Best** — наилучшее.

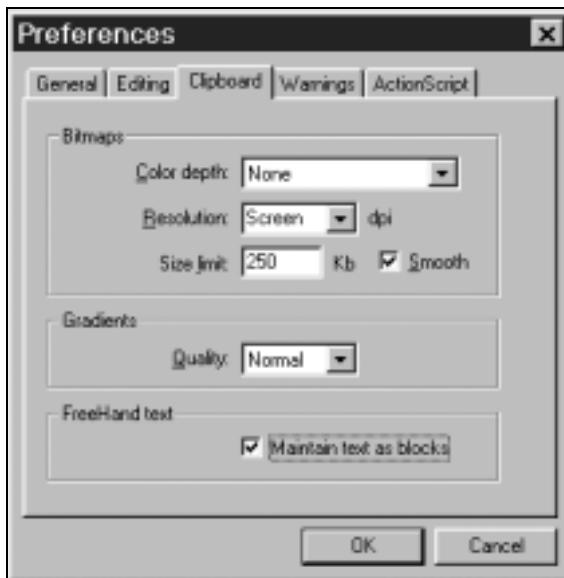


Рис. 3.10. Вкладка **Clipboard**
окна настройки рабочей среды **Preferences**

- **Maintain text as blocks** (В виде текстового блока) — этот параметр позволяет импортировать текст из программы Freehand в виде редактируемых текстовых блоков. Это, однако, не распространяется на текст по траектории.
- Вкладка **Warnings** (Предупреждения) содержит следующие параметры (рис. 3.11).
 - **Warn on save for Macromedia Flash MX Compatibility** — предупреждать о сохранении файлов, содержащих новые возможности работы в формате документа для более ранней версии — Flash MX.

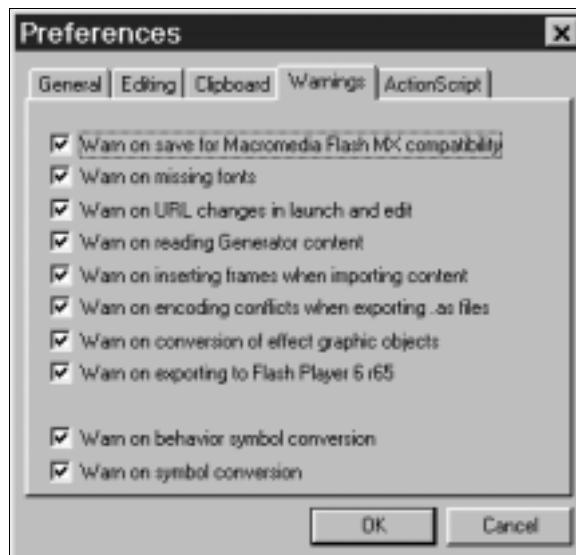


Рис. 3.11. Вкладка **Warnings**
окна настройки рабочей среды **Preferences**

- **Warn on missing fonts** — предупреждать об отсутствии в системе пользователя шрифтов, используемых в открытом документе.
- **Warn on URL changes in launch and edit** — предупреждать об изменении URL документа с момента последнего редактирования.
- **Warn on reading Generator content** — предупреждать о том, что объекты **Generator** не поддерживаются Flash MX 2004. В рабочей среде программы они будут помечены красным крестиком.
- **Warn on inserting frames when importing content** — предупреждать об автоматическом добавлении кадров на монтажную линейку при импорте видео и звуков.
- **Warn on encoding conflicts when exporting .as files** — предупреждать о конфликтах кодировок при импорте файлов сценариев ActionScript (файлы с расширением as).
- **Warn on conversion of effect graphic objects** — предупреждать о попытке редактирования объекта, к которому были применены эффекты монтажной линейки (Timeline effects).
- **Warn on exporting to flash player 6 r65** — предупреждать о публикации для более ранней версии Flash Player 6.

- **Warn on behavior symbol conversion** — предупреждать об изменении поведения клипа.
 - **Warn on symbol conversion** — предупреждать об изменении типа символа.
- Вкладка **ActionScript** содержит следующие параметры, позволяющие сделать процесс написания сценария более удобным (рис. 3.12).
- **Editing options.**
 - ◊ **Automatic indentation** — автоматический отступ абзаца при вводе соответствующей команды.
 - ◊ **Code hints** — всплывающие подсказки для элементов кода.

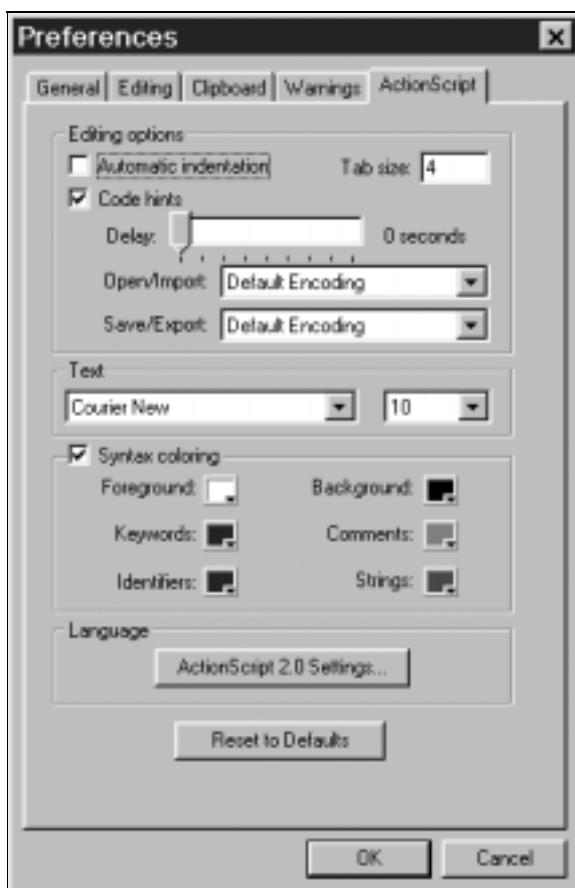


Рис. 3.12. Вкладка **ActionScript**
окна настройки рабочей среды **Preferences**

- ◊ **Tab size** — величина отступа строки текста.
- ◊ **Delay** — время, через которое появляется всплывающая подсказка.
- ◊ **Open/Import** — кодировка текста при открытии/импорте.
- ◊ **Save/Export** — кодировка текста при сохранении/экспорте.
- **Text** (Текст) — шрифт, используемый при создании сценария в редакторе ActionScript.
- **Syntax coloring** (Подсветка синтаксиса) — возможность отображения различных элементов синтаксиса цветом для быстрой ориентации в тексте сценария.
 - ◊ **Foreground** — основной текст.
 - ◊ **Keywords** — ключевые слова.
 - ◊ **Identifiers** — идентификаторы.
 - ◊ **Background** — фон.
 - ◊ **Comments** — комментарии.
 - ◊ **Strings** — строки.
- **Language** (Язык) — при помощи кнопки **ActionScript 2.0 Settings** создается список директорий (Classpath), содержащих файлы с расширением as с определением классов.
- **Reset to Defaults** (Восстановить по умолчанию) — восстановить настройки по умолчанию.

Настройки горячих клавиш

Горячие клавиши — это клавиатурный эквивалент выполнения команд рабочей среды, вызова инструментов, работы с кадрами и т. д. Горячие клавиши служат для быстрого доступа к необходимой операции. Это инструмент, позволяющий оптимизировать рабочий процесс. В *приложении 1* представлены клавиатурные эквиваленты всех команд главного меню программы. Этот набор является стандартным и установлен по умолчанию.

Кроме использования стандартного набора, Flash MX 2004 позволяет выбрать другой набор горячих клавиш либо создать свой собственный. Настройки выполняются при помощи команды главного меню **Edit>Keyboard Shortcuts**. В результате выполнения этой команды появляется одноименное диалоговое окно, представленное на рис. 3.13.

Раскрывающийся список **Current set** (Текущий набор) позволяет выбрать один из стандартных наборов горячих клавиш, используемых в более ранней версии Flash или в других приложениях, таких как FreeHand, Illustrator,

Photoshop и т. д. Кнопка **Duplicate set** позволяет создать копию набора, сохранив ее под новым именем, в которой позднее можно будет изменить любые сочетания клавиш. Кнопка **Rename Set** позволяет переименовать набор, а кнопка **Delete Set** удаляет его. Список **Commands** содержит группы операций, для которых можно задать или изменить клавиатурный эквивалент. Рядом с надписью **Description** указывается краткое описание команды. Поле **Shortcuts** служит для выбора клавиатурного эквивалента, который необходимо заменить, или для добавления нового сочетания клавиш для вызова соответствующей команды. Для добавления нового сочетания клавиш можно воспользоваться кнопкой **Add Shortcut**; кнопка **Remove Shortcut** удаляет выделенный клавиатурный эквивалент. Для того чтобы назначить горячие клавиши не имеющей их команде, нужно добавить пустое поле, нажав кнопку **Add Shortcut**. Затем в поле **Press key** простым нажатием требуемых клавиш вводится новое сочетание клавиш. Для того чтобы замена (или назначение) была установлена, необходимо нажать кнопку **Change**.



Рис. 3.13. Диалоговое окно **Keyboard Shortcuts**

Настройка глобальных параметров фильма

К глобальным параметрам фильма относятся следующие:

- размер рабочей области;
- цвет фона;
- скорость воспроизведения монтажной линейки;
- единицы измерения рабочей среды.

Для настройки всех перечисленных параметров используется команда меню **Modify>Document** или сочетание клавиш <Ctrl>+<J>. Выполнение данной команды приводит к появлению окна **Document Properties**, представленного на рис. 3.14.

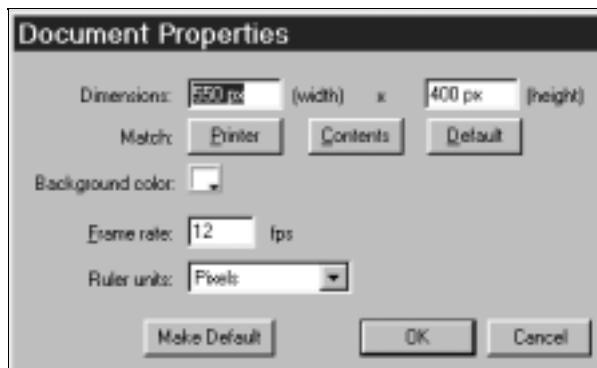


Рис. 3.14. Окно настройки глобальных параметров фильма **Document Properties**

Задать глобальные параметры фильма также можно при помощи Инспектора свойств, содержащего аналогичные настройки при отсутствии выделенных объектов (графики или кадров). Для отображения настроек глобальных параметров фильма в Инспекторе свойств необходимо щелкнуть инструментом **Selection** (Выделение) по свободному участку сцены, сняв выделение со всех объектов. Нажатие кнопки **Size** (Размер) Инспектора свойств также приводит к появлению окна **Document Properties**. Окно содержит следующие настройки.

- Dimensions** — размеры рабочей области по ширине (**width**) и высоте (**height**), задаются в текущих единицах измерения. Диапазон допустимых значений в пикселях — от 1 до 2880.
 - **Printer** — автоматическая установка размеров рабочей области равными размерам области печати принтера.

- **Contents** — размер рабочей области устанавливается таким образом, что все объекты, расположенные в положительном квадранте (т. е. правее и ниже левого верхнего угла рабочей области), оказываются в пределах рабочей области. Для того чтобы точно подогнать размер рабочей области к размеру размещённой на ней композиции, нужно совместить левый верхний угол ограничивающей рамки изображения и левый верхний угол рабочей области (см. разд. "Панель Align", гл. 4) и нажать кнопку **Contents**.
 - **Default** — устанавливает размер рабочей области, заданный по умолчанию.
- Background color** — выбор цвета фона из текущего каталога цветов. В качестве цвета фона может быть использован *только* сплошной (solid) цвет, однако он может содержать прозрачность.
- Frame rate <...> fps** — скорость воспроизведения монтажной линейки. Измеряется в единицах кадр в секунду (frames per second) и задается в диапазоне от 0,01 до 120 кадров в секунду (fps). Наиболее часто используемые значения — 12 или 24 fps.
- Ruler units** — единицы измерения рабочей среды. Вся информация о размерах отображается в этих единицах.
- Make Default** — сделать текущие настройки настройками, используемыми по умолчанию.

Использование учебных ресурсов Flash MX 2004

Вместе с приложением Flash MX 2004 поставляется большое количество справочной информации, включающей описание возможностей программы, пошаговые инструкции, справочник по ActionScript и примеры исходных файлов. Кроме того, на сайте компании Macromedia в разделе Support содержится множество справочных материалов.

Для работы со встроенной справочной системой используется главное меню **Help** (Справка). Оно содержит следующие команды:

- Help** — открывает одноименную панель для работы со встроенной справочной системой;
- How Do I** — переход к разделам справочной системы, позволяющим ознакомиться с основными элементами Flash-технологии;
- What's New** — новые возможности Flash MX 2004;
- Using Flash** — доступ ко всем разделам справочной системы;
- ActionScript Dictionary** — переход к разделу, содержащему словарь ActionScript, включающий достаточно полное описание всех элементов языка;

- Using Components** — переход к разделу, содержащему информацию об использовании компонентов;
- Samples** — примеры исходных файлов. При работе с операционной системой Windows 2000 или Windows XP примеры содержатся в папке по адресу: <drive>/Documents and Settings/<username>/Local Settings/Application Data/Macromedia/Flash MX 2004/En/Configuration/Samples. При работе с операционной системой Windows 98 примеры содержатся в папке по адресу: <drive>/Windows/Application Data/Macromedia/Flash MX 2004/En/ Configuration/Samples;
- Flash Exchange** — переход к разделу сайта компании Macromedia (http://www.macromedia.com/go/flash_exchange);
- Manage Extensions** — справка по использованию установленных расширений;
- Flash Support Center** — переход к разделу сайта компании Macromedia (http://www.macromedia.com/go/flash_support);
- About Flash** — информация о текущей версии продукта.

Панель **Help**, представленная на рис. 3.15, состоит из двух областей.

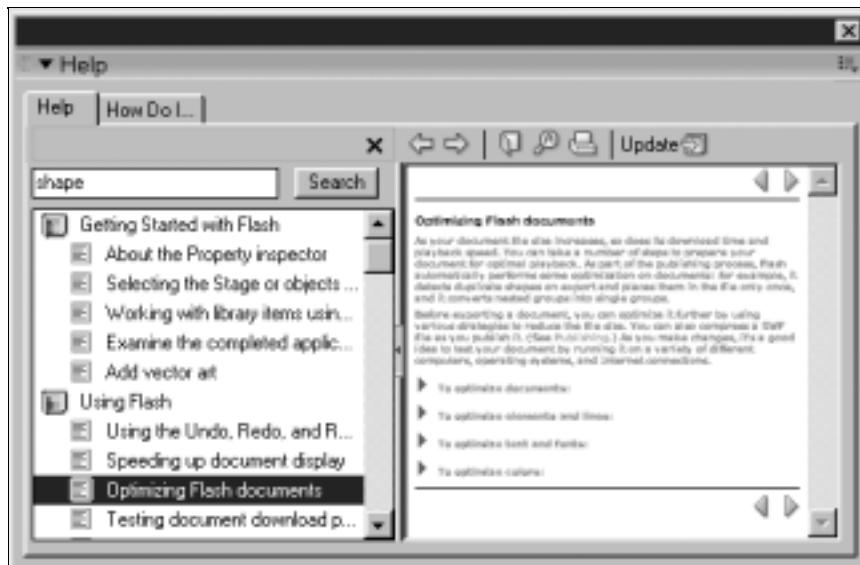


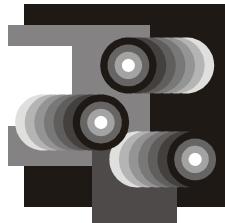
Рис. 3.15. Панель справочной системы **Help**

Слева находится древовидный перечень всех доступных разделов справки. Пиктограмма закрытой книги представляет собой раздел, содержащий под-

разделы. Двойной щелчок разворачивает вложенные пункты, повторный щелчок вновь сворачивает содержимое раздела. Справа расположено окно, в котором выводится содержание справки. Для того чтобы скрыть (открыть) левую часть панели, можно воспользоваться стрелкой, расположенной на середине границы, разделяющей соседние области, или щелкнуть на крестике, расположенном над перечнем разделов справа вверху.

Панель **Help** содержит следующие элементы управления:

- History Back** — переход к предыдущему разделу;
- History Forward** — переход к следующему разделу;
- Table of Contents** — содержание справки;
- Search** — поиск в справочной системе;
- Print** — вывод справки на печать;
- Download Help Content** — автоматическое обновление справочной системы через Интернет.



Глава 4

Инструментарий векторного редактора Flash MX 2004

Форма остается абстрактной, т. е. она не обозначает никакого реального предмета, а является совершенно абстрактным существом.

Василий Кандинский, "О духовном в искусстве"

Понятие о векторной графике

Для представления графической информации в компьютерной среде используются два совершенно разных (с точки зрения технологии реализации) подхода: *векторная* графика и *растровая* графика. Каждый из этих подходов имеет свои характерные особенности и, как следствие, свою область применения.

Любое векторное изображение состоит из *контуров* (*paths*), в свою очередь представляющих собой совокупность математических объектов, которые могут быть описаны аналитически. Объекты векторной графики очень легко поддаются трансформации, поскольку результат трансформации вычисляется путем изменения значений соответствующих коэффициентов уравнений, описывающих трансформируемый объект.

В частности, именно это свойство векторной графики объясняет *независимость изображения от масштаба*. Так, если мы имеем дело с кругом, то для векторной графики совершенно неважно, каков диаметр этого круга, изменение масштаба никак не повлияет на внешний вид объекта. Таким образом, у векторного изображения не существует жесткой привязки к размеру. На практике это означает, что внешний вид изображения определяется только параметрами устройства вывода (монитора, принтера и т. д.). Векторное изображение на экране монитора будет выглядеть настолько четко и аккуратно, насколько позволяют возможности монитора (разрешение, цветовой охват и т. д.).

В векторной графике форма превалирует над цветом, т. е. форма *первична*. Векторное изображение состоит из отдельных объектов, каждый из которых является независимым от остальных. Это дает широчайшие возможности манипуляции объектами, их выделения, перемещения, изменения различных параметров. Процесс работы с векторными объектами чрезвычайно гибок и подвижен. Их можно редактировать без опасения сделать необратимые преобразования.

Векторный формат представления графических данных, как правило, позволяет получать конечные *файлы небольшого размера*. Это связано с тем, что при записи сохраняется не само изображение, а определенный набор данных, описывающих входящие в его состав формы и их атрибуты. Таким образом, для вывода графической информации всякий раз при воспроизведении машина производит расчет, результатом которого является формируемое изображение. Размер файла, содержащего векторную графику, главным образом зависит от количества контуров, описывающих изображение. В некоторых случаях, если детализация слишком высока, размер конечного файла может занимать огромный объем.

Векторный формат позволяет *интегрировать текст в изображение*, представляя при этом широкие возможности форматирования. Это позволяет обращаться с текстом, используя все преимущества работы с векторными объектами. При этом качество отображения текста будет оптимальным.

Однако векторная графика имеет и свои ограничения. Сложность принципа реализации векторной графики не позволяет добиться полной реалистичности изображения. Несмотря на то, что современные векторные редакторы располагают широкими возможностями использования различных визуальных эффектов, свойственных скорее раstralной графике, в векторных изображениях трудно добиться фотографичности при передаче тонких нюансов.

Другое важное ограничение — отсутствие адекватного механизма ввода графической информации с автоматическим получением векторного изображения. На сегодняшний день совершенного векторного сканера не существует. Разнообразие природных форм безгранично, и вряд ли возможно найти алгоритм, позволяющий точно описывать их при помощи математического аппарата. Единственная возможность автоматизировать процесс ввода — использование автоматической трассировки. Однако трассировка далеко не всегда дает удовлетворительный результат, особенно в случае ее применения к фотoreалистичным изображениям.

Flash использует векторный формат представления графики. Это означает, что, находясь в рабочей среде, мы создаем изображения, используя векторную технологию. Однако Flash также содержит достаточно широкие возможности работы с раstralными изображениями (и даже с видеофайлами). При этом раstralное изображение обрабатывается как особый объект.

Элементы векторного объекта

В общем случае векторный объект состоит из двух элементов — обводки и заливки. *Обводка (Stroke)* — это линия (контур), которая может быть замкнута и ограничивает внутреннюю область (что и отражает название) или разомкнута (в этом случае никакой внутренней области обводка не содержит). *Заливка (Fill)* — это окрашенная внутренняя область (векторная форма), ограниченная обводкой (рис. 4.1).

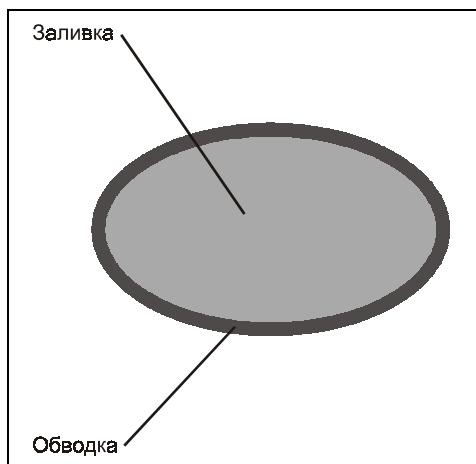


Рис. 4.1. Элементы векторного объекта

Далее в этой книге термины *контур* и *обводка* являются взаимозаменяемыми. Разделение на обводку и заливку вызвано тем, что поведение этих элементов и технология работы с ними различны.

Как частный случай, векторный объект может состоять только из одной обводки или, наоборот, представлять собой только векторную форму, т. е. заливку. Такие объекты могут являться результатом использования инструментов, создающих только контуры или только формы, если изначально задано отсутствие определенного элемента либо он намеренно удален.

Инструменты рисования и выделения

Блок управления цветом

Цвет элементов векторного объекта можно задать при помощи блока управления цветом, расположенного в разделе **Colors** (Цвет) панели инструментов, который представлен на рис. 4.2. Блок управления цветом включает в

себя два элемента управления, при помощи которых можно по отдельности задавать цвета обводки и заливки. Чтобы задать цвет обводки, нужно щелкнуть на цветовом образце рядом с пиктограммой с изображением карандаша и из появившегося текущего каталога цветов выбрать необходимый оттенок. Для указания цвета заливки используйте цветовой образец, расположенный рядом с пиктограммой с изображением ведра заливки. Обратите внимание, что в качестве цвета заливки может быть выбран *сплошной (Solid)* цвет или *градиент (Gradient)*, в то время как контур может быть окрашен только сплошным цветом. О синтезе градиента и его настройке рассказывается в гл. 6.

В нижней части блока управления цветом расположены три кнопки. Их назначение (слева направо):

- установка цветов по умолчанию — черного для обводки, белого для заливки;
- установка атрибута отсутствия цвета, приводящая к созданию объекта без обводки или без заливки. Альтернативой использованию этой кнопки является кнопка, расположенная в верхнем правом углу текущего каталога цветов, появляющаяся при щелчке на цветовом образце;
- обмен цветов обводки и заливки местами.

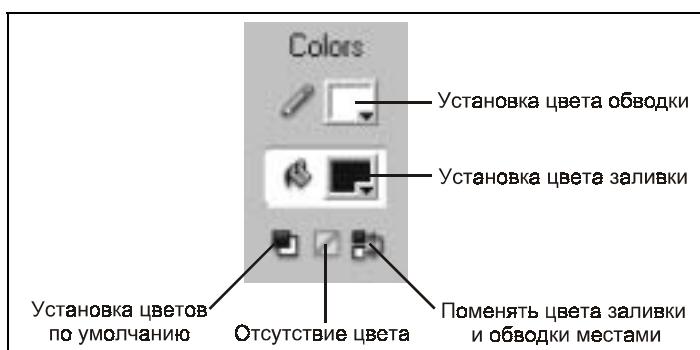


Рис. 4.2. Блок управления цветом **Colors**

Для того чтобы нарисовать объект, состоящий только из одного элемента (только обводка или только заливка), необходимо активизировать любой рисующий инструмент и перед началом рисования в блоке управления цветом на панели инструментов в качестве цвета отсутствующего элемента установить атрибут отсутствия цвета. При этом рядом с соответствующей пиктограммой в поле цветового образца появится белый квадрат, перечеркнутый красной полосой. В результате произведенных действий нарисованный вслед за этим объект будет содержать либо только заливку, либо только обводку.

Далее в этой главе будут рассмотрены основные инструменты, при помощи которых осуществляется работа по созданию графики и выполняются манипуляции с графическими объектами.

Некоторые инструменты содержат *модификаторы* (*Modifiers*), позволяющие задать режим и параметры работы инструмента. Модификаторы становятся доступны при активизации инструмента и располагаются в разделе модификаторов **Options** панели инструментов.

Инструмент *Line*

Line (Линия)  — это инструмент, предназначенный для создания контуров произвольной формы, состоящих из прямолинейных сегментов. Рабочий цвет для этого инструмента устанавливается при помощи блока управления цветом как цвет обводки. Для рисования необходимо щелкнуть на рабочем поле в начальной точке линии и протянуть курсор в направлении ее распространения. Если в процессе рисования удерживать клавишу <Shift>, то линия будет располагаться с дискретностью 45 градусов. Таким образом можно получить правильные вертикальные, горизонтальные и диагональные линии.

Чтобы нарисовать контур, состоящий из нескольких сегментов, нужно последовательно применять инструмент **Line**, так чтобы начальная точка нового сегмента совпадала с конечной точкой предыдущего сегмента. При этом важно добиться того, чтобы сегментыстыковались без зазоров. Возможные разрывы контура могут привести к тому, что он не будет заливаться цветом. Для того чтобы гарантироватьстыковку соседних сегментов без зазоров, можно применить модификатор **Snap to Objects** инструмента **Line** или активизировать режим притягивания к объектам при помощи команды меню **View>Snapping>Snap to Objects**. О том, что данный режим активен, свидетельствует галочка в меню рядом с его названием. В режиме притягивания при соединении соседних сегментов или замыкании контура рядом с курсором появляется пиктограмма кружка, увеличивающегося в размерах в момент притягивания.

Инструмент **Line** создает контуры или *обводки*. У обводки имеются три атрибута, определяющих ее внешний вид. Это *цвет*, *толщина* и *стиль*. Значения этих атрибутов задаются при помощи Инспектора свойств. Для того чтобы это сделать, необходимо активизировать любой инструмент, создающий контуры, или выделить имеющийся контур. Внешний вид Инспектора свойств представлен на рис. 4.3.

Щелчок на цветовом образце Инспектора свойств позволяет выбрать из текущего каталога цветов цвет обводки. Это альтернатива использованию блока управления цветом панели инструментов.



Рис. 4.3. Задание атрибутов обводки. Инспектор свойств

Следующее поле ввода предназначено для задания толщины контура в пикселях. Можно либо вручную ввести значения в диапазоне от 0,1 до 10, либо воспользоваться вертикальным слайдером, расположенным справа.

Раскрывающийся список используется для задания стиля обводки. Здесь можно выбрать тип штриха. По умолчанию используется значение **Solid** (Сплошной), дающее в результате сплошную линию, толщину которой можно варьировать в указанном диапазоне.

Следующее значение **Hairline** (Волосяная линия) позволяет нарисовать линии минимально возможной при данном разрешении экрана толщины. Причем интересной особенностью данного стиля является то, что такая линия всегда остается предельно тонкой вне зависимости от масштаба просмотра (рис. 4.4).

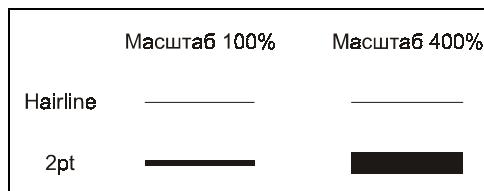


Рис. 4.4. Линия типа **Hairline**
не зависит от масштабирования

Далее последовательно в раскрывающемся списке представлены стили линий **Dashed** (Пунктирный), **Dotted** (Точечный), **Ragged** (Рваный), **Stipple** (Точечный пунктири), **Hatched** (Поперечно-штриховой). Толщину линии можно менять для каждого типа, за исключением волосяной. Кроме того, имеется возможность настроить каждый из данных стилей, изменяя определенные параметры, индивидуальные для каждого из них. Чтобы настроить соответствующий стиль, нужно выбрать его из списка и щелкнуть на кнопке **Custom** (Заказной). Читатель может самостоятельно поэкспериментировать с этой возможностью.

Следует иметь в виду, что использование сложных стилей может очень существенно (иногда в десятки раз!) увеличить конечный объем файла, поэтому не стоит использовать их без необходимости.

Инструмент *Oval*

Инструмент **Oval** (Овал)  — это инструмент для создания простейших геометрических фигур — овалов и кругов. В общем случае овал (круг) содержит и контур, и заливку. Перед началом рисования необходимо задать атрибуты обводки и заливки при помощи Инспектора свойств. Цвета элементов могут быть также установлены в блоке управления цветом панели инструментов. После этого необходимо активизировать инструмент (курсор при этом принимает вид перекрестия при внесении в рабочее поле) и, щелкнув в месте начала рисования, протянуть курсор в направлении распространения фигуры. Следует обратить внимание на то, что овал рисуется не из центра, а начиная от одного из углов ограничивающей его воображаемой прямоугольной рамки.

Если в процессе рисования удерживать клавишу <Shift>, можно получить правильную окружность.

Инструмент **Oval** содержит модификатор **Snap to Objects** , позволяющий в процессе рисования использовать режим притягивания к объектам.

Инструмент *Rectangle*

Инструмент **Rectangle** (Прямоугольник)  позволяет рисовать прямоугольники или правильные квадраты. Прямоугольник, так же как и овал, может содержать и заливку, и обводку. Принцип его использования очень похож на инструмент **Oval**. Для рисования правильных квадратов необходимо удерживать клавишу <Shift>.

У инструмента **Rectangle** имеется модификатор, который называется **Round Rectangle Radius** (Радиус скругления) . Он расположен в разделе модификаторов (**Options**) панели инструментов, и с его помощью можно создавать прямоугольники со скругленными углами. Задать радиус скругления можно двумя способами.

- Перед рисованием активизировать инструмент **Rectangle**, щелкнуть на модификаторе **Round Rectangle Radius** (или произвести два щелчка на инструменте **Rectangle**) и в появившееся диалоговое окно **Rectangle Settings** (Настройка прямоугольника) ввести значение в диапазоне от 0 до 999.
- Непосредственно во время рисования удерживать клавишу <↑> или <↓>, чтобы соответственно уменьшить или увеличить радиус скругления.

Все прямоугольники, нарисованные после установки радиуса скругления, будут иметь скругленные углы. Чтобы отменить действие этого модификатора, необходимо ввести нулевое значение в диалоговом окне **Rectangle Settings**.

Нужно отметить, что изменить радиус скругления уже нарисованного прямоугольника невозможно. Для этого придется создать новый объект, предварительно задав необходимое значение.

Инструмент **Rectangle** (Прямоугольник) содержит модификатор **Snap to Objects** , позволяющий в процессе рисования использовать режим притягивания к объектам.

Инструмент **Polystar**

Инструмент **Polystar** (Многоугольник)  применяется для создания правильных многоугольников и звезд. Он расположен в одной группе с инструментом **Rectangle**.

После того как инструмент **Polystar** (Многоугольник) активирован и заданы соответствующие атрибуты обводки и заливки, необходимо настроить режим его работы. Для этого используется кнопка **Options** Инспектора свойств. Нажатие на эту кнопку приводит к появлению диалогового окна **Tool Settings**, содержащего следующие элементы:

- Style** — раскрывающийся список, задающий тип фигуры **Polygon** (Многоугольник) или **Star** (Звезда);
- Number of Sides** — поле, в которое вводится количество вершин многоугольника или звезды. Допустимые значения лежат в диапазоне от 3 до 32;
- Star point size** — поле, позволяющее задать соотношение внутреннего и внешнего радиуса звезды в виде числа в диапазоне от 0 до 1. Чем ближе введенное число к 0, тем более остроконечной будет результирующая фигура, и, наоборот, чем ближе значение к 1, тем более округлой будет форма получившейся звезды.

Инструмент **Polystar** (Многоугольник) содержит модификатор **Snap to Objects** , позволяющий в процессе рисования использовать режим притягивания к объектам.

Инструмент **Selection**

Selection (Выделение)  — это многофункциональный инструмент, используемый для выполнения следующих операций:

- выделение объектов и их частей;
- перемещение, копирование объектов;
- редактирование формы объектов.

Необходимо отметить, что выделение частей объектов и редактирование их формы при помощи инструмента **Selection** возможно только для объектов

рабочего уровня (*подробнее об иерархии графических объектов см. далее в этой главе*).

Быстро активировать инструмент **Selection**, работая с другим инструментом, можно при помощи клавиши <Ctrl>.

Выделение объектов и их частей

Для того чтобы выделить объект (или его часть), можно действовать двумя способами.

Первый способ позволяет выделять объекты, при помощи щелчков на составляющих их элементах. Здесь необходимо запомнить несколько правил.

- Для того чтобы выделить *фрагмент контура* (обводки) фигуры, необходимо просто щелкнуть на нем. Выделенный фрагмент будет подсвечен. В данном случае фрагментом считается участок контура, лежащий между двумя соседними угловыми точками, либо концевая часть контура от точки, в которой произведен щелчок, до ближайшей угловой точки.
- Для того чтобы выделить *контуру* целиком, необходимо щелкнуть на нем два раза.
- Для выделения *заливки* нужно щелкнуть на ней. Выделенная заливка подсвечивается сплошной сеткой выделения (эта особенность имеет место только для объекта рабочего уровня).
- Чтобы выделить *контуру и заливку* одновременно, необходимо два раза щелкнуть на заливке.

Для того чтобы последовательно выделить несколько объектов, необходимо удерживать клавишу <Shift>, добавляя таким образом новые объекты к уже выделенным. В некоторых случаях бывает удобно включить режим, при котором последовательное выделение объектов осуществляется без нажатия дополнительных клавиш. Для этого в меню **Edit>Preferences** на вкладке **General** необходимо снять флажок **Shift Select**.

Другой способ выделения предполагает заключение выделяемых объектов или их частей в прямоугольную область выделения. Чтобы выполнить эту процедуру, нужно щелкнуть инструментом **Selection** в соответствующей точке сцены и, не отпуская правой кнопки мыши, протянуть курсор в нужном направлении. При этом становится видна прямоугольная рамка, указывающая на границы области выделения. При отпускании кнопки мыши все объекты (или их части), попадающие в эту область, будут выделены. Для выделения нескольких объектов этим способом также можно воспользоваться клавишей <Shift>.

Кроме того, чтобы выделить сразу все объекты на сцене, можно использовать команду главного меню **Edit>Select All** (<Ctrl>+<A>).

Для снятия выделения с объекта нужно выделить другой объект либо щелкнуть на свободном участке сцены. Если необходимо снять выделение лишь с некоторых из выделенных объектов, удерживайте клавишу <Shift>, последовательно щелкая на них. Для снятия выделения сразу со всех объектов сцены используется команда главного меню **Edit>Deselect All** (<Ctrl>+<Shift>+<A>).

Для того чтобы отключить отображение выделения объекта, нужно выполнить команду **View>Hide Edges** (Скрыть выделение). При этом выделение с объектов не снимается. Выполнение этой команды бывает полезно в случаях, когда необходимо увидеть, как выделенный объект (или несколько объектов) стыкуется с остальными объектами и с фоном. Для того чтобы каждый раз не снимать выделение, а затем вновь выделять объект, можно просто скрыть выделение. Для того чтобы вновь включить выделение, нужно снять флагок в меню **View>Hide Edges**.

С выделенными объектами можно выполнять целый ряд различных манипуляций. К их числу относятся:

- перемещение;
- копирование;
- вырезание в буфер обмена;
- вставка из буфера обмена;
- удаление;
- изменение цвета и параметров элементов объектов;
- операции трансформации.

Перемещение

Перемещение объектов осуществляется при помощи инструмента **Selection** либо при помощи клавиш перемещения курсора (клавиатуры). Использование курсоров позволяет смещать объект с определенной дискретностью, при этом значение шага смещения зависит от текущего масштаба просмотра. Так, при масштабе просмотра 100% объект смещается с дискретностью в один пиксел. Если при этом удерживать клавишу <Shift>, то смещение выполняется с дискретностью в 10 пикселов. Уменьшение масштаба просмотра приводит к увеличению шага смещения объекта, а увеличение масштаба — к уменьшению шага смещения объекта. Следует иметь в виду, что для перемещения объекта при помощи клавиатуры его предварительно необходимо выделить. Если речь идет о перемещении заливки или объекта наложенного уровня при помощи курсора мыши, предварительно выделять объект не обязательно. Достаточно разместить курсор над объектом и убедиться в том, что рядом с ним появляется пиктограмма крестообразной стрелки () , указывающая на то, что выполняется перетаскивание.

Копирование

Копирование объектов в буфер может осуществляться различными способами. Копируемый объект (или несколько объектов) должен быть выделен.

Первый способ предполагает использование команды главного меню **Edit>Copy** или использование ее клавиатурного эквивалента **<Ctrl>+<C>**.

Второй способ предполагает использование аналогичной команды **Copy** контекстного меню самого объекта, которое появляется при щелчке на объекте правой кнопкой мыши.

В результате выполнения любой из данных команд скопированный объект попадает в буфер обмена.

Вырезание объектов в буфер обмена выполняется либо при помощи команды главного меню **Edit>Cut** (ее клавиатурный эквивалент **<Ctrl>+<X>**), либо при помощи аналогичной команды **Cut** контекстного меню самого объекта.

В результате вырезания объект попадает в буфер обмена, исчезая при этом со сцены.

Вставка скопированного или вырезанного объекта из буфера обмена выполняется либо при помощи команды главного меню **Edit>Paste in Center** (ее клавиатурный эквивалент **<Ctrl>+<V>**), либо при помощи аналогичной команды **Paste** контекстного меню сцены, которое появляется при щелчке правой кнопкой мыши в любой точке сцены. В результате выполнения данной команды скопированный объект будет вставлен в центр рабочей области.

Кроме того, при помощи команды главного меню **Edit>Paste in Place** (**<Ctrl>+<Shift>+<V>**) можно вставить объект из буфера в ту же точку (относительно текущего начала координат), откуда он был скопирован или вырезан. Эта команда бывает весьма полезна при работе с объектами в нескольких кадрах.

Две команды — копирование и вставку — можно заменить одной процедурой, в результате выполнения которой объект будет скопирован в буфер и вставлен в указанное место. Выполнить эту процедуру можно, либо перемещая объект инструментом **Selection**, одновременно удерживая клавишу **<Ctrl>** (при этом рядом с курсором появляется пиктограмма "плюс"), либо выполнив команду главного меню **Edit>Duplicate** (**<Ctrl>+<D>**). В последнем случае скопированный объект разместится правей и ниже оригинала, частично перекрывая его. Если операция дублирования будет применена к объектам рабочего уровня, то в результате частичного перекрывания объектов они провзаимодействуют между собой, объединяясь или сегментируя друг друга (см. разд. "Понятие о рабочем и наложженном уровнях" данной главы).

Удаление

Удаление объекта приводит к его исчезновению со сцены, минуя буфер обмена. Чтобы удалить объект, его необходимо выделить и далее либо использовать команду главного меню **Edit>Clear** (<Backspace>), либо нажать клавишу <Delete>.

Изменение

цвета контура и заливки

Изменение цвета выделенных контура и заливки, а также изменение значения толщины и стиля выделенного контура осуществляются при помощи блока управления цветом (изменение цвета) или Инспектора свойств (изменение цвета и атрибутов контура). Для выполнения этих процедур необходимо выделить соответствующий элемент и задать новые параметры.

Редактирование формы

Редактирование формы при помощи инструмента **Selection** заключается в изменении длины и кривизны сегментов контура. Эти операции в равной степени могут быть применены как к обводкам, так и к заливкам. Редактирование формы инструментом **Selection** возможно только для объекта рабочего уровня.

Для того чтобы отредактировать форму векторного объекта, необходимо убедиться в том, что выделение снято, и подвести курсор к границе формы. При этом рядом с курсором появится пиктограмма дуги (если он подведен к гладкому участку контура) или угла (если курсор подведен к угловой точке). Воздействуя на соответствующий участок, можно трансформировать форму, например, изменить длину прямой линии или превратить круг в эллипс. Для того чтобы в процессе редактирования формы добавить в контур новую угловую точку, нужно использовать клавишу <Ctrl>. Таким образом можно, например, из круга сделать фигуру в виде сердца (рис. 4.5).

Модификатор **Snap to Objects** (Притягивание)  инструмента **Selection** позволяет активизировать режим притягивания к объектам. Использование данного модификатора является эквивалентом выполнения команды меню **View>Snapping>Snap to Objects**. Если режим притягивания активен, то при перемещении объектов при помощи инструмента **Selection** они притянутся к граням, вершинам или геометрическому центру других объектов. Для выполнения притягивания объект нужно перемещать за грань, вершину или геометрический центр. В этом случае рядом с курсором появляется пиктограмма кружка, который увеличивается в момент притягивания.

Инструмент **Selection** содержит еще два модификатора: **Straighten** (Спрямление) и **Smooth** (Сглаживание). Подробнее об их использовании будет рассказано в разд. "Оптимизация кривых" гл. 5.

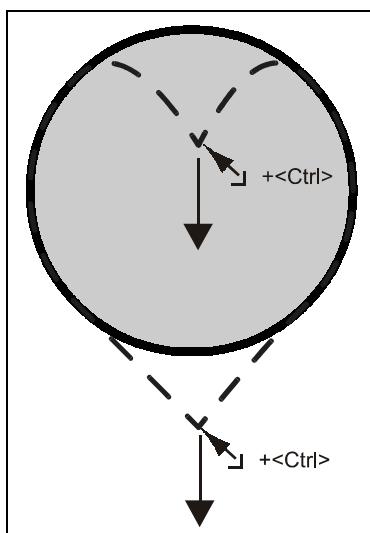


Рис. 4.5. Редактирование формы при помощи инструмента **Selection**

Инструмент *Lasso*

Lasso (Лассо) — инструмент выделения, позволяющий создавать области выделения произвольной формы.

Принцип использования **Lasso** (Лассо) достаточно прост. Необходимо вручную нарисовать область выделения требующейся формы. Все объекты, попавшие внутрь этой области, будут выделены.

Модификатор **Polygon Mode** (Многоугольное лассо) позволяет создавать произвольные области выделения, состоящие из прямолинейных сегментов. Для этого достаточно просто расставить точки, соответствующие вершинам, которые автоматически будут соединены прямыми отрезками. Чтобы замкнуть область выделения, необходимо совместить начальную точку с конечной точкой или просто пересечь начальный сегмент. Для быстрой активизации этого модификатора нужно удерживать клавишу <Alt>.

Два других модификатора — **Magic Wand** (Волшебная палочка) и **Magic Wand Properties** (Свойства Волшебной палочки) применяются при работе с растровой графикой (*подробную информацию см. в гл. 7*).

Инструмент *Pencil*

Pencil (Карандаш) — это инструмент для создания контуров произвольной формы от руки. У инструмента **Pencil** имеется модификатор **Pencil Mode** (Режим Карандаша), при помощи которого можно активизировать возможности автоматического распознавания формы контура (рис. 4.6).

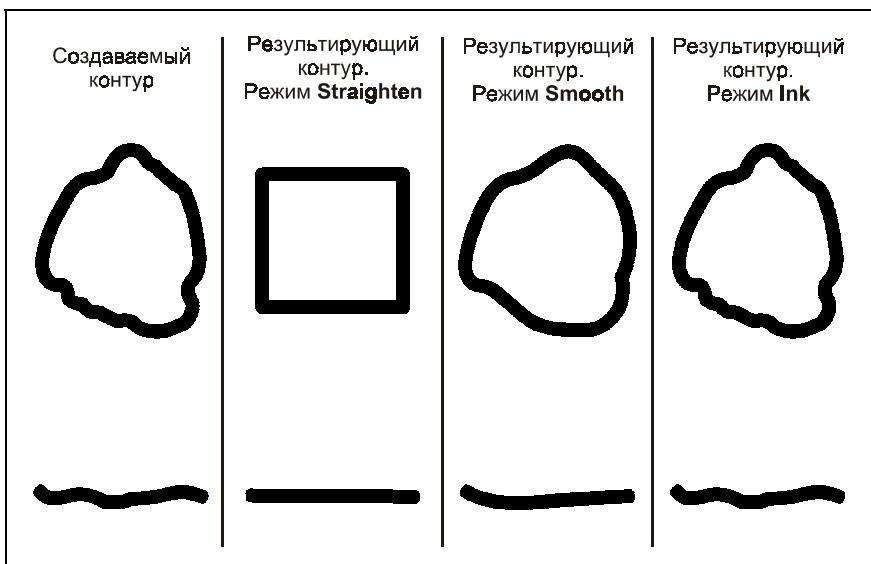


Рис. 4.6. Режимы инструмента **Pencil**

Шелчок на кнопке модификатора в разделе **Options** панели инструментов приводит к появлению списка, содержащего три режима: **Straighten**, **Smooth** и **Ink**.

Использование режима **Straighten** (Спрямление) приводит к тому, что по окончании рисования созданный контур принимает более правильные геометрические очертания. Так, если в режиме **Straighten** нарисовать замкнутый контур, хотя бы отдаленно напоминающий овал, он автоматически будет преобразован в овал, круг или прямоугольник.

Smooth (Сглаживание) — в этом режиме автоматическое распознавание формы приводит к сглаживанию сегментов нарисованного контура и уменьшению количества угловых точек. В результате линии становятся более гладкими и плавными.

Ink (Чернила)  — в этом режиме автоматическое распознавание формы контура отключено. Линии остаются в точности такими, как их нарисовали, корректировки не происходят. Как правило, использование режима **Ink** приводит к созданию контура, содержащего большое число опорных точек.

◀ Примечание ▶

При использовании инструмента **Pencil** замыкание контуров осуществляется вручную, режим притягивания к объектам **View>Snapping>Snap to Objects** здесь не действует. Если созданный вами контур предполагается впоследствии залить цветом (добавить заливку), необходимо удостовериться, что он замкнут (двойной щелчок на контуре должен выделять его целиком).

Инструмент **Brush**

Brush (Кисть)  — инструмент, при помощи которого можно создавать произвольные векторные формы (заливки).

У инструмента **Brush** имеется целый ряд модификаторов (рис. 4.7), позволяющих задать параметры и режимы его работы.

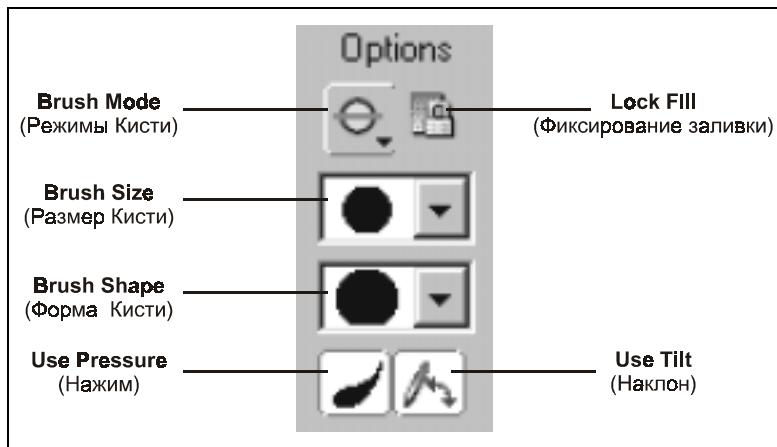


Рис. 4.7. Модификаторы инструмента **Brush**

Раскрывающийся список **Brush Size** (Размер Кисти) позволяет установить размер кисти, определяющий ширину ее мазка.

При помощи раскрывающегося списка **Brush Shape** (Форма Кисти) задается форма кисти. Эта установка влияет на характер мазка. Можно, например, выбрать каллиграфическую кисть, получив мазок переменной ширины.

Модификатор **Use Pressure** (Нажим) позволяет активизировать чувствительность к нажиму. Эта опция применяется только при работе с графическим планшетом, чувствительным к нажиму (например, Wacom Tablet). Получившиеся в результате мазки напоминают работу с обычной кистью, когда имеется возможность варьировать ширину мазка в зависимости от силы нажатия.

Модификатор **Use Tilt** (Наклон) позволяет варьировать угол наклона мазка в зависимости от угла наклона электронного пера. Эта опция также применяется только при работе с графическим планшетом. Угол наклона определяется как угол между плоскостью планшета и осью электронного пера.

Модификатор **Brush Mode** (Режим Кисти) позволяет установить один из пяти режимов работы этого инструмента (рис. 4.8).

- Paint Normal** (Нормальный режим) — в этом режиме мазки кисти перекрывают контур и заливку векторного объекта.

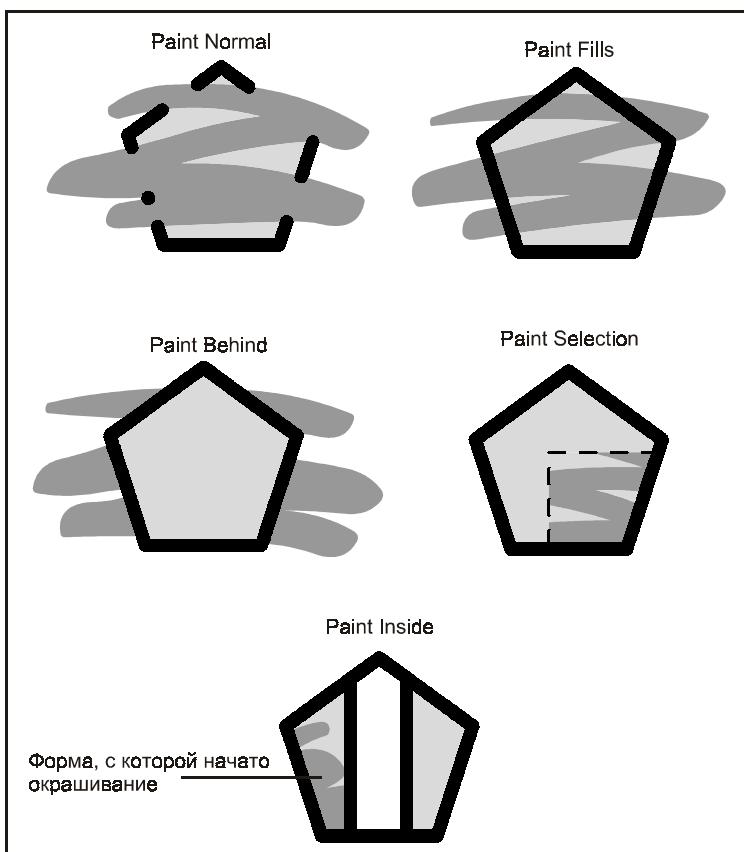


Рис. 4.8. Режимы работы инструмента **Brush**

- Paint Fills** (Окраска заливок) — в этом режиме окрашиваются только векторные формы (заливки) и фон. Контуры остаются без изменений.
- Paint Behind** (Окраска позади) — окрашивается только фон под векторным объектом.
- Paint Selection** (Окраска выделенного) — окрашиваются только векторные формы (заливки), находящиеся внутри выделенной области.
- Paint Inside** (Окраска внутри) — в этом режиме окрашивается только та векторная форма, с которой начато действие. Если рядом располагаются несколько не связанных или разноцветных векторных форм, то окрашена будет только та из них, к которой кисть была применена первой.

При помощи поля **Smoothing** (Сглаживание) Инспектора свойств можно задать степень автоматического сглаживания векторных форм, создаваемых кистью. В результате сглаживания уменьшается количество опорных точек, формы становятся более лаконичными. Диапазон допустимых значений для параметра **Smoothing** — от 0 до 100. Соответствующее значение вводится либо вручную, либо при помощи вертикального слайдера, расположенного справа. Чем больше вводимое значение, тем большая степень сглаживания будет достигнута.

Инструмент **Brush** имеет еще один модификатор **Lock Fill** (Фиксация заливки). Он используется только при работе с градиентными или раstroвыми заливками (*подробнее о его использовании будет рассказано в гл. 6*).

Примечание

Ширина мазка кисти остается неизменной вне зависимости от масштаба просмотра. Так, мазок, созданный кистью соответствующей ширины при масштабе отображения 100%, будет выглядеть в два раза тоньше, чем мазок, выполненный кистью той же ширины при масштабе просмотра 50%. Однако изменение масштаба просмотра не изменяет размер уже существующих мазков.

Инструмент *Ink Bottle*

Ink Bottle (Чернильница)  — инструмент, позволяющий изменить параметры уже существующего контура или добавить обводку векторной форме. Чернильница — инструмент, работающий с контурами. Чтобы изменить значение одного из атрибутов контура или добавить обводку векторной форме, не имеющей контура, необходимо активизировать чернильницу, задать соответствующие параметры в блоке управления цветом или на панели Инспектора свойств и щелкнуть на векторной форме.

Поскольку изменить параметры контура можно также при помощи инструмента **Selection** (см. выше), основное назначение Чернильницы состоит именно в создании обводки для формы, не имеющей таковой.

Инструмент **Paint Bucket**

Инструмент **Paint Bucket** (Ведро Заливки)  служит для создания заливок. С его помощью выполняется заполнение контуров или изменение цвета и типа заливки.

Для применения **Paint Bucket** нужно выбрать подходящий цвет (и тип) заливки в блоке управления цветом, активизировать инструмент и щелкнуть внутри контура. Для того чтобы можно было залить контур, он *должен* быть замкнут.

Однако это правило не является строгим, поскольку у инструмента **Paint Bucket** имеется модификатор **Gap Size** (Величина Зазора)  , позволяющий задать определенный допуск на величину разрыва, при котором контур все же будет залит. При щелчке на кнопке модификатора появляется выпадающее меню, содержащее четыре значения допуска, которые отличаются по степени толерантности к величине разрыва.

- Don't Close Gaps** (Не заливать зазоры) — в этом режиме контур, содержащий зазор, залит не будет.
- Close Small Gaps** (Заливка небольших зазоров) — заливаются только контуры, содержащие небольшие зазоры.
- Close Medium Gaps** (Заливка средних зазоров) — заливаются контуры, содержащие зазоры средней величины.
- Close Large Gaps** (Заливка больших зазоров) — возможна заливка контура, имеющего большой зазор.

Использование модификатора **Gap Size** имеет своеобразную особенность. Поскольку величина допустимого зазора количественно никак не регламентируется, а представляет собой качественную характеристику, работа с этими режимами зависит от текущего масштаба просмотра. Так, контур, содержащий зазор, который при масштабе просмотра 100% может быть залит только в режиме **Close Large Gaps**, при масштабе просмотра 10% заливается даже в режиме **Close Small Gaps**.

Как и инструмент **Brush** (Кисть), **Paint Bucket** (Ведро Заливки) имеет модификатор **Lock Fill** (Фиксация заливки). Подробнее о его использовании будет рассказано в гл. 6.

Инструмент *Eyedropper*

Eyedropper (Пипетка)  — это вспомогательный инструмент, позволяющий взять образец заливки или обводки одного объекта и применить его к другому объекту. При наведении инструмента **Eyedropper** на соответствующий элемент векторного объекта рядом с курсором появляется пиктограмма кисти (заливка) или карандаша (контур). Щелчок на элементе автоматически меняет пипетку на ведро заливки или чернильницу соответственно, причем в качестве текущих устанавливаются параметры образца (цвет, толщина и стиль линии).

Если при использовании инструмента **Eyedropper** удерживать клавишу **<Shift>**, то взятый в качестве образца цвет назначается в качестве текущего и для обводки, и для заливки, кроме того, смены пипетки на другой инструмент не происходит.

Примечание

При помощи пипетки можно "подобрать" цвет с любой точки экрана, даже за пределами окна текущего приложения. Чтобы сделать это, необходимо щелкнуть на цветовом образце в блоке управления цветом панели инструментов или Инспектора свойств и, не отпуская кнопки мыши, увести курсор, принявший вид пипетки, в любую точку экрана. При этом в качестве цветового образца будет отображаться оттенок той точки, над которой курсор находится в данный момент.

Инструмент *Eraser*

Eraser (Ластик)  — инструмент, являющийся антагонистом **Brush** (Кисть), предназначенный для стирания элементов векторного объекта.

Инструмент **Eraser** содержит несколько модификаторов.

Модификатор **Eraser Shape** (Форма Ластика), представляющий собой выпадающий список, позволяет установить размер и форму ластика.

Как и инструмент **Brush**, ластик содержит модификатор режима **Eraser Mode** (Режим Стирания) . Режимы стирания ластика идентичны режимам работы кисти, с той лишь разницей, что кисть создает формы, а ластик удаляет элементы векторных объектов.

- Erase Normal** (Нормальный режим) — стирание и контуров, и заливок.
- Erase Fills** (Стирание заливок) — стирание только заливок.
- Erase Lines** (Стирание контуров) — стирание только контуров.

- Erase Selected Fills** (Стирание выделенных областей) — стирание только выделенных областей заливок.
- Erase Inside** (Стирание внутри) — стирается только та заливка, с которой начато стирание.

Модификатор **Faucet** (Кран)  используется для стирания элементов векторных объектов целиком. В этом режиме щелчок на контуре или заливке удалит их полностью.

Кроме того, при помощи ластика можно быстро очистить сцену — для этого достаточно два раза щелкнуть на его пиктограмме в панели инструментов.

Трансформация объектов

Термин "трансформация" охватывает целую группу операций, которые могут производиться над объектами, изменения их внешний вид. Можно выделить следующие операции трансформации:

- масштабирование (Scale);
- зеркальное отражение (Flip);
- вращение (Rotate);
- скос (Skew);
- искажение (Distort);
- редактирование при помощи огибающих (Envelope).

Как и многие другие операции Flash, операции трансформации могут выполняться различными способами. Далее все они будут подробно описаны.

Инструмент *Free Transform*

Инструмент **Free Transform** (Свободное трансформирование)  позволяет выполнять практически все виды трансформаций векторных объектов.

Чтобы применить инструмент **Free Transform** (Свободное трансформирование), объект (или несколько объектов) либо должен быть предварительно выделен, либо заключен в прямоугольную область, созданную самим инструментом (сделать это можно, просто щелкнув на объекте). Трансформировать можно, как весь объект целиком, так и его выделенные отдельные части (в случае, если это объект рабочего уровня). Во время сеанса трансформации объект находится внутри прямоугольной рамки, которая называется **Bounding Box** (Ограничивающая рамка). В углах и на серединах граней Ограничиваю-

щей рамки расположены маркеры трансформации, воздействуя на которые, можно выполнять те или иные операции (рис. 4.9).



Рис. 4.9. Ограничивающая рамка инструмента **Free Transform**

Все операции трансформации выполняются относительно некой точки, называемой *центром трансформации* (Transformation Point). Во время сеанса трансформации точка трансформации отображается белым круглым маркером. По умолчанию она совпадает с геометрическим центром объекта. Сместив точку трансформации, можно добиться того, что все операции будут выполняться относительно ее нового местоположения (рис. 4.10). Однако следует иметь в виду, что при трансформации объекта рабочего уровня смещение точки трансформации действует только в течение сеанса трансформации. При переключении на другой объект или при выборе другого инструмента точка трансформации возвращается в геометрический центр.

Для *перемещения* объекта во время сеанса трансформации можно взяться за любой его участок, свободный от маркеров, или воспользоваться курсорами клавиатуры.

Рассмотрим операции, которые выполняются самим инструментом **Free Transform** (без участия модификаторов).

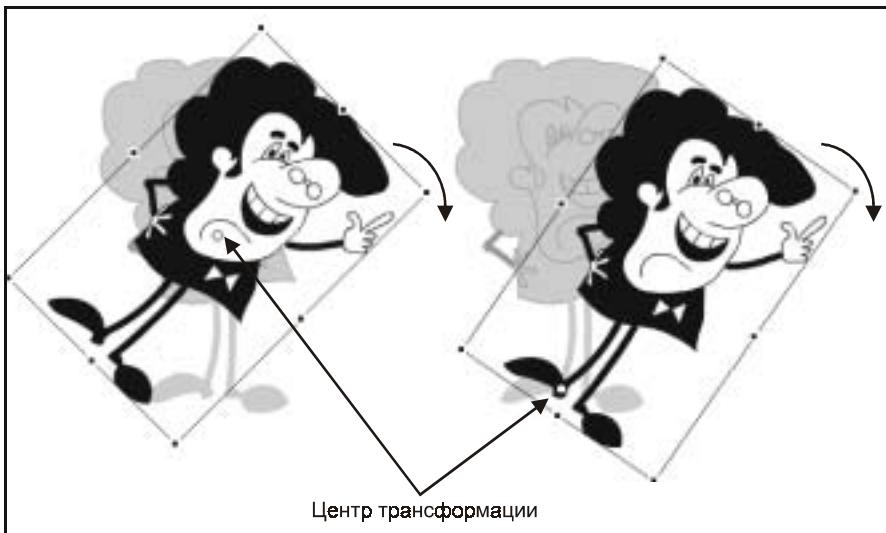


Рис. 4.10. Изменение положения точки трансформации

Масштабирование (Scale)

Существуют два типа масштабирования: пропорциональное и непропорциональное. Для того чтобы изменить масштаб объекта, необходимо потянуть за любой маркер ограничивающей рамки. При этом курсор должен принять вид двунаправленной стрелки . Для пропорционального масштабирования нужно удерживать клавишу <Shift>.

По умолчанию масштабирование объектов рабочего уровня выполняется относительно противоположного маркера ограничивающей рамки. Для масштабирования относительно центра нужно удерживать клавишу <Alt>. Объекты наложенного уровня, наоборот, масштабируются относительно центра трансформации. Для их масштабирования относительно противоположного маркера ограничивающей рамки нужно удерживать клавишу <Alt>.

Зеркальное отражение (Flip)

Для того чтобы вручную отразить объект по горизонтали или вертикали, нужно взяться за маркер, расположенный на середине вертикальной или горизонтальной грани ограничивающей рамки, и потянуть до пересечения формы с самой собой.

Вращение (Rotate)

Для поворота объекта необходимо воздействовать на угловой маркер трансформации. Курсор должен быть размещен в непосредственной близости

от маркера (а не прямо на нем) вне ограничивающей рамки. При этом он принимает вид дугообразной стрелки  . Если во время вращения удерживать клавишу <Shift>, то вращение будет выполняться с дискретностью в 45 градусов.

По умолчанию вращение выполняется относительно точки трансформации объекта. Для того чтобы выполнить вращение относительно противоположной вершины ограничивающей рамки, необходимо удерживать клавишу <Alt>.

Чем дальше находится курсор от объекта во время поворота (чем больше плечо рычага), тем меньше дискретность смещения и, соответственно, тем точнее можно позиционировать объект.

Скос (Skew)

Для выполнения скоса нужно воздействовать на одну из граней ограничивающей рамки между маркерами трансформации. При этом курсор принимает вид двух разнонаправленных стрелок  . При воздействии на боковые грани ограничивающей рамки курсор нужно смещать в вертикальном направлении. В этом случае выполняется скос по вертикали. При воздействии на нижнюю или верхнюю грани ограничивающей рамки курсор нужно смещать в горизонтальном направлении. В этом случае выполняется скос по горизонтали.

По умолчанию скос выполняется относительно противоположной грани ограничивающей рамки. Для того чтобы выполнить скос относительно точки трансформации, необходимо удерживать клавишу <Alt>.

Искажение (Distort)

Для выполнения искажения необходимо воздействовать на маркер трансформации, одновременно удерживая нажатой клавишу <Ctrl>. При этом курсор принимает вид белой стрелки  . Если таким образом воздействовать на угловой маркер трансформации, то происходит собственно искажение формы. Если перемещать маркер трансформации, расположенный на середине грани, то будет выполняться скос одновременно с масштабированием.

Если при перемещении углового маркера вместе с клавишей <Ctrl> удерживать клавишу <Shift>, то результатом будет эффект трапеции (taper) или перспективы. Суть этой операции заключается в том, что два соседних маркера трансформации, расположенные в вершинах ограничивающей рамки, смещаются на одно и то же расстояние, но в противоположных направлениях, благодаря чему сама рамка принимает вид трапеции.

Применение искажения возможно только к объектам рабочего уровня (см. разд. "Понятие о рабочем и наложенным уровнях" данной главы).

Для выполнения операций масштабирования, вращения, скоса, искажения можно соответственно использовать модификаторы инструмента **Free Transform**: **Scale** (Масштабирование), **Rotate and Skew** (Поворот и Скос) и **Distort** (Искажение). Таким образом, данные модификаторы дублируют возможности самого инструмента.

Редактирование при помощи огибающих (**Envelope**)

Для выполнения этого вида трансформации необходимо использовать модификатор **Envelope** (Оболочка). Его активизация приводит к тому, что на ограничивающей рамке появляются касательные к маркерам трансформации линии с управляемыми точками на концах, помеченными круглыми маркерами. Эти касательные называются *управляющими линиями*. В свою очередь, управляющие линии состоят из двух отрезков. Редактирование формы выполняется при помощи воздействия на маркер трансформации или на управляющую точку.

При смещении маркера трансформации выполняется параллельный перенос управляющей линии. Это приводит к изменению радиуса кривизны прилежащих сегментов ограничивающей рамки, что, в свою очередь, отражается на редактируемой форме (рис. 4.11).



Рис. 4.11. Редактирование при помощи огибающих (модификатор **Envelope**)

Перемещая управляющую точку на одном из отрезков касательных, можно изменять характер кривизны прилежащих сегментов ограничивающей рамки, создавая точку перегиба, что, в свою очередь, отражается на редактируемой форме.

Для того чтобы редактировать только один из отрезков управляющих линий, необходимо перемещать соответствующую угловую точку, удерживая при этом нажатой клавишу <Alt>. В результате редактируется только один прилежащий сегмент ограничивающей рамки, что, в свою очередь, отражается на редактируемой форме.

Редактировать при помощи огибающих можно только объекты рабочего уровня (см. разд. "Понятие о рабочем и наложенным уровнях" данной главы).

Сводная табл. 4.1 иллюстрирует возможности применения инструмента **Free Transform** и его модификаторов.

Таблица 4.1. Инструмент *Free Transform* и его модификаторы

Инструмент/ модификатор	Допустимые операции	Способ выполнения
Инструмент Free Transform	Перемещение объекта	Поместить курсор внутрь ограничивающей рамки (Bounding box) не на центр трансформации
	Изменение центра транс- формации	Перетащить центр трансформации (белый маркер) в новую точку
	Поворот	Поместить курсор снаружи углового маркера ограничивающей рамки. Объект будет вращаться вокруг своего центра трансформации. С <Shift> — вращение с дискретностью 45 градусов. С <Alt> — вращение вокруг противоположного угла ограничивающей рамки
	Масштабиро- вание	Потянуть за угловой маркер для масштабирования по вертикали и горизонтали одновременно. С <Shift> — пропорциональное масштабирование.
	Зеркальное отражение	Потянуть за маркер, расположенный на стороне ограничивающей рамки, для масштабирования в горизонтальном или вертикальном направлении
		Взяться за любой маркер ограничивающей рамки и потянуть до пересечения формы с самой собой

Таблица 4.1 (продолжение)

Инструмент/ модификатор	Допустимые операции	Способ выполнения
	Искажение	Удерживая клавишу <Ctrl>, потянуть за угловой или боковой маркер ограничивающей рамки. Потянуть за угловой маркер, удерживая <Shift>+<Ctrl>, для перемещения соседних углов на одно и то же расстояние, но в противоположном направлении (эффект трапеции)
Модификатор Rotate and Skew (Поворот и Скос) 	Поворот	Переместить угловой маркер. Дискретность вращения зависит от расстояния от курсора до углового маркера. С <Shift> — вращение с дискретностью 45 градусов. С <Alt> — вращение вокруг противоположного угла
	Скос	Потянуть за маркер, расположенный на стороне ограничивающей рамки
	Изменение центра трансформации	Перетащить центр трансформации в новое положение
Модификатор Scale (Масштаб) 	Масштабирование	Потянуть за угловой маркер для выполнения пропорционального масштабирования или за боковой для масштабирования по вертикали или горизонтали. Потянуть за угловой маркер с <Shift> для непропорционального масштабирования по вертикали и горизонтали одновременно. С <Alt> — масштабирование объектов рабочего уровня от центра и объектов наложенного уровня относительно противоположно-го маркера ограничивающей рамки
	Зеркальное отражение	Взяться за любой маркер ограничивающей рамки и потянуть до пересечения формы с самой собой
	Изменение центра трансформации	Перетащить центр трансформации в новое положение
	Масштабирование	Потянуть за угловой маркер для выполнения пропорционального масштабирования или за боковой для масштабирования по вертикали или горизонтали. Потянуть за угловой маркер с <Shift> для непропорционального масштабирования по вертикали и горизонтали одновременно. С <Alt> — масштабирование от центра

Таблица 4.1 (окончание)

Инструмент/ модификатор	Допустимые операции	Способ выполнения
Модификатор Distort (Искажение)	Искажение	Потянуть за угловой (с <Ctrl> — для перемещения соседних углов на одно и то же расстояние, но в противоположном направлении) или боковой маркер ограничивающей рамки
		
Модификатор Envelope (Огибающие)	Редактирование формы объекта	При помощи управляющих точек перемещать управляющие линии сегментов кривых, изменяя их радиус и кривизну. С <Alt> можно изменить угол между отрезками управляющих линий. С <Ctrl> — смещается вся грань
		

Меню **Modify>Transform**

Для трансформации выделенных объектов можно также использовать раздел главного меню **Modify>Transform**. Здесь содержатся следующие команды:

- Free Transform** — активизирует инструмент **Free Transform**;
- Distort** — активизирует модификатор **Distort** (Искажение);
- Envelope** — активизирует модификатор **Envelope** (Огибающие);
- Scale** — активизирует модификатор **Scale** (Масштабирование);
- Rotate and Skew** — активизирует модификатор **Rotate and Skew** (Вращение и Скос);
- Scale and Rotate** — позволяет задать точное значение масштаба в процентах и угла поворота в градусах; положительному значению соответствует поворот по часовой стрелке, отрицательному — против часовой стрелки;
- Rotate 90 CW** — поворот по часовой стрелке на 90 градусов (clockwise);
- Rotate 90 CCW** — поворот против часовой стрелки на 90 градусов (counterclockwise);
- Flip Vertical** — точное зеркальное отражение относительно горизонтальной оси симметрии;
- Flip Horizontal** — точное зеркальное отражение относительно вертикальной оси симметрии;
- Remove Transform** — отмена трансформации.

Примечание

Первые три из этих команд можно также выполнить, используя контекстное меню самого объекта, которое появляется при щелчке на нем правой кнопкой мыши. При работе с объектом наложенного уровня в контекстном меню доступна только команда **Free Transform**.

Палитра **Transform**

Точное трансформирование выполняется при помощи палитры **Transform** меню **Window>Design Panels** (<Ctrl>+<T>). Внешний вид палитры представлен на рис. 4.12.

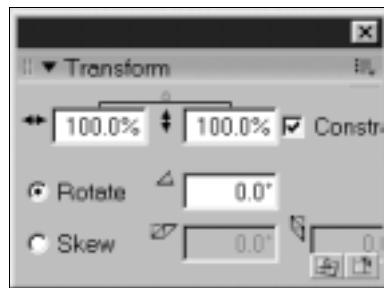


Рис. 4.12. Палитра **Transform**

Два верхних поля предназначены для задания масштаба по горизонтали и вертикали в процентах. Для пропорционального масштабирования нужно установить флажок **Constrain** (Сохранять пропорции). В этом случае значение масштаба достаточно задать только в одном из полей.

Следующие два поля **Rotate** (Поворот) и **Skew** (Скос) используются для задания точных значений угла поворота или скоса по горизонтали и по вертикали. Эти значения задаются в градусах, причем положительному значению соответствует поворот (скос) по часовой стрелке, отрицательному — поворот против часовой стрелки (рис. 4.13).

Кнопка **Copy and Apply Transform** (Трансформировать копию) в нижней части палитры позволяет скопировать объект и применить трансформацию к копии. При помощи этой функции можно создавать довольно сложные геометрические фигуры (рис. 4.14).

Кнопка **Reset** (Восстановить) отменяет все сделанные трансформации, возвращая объект в первоначальное состояние.

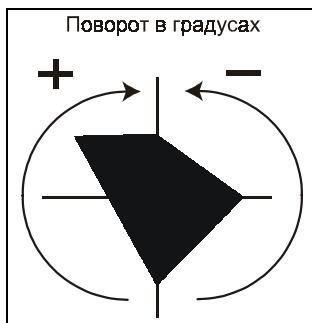


Рис. 4.13. Правило знаков при повороте и скосе объектов

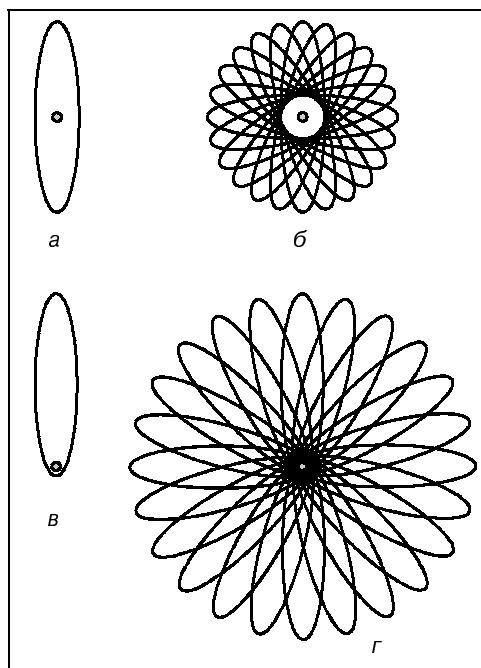


Рис. 4.14. Применение трансформирования к копии объекта:
а — начальный объект; б — копирование с поворотом на 15 градусов;
в — смещение центра трансформации объекта;
г — копирование с поворотом на 15 градусов

Палитра *Info*

Данная палитра предназначена для задания абсолютных размеров объекта и его точного позиционирования в координатах сцены. Кроме того, палитра **Info** содержит еще ряд дополнительных сведений.

Для вызова этой палитры необходимо выполнить команду **Window>Design Panels>Info** или использовать сочетание клавиш <Ctrl>+<I>. Внешний вид палитры представлен на рис. 4.15. Поля ввода палитры активизируются при выделении объекта на сцене.

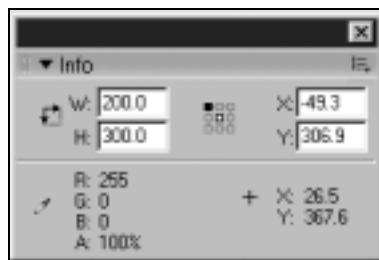


Рис. 4.15. Палитра Info

Поля **Width of Instance** (Ширина) и **Height of Instance** (Высота) используются для задания точных размеров объекта по ширине и высоте соответственно в текущих единицах измерения, устанавливаемых в меню **Modify>Document**.

Поля **X location of instance** (X-координата) и **Y location of instance** (Y-координата) позволяют задать точные координаты объекта относительно начала координат. Начало координат сцены совпадает с левым верхним углом рабочей области. Положительная полуось абсцисс (ось X) направлена вправо, а положительная полуось ординат (ось Y) направлена вниз; таким образом, вся рабочая область лежит в положительном квадранте (рис. 4.16).

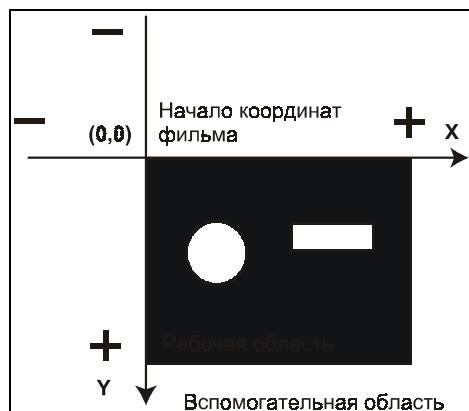


Рис. 4.16. Отсчет координат фильма

В качестве точки регистрации объекта, т. е. точки, принадлежащей объекту, координаты которой ассоциируются с координатами самого объекта, можно использовать либо левый верхний угол воображаемой ограничивающей рамки объекта, либо его геометрический центр. Указать местоположение точки регистрации можно при помощи расположенной слева от полей координат пиктограммы **Symbol position** (Точка регистрации). На пиктограмме необходимо выделить либо верхний левый угол, либо центральную точку. Выделенная точка отображается черным квадратиком. Положение выделенной точки указывает на местоположение точки регистрации.

Поля, расположенные в верхней половине панели **Info**, предназначены как для чтения, так и для записи. Таким образом, они служат и для вывода информации, и для изменения соответствующих значений. Нижняя часть панели **Info** служит только для чтения, здесь нельзя производить никакие изменения.

Поле **Color at cursor location** (Цвет в местоположении курсора) — содержит координаты цвета и степень прозрачности точки, над которой в данный момент находится курсор (работает только для сплошных заливок).

В поле **Cursor location** (Координаты курсора) — указываются текущие координаты курсора относительно начала координат.

Примечание

Начало координат сцены находится в левом верхнем углу рабочей области. В среде редактирования символа (см. гл. 10) начало координат находится в его точке регистрации.

Абсолютные размеры объектов и их координаты можно также задать при помощи Инспектора свойств. Здесь имеются четыре поля, аналогичные соответствующим полям панели **Info**. Данные поля становятся активными всякий раз при выделении объекта. Значок замка справа позволяет включить режим пропорционального масштабирования (рис. 4.17). В этом случае замок закрывается, сверху и снизу от него появляются уголки.

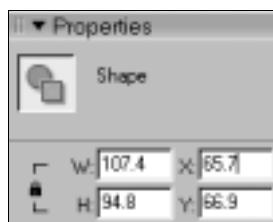


Рис. 4.17. Задание свойств объекта при помощи Инспектора свойств

Информация о способах преобразования объектов обобщена в табл. 4.2.

Таблица 4.2. Способы преобразования объектов

Операция/ Способ	Инструмент (ручное преобразование)	Галитра (точное преобразование)	Команда меню
Move (Перемещение)	1. Инструмент Selection (Выделение), <V> 2. Управляющие клавиши ("стрелки") на клавиатуре, с <Shift> шаг 10 пикселов 3. Инструмент Free Transform	Галитра Info , ввод координат X Y, относительно левого верхнего угла или центральной точки объекта	Палитра Выравнивания и расположения объектов Window>Design Panels>Align , <Ctrl>+<K>
Copy (Копирование)	1. Инструмент Selection (Выделение)+<Alt> или <Ctrl> 2. <Ctrl>+<D> 3. Через буфер обмена 4. Контекстное меню объекта	Галитра Transform , кнопка Copy	Edit>Copy , <Ctrl>+<C> Edit>Duplicate Edit>Paste in Center Вставить в центр Edit>Paste in Place
Scale (Масштабирование)	Инструмент Free Transform , модификатор Scale	Галитра Info , ввод W, H (ширины и высоты). Палитра Transform ввод W, H (%) по высоте и ширине, пропорциональное (Constrain) и непропорциональное масштабирование	Modify>Transform>Scale Палитра Выравнивания Window>Design Panels>Align , <Ctrl>+<K> Выровнять по размеру (Match Size)

Таблица 4.2 (окончание)

Операция/ Способ	Инструмент (ручное преобразование)	Панельра (точное преобразование)	Команда меню
Rotate (Поворот)	Инструмент Free Transform , монтификатор Rotate , перенос диагональных маркеров, с <Shift> на углы кратные 45	Панельра Transform , ввод угла поворота в градусах, по часовой стрелке – положительное число, против часовой – отрицательное	Modify>Transform>Rotate 90 CW (Поворот на 90 по часовой стрелке) Modify>Transform>Rotate 90 CCW (Поворот на 90 против часовой стрелки)
Skew (Наклон и Скос)	Инструмент Free Transform , монтификатор Rotate , перенос боковых и вертикальных маркеров	Панельра Transform , ввод угла скоса W, H по горизонтали и вертикали	Modify>Transform>Rotate and Skew
Distort (Искажение)	Инструмент Free Transform , монтификатор Distort	Нет	Modify>Transform>Distort
Envelope (Оболочка)	Инструмент Free Transform , монтификатор Envelope	Нет	Modify>Transform>Envelope
Flip (Зеркальное отображение)	Инструмент Free Transform , монтификатор Scale – перенос боковых и диагональных маркеров через объект	Нет	Modify>Transform>Flip Horizontal Modify>Transform>Flip Vertical
Delete (Удаление)	Выделить объект и нажать клавишу <Delete>	Нет	Edit>Clear, <Backspace>

Понятие о рабочем и наложенном уровнях

В рабочей среде Flash имеет место строгая иерархия графических объектов. Любой графический объект принадлежит либо к *рабочему уровню*, либо к *наложенному уровню*. Принадлежность к одному или к другому уровню определяет характерные особенности объекта и набор действий, которые могут быть с ним произведены. Разделение на рабочий и наложенный уровни носит несколько абстрактный характер, поскольку сами уровни не имеют визуального выражения (как, например, слои), а о принадлежности объекта к одному или другому уровню свидетельствуют косвенные признаки. Тем не менее, введение этих понятий является целесообразным, поскольку позволяет легче освоить технологию процесса создания графики в рабочей среде программы. Приведенные ниже сведения обобщены и проиллюстрированы в табл. 4.3.

Рабочий уровень

Характерными особенностями рабочего уровня являются:

- возможность изменения базовых свойств объектов (цвет составляющих их элементов, форма);
- взаимодействие объектов между собой.

Возможность изменения базовых свойств объекта позволяет выполнять следующие процедуры:

- выделять объекты целиком и по частям (обводку, заливку или части этих элементов);
- изменять цвет (и тип заливки) элементов векторного объекта;
- изменять форму объекта при помощи инструмента **Selection** или любых операций трансформации.

Взаимодействие объектов между собой выражается в том, что при наложении (пересечении) объекты определенным образом воздействуют друг на друга, что приводит к их изменению. Это взаимодействие может носить различный характер и включает в себя две группы операций.

- Объединение — векторные формы без контура (заливки) одного цвета при пересечении объединяются в одну векторную форму. Для объединения объектов необходимо выделить одну фигуру и перетащить ее поверх другой. После снятия выделения они объединятся в одну фигуру.
- Сегментирование — векторные формы разного цвета и контуры при пересечении сегментируются одна другой. Объект может либо делиться на части другой фигурой, либо при помощи контура от него может быть отрезана часть. В результате сегментирования образуются новые формы и/или контуры. Контуры также могут быть сегментированы векторными формами.



Примечание

Любой рисующий инструмент, рассмотренный выше, создает объекты рабочего уровня.

Наложенный уровень

Наложенный уровень характеризуется следующими особенностями:

- на нем отсутствуют возможности изменения базовых свойств объектов (цвет составляющих их элементов, форма);
- на нем отсутствуют взаимодействия между объектами;
- все объекты наложенного уровня выделяются прямоугольными рамками;
- объекты наложенного уровня всегда располагаются над объектами рабочего уровня (что и отражено в названии).

На наложенном уровне отсутствует непосредственная возможность редактирования формы и цвета элементов векторного объекта. Операции трансформации, доступные на наложенном уровне, также ограничены и включают в себя: масштабирование, вращение, скос, отражение. Как мы видим, на наложенном уровне нельзя выполнять искажение и редактирование при помощи огибающих, поскольку эти операции изменяют базовые свойства объекта.

Существует целый ряд объектов наложенного уровня. К их числу относятся: группа, текстовый блок, растровое изображение, видеоролик, экземпляр символа.

Объекты наложенного уровня размещаются (плавают) друг над другом в стопке, не взаимодействуя друг с другом.

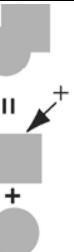
Группировка объектов

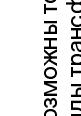
Группа представляет собой контейнер, который может содержать любые графические объекты (в том числе другие группы). Этот контейнер изолирован от других объектов и его содержимое извне не может подвергнуться каким-либо модификациям, связанным с изменением их базовых свойств.

Основные цели группирования объектов:

- изоляция объектов, запрет взаимодействия между объектами рабочего уровня после создания законченного рисунка;
- коллективное преобразование нескольких объектов, входящих в состав группы графических объектов (например, выравнивание сразу нескольких объектов);
- упорядочение сложных структур (вложение групп друг в друга).

Таблица 4.3. Иерархия графических объектов Flash

Тип объектов	Выделение объекта инструментом Selection	Редактирование объектов	Взаимодействие объектов	Трансформация объектов
Векторные формы и контуры, созданные инструментами рисования	 Выделен сегмент контура	 Выделен весь контур (2 щелчка по контуру)	 Объединение объектов одного цвета без контура	 Возможны все виды трансформации над выделенным объектом, его частью или несколькими объектами: перемещение, масштабирование, зеркальное отражение, поворот, скос, искажение, редактирование при помощи огибающих

	Выделены контур и заливка (2 щелчка по заливке)	 Изменение кривизны отрезка	
	Выделение рамкой	 Добавить вершину (потянуть с <Ctrl> за отрезок)	

Операция *группировки* используется для перемещения объекта рабочего уровня на наложенный уровень. Чтобы сгруппировать объект (или несколько объектов), можно действовать двумя способами. В соответствии с первым, необходимо выделить объект и выполнить команду меню **Modify>Group** ($<\text{Ctrl}>+<\text{G}>$). Полученная группа будет выделена прямоугольной рамкой.

Другой способ создания группы заключается в создании пустой группы (пустого контейнера) и наполнении ее изнутри. Для этого необходимо убедиться в том, что на сцене нет выделенных объектов, и выполнить все ту же команду **Modify>Group**. В результате этого будет создана новая группа и активизирован режим ее редактирования. Режим редактирования группы позволяет выполнять любые преобразования с ее содержимым, как бы находясь внутри этого контейнера. О том, что режим редактирования группы активен, свидетельствуют два факта.

- Все остальное содержимое сцены отображается с пониженной степенью насыщенности, указывая на то, что пользователь находится в изолированной среде.
- Над сценой на полосе редактирования появляются надпись **Group** (Группа) и цветная пиктограмма.

Для завершения создания группы необходимо ее наполнить. При этом можно использовать любые рисующие инструменты, как будто речь идет о работе на рабочем уровне. Чтобы выйти из режима редактирования группы, можно щелкнуть на надписи **Scene** (Сцена) на полосе редактирования либо использовать стрелку, находящуюся там же, чуть левее, или же просто два раза щелкнуть на любом участке сцены, свободном от объектов данной группы.

Создание группы из пустой группы применяется, например, в ситуации, когда необходимо нарисовать и разместить объект поверх уже имеющихся объектов рабочего уровня. Если рисовать его непосредственно сверху, то объекты провзаимодействуют. Использование пустой группы исключает эту возможность, в то же время позволяя видеть все объекты сцены.

Для того чтобы войти в режим редактирования имеющейся группы, необходимо либо два раза щелкнуть на ней, либо выполнить команду главного меню **Edit>Edit Selected** (Редактировать выделенное) или контекстного меню самой группы (щелкнув на ней правой кнопкой мыши).

При организации групп часто применяется иерархический подход, когда группы группируются друг с другом и с другими объектами рабочего и наложенного уровней. В результате образуются группы, содержащие другие группы, называемые *вложенными* (*nesting groups*). Так, чтобы сгруппировать две имеющиеся группы, нужно выделить их и применить команду **Modify>Group** ($<\text{Ctrl}>+<\text{G}>$). В результате будет получена группа более высокого уровня, содержащая в себе две другие группы. Количество уровней вложен-

ности групп не ограничено, т. е. группа может содержать любое число вложенных групп. Для того чтобы изменить содержимое вложенной группы, необходимо последовательно входить в режим редактирования групп более высокого уровня, при этом число пиктограмм с надписью **Group**, появляющихся на полосе редактирования, будет соответствовать текущему уровню вложенности.

Как было сказано выше, объекты наложенного уровня не взаимодействуют и размещаются в стопке друг над другом в порядке их появления. Также отмечалось, что объект рабочего уровня всегда будет располагаться *под* объектом наложенного уровня, даже если он создан позднее.

Монтаж объектов в стопке

Изменение порядка следования объектов наложенного уровня друг относительно друга носит название *монтажа в стопке*. Монтаж в стопке выполняется при помощи команд меню **Modify>Arrange**. Здесь имеются следующие команды:

- Bring to Front** (На передний план) — делает текущий объект самым верхним в стопке;
- Bring Forward** (На шаг вперед) — перемещает объект на один уровень вверх;
- Send Backward** (На шаг назад) — перемещает объект на один уровень назад;
- Send to Back** (На задний план) — делает текущий объект самым нижним в стопке.

Меню **Modify>Arrange** содержит также команду **Lock** (Блокировка), при помощи которой запрещается доступ к выделенному объекту (или объектам) наложенного уровня. Используйте эту команду в случае, когда любые действия с объектом являются нежелательными. Заблокированный объект не может быть выделен и, соответственно, с ним нельзя выполнять никакие действия. Для добавления к заблокированным объектам новых объектов нужно вновь повторить эту же процедуру. Заблокированный объект специально никак не помечается.

Снять блокировку можно только со всех объектов сразу при помощи команды **Unlock All** (Разблокировать все) меню **Modify>Arrange**.

Примечание

Меню **Arrange** также можно вызвать, щелкнув правой кнопкой мыши по объекту наложенного уровня.

Разгруппировать группу можно при помощи команды меню **Modify>Ungroup** (Разгруппировать) (<Ctrl>+<Shift>+<G>). В результате этой операции группа распадается на составляющие ее элементы. Разгруппирование также мо-

жет быть выполнено при помощи команды меню **Modify>Break Apart** (Разбиение) (<Ctrl>+).

Выравнивание и распределение объектов

В процессе организации и упорядочения графических объектов перед дизайнером часто стоит задача равномерного позиционирования объектов друг относительно друга и относительно границ изображения. Для того чтобы придать композиции уравновешенность и гармоничность, часто прибегают к выравниванию отдельных ее элементов относительно некой общей оси. Flash MX 2004 предлагает несколько возможностей выравнивания и распределения объектов друг относительно друга и относительно сцены. К их числу относятся:

- палитра **Align**;
- использование режима **Snap Align**;
- использование режима **Snap to Objects**;
- вспомогательные элементы — **Rulers** (Линейки), **Grid** (Сетка), **Pixel Grid** (Пиксельная сетка), **Guides** (Направляющие).

Палитра **Align**

Палитра **Align** (рис. 4.18) используется для выполнения целой группы различных операций:

- для выравнивания объектов;
- для распределения объектов;
- для приведения размеров объектов;
- для установления равных промежутков между объектами.

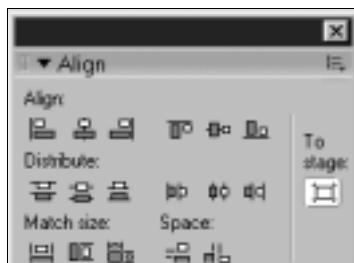


Рис. 4.18. Палитра **Align**

Для вызова палитры **Align** нужно выполнить команду **Window>Design Panels>Align** (<Ctrl>+<K>). Палитра включает в себя четыре набора кнопок, сгруппированных по выполняемым ими действиям и расположенных соответственно в разделах: **Align** (Выравнивание), **Distribute** (Распределение), **Match size** (Приведение размеров), **Space** (Установка равных промежутков). Кроме того, в правой части палитры расположена кнопка **To stage** (Относительно рабочей области), активизация которой приводит к тому, что все команды данной палитры разрешаются относительно рабочей области.

Для применения команд палитры **Align** к объектам их необходимо предварительно выделить.

Примечание

Применение некоторых команд палитры **Align** к объектам рабочего уровня может привести к их взаимному пересечению и вызвать взаимодействие их друг с другом.

Выравнивание объектов (Align)

Операции выравнивания позволяют разместить объекты таким образом, чтобы их грани или центры располагались на одной линии либо совпадали. Выравнивание включает шесть различных вариантов (рис. 4.19).

- **Align left edge** — выравнивание по левой грани базового объекта. В качестве базового объекта в данном случае рассматривается объект, левая грань которого находится левее всех остальных (рис. 4.19, а). Применение этой команды в режиме **To stage** приведет к выравниванию объектов относительно левой границы рабочей области. В результате левые грани объектов совместятся с левой границей рабочей области.
- **Align horizontal center** — выравнивание по среднегеометрическому горизонтальному центру (рис. 4.19, б). Применение этой команды в режиме **To stage** приведет к выравниванию объектов относительно горизонтального центра рабочей области. В результате горизонтальные центры объектов совместятся с горизонтальным центром рабочей области.
- **Align right edge** — выравнивание по правой грани базового объекта. В качестве базового объекта в данном случае рассматривается объект, правая грань которого находится правее всех остальных (рис. 4.19, в). Применение этой команды в режиме **To stage** приведет к выравниванию объектов относительно правой границы рабочей области. В результате правые грани объектов совместятся с правой границей рабочей области.
- **Align top edge** — выравнивание по верхней грани базового объекта. В качестве базового объекта в данном случае рассматривается объект, верхняя грань которого находится выше всех остальных (рис. 4.19, г). Примене-

ние этой команды в режиме **To stage** приведет к выравниванию объектов относительно верхней границы рабочей области. В результате верхние грани объектов совместятся с верхней границей рабочей области.

- Align vertical center** — выравнивание по среднегеометрическому вертикальному центру (рис. 4.19, *д*). Применение этой команды в режиме **To stage** приведет к выравниванию объектов относительно вертикального центра рабочей области. В результате вертикальные центры объектов совместятся с вертикальным центром рабочей области.
- Align bottom edge** — выравнивание по нижней грани базового объекта. В качестве базового объекта в данном случае рассматривается объект, нижняя грань которого находится ниже всех остальных (рис. 4.19, *е*). Применение этой команды в режиме **To stage** приведет к выравниванию объектов относительно нижней границы рабочей области. В результате нижние грани объектов совместятся с нижней границей рабочей области.

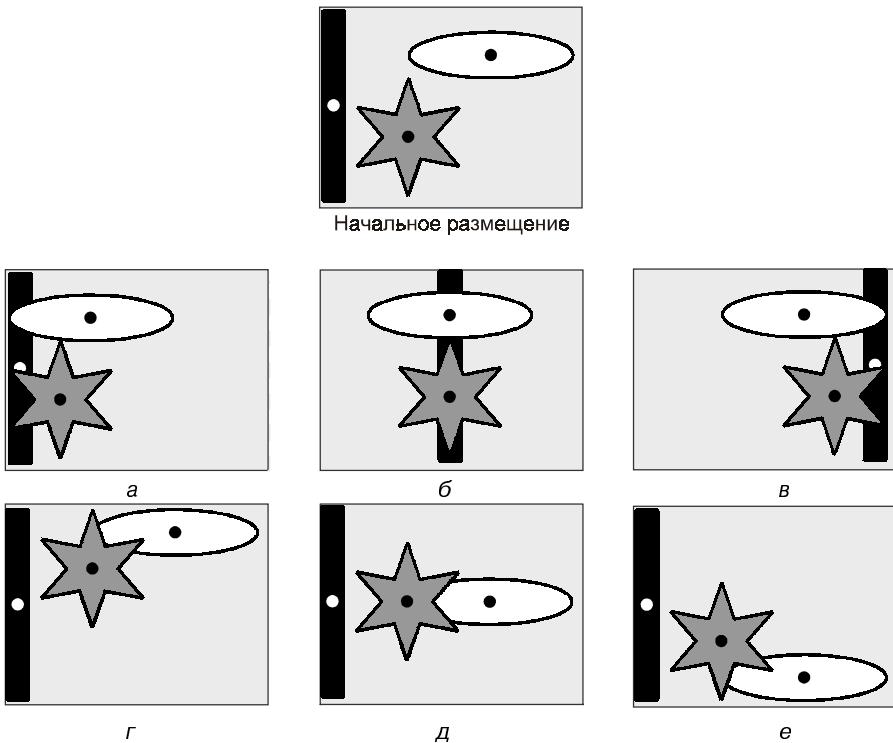


Рис. 4.19. Операции выравнивания

Распределение объектов (Distribute)

При распределении объекты размещаются таким образом, чтобы расстояния между их гранями или центрами были равны. Как и выравнивание, распределение включает шесть различных вариантов (рис. 4.20).

- **Distribute top edge** — установка равных расстояний между верхними гранями объектов. В качестве базового объекта в данном случае рассматривается объект, верхняя грань которого находится выше всех остальных (рис. 4.20, а). Применение этой команды в режиме **To stage** приведет к распределению объектов по вертикали на протяжении всей рабочей области. В результате верхняя грань самого верхнего объекта совместится с верхней границей рабочей области, а нижняя грань самого нижнего объекта — с нижней границей рабочей области.
- **Distribute vertical center** — установка равных расстояний между вертикальными центрами объектов. Самый верхний и самый нижний объекты остаются на том же месте (рис. 4.20, б). Применение этой команды в режиме **To stage** приведет к распределению объектов по вертикали на протяжении всей рабочей области. В результате верхняя грань самого верхнего объекта совместится с верхней границей рабочей области, а нижняя грань самого нижнего объекта — с нижней границей рабочей области.
- **Distribute bottom edge** — установка равных расстояний между нижними гранями объектов. В качестве базового объекта в данном случае рассматривается объект, нижняя грань которого находится ниже всех остальных (рис. 4.20, в). Применение этой команды в режиме **To stage** приведет к распределению объектов по вертикали на протяжении всей рабочей области. В результате верхняя грань самого верхнего объекта совместится с верхней границей рабочей области, а нижняя грань самого нижнего объекта — с нижней границей рабочей области.
- **Distribute left edge** — установка равных расстояний между левыми гранями объектов. В качестве базового объекта в данном случае рассматривается объект, левая грань которого находится левее всех остальных (рис. 4.20, г). Применение этой команды в режиме **To stage** приведет к распределению объектов по горизонтали на протяжении всей рабочей области. В результате верхняя грань самого верхнего объекта совместится с верхней границей рабочей области, а нижняя грань самого нижнего объекта — с нижней границей рабочей области.
- **Distribute horizontal center** — установка равных расстояний между горизонтальными центрами объектов. Самый правый и самый левый объекты остаются на том же месте (рис. 4.20, д). Применение этой команды в режиме **To stage** приведет к распределению объектов по горизонтали на протяжении всей рабочей области. В результате правая грань самого правого объекта совместится с правой границей рабочей области, а левая грань самого левого объекта — с левой границей рабочей области.

- **Distribute right edge** — установка равных расстояний между правыми гранями объектов. В качестве базового объекта в данном случае рассматривается объект, правая грань которого находится правее всех остальных (рис. 4.20, *e*). Применение этой команды в режиме **To stage** приведет к распределению объектов по горизонтали на протяжении всей рабочей области. В результате правая грань самого правого объекта совместится с правой границей рабочей области, а левая грань самого левого объекта — с левой границей рабочей области.

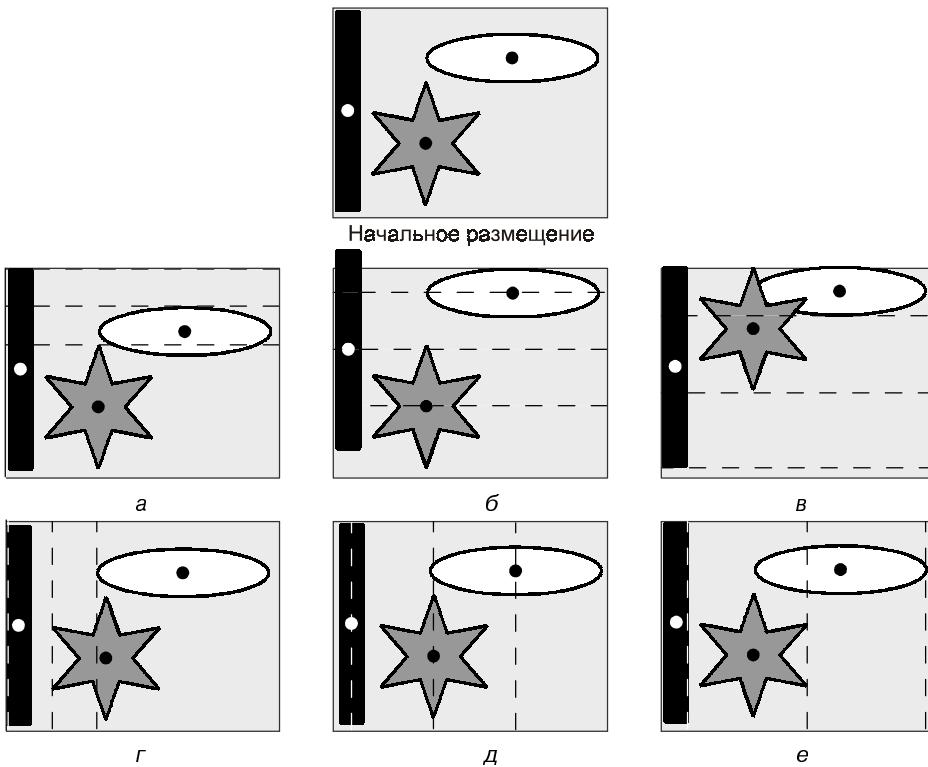


Рис. 4.20. Операции распределения

Приведение размеров (Match size)

Эта операция позволяет привести размеры объектов к одной величине. Данная процедура включает три варианта (рис. 4.21).

- **Match width** — приведение размеров по вертикали к максимальному среди объектов вертикальному размеру. В результате высоты всех объектов устанавливаются равными наибольшей высоте (рис. 4.21, *a*). Применение

данной команды в режиме **To Stage** приводит к установлению вертикальных размеров объектов равными высоте рабочей области.

- Match height** — приведение размеров по горизонтали к максимальному среди объектов горизонтальному размеру. В результате значения ширины всех объектов устанавливаются равными максимальной ширине (рис. 4.21, *б*). Применение данной команды в режиме **To Stage** приводит к установлению горизонтальных размеров объектов равными ширине рабочей области.
- Match width and height** — одновременное выполнение команд **Match width** и **Match height** (рис. 4.21, *в*).

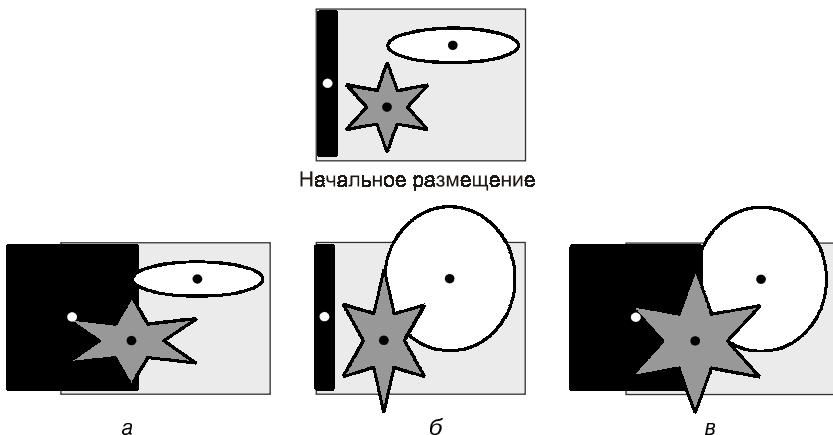


Рис. 4.21. Приведение размеров

Установление равных промежутков между объектами (Space)

Эта команда позволяет разместить объекты таким образом, что промежутки между ними будут равны. Имеется два варианта данной операции.

- Space evenly vertically** — установка равных промежутков по вертикали. Применение данной команды в режиме **To Stage** приведет к тому, что верхняя грань самого верхнего объекта совместится с верхней границей рабочей области, а нижняя грань самого нижнего объекта — с нижней границей рабочей области. При этом вертикальные промежутки между объектами будут иметь одинаковый размер.
- Space evenly horizontally** — установка равных промежутков по горизонтали. Применение данной команды в режиме **To Stage** приведет к тому, что правая грань самого правого объекта совместится с правой границей рабочей области, а левая грань самого левого объекта — с левой границей рабочей области. При этом горизонтальные промежутки между объектами будут иметь одинаковый размер.

Примечание

Для того чтобы использовать произвольный прямоугольный объект в качестве фона, можно выполнить следующие действия: привести его размеры к размерам рабочей области (режим **To stage**, команда **Match width and height**), отцентрировать объект относительно рабочей области (режим **To stage**, команды **Align horizontal center+Align vertical center**).

Использование режима **Snap Align**

Режим **Snap Align** (Притягивание с выравниванием) используется при ручном позиционировании объектов. Притягивание с выравниванием позволяет точно выравнивать объекты относительно их граней, центров, а также границ рабочей области.

Для активизации режима **Snap Align** нужно выполнить команду **View>Snapping>Snap Align**. Во время перемещения объектов по сцене в этом режиме при заданном приближении к граням или центрам объектов, а также к границам рабочей области они притягиваются к появляющимся пунктирным направляющим линиям (рис. 4.22).

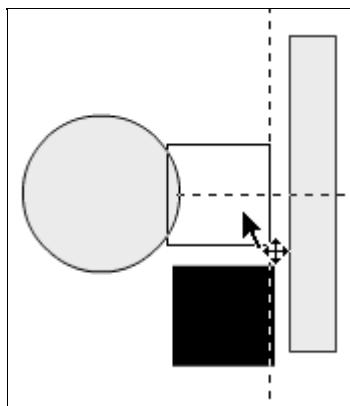


Рис. 4.22. Перемещение объекта в режиме **Snap Align**

Для задания параметров работы этого режима используется команда **View>Snapping>Edit Snap Align**, в результате выполнения которой появляется диалоговое окно **Snap Align**, содержащее следующие настройки:

- **Movie border** (Поля фильма) — расстояние между гранями воображаемой ограничивающей рамки объекта и границами рабочей области, на котором выполняется притягивание;

- Horizontal** (Горизонтальный допуск) — расстояние между вертикальными гранями воображаемых ограничивающих рамок объектов, на котором выполняется притягивание;
- Vertical** (Вертикальный допуск) — расстояние между горизонтальными гранями воображаемых ограничивающих рамок объектов, на котором выполняется притягивание;
- Show horizontal guides** (Включить горизонтальные направляющие) — включение горизонтальных пунктирных направляющих, проходящих через горизонтальные центры объектов;
- Show vertical guides** (Включить вертикальные направляющие) — включение вертикальных пунктирных направляющих, проходящих через вертикальные центры объектов.

Использование режима *Snap to Objects*

Режим **Snap to Objects** (Притягивание к объектам), так же как и **Snap Align**, используется при ручном позиционировании объектов. Притягивание к объектам позволяет без зазоров замыкать контуры, выравнивать объекты относительно границ их форм, контуров и центров (рис. 4.23). Особенностью данного режима является возможность использования притягивания в момент рисования.

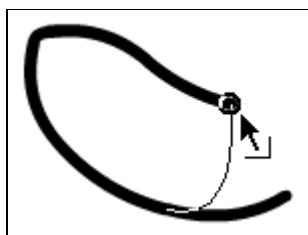


Рис. 4.23. Замыкание контура
в режиме **Snap to Objects**

Для активизации режима **Snap to Objects** нужно выполнить команду **View>Snapping>Snap to Objects**. Некоторые рисующие инструменты содержат модификатор **Snap**, также позволяющий активизировать данный режим. Во время рисования объектов или их перемещения по сцене в этом режиме при определенном приближении к границам или центрам объектов происходит притягивание. При этом рядом с курсором появляется изображение кружка, увеличивающегося в размерах в момент притягивания. Для того чтобы режим притягивания к объектам работал при их перемещении, необ-

ходимо, чтобы курсор располагался достаточно близко к границе формы или находился в геометрическом центре объекта.

Примечание

Использование режима притягивания к объектам бывает весьма полезно при необходимости замыкать контур или соединять сегменты контура непосредственно в момент рисования или впоследствии при помощи инструмента **Selection**.

Вспомогательные элементы: **Rulers, Grid, Guides**

Применение вспомогательных элементов бывает очень полезно в тех случаях, когда возникает необходимость упорядочивания размещения объектов вручную. При помощи вспомогательных элементов можно сформировать макет композиции, позволяющий точно позиционировать все ее составляющие. При этом сами вспомогательные элементы в конечном фильме видны не будут. Они присутствуют только в рабочей среде.

Линейки (Rulers)

Для включения линеек необходимо выполнить команду меню **View>Rulers**. Линейки расположены вдоль левой и верхней граней сцены. Нулевая точка линеек совпадает с началом координат фильма (левый верхний угол рабочей области). Линейки проградуированы в текущих единицах измерения, которые задаются при настройках глобальных параметров фильма в меню **Modify>Document** (см. разд. "Настройка глобальных параметров фильма" гл. 3).

Сетка (Grid)

Сетка — это вспомогательный элемент, который может быть использован для выравнивания и равномерного распределения объектов. У сетки имеется режим **Snap to Grid** (Притягивание к сетке), позволяющий организовать автоматическое притягивание объектов к ее узлам при рисовании и перемещении. Для включения или выключения отображения сетки применяется команда меню **View>Grid>Show Grid**.

Для настройки сетки служит команда меню **View>Grid>Edit Grid**. Здесь можно задать следующие параметры:

- Color** (Цвет) — установка цвета сетки. В качестве цвета сетки может быть выбран любой цвет из текущего каталога цветов;
- Show grid** (Отображать сетку) — включение отображения сетки;
- Snap to grid** (Притягивание к сетке) — активизация режима притягивания к сетке;

- Шаг по горизонтали** — размер ячейки сетки по горизонтали в пикселях;
- Шаг по вертикали** — размер ячейки сетки по вертикали в пикселях;
- Snap accuracy** (Чувствительность к притягиванию) — при помощи этого раскрывающегося списка можно указать степень чувствительности сетки, определяемой допуском на расстояние от объекта до узла сетки. Список содержит следующие опции:
 - **Must be close** (Близко) — притягивание к узлу сетки происходит только в случае, если объект находится достаточно близко к нему;
 - **Normal** (Нормально) — разрешается допуск больший, чем при значении **Must be close**, но меньший, чем при **Can be distant**;
 - **Can be distant** (Отдаленно) — притягивание к узлу сетки выполняется даже при значительном удалении объекта от узла сетки;
 - **Always Snap** (Всегда притягивать) — притягивание выполняется всегда вне зависимости от расстояния до узла сетки.

Кнопка позволяет сохранить текущие настройки сетки в качестве настроек по умолчанию. Так, если отображение сетки должно быть включено по умолчанию всякий раз при создании нового документа, необходимо установить флажок **Snap to grid** и нажать кнопку **Save Default**.

Для того чтобы режим притягивания к сетке работал при перемещении объектов, необходимо, чтобы курсор располагался достаточно близко к границе формы или находился в геометрическом центре объекта.

Примечание

При включенном режиме **Snap to Grid** притягивание к сетке выполняется, даже если режим отображения сетки отключен. Несмотря на то, что за пределами рабочей области сетка не видна, притягивание к ней продолжает выполняться.

Пиксельная сетка (Pixel Grid)

Пиксельная сетка — это сетка с размером ячеек 1×1 пикселей. С использованием пиксельной сетки можно выполнять позиционирование объектов с точностью до 1 пикселя. Для включения режима притягивания к пиксельной сетке необходимо выполнить команду меню **View>Snapping>Snap to Pixels**. Пиксельная сетка становится видна только при масштабе просмотра не менее 400%.

Направляющие (Guides)

Направляющие используются для выравнивания объектов вдоль горизонтальной или вертикальной оси. Направляющие, как и сетка, имеют режим **Snap to guides** (Притягивание к направляющим), позволяющий организовать

автоматическое притягивание объектов при рисовании и перемещении к их осям. Кроме этого, направляющие имеют режим **Lock guides** (Фиксирование направляющих), при котором направляющие не могут быть смещены и остаются заблокированными. Это позволяет избежать случайного смещения направляющей во время работы с объектами.

Для того чтобы поместить горизонтальную или вертикальную направляющую на сцену, необходимо щелкнуть на соответствующей линейке (Rulers) и, не отпуская курсора мыши, прямо из нее вытащить направляющую. Последовательно выполняя эти действия, можно разместить на сцене столько направляющих, сколько необходимо. Чтобы удалить направляющую, необходимо перенести ее за пределы сцены.

Для включения или отключения отображения направляющих используется команда меню **View>Guides>Show Guides**.

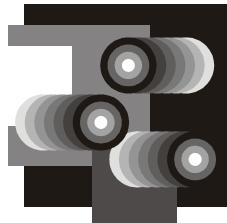
Чтобы заблокировать направляющую, можно использовать команду меню **View>Guides>Lock Guides**. Для настройки направляющих используется команда меню **View>Guides>Edit Guides**, выполнение которой приводит к появлению диалогового окна **Guides** (Направляющие), при помощи которого можно задать следующие параметры:

- Color** (Цвет) — установка цвета направляющих. В качестве цвета направляющих может быть выбран любой цвет из текущего каталога цветов;
- Show guides** (Отображать направляющие) — включение/выключение отображения;
- Snap to guides** (Притягивание к направляющим) — активизация режима притягивания к направляющим;
- Lock guides** (Задерживать направляющие) — фиксация направляющих;
- Snap accuracy** (Чувствительность к притягиванию) — степень чувствительности направляющих — параметр, аналогичный соответствующему параметру для сетки (см. выше).

Кнопка **Clear All** позволяет сразу удалить со сцены все направляющие, а кнопка **Save Default** — сохранить текущие настройки направляющих в качестве настроек по умолчанию.

Примечание

Активизировать вспомогательные инструменты или включить необходимый режим притягивания можно при помощи команд контекстного меню сцены, появляющегося при щелчке правой кнопкой мыши на ее свободном участке.



Глава 5

Работа с контурами

От кривой ветви — кривая тень.

Японская пословица

Понятие о кривых

Любое векторное изображение состоит из совокупности контуров, которые, в свою очередь, состоят из частей кривых, описываемых полиномиальными уравнениями разного порядка. Наиболее широкое распространение в программах векторной графики получили кривые Безье — кривые, описываемые уравнениями различного порядка. Они получили свое название в честь французского математика Пьера Безье, который в 60—70-х годах XX века работал на автомобильную компанию "Рено" и вместе с группой разработчиков занимался созданием метода представления пространственных кривых и разработкой соответствующего программного обеспечения. В их задачу входила разработка метода управления высокоточными режущими машинами и модернизация процесса производства машин. Безье разработал систему создания кривых, получившую его имя.

Пространственное размещение кривой Безье однозначно задается положением четырех точек. Две из них принадлежат самой кривой — *опорные точки* (anchor points), а другие две — *управляющие точки* (control points) — размещаются на концах отрезков касательных к кривой, проходящих через опорные точки. Отрезки касательных носят название *управляющих линий* (control handles) (рис. 5.1).

Опорные точки располагаются в начале и в конце сегмента кривой и называются соответственно начальной и конечной опорной точками. Опорные точки бывают *гладкими* (smooth) и *угловыми* (corner). Гладкой опорной точкой называется точка, лежащая в месте гладкого сопряжения двух криволинейных сегментов. Угловой точкой является точка, расположенная в вершине угла, представляющего собой соединение двух прямолинейных сегментов, либо точка, отрезки управляющих линий которой расположены не на одной прямой. В первом случае угловая точка вообще не содержит управляющих линий (рис. 5.2).

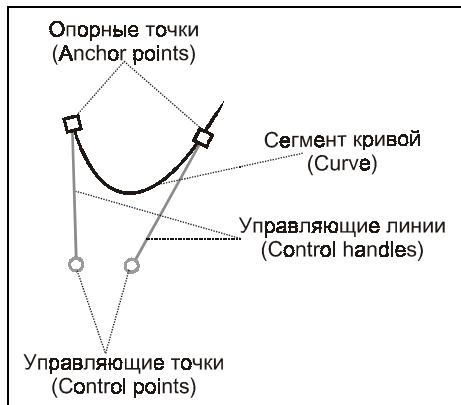


Рис. 5.1. Элементы кривой Безье

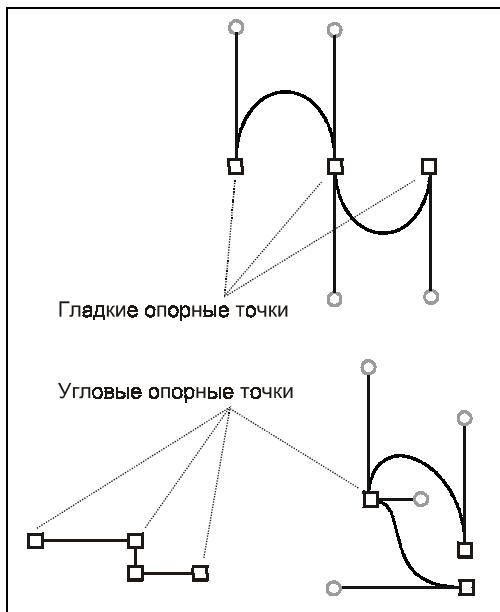


Рис. 5.2. Виды опорных точек

Положение и характер сегмента кривой могут изменяться как при воздействии на опорные точки, так и при перемещении управляемых точек. Управляющие линии сами по себе не выполняют никакой функции, кроме указания того, с какой опорной точкой соединена данная управляемая точка.

Перемещение опорной точки приводит к изменению амплитуды сегмента, влияя на расстояние между опорными точками и направлением распростра-

нения сегмента (горизонтальное, вертикальное и т. д.). Перемещение управляющих точек приводит к изменению характера кривизны сегмента. Смещение управляющей точки в большей степени влияет на ту половину сегмента кривой, которая прилегает к данной управляющей линии; таким образом, управляя взаимным расположением управляющих точек можно получать выпуклые или S-образные сегменты.

Инструменты *Pen* и *Subselection*

В Flash имеются два инструмента для профессиональной работы с контурами на уровне опорных точек. Они позволяют с высокой точностью создавать и редактировать контуры любой сложности.

Инструмент *Pen*

Инструмент **Pen** (Перо)  служит для создания и выполнения некоторых операций редактирования контуров. Все настройки работы этого инструмента задаются в меню **Edit>Preferences** на закладке **Editing** в разделе **Pen tool** (см. разд. "Настройки программы" гл. 3). Атрибуты контура, создаваемого пером, задаются при помощи Инспектора свойств, аналогично другим инструментам создания контуров (обводок). В процессе использования инструмента **Pen** для отображения создаваемого контура с расположенными на нем опорными точками используется цвет слоя.

Контуры, создаваемые инструментом, **Pen** можно разделить на контуры, состоящие из прямолинейных сегментов, контуры, состоящие из криволинейных сегментов, и комбинированные контуры. Данная классификация объясняется различиями в способе создания и редактирования сегментов этих контуров.

Создание контуров, состоящих из прямолинейных сегментов

Для создания контура, состоящего из прямолинейных сегментов, достаточно активизировать инструмент **Pen** и просто расставлять опорные точки, соответствующие вершинам, щелкая в области сцены. При этом соседние опорные точки будут автоматически соединены прямолинейными отрезками. В результате получится контур, состоящий исключительно из угловых опорных точек. Сегменты такого контура не содержат управляющих точек.

Для замыкания контура необходимо подвести курсор к начальной опорной точке (при этом рядом с его пиктограммой появится маленький кружок ) и щелкнуть в ней. Если в качестве цвета заливки установлен соответствую-

щий оттенок, контур автоматически зальется при замыкании. Для того чтобы точнее замкнуть контур, удобно включить режим притягивания к объектам **View>Snapping>Snap to Objects**.

Чтобы прекратить рисование, не замыкая контур, нужно выполнить одно из следующих действий:

- два раза щелкнуть инструментом **Pen** в области сцены. При этом будет создана новая опорная точка, после чего рисование прекратится;
- один раз щелкнуть в области сцены, удерживая при этом клавишу <Ctrl>. Рисование прекратится без установки новой опорной точки;
- выбрать любой другой инструмент на панели инструментов.

Если инструмент **Pen** неактивен, т. е. создание контура либо еще не начиналось, либо уже закончено, рядом с его пиктограммой появляется крестик .

Для того чтобы добавить *новый прямолинейный* сегмент к уже существующему контуру, нужно подвести перо к его конечной опорной точке, щелкнуть в ней, после чего щелкнуть в области сцены, установив новую опорную точку. Обе точки будут автоматически соединены прямой линией.

Создание контуров, состоящих из криволинейных сегментов

Процесс создания криволинейного контура представлен на рис. 5.3. Для создания криволинейного сегмента контура необходимо щелкнуть инструментом **Pen** в области сцены, установив первую опорную точку, и, не отпуская кнопки мыши, протянуть курсор в направлении распространения сегмента. При этом из опорной точки будут "вытащены" управляющие линии, которыми можно манипулировать при помощи управляющей точки, а курсор примет вид белой стрелки  (рис. 5.3, *а*). Далее необходимо отпустить кнопку мыши и установить новую опорную точку. Криволинейный сегмент будет нарисован (рис. 5.3, *б*). Для создания нескольких таких сегментов необходимо после установки новой опорной точки, не отпуская кнопки мыши, "вытаскивать" из нее управляющие линии, создавая, таким образом, новые криволинейные сегменты контура (рис. 5.3, *в*).

Примечание

Для того чтобы сделать процесс рисования контура более контролируемым и наглядным, можно включить режим **View>Preferences>Show pen preview** вкладки **Editing**. Это позволит увидеть новый сегмент контура еще до установки его конечной опорной точки.

Если в момент рисования удерживать клавишу <Shift>, то управляющие линии будут располагаться с дискретностью в 45 градусов, т. е. строго перпен-

дикулярно, горизонтально или диагонально. Это позволяет получить контур более правильной формы (рис. 5.4).

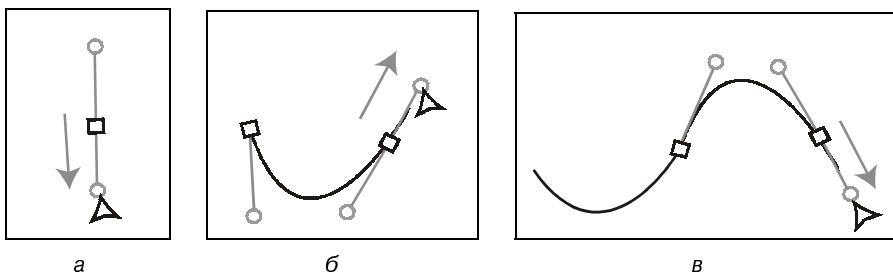


Рис. 5.3. Создание контура, состоящего из криволинейных сегментов

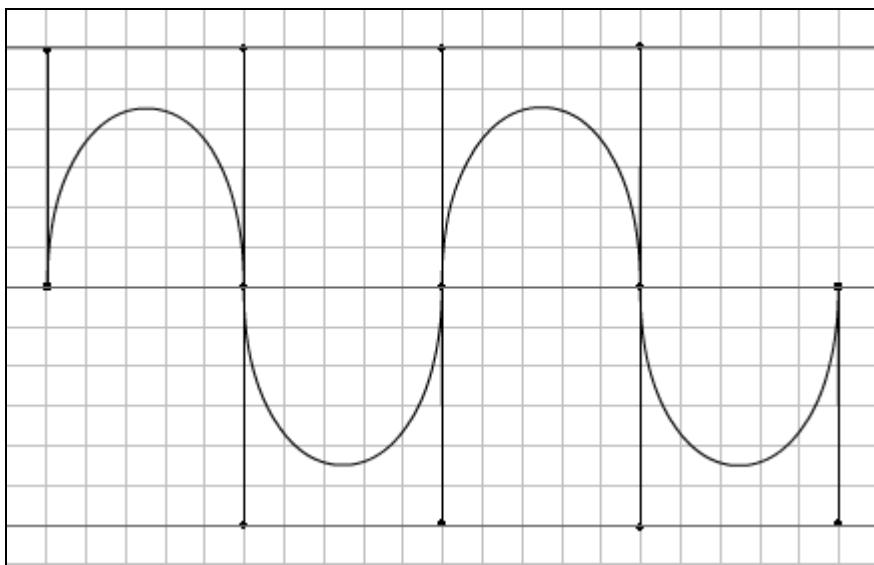


Рис. 5.4. Построение правильной синусоидальной кривой

В результате описанных действий можно получить криволинейные сегменты двух типов — гладкие (параболические) и S-образные, имеющие точку перегиба (рис. 5.5).

Для создания *гладкого* сегмента необходимо выполнить следующие действия.

Вначале нужно установить начальную опорную точку и, не отпуская кнопки мыши, протянуть курсор в направлении выпуклости, вытягивая из точки управляющие линии. После этого надо отпустить кнопку мыши, подвести курсор к месту расположения конечной опорной точки, и, установив ее, не

отпуская кнопки мыши, протянуть курсор в *диаметрально противоположном* направлении (рис. 5.5, а). Повторно применяя эту последовательность действий можно нарисовать контур, напоминающий синусоиду.

Для создания *S-образного* сегмента необходимо выполнить следующие действия.

Вначале надо установить начальную опорную точку и, не отпуская кнопки мыши, протянуть курсор в направлении выпуклости, вытягивая управляющие линии. После этого нужно отпустить кнопку мыши, подвести курсор к месту расположения конечной опорной точки, и, установив ее, не отпуская кнопки мыши, протянуть курсор *в том же самом* направлении (рис. 5.5, б).

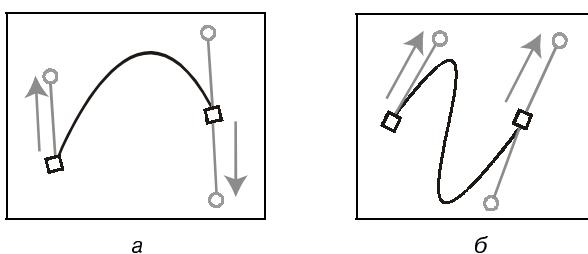


Рис. 5.5. Создание гладкого и S-образного сегментов

Для того чтобы добавить *новый криволинейный* сегмент к уже существующему контуру, нужно подвести перо к его конечной опорной точке, щелкнуть в ней, и, не отпуская кнопки мыши, протянуть курсор в направлении распространения следующего сегмента, после чего отпустить кнопку мыши и щелкнуть в области сцены, установив новую опорную точку.

Редактирование контуров при помощи инструмента *Pen*

При помощи инструмента **Pen** (Перо) можно удалять опорные точки из контура, добавлять опорные точки, трансформировать гладкие опорные точки в угловые. Рассмотрим эти операции.

- **Удаление.** Чтобы удалить угловую опорную точку из контура, необходимо щелкнуть в ней инструментом **Pen**. При подведении курсора к угловой точке рядом с ним появляется знак минуса . Чтобы удалить гладкую опорную точку из контура, необходимо дважды щелкнуть в ней инструментом **Pen**.
- **Добавление.** Для того чтобы добавить новую опорную точку в контур, необходимо щелкнуть на соответствующем участке. При подведении курсора к криволинейному сегменту рядом с ним появляется знак плюса .

Новую опорную точку при помощи инструмента Перо можно добавить только на криволинейный сегмент. Для того чтобы добавить новую опорную точку на прямолинейный сегмент, необходимо использовать инструмент **Selection** (см. разд. "Редактирование формы" гл. 4).

- **Преобразование гладкой опорной точки в угловую.** Для выполнения этой операции необходимо подвести курсор к гладкой точке и щелкнуть в ней один раз. При подведении курсора к гладкой опорной точке рядом с ним появляется пиктограмма угла

Инструмент **Subselection**

Инструмент **Subselection** (Частичное выделение) используется для выделения и редактирования опорных точек и сегментов контура. Его можно быстро активизировать непосредственно при работе с пером, нажав клавишу <Ctrl>. При помощи данного инструмента можно производить ряд различных операций. Рассмотрим их.

- **Перемещение всего контура.** Для выполнения этой процедуры необходимо подвести инструмент Частичное выделение к любому участку контура, не содержащему опорных точек (при этом рядом с курсором появится черный квадрат) , и, взявшись за него, переместить контур.
- **Выделение опорных точек контура.** Для того чтобы выделить опорную точку, необходимо подвести к ней курсор (при этом он принимает вид) и щелкнуть инструментом Частичное выделение. По умолчанию невыделенная опорная точка отображается пустым кружком, а выделенная — заполненным. Для изменения этого режима можно использовать пункт **Show solid points** вкладки **Editing** меню **Edit>Preferences** (см. разд. "Настройки программы" гл. 3). Для выделения отдельной точки нужно просто щелкнуть в ней. Для выделения нескольких точек можно либо заключить их инструментом Частичное выделение в прямоугольную область выделения, щелкнув в области сцены и протянув курсор, либо, используя клавишу <Shift>, щелчками последовательно выделять точки.
- **Редактирование контура** включает несколько операций.
- **Удаление опорных точек.** Чтобы удалить точку, ее необходимо выделить и затем либо нажать клавишу <Delete>, либо выполнить команду **Edit>Clear**.
 - **Перемещение опорных точек.** Чтобы переместить одну точку, можно взяться за нее инструментом Частичное выделение и перенести в другое место. Чтобы переместить сразу несколько точек (не обязательно расположенных друг возле друга), необходимо выделить их и затем воспользоваться управляемыми клавишами <Up>, <Down>,

<Left>, <Right>. Если при этом воспользоваться клавишей <Shift>, смещение будет выполняться с большей дискретностью. Перемещение опорных точек приводит как бы к "растеканию" контура.

- *Перемещение управляющих точек.* Для того чтобы переместить управляющую точку, необходимо выделить соответствующую ей опорную точку. Перемещение управляющей точки вдоль направления управляющей линии приводит к изменению глубины сегмента кривой, а вращение управляющей линии изменяет характер кривизны сегмента. По умолчанию угол между отрезками управляющих линий всегда составляет 180 градусов. При повороте одного отрезка другой смещается на тот же угол. Для того чтобы редактировать только один отрезок управляющей линии, необходимо удерживать клавишу <Alt>. Это приводит к появлению угловой точки в месте сопряжения двух криволинейных сегментов. Такая точка называется *криволинейной угловой точкой*.
- *Преобразование угловой точки в гладкую.* Для выполнения этого преобразования необходимо выделить угловую точку и "вытянуть" из нее управляющие линии, удерживая клавишу <Alt>.

Оптимизация кривых

Объем файла, содержащего векторное изображение, в большой степени определяется количеством опорных точек. Чем сложнее очертания форм, тем больше опорных точек требуется для описания контура и тем, соответственно, больше конечный объем файла. Кроме того, большое количество опорных точек сильно затрудняет процесс редактирования контура.

При работе с контурами часто возникает задача оптимизации кривых. Flash располагает несколькими возможностями, позволяющими автоматически оптимизировать контуры и векторные формы. Рассмотрим их.

- *Автоматическое распознавание форм.* При работе с инструментом **Pencil** существует режим, позволяющий распознавать форму контура или сглаживать его (см. гл. 4, разд. "Инструмент Pencil").
- *Автоматическое сглаживание мазков кисти.* При работе с инструментом Кисть можно задать степень сглаживания результирующих векторных форм (см. гл. 4, разд. "Инструмент Brush").
- *Использование команд Smooth (Сглаживание) Straighten (Спрямление).* Для выполнения этих команд можно воспользоваться меню **Modify>Shape** или употребить одноименные модификаторы инструмента **Selection**. Команда **Smooth** сглаживает сегменты контура, упрощая его форму. В результате мелкие детали и подробности контура могут пропадать. Команда

Straighten делает формы более правильными, спрямляя криволинейные участки. Применение команд сглаживания и спрямления к мазкам, созданным инструментом Кисть при помощи графического планшета, чувствительного к нажиму, позволит получить красивые стилизованные формы (рис. 5.6).



Исходный рисунок

Применение сглаживания
(Smooth)Применение спрямления
(Straighten)Рис. 5.6. Использование команд **Smooth** и **Straighten**

Данные команды могут быть применены как к контурам, так и к векторным формам. Для того чтобы воспользоваться ими, объект необходимо предварительно выделить. Команды обладают *кумулятивным эффектом*, т. е. их последовательное выполнение усиливает эффект. Однако оптимизация не может выполняться до бесконечности, поэтому на определенном этапе дальнейшее применение сглаживания или спрямления не будет иметь никакого эффекта.

- **Автоматическая оптимизация кривых.** Выполнение команды меню **Modify>Shape>Optimize** приводит к появлению окна **Optimize Curves** (Оптимизация кривых), позволяющего задать следующие параметры оптимизации:

- **Smoothing** (Сглаживание) — степень сглаживания при оптимизации (**None** — сглаживание отключено, **Maximum** — максимальное сглаживание);
- **Use multiple passes** (Полная оптимизация) — оптимизация сразу будет проведена в максимальной степени до тех пор, пока дальнейшее упрощение формы станет невозможным. Эта опция может привести к неожиданным результатам, поскольку первоначальная форма может сильно измениться.
- **Show totals message** (Вывести отчет о результатах оптимизации) — по окончании оптимизации будет выведен отчет, содержащий сведения о количестве кривых до и после оптимизации. Если в отчете указывается нулевой результат (0% reduction), это означает, что контур уже больше не может быть оптимизирован.

Примечание

При работе с кривыми можно заметить, что Flash самостоятельно добавляет новые опорные точки. Это связано с тем, что в Flash все кривые реализуются при помощи уравнений второго порядка. Это сделано для того, чтобы увеличить скорость их обработки проигрывателем Flash. Для удобства работы в среде редактирования кривые отображаются в представлении кривых третьего порядка, содержащих две управляемые точки, однако по завершении сеанса работы информация о размещении опорных точек таких кривых теряется. При новом обращении к данному контуру его сегменты вновь приводятся к виду кривых Безье третьего порядка, однако число точек при этом увеличивается. Дополнительные точки не влияют на размер файла, но существенно затрудняют редактирование контура. Для того чтобы избежать добавления точек, следует воздержаться от перемещения контура инструментом **Selection**. При копировании и дублировании объекта в копии контура также появятся новые точки. Для того чтобы без изменений перенести контур в другой кадр, необходимо скопировать весь кадр.

Дополнительные возможности работы с векторными формами

Меню **Modify>Shape** содержит еще несколько команд, позволяющих выполнять различные манипуляции с контурами и векторными формами.

□ **Convert Lines to Fills** — эта команда преобразует выделенный контур в векторную форму, при этом никаких визуальных изменений объекта не происходит. Как известно, контур может быть окрашен только сплошным (solid) цветом. Если все-таки существует объективная необходимость закрасить контур градиентом или даже растром, то для этого его необходимо предварительно превратить в форму. После выполнения данной команды контур, внешне никак не изменяясь, превращается в векторную форму (заливку), что позволяет применять к нему все операции, используемые при работе с заливками. Трансформация контура в векторную форму исключает дальнейшую возможность изменения таких атрибутов, как толщина и стиль. Зато появляется возможность использования любого типа заливки и применения всех инструментов для работы с формами.

□ **Expand Fill** — эта команда позволяет растянуть или сжать векторную форму, соответственно увеличивая или уменьшая ее. Перед применением команды форма должна быть выделена. Появившееся окно **Expand Fill** позволяет задать следующие параметры:

- **Distance** — расстояние в текущих единицах измерения, на которое увеличится или уменьшится форма. Если в качестве единиц измерения выбраны пиксели, то диапазон задаваемых значений может находиться в интервале от 0,05 до 144 пикселов;
- **Direction** — направление действия команды. Значение **Expand** (Наружу) приводит к увеличению площади формы, а значение **Inset** (Внутрь) к ее уменьшению.

Данная команда применяется только к векторным формам. Попытка ее применения к контуру приведет к его исчезновению. При задании достаточно большого значения параметра **Distance** можно получить нежелательные эффекты (потеря детализации сложной границы формы, скругление углов прямоугольника и т. д.).

□ **Soften Fill Edges** — команда, позволяющая реализовать смягчение границы формы, вызвав своеобразный эффект размытия или растушевки. Смягчение краев формы выполняется за счет размещения вдоль ее границ концентрических векторных форм, окрашенных одним и тем же цветом, но с постепенным усилением прозрачности (рис. 5.7). При выполнении данной команды в появляющемся окне **Soften Fill Edges** можно задать следующие параметры:

- **Distance** (Расстояние) — расстояние, на протяжении которого выполняется растушевка;

- **Number of steps** (Количество шагов) — количество концентрических векторных форм, создаваемых для имитации растушевки;
- **Direction** (Направление) — направление растушевки **Expand** (Наружу) или **Inset** (Внутрь);

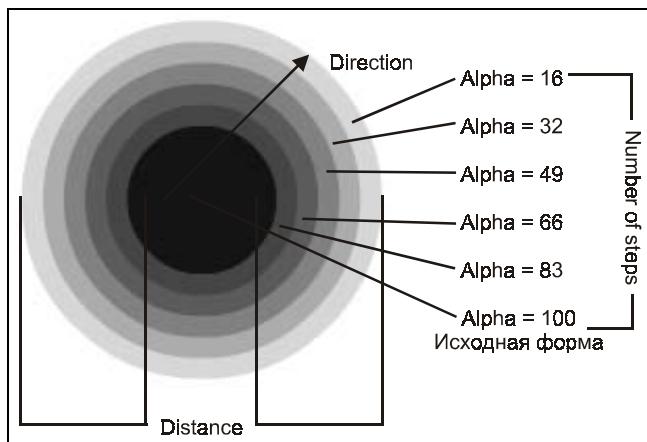


Рис. 5.7. Применение команды **Softens Fill Edges**

Параметры **Distance** и **Number of steps** в совокупности определяют то, насколько гладким будет переход. Так, при расстоянии 100 px и количестве шагов 5 для растушевки будет создано 5 форм, каждая шириной 20 px. Если же значение параметра **Number of steps** задать равным 20, то для имитации растушевки будет создано уже 20 векторных форм и ширина каждой составит 5 px. Очевидно, что во втором случае смягчение краев будет более плавным и аккуратным. Однако при этом стоит помнить о том, что большое число векторных форм, образованных при установлении высоких значений параметра **Number of steps**, приведет к увеличению объема файла, и процесс выполнения команды может занять слишком много времени. Поэтому не стоит злоупотреблять чрезесчур тщательным смягчением краев.

Как и команда **Expand Fill**, данная команда применяется только к векторным формам. Если векторная форма, к которой применяется команда смягчения краев, содержит обводку, то в режиме **Inset** обводка останется на месте, а в режиме **Expand** в результате взаимодействия с созданными векторными формами исчезнет. Применение данной команды к сложным формам с большим числом деталей может дать непредсказуемый эффект.

На практике эту команду можно применять для различных целей. Во-первых, смягчение краев используется для того, чтобы добиться эффекта

реалистичности при передаче объема. Впечатляющие результаты при рисовании сложных поверхностей, фасок, бликов, рефлексов и т. д. позволяет получить применение градиентных заливок наряду с эффектами смягчения краев.

Во-вторых, команда смягчения краев бывает чрезвычайно удобна в ситуации, когда необходимо нарисовать концентрические рамки, повторяющие все очертания формы (рис. 5.8). Для того чтобы это сделать, необходимо применить к форме команду **Soften Fill Edges**, задав все соответствующие параметры. Получившиеся в результате векторные формы можно перекрашивать и удалять, создавая бордюры. Кроме того, при помощи инструмента Чернильница можно добавить векторным формам обводки, получив необходимое число концентрических контуров.

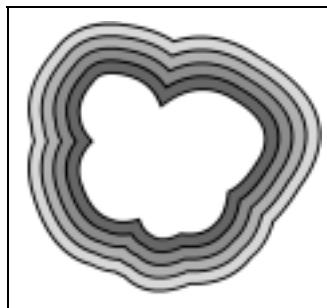


Рис. 5.8. Создание концентрических рамок при помощи команды **Soften Fill Edges**

Импорт векторных графических объектов и последовательностей в Flash

В дополнение к созданию графики средствами встроенного в Flash векторного редактора существует возможность импорта файлов векторной графики, созданных во внешних приложениях.

Импортировать файл можно тремя способами:

□ при помощи команды главного меню Flash **File>Import**. Команда **Import** предлагает следующий выбор:

- **Import to Stage** (Импортировать на сцену) — импортированное изображение помещается на сцену, при этом в библиотеку могут добавляться импортированные символы;
- **Import to Library** (Импортировать в библиотеку) — импортированное изображение помещается в библиотеку, без размещения на сцене;

- через *буфер обмена*. Во внешнем векторном редакторе скопировать векторный объект в буфер и вставить его в рабочей среде Flash при помощи команды меню **Edit>Paste** (<Ctrl>+<V>);
- метод **Drag and Drop**. Если уменьшить размер окна Flash-приложения и рядом открыть окно другого векторного редактора, то можно курсором перетащить векторный объект из одного приложения в другое.

Flash MX 2004 поддерживает возможности импорта векторной графики большинства наиболее широко используемых форматов. К их числу относятся:

- Flash Movie (файлы с расширением swf);
- Free Hand (fh, ft) версия FreeHand MX и ниже;
- Adobe Illustrator (ai) версия Adobe Illustrator 10 и ниже;
- EPS (eps) любая версия (не поддерживается формат EPS, созданный в Photoshop);
- PDF (pdf) версия 1.4 и ниже;
- Enhanced Metafile (emf);
- Windows Metafile (wmf);
- AutoCAD DXF (dxf) (поддерживаются только двоичные двумерные файлы).

Помимо одиночных векторных изображений можно импортировать и целые последовательности. Последовательность представляет собой набор файлов, имя которых заканчивается порядковым номером, например: sequence01.eps, sequence02.eps, sequence03.eps и т. д. Последовательности, как правило, содержат отдельные фазы движения объекта. При импорте такой последовательности в Flash составляющие ее файлы будут автоматически распределены по соответствующим кадрам монтажной линейки, в результате чего получится готовый анимационный ролик. При попытке импорта одного из набора файлов последовательности Flash запросит уведомление о том, нужно ли импортировать всю последовательность.

Импорт файлов Flash Movie

Файл SWF может содержать графику, анимацию и интерактивность. Поскольку в процессе публикации документа Flash происходит компиляция файла, в результате чего вся информация реорганизуется, при импорте Flash-ролика обратно в рабочую среду происходит утрата значительной части его функциональности. Файл SWF импортируется как последовательность векторных изображений, каждое из которых размещается в соответствующем кадре. В случае статичного файла (рисунка) изображение импортируется без всяких потерь. Анимация основной монтажной линейки импортируется в виде последовательности, причем автоматическая анимация заменяется

покадровой. Вся анимация монтажных линеек символов, звук и интерактивность будут потеряны. Таким образом, при импорте файла этого типа можно получить только набор графики в виде кривых, групп, символов типа **Graphic** и текста. Структура слоев основной монтажной линейки также может быть сохранена.

В случае если автор файла запретил возможность импорта, при попытке импортировать ролик в рабочую среду будет выведено сообщение о невозможности выполнения этой операции.

Импорт файлов FreeHand

Компания Macromedia позаботилась о хорошей совместимости двух своих продуктов, поэтому файлы приложения FreeHand, начиная с 7 версии, импортируются во Flash напрямую. При этом имеется возможность сохранить структуру палитры слоев, редактируемый текст и символы. Поскольку FreeHand позволяет создавать многостраничные документы, существует несколько вариантов их размещения в рабочей среде Flash.

Поскольку векторный редактор FreeHand в силу своего основного назначения обладает более изощренными возможностями создания графики, существует несколько нюансов, которые следует иметь в виду:

- градиенты Flash могут содержать не более восьми цветовых порогов. При импорте изображения, включающего сложные градиентные заливки (например, созданные с использованием градиентной сетки), которые содержат более восьми цветовых порогов, они будут разделены на множество мелких групп. Каждая такая группа будет содержать векторную форму, залитую сплошным цветом. Поскольку количество групп может быть очень велико, это чрезвычайно затруднит работу и вызовет существенное увеличение объема файла. Поэтому при разработке графики, которую предполагается в дальнейшем импортировать в Flash, не стоит использовать сложные цветовые переходы без большой необходимости;
- при импорте переходов (**Blends**) каждый шаг перехода преобразуется в отдельный контур. Если переход содержит большое число шагов, то объем файла может очень сильно вырасти;
- при импорте контуров со срезанными или квадратными концами (**caps**) во Flash происходит скругление концов;
- изображения в цветовой модели CMYK преобразуются в RGB.

После выполнения команды **File>Import** и выбора необходимого файла FreeHand появляется диалоговое окно **FreeHand Import**, представленное на рис. 5.9. Здесь содержатся следующие настройки:

- Pages** (Страницы) — настройки, задающие вариант обработки страниц документа FreeHand во Flash.

- **Scenes** — каждая страница документа FreeHand разместится на отдельной сцене Flash;
- **Keyframes** — каждая страница документа FreeHand разместится в отдельном ключевом кадре основной монтажной линейки Flash;



Рис. 5.9. Диалоговое окно **FreeHand Import**

- **Layers** (Слои) — настройки, задающие вариант обработки слоев документа FreeHand во Flash:
- **Layers** — содержимое каждого слоя документа FreeHand разместится на соответствующих слоях Flash, которые будут созданы автоматически. Таким образом, структура слоев будет сохранена;
 - **Keyframes** — содержимое каждого слоя документа FreeHand разместится в отдельном ключевом кадре основной монтажной линейки Flash;
 - **Flatten** — содержимое со всех слоев документа FreeHand разместится в пределах одного слоя Flash;
- **All** (Все) — импортируются все страницы документа FreeHand;
- **From (От) To (До)** — импортируется только заданный диапазон страниц документа FreeHand;
- **Options (Опции)** — дополнительные параметры:
- **Include Invisible Layers** — импортировать содержимое всех слоев документа FreeHand, в том числе тех, отображение которых отключено;

- **Include Background Layer** — импортировать содержимое фонового слоя;
- **Maintain Text Blocks** — импортировать текст как текстовый блок, допускающий возможность редактирования. Эта опция не работает в случае импорта текста по траектории — он будет переведен в контуры.

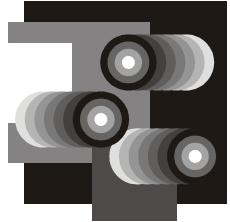
Импорт файлов Adobe Illustrator

Flash MX 2004 может напрямую импортировать файлы Adobe Illustrator, а также файлы в формате EPS и PDF. При импорте файлов Adobe Illustrator изображение представляет собой набор групп, которые могут быть автоматически распределены по соответствующим слоям. После разгруппировки (**Modify>Ungroup**) или разбиения (**Modify>Break Apart**) с импортированной графикой можно работать как с объектами рабочего уровня.

После выполнения команды **File>Import** и выбора необходимого файла Adobe Illustrator появляется диалоговое окно **Import Options**, содержащее параметры настройки, практически идентичные настройкам импорта файлов FreeHand. В дополнение к приведенным выше возможностям здесь имеется возможность *растеризации*, т. е. конвертации векторного изображения в растровое. Для выполнения этой процедуры необходимо установить флажок **Rasterize everything** (Растрировать все) и задать соответствующее разрешение растеризации в поле **Rasterization resolution** (Разрешение растеризации).

Примечание

При использовании коллекций готовых векторных рисунков (*клипартов*) необходимо тщательно оптимизировать контуры (уменьшать количество опорных точек, удалять лишние сегменты кривых и т. п.) и градиентные заливки в целях уменьшения объема конечного файла.



Глава 6

Работа с цветом

Как при прикосновении ко льду можно испытать только ощущение физического холода и это ощущение забывается при согревании пальца, так забывается и физическое воздействие цвета, когда от него отвернешься. Но как физическое ощущение ледяного холода, если оно проникнет глубже, вызывает более глубокие чувства и может вызвать целую цепь психических переживаний, так же и поверхностное впечатление от цвета может развиться в переживание.

Василий Кандинский, "О духовном в искусстве"

Текущий каталог цветов Flash по умолчанию содержит стандартный набор из 216 цветов, однако в действительности возможности использования цвета гораздо шире. В рабочей среде Flash можно синтезировать любой из более чем 16 миллионов оттенков, доступных в цветовой системе True Color. Кроме того, на стадии синтеза цвета можно задать степень прозрачности, что позволяет получать интересные визуальные эффекты, связанные с перекрыванием частично прозрачных объектов. Имеется возможность загрузки внешних цветовых палитр и создания собственных, которые могут храниться в отдельном файле. Таким образом, при работе над большим проектом несколько документов могут использовать общие цветовые палитры, что позволяет обеспечить единое цветовое решение различных его разделов.

В Flash существуют три типа заливок:

- сплошная (Solid);
- градиентная (Gradient);
- растровая (Bitmap).

Как известно, обводки могут быть окрашены только сплошным цветом, в то время как заливки могут содержать любой из вышеперечисленных типов цветового заполнения. Необходимо также помнить, что цвет, как базовое свойство векторного объекта, может быть изменен только на рабочем уровне. Для того чтобы изменить цвет содержимого группы, необходимо войти в режим ее редактирования.

Подробная информация об использовании растровой заливки содержится в гл. 7 разд. "Растровое изображение как объект рабочего уровня".

Понятие о цветовых моделях RGB и HSB

В Flash MX 2004 существует несколько альтернативных способов синтеза цвета. Один из них предполагает задание точных координат цвета в виде числовых значений составляющих соответствующей цветовой модели. Этот способ позволяет абсолютно точно синтезировать нужный цвет при условии, что его координаты известны. Для приблизительного подбора цвета на основе данного метода необходимо иметь представление о закономерностях, присущих той или иной цветовой модели. Синтез цвета в Flash осуществляется на основе одной из двух цветовых моделей. Это модели RGB и HSB.

Другой способ синтеза цвета предполагает выбор необходимого цветового оттенка "на глаз" при помощи цветового поля, содержащего все доступные оттенки цвета. Этот способ является более наглядным, однако подобрать цвет в точности такой же, как цвет образца, таким образом представляется довольно непростой задачей. Выбор цвета "на глаз" применяется в случае, когда нет необходимости в подборе его точного значения. Например, если при разработке цветового решения важно найти гармоничное сочетание цветов элементов композиции, то при этом анализируются соотношения цветов, а не их координаты. И наоборот, если цветовое решение уже жестко определено, необходимо точно контролировать значения цветовых составляющих.

Кроме того, синтезировать цвет на основе образца можно, совершенно механически "подобрав" его с любой точки экрана пипеткой (*данная процедура описывается в прим. к разд. "Инструмент Eyedropper" гл. 4*).

Далее будут представлены базовые сведения о принципах использования цветовых моделей RGB и HSB.

Цветовая модель RGB

В основе цветовой модели лежат три *первичных*, или *основных* цвета: Red (красный), Green (зеленый), Blue (синий). В качестве первичных выделяются три цвета, позволяющие при смешении воспроизвести наиболее широкий цветовой охват. При этом характерной особенностью первичных цветов является то, что ни один из них не может быть получен путем смешения двух других. Свое название модель RGB получила по первым буквам ее составляющих.

Любой из шестнадцати миллионов цветов, которые могут быть отображены на экране монитора в режиме True Color, может быть получен в результате

смешения трех первичных цветов. Поэтому модель RGB называется *аддитивной*.

В рамках данной модели каждый первичный цвет может иметь 256 значений интенсивности (яркости) от 0 до 255, и в результате смешения можно получить $256 \times 256 \times 256 \approx 16,7$ миллионов цветов. При нулевом значении параметра яркости излучение света отсутствует, при значении 255 излучение света максимально. Если значения всех трех составляющих равны нулю, получается черный цвет; если значения всех трех составляющих максимальны, результирующий цвет будет белым. При равных числовых значениях всех трех яркостных составляющих получаются оттенки серого цвета.

Хроматические (т. е. имеющие цветовой тон) цвета могут быть получены при смешении трех составляющих в разных соотношениях. Основные закономерности такого смешения представлены ниже.

Смешение **красного** и **зеленого** цветов при нулевом значении синего дает желтый цвет. При уменьшении интенсивности красного получается желто-зеленый цвет, а уменьшение интенсивности зеленого дает оранжевый.

Смешение **зеленого** и **синего** цветов при нулевом значении красного дает голубой цвет. С уменьшением интенсивности зеленого получается синеватый цвет; соответственно уменьшая интенсивность синего, можно получить зеленоватый цвет.

Смешение **синего** и **красного** дает фиолетовый цвет. При ослаблении синей составляющей цветовой тон смещается в сторону розового, а при уменьшении интенсивности — в сторону пурпурного.

Для получения чистых красного, зеленого или синего цветов необходимо задать максимальную интенсивность соответствующих составляющих (255), при этом установив значения интенсивности двух других составляющих равными нулю.

Модель RGB имеет большое значение, поскольку она используется для воспроизведения цвета на дисплее компьютера.

Понятие о шестнадцатеричном представлении цвета (HEX)

Вместо задания числовых значений составляющих модели RGB в диапазоне от 0 до 255 можно воспользоваться их шестнадцатеричным представлением (**Hexadecimal**).

В основе десятичной системы счисления лежит число 10; здесь числа изменяются от 0 до 9, после чего повторяются с добавлением нового разряда 1, 2, 3 и т. д. Таким образом, в десятичной системе существует 10 цифр. В основе шестнадцатеричной системы счисления лежит число 16. Шесть недостающих цифр заменяются первыми буквами латинского алфавита от A до F,

соответственно, числа изменяются в диапазоне от 0 до F. Таким образом, десятичному числу 15 соответствует шестнадцатеричное число F. После числа F вновь начинается повторение, но при этом добавляется новый разряд: 10, 11, 12, ..., 1F. В двузначном шестнадцатеричном числе цифра левого разряда указывает, сколько раз по 16 содержится в числе, а цифра второго разряда соответствует единицам, т. е. неполной части шестнадцати. Например, число 1C переводится в десятичную систему счисления следующим образом: $1C = 1 \times 16 + 12 = 28$.

Значение каждой цветовой составляющей модели RGB может быть представлено двузначным шестнадцатеричным числом, изменяющимся в диапазоне от 00 (десятичное число 0) до FF (десятичное число 256). Таким образом, белый цвет в шестнадцатеричном представлении задается как FFFFFF, черный — 000000, чистый красный — FF0000 и т. д.

Такой способ задания цвета получил широкое распространение в сети Интернет.

Цветовая модель HSB

Цветовая модель HSB позволяет реализовать интуитивно наиболее понятный способ синтеза цвета. Использование данной модели напоминает процесс подбора цвета при работе с реальными красками, когда сначала выбирается необходимый цветовой тон, после чего он может быть освещен или затемнен, и, наконец, краска наносится на полотно с той или иной степенью интенсивности, или насыщенности.

Свое название модель HSB получила по первым буквам ее составляющих: **Hue** (Тон), **Saturation** (Насыщенность) **Brightness** (Яркость). Рассмотрим эти составляющие.

Hue (Тон) — цветовой тон, характеристика цвета, определяемая длиной световой волны. Поскольку визуально модель HSB представляет собой круг, по краю которого расположены цвета максимальной интенсивности, значение параметра **Hue** задается в градусах в диапазоне от 0 до 360. Так, чистому красному цвету соответствует значение 0, чистому зеленому — 120, а чистому синему — 240 градусов.

Saturation (Насыщенность) — данный параметр характеризует чистоту цвета. Уменьшение значения этого параметра приводит к разбелению цвета. При нулевой насыщенности любой цветовой тон превратится в белый цвет. Наборот, максимальная степень насыщенности дает наиболее интенсивный (чистый) цвет. Значение параметра насыщенности задается в процентах и лежит в диапазоне от 0 до 100.

Brightness (Яркость) — параметр, задающий степень освещенности цвета. При нулевой яркости освещение отсутствует, и любой цветовой тон превращается в черный цвет. Максимальная яркость соответствует хорошо ос-

вещенной цветной поверхности. Значение яркости задается в процентах, в диапазоне от 0 до 100.

Таким образом, синтез цвета при помощи модели HSB предполагает последовательность действий, включающую выбор цветового тона и установку значений насыщенности и яркости.

Синтез сплошного цвета

Для синтеза цвета в рабочей среде Flash используется панель **Color Mixer** (Синтез цвета). Цвет может быть синтезирован на основе одной из представленных цветовых моделей путем задания его точных координат либо выбран "на глаз" при помощи цветового поля, содержащего все возможные цветовые оттенки. Кроме того, синтезировать сплошной цвет можно "подобрав" его с любой точки экрана пипеткой (*данная процедура описывается в прим. к разд. "Инструмент Eyedropper" гл. 4*).

Использование панели **Color Mixer**

Для того чтобы открыть панель **Color Mixer** (Синтез цвета), необходимо выполнить команду главного меню **Window>Design Panels>Color Mixer** или нажать сочетание клавиш <Shift>+<F9>. Внешний вид панели представлен на рис. 6.1.

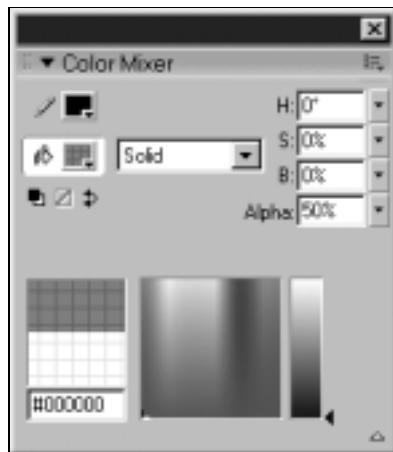


Рис. 6.1. Панель синтеза цвета **Color Mixer**

В левом верхнем углу панели **Color Mixer** находится блок управления цветом, аналогичный блоку управления цветом панели инструментов (см. разд.

"Блок управления цветом", гл. 4). Поле **Fill Style** позволяет выбрать один из возможных типов заливки.

Три поля ввода, расположенные в правом верхнем углу панели, предназначены для задания числовых значений составляющих одной из двух цветовых моделей RGB или HSB. Числовые значения могут быть введены вручную или при помощи вертикального слайдера, расположенного справа от каждого поля. Для переключения между цветовыми моделями используются соответствующие команды контекстного меню палитры, которое можно открыть, щелкнув на стандартной пиктограмме, расположенной справа в полосе заголовка панели.

Поле **Alpha** позволяет задать степень прозрачности для цвета. Прозрачность задается в процентах, в диапазоне от 0 до 100. При значении **Alpha**, равном нулю, цвет полностью прозрачен, при **Alpha** = 100 цвет полностью непрозрачен. Все промежуточные значения позволяют задать некоторую степень прозрачности. При установлении значения прозрачности, отличного от 100, в поле цветового образца, содержащего синтезируемый цвет, появляется светло-серая сетка, свидетельствующая о его прозрачности. Возможность назначения цвету прозрачности непосредственно на стадии его синтеза позволяет, например, окрашивать векторные формы и контуры частично прозрачным цветом, использовать прозрачный текст и прозрачный фон рабочей области.

Примечание

При перекрывании нескольких объектов, окрашенных прозрачным цветом, их необходимо сгруппировать, чтобы избежать взаимодействия. В противном случае вышележащая форма (или контур) провзаимодействует с нижележащей, в результате чего произойдет сегментирование (см. разд. "Понятие о рабочем и наложенном уровнях" гл. 4).

Нижняя часть панели **Color Mixer** может представлять собой цветовую полосу (компактный вид) или содержать цветовое поле и другие элементы. Для того чтобы развернуть или, наоборот, привести панель к более компактному виду используется пиктограмма треугольника справа внизу.

В развернутом виде панель синтеза цвета предоставляет возможность выбора цвета вручную при помощи цветового поля. На цветовом поле можно выбрать оттенок цвета, перемещая по нему курсор в виде перекрестия. При этом перемещение по горизонтали изменяет цветовой тон (Hue), а перемещение по вертикали — насыщенность (Saturation) и яркость (Brightness). При помощи расположенного справа от цветового поля регулятора устанавливается требуемое соотношение между яркостью и насыщенностью синтезируемого цвета. Верхняя грань цветового поля содержит чистые цветовые тона (максимальная насыщенность), а нижняя грань — серый цвет (нулевая насыщенность). Следует иметь в виду, что при уста-

новке крайних значений яркости посредством вертикального слайдера (верхнее или нижнее положение регулятора) вне зависимости от выбранного цветового оттенка и степени насыщенности будут получаться ахроматические, т. е. белый или черный цвета.

При выборе цвета вручную в поле цветового образца, расположенного в нижней части панели, отображаются предыдущий и текущий цвета. Это дает возможность оценить синтезируемый цвет в отношении к другому цвету, что бывает необходимо при поиске наиболее гармоничного цветового решения.

Поле **HEX** позволяет задать (или получить) координаты цвета в шестнадцатеричном представлении.

Синтезированный цвет можно сохранить, добавив его в текущий каталог цветов при помощи команды **Add Swatch** (Добавить Образец) контекстного меню панели **Color Mixer**. Новый образец цвета будет помещен в конец текущего каталога цветов.

Панель **Color Mixer** может быть использована для назначения цветов элементам векторного объекта. Для выполнения этой процедуры необходимо предварительно выделить соответствующий элемент на сцене. После этого в блоке управления цветом на панели синтеза нужно щелкнуть на пиктограмме карандаша (в случае контура) или ведра заливки (в случае векторной формы) и установить цвет любым из описанных методов. Таким образом можно наблюдать изменение цвета обводки или заливки непосредственно на самом векторном объекте.

Кроме синтеза сплошного цвета, на панели **Color Mixer** можно выполнить синтез градиента и выбрать растровую заливку. Об этом будет подробно рассказано далее в настоящей главе.

Использование системной палитры **Color**

Альтернативой панели является системная палитра **Color**, также позволяющая синтезировать цвет на основе одной из двух цветовых моделей RGB или HSL (Hue Saturation Luminosity — другое название системы HSB). Системная палитра содержит свой собственный мини-каталог сплошных цветов, позволяя добавлять туда новые образцы цвета.

Для того чтобы открыть палитру **Color**, необходимо щелкнуть на цветовом образце в блоке управления цветом (на панели инструментов или на панели синтеза цвета) и в появившемся каталоге цветов нажать кнопку с пиктограммой цветового круга, расположенную в правом верхнем углу .

Окно **Color** представлено на рис. 6.2. В левой части окна расположен каталог цветов, состоящий из двух разделов. Раздел **Basic colors** (Основные цвета) содержит набор из 48 цветов, которые не могут быть заменены. Раздел **Custom colors** (Заказные цвета) служит для формирования пользовательского каталога цветов.

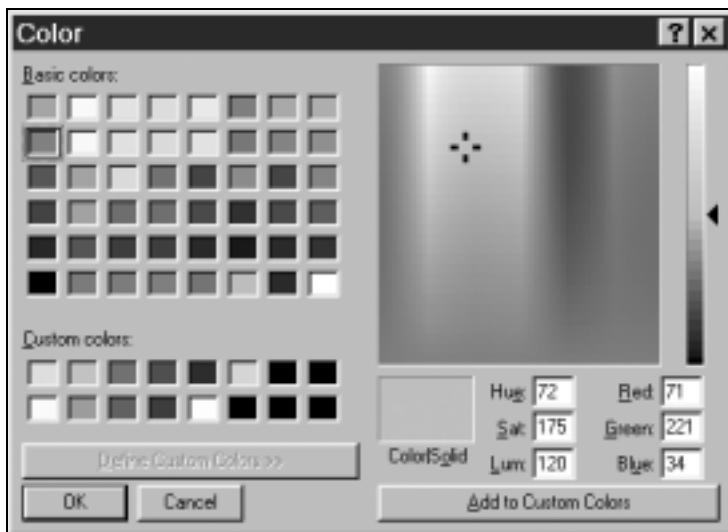


Рис. 6.2. Системная палитра **Color**

Синтез цвета осуществляется так же как и на панели **Color Mixer** при помощи аналогичного цветового поля или полей, предназначенных для ввода точных числовых значений составляющих моделей RGB или HSL. Здесь ввод значений составляющих RGB осуществляется в десятичном представлении (в диапазоне от 0 до 255), а значения составляющих HSL задаются в диапазоне от 0 до 240. Для добавления текущего цвета в каталог палитры **Color** нужно нажать кнопку **Add to Custom Color** (Добавить Заказной Цвет).

Работа с каталогом цветов **Color Swatches**

Каталог цветов **Color Swatches** выполняет в рабочей среде Flash функцию палитры, содержащей фиксированный набор цветов и обеспечивающей быстрый доступ к ним при работе с любым рисующим инструментом. Синтезированный на панели **Color Mixer** сплошной цвет или градиент при сохранении помещается в текущий каталог цветов.

Чтобы открыть панель **Color Swatches**, нужно выполнить команду главного меню **Window>Design Panels>Color Swatches** или использовать сочетание клавиш **<Ctrl>+<F9>**. Внешний вид панели представлен на рис. 6.3.

Панель каталога цветов состоит из двух отделов. В верхнем отделе содержатся сплошные цвета, в нижнем — градиенты. При добавлении нового цвета он размещается в конце соответствующего раздела каталога. В том случае, если каталог содержит большое количество цветов, для просмотра его содержи-

МОГО можно воспользоваться полосами прокрутки. При необходимости можно, взявшись за границы панели, растянуть ее, в результате чего все цветовые образцы будут увеличены.

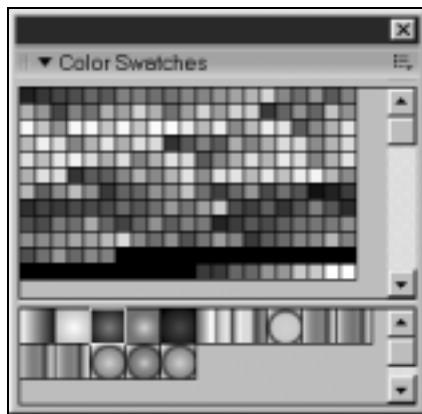


Рис. 6.3. Палитра **Color Swatches**

Контекстное меню панели каталога цветов содержит следующие команды, позволяющие выполнять различные манипуляции с цветовыми образцами и наборами цветов:

- **Duplicate Swatch** (Дублировать Образец) — создать копию текущего цветового образца. Для того чтобы скопировать цветовой образец, можно также щелкнуть на нем курсором, принявшим вид пипетки, и затем щелкнуть в свободной области каталога цветов. В свободной области курсор принимает вид ведра заливки. Копия образца помещается в конец каталога цветов;
- **Delete Swatch** (Удалить Образец) — удалить текущий образец из каталога цветов;
- **Add Colors** (Импорт Палитры) — загрузить цветовую палитру из внешнего файла. Flash может импортировать следующие форматы цветовых палитр: Flash Color Set с расширением clr, Color Table с расширением act и файлы в формате GIF с расширением gif. Импорт градиентов может быть осуществлен только при использовании формата Flash Color Set (clr). В результате выполнения этой команды новые цветовые образцы будут добавлены к уже имеющимся и размещены в конце каталога цветов;
- **Replace Colors** (Заменить Палитру) — заменить текущий каталог цветов палитрой из внешнего файла. В результате выполнения этой команды новые цветовые образцы заместят имеющиеся;

- Load Default Colors** (Загрузить Палитру по умолчанию) — загрузить набор цветов по умолчанию;
- Save Colors** (Экспорт Палитры) — сохранить текущий каталог цветов во внешнем файле. Для сохранения можно использовать форматы Flash Color Set (clr) или Color Table (act). Градиенты могут быть сохранены только в формате CLR;
- Save as Default** (Сохранить как Палитру по умолчанию) — сохранить текущий каталог цветов в качестве палитры по умолчанию. Данная палитра всегда будет использоваться по умолчанию при создании нового документа;
- Clear Colors** (Очистить Каталог) — очистить каталог цветов. В результате выполнения этой команды в каталоге останутся только два цвета: черный и белый;
- Web 216** (Безопасная палитра) — загрузить безопасную (инвариантную) палитру, состоящую из 216 цветов. Цвета, входящие в состав этой палитры, без искажений отображаются на различных платформах и в различных браузерах. Кроме того, использование безопасной палитры позволяет корректно отображать цветовую информацию выводными устройствами, способными работать с ограниченным количеством цветов. К числу таких устройств относятся устаревшие модели мониторов, а также некоторые мобильные устройства (сотовые телефоны, карманные ПК и т. д.);
- Sort by Color** (Сортировать по цвету) — распределить цвета в каталоге по цветовому тону. Эта команда не приводит к добавлению или удалению цветов из каталога, она только изменяет порядок их размещения. Для того чтобы восстановить прежний порядок размещения цветов, можно загрузить палитру по умолчанию или другую внешнюю палитру.

Таким образом, процесс синтеза сплошного цвета включает в себя следующую последовательность действий:

1. Открыть палитру **Color Mixer** при помощи команды **Window>Design Panels>Color Mixer**.
2. Выбрать тип заливки **Solid** (Сплошная).
3. Выбрать цветовую модель (RGB или HSB) при помощи контекстного меню палитры.
4. Выполнить одно из следующих действий:
 - ввести числовые значения цветовых составляющих данной цветовой модели;
 - на цветовом поле выбрать нужный оттенок, степень его насыщенности и задать яркость при помощи расположенного рядом регулятора;
 - ввести шестнадцатеричное значение цвета в поле **HEX**.

5. Задать степень прозрачности цвета **Alpha**.
6. Сохранить цветовой образец в текущем каталоге цветов при помощи команды **Add Swatch** (Добавить образец) контекстного меню палитры **Color Mixer**.

Данные об основных операциях с цветом представлены в табл. 6.1.

Таблица 6.1. Операции с цветом

Операция	Способ
Синтез цвета	Панель Color Mixer
Хранение цвета	Панель Color Swatches
Назначение цвета по умолчанию	Кнопка на панели Color Mixer ; блок управления цветом Color панели Tools — черная обводка и белая заливка
Назначение цвета заливке	Инструмент Paint Bucket ; панели Color Mixer , Color Swatches , Properties ; блок управления цветом Color , системная палитра Color
Назначение цвета обводке	Инструмент Ink Bottle ; панели Color Mixer , Color Swatches , Properties ; блок управления цветом Color ; системная палитра Color . Обводка не может быть окрашена градиентом или расцветкой
Подбор образца цвета	Инструмент Eyedropper

Синтез градиентной заливки

Градиент представляет собой плавную растяжку (переход) цветов. Основные цвета, входящие в градиент, называются *цветовыми порогами* градиента. В Flash градиент может содержать не более восьми цветовых порогов. Градиент может быть *линейным* (Linear) или *радиальным* (Radial). В линейном градиенте переход цветов осуществляется вдоль прямолинейной оси, совпадающей с направлением градиента. Цветовые пороги радиального градиента представляют собой концентрические окружности с общим центром, диаметр которых увеличивается по мере удаления от центра. Цветовой переход в этом случае осуществляется вдоль радиусов (рис. 6.4).

Градиентная заливка может быть применена только к векторным формам. Для того чтобы окрасить градиентом контур, его предварительно необходимо трансформировать в форму (см. разд. "Дополнительные возможности работы с векторными формами" гл. 5).

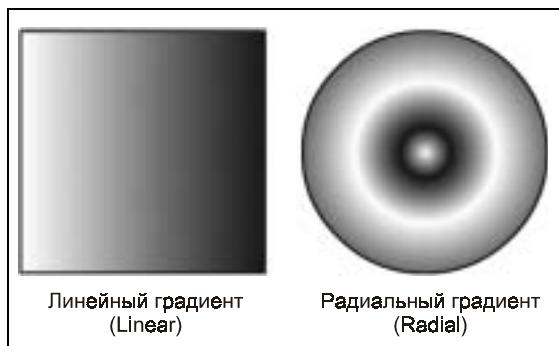


Рис. 6.4. Типы градиентов Flash

Создание градиента

Синтез градиента осуществляется при помощи панели **Color Mixer**. Создание линейного или радиального градиентов выполняется единообразно.

Для того чтобы создать градиент, необходимо выполнить следующие действия:

1. Открыть панель **Color Mixer** при помощи команды **Window>Design Panels>Color Mixer**.
2. Выбрать тип заливки **Linear** (Линейный) или **Radial** (Радиальный). На панели появится полоса редактирования градиента. Под полосой градиента расположены маркеры редактирования градиента. Каждый маркер соответствует цветовому порогу.
3. Развернуть панель **Color Mixer**, щелкнув на пиктограмме треугольника в правом нижнем углу. Панель должна принять вид, изображенный на рис. 6.5.
4. Для того чтобы изменить цвет порога градиента, необходимо щелкнуть на маркере, выделив его. Выделенный маркер отмечен черным треугольником.
5. Задать цвет порога можно одним из следующих способов:
 - щелкнуть на цветовом образце на панели **Color Mixer** и из появившегося текущего каталога цветов выбрать необходимый оттенок;
 - ввести числовые значения цветовых составляющих данной цветовой модели;
 - на цветовом поле выбрать нужный оттенок, степень его насыщенности и задать яркость при помощи расположенного рядом регулятора;
 - ввести шестнадцатеричное значение цвета в поле **HEX**.



Рис. 6.5. Синтез градиента.
Панель **Color Mixer**

6. Задать степень прозрачности цветового порога градиента можно, введя значение в диапазоне от 0 до 100 в поле **Alpha**.
7. Для того чтобы добавить новый цветовой порог в градиент, необходимо щелкнуть курсором на полосе редактирования, при этом рядом с курсором должна появиться пиктограмма с плюсом. Градиент не может содержать более восьми цветовых порогов.
8. Перемещая маркеры по горизонтали, можно изменять расстояние между цветовыми порогами градиента. Если маркер отодвинут от боковой границы полосы редактирования, то край градиента будет заполнен сплошным цветом, соответствующим цвету крайнего маркера.
9. Для того чтобы удалить цветовой порог из градиента, нужно взяться за соответствующий маркер редактирования и оттащить его за пределы зоны редактирования.
10. Для того чтобы изменить тип синтезированного градиента, нужно выбрать в списке типов заливки необходимое значение. При этом в поле предварительного просмотра отобразится измененный результат.
11. Сохранить созданный градиент в текущем каталоге цветов, выполнив команду **Add Swatch** (Добавить образец) контекстного меню палитры **Color Mixer**.

◀▶ Примечание

Если созданный градиент не будет сохранен в текущем каталоге цветов, то при использовании другого градиента предыдущий будет утерян. Если градиент предполагается использовать многократно, он должен быть сохранен.

Окрашивание градиентом

Для того чтобы *нарисовать* векторную форму, залитую градиентом, необходимо активизировать любой инструмент, создающий формы. После этого в блоке управления цветом в качестве цвета заливки нужно выбрать любой градиент из текущего каталога цветов и приступить к рисованию. Получившаяся в результате форма будет окрашена выбранным градиентом.

Для того чтобы *переопределить* имеющийся тип заливки или заменить текущую градиентную заливку новой, можно выделить ее и выбрать другой градиент в текущем каталоге цветов. Предыдущая заливка будет заменена новой.

Если в процессе синтеза градиента на сцене имеется выделенная векторная форма, то на ней будут отражаться все изменения, связанные с редактированием градиента. Таким образом, можно осуществлять синтез градиента, тут же применяя его к соответствующему объекту сцены.

Для работы с заливками используется инструмент **Paint Bucket** (Ведро Заливки). При помощи этого инструмента осуществляется заливка контуров или переопределение цветового заполнения векторных форм. При работе с градиентом **bycнhevtynjv** **Paint Bucket** позволяет также задать направление линейного градиента или центр радиального.

Для того чтобы задать направление линейного градиента непосредственно в момент заливки, необходимо щелкнуть инструментом **Paint Bucket** на соответствующей векторной форме (или внутри замкнутого контура) и, не отпуская кнопку мыши, протянуть курсор в направлении распространения градиента. При этом направление градиента указывается прямой линией, соединяющей начальную точку заливки с курсором. Для того чтобы направление градиента изменялось с дискретностью 45 градусов, нужно удерживать клавишу **<Shift>**.

Длина линии направления градиента определяет расстояние между его крайними цветовыми порогами. Если линия целиком размещается внутри формы, то за ее пределами будет находиться сплошной цвет, соответствующий крайним цветовым порогам градиента. Если же линия, наоборот, выходит за границы формы, то градиент будет срезан, т. е. часть его цветовых порогов не будет видна.

Работая с радиальным градиентом при помощи инструмента **Paint Bucket**, можно установить местоположение его центра. Для этого необходимо щелкнуть инструментом **Paint Bucket** на форме или внутри контура и, не отпуская кнопки мыши, сместить курсор с появившимся рядом кружком в точку, соответствующую центру градиента.

Градиент можно успешно применять для передачи *эффекта объема*. Радиальный градиент может также служить для имитации световых бликов или рефлексов цвета. Так, для имитации блика левый цветовой порог радиального

градиента должен содержать светлый оттенок (освещенная область), а правый — тот же цвет, но с пониженной степенью яркости. Для более сложного перехода можно добавить несколько промежуточных цветовых порогов.

Линейный градиент можно использовать для имитирования освещенной поверхности или фона (рис. 6.6).



Рис. 6.6. Использование градиента
для передачи объема и эффектов освещения

◀ Примечание ▶

Для хранения градиентной заливки требуется больший объем памяти, чем для сплошной. Область, заполненная градиентом, занимает примерно на 50 байт больше, чем та же область, содержащая сплошной цвет. Использование большого числа градиентов может привести к существенному увеличению конечного объема файла, поэтому не следует злоупотреблять ими без необходимости.

Редактирование градиента. Инструмент *Fill Transform*

Для редактирования градиентной заливки используется инструмент **Fill Transform** (Трансформация Заливки)

- масштабирование;
- поворот;

- перенос центра градиента;
- скос (для радиального градиента).

Трансформация линейного градиента

Для того чтобы трансформировать заливку, представляющую собой линейный градиент, необходимо выполнить следующие действия:

1. Активизировать инструмент **Fill Transform** (Трансформация Заливки) на панели инструментов.
2. Щелкнуть на векторной форме, содержащей линейный градиент. Рядом с векторной формой появятся маркеры трансформации градиента (рис. 6.7). В том случае, если их не видно, необходимо уменьшить масштаб просмотра.

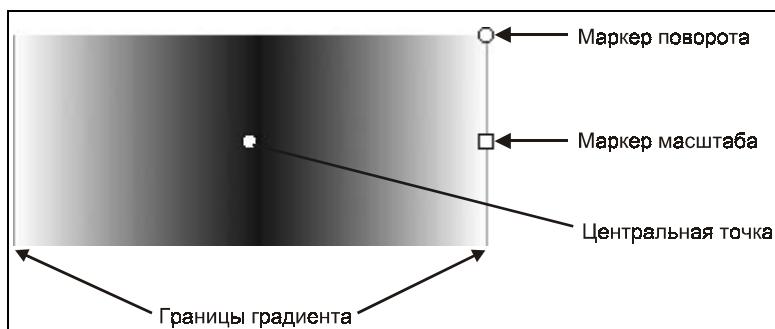


Рис. 6.7. Редактирование линейного градиента

3. Для того чтобы изменить масштаб градиента, необходимо переместить квадратный маркер. При этом курсор принимает вид двунаправленной стрелки. Выполнение этого действия приведет к пропорциональному изменению расстояний между цветовыми порогами градиента.
4. Для того чтобы повернуть градиент, необходимо переместить боковой круглый маркер. При этом курсор принимает вид круговых стрелок. Выполнение этого действия приведет к изменению направления градиента.
5. Для того чтобы перенести центр градиента, необходимо переместить центральный круглый маркер. При этом курсор принимает вид крестообразной стрелки. Выполнение этого действия приведет к смещению всего градиента внутри векторной формы.

Трансформация радиального градиента

Для того чтобы трансформировать заливку, представляющую собой радиальный градиент, необходимо выполнить следующие действия:

1. Активизировать инструмент **Fill Transform** (Трансформация Заливки) на панели инструментов.
2. Щелкнуть на векторной форме, содержащей радиальный градиент. Вокруг векторной формы появится окружность с маркерами трансформации градиента (рис. 6.8). В том случае, если их не видно, необходимо уменьшить масштаб просмотра.

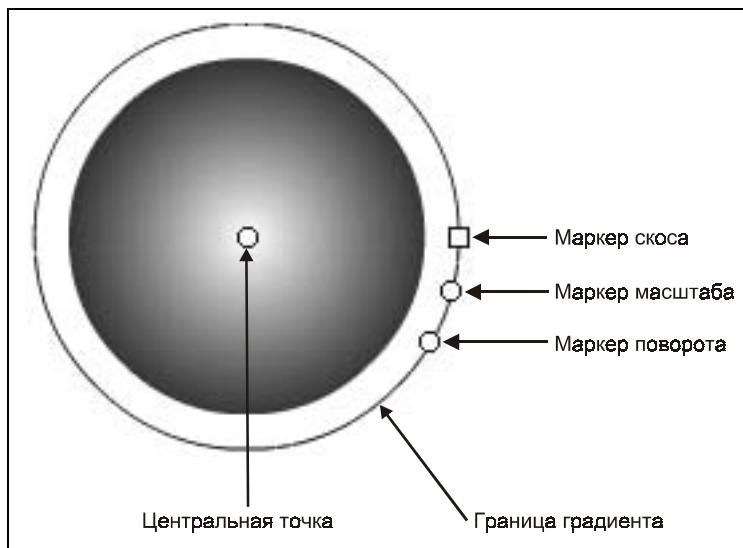


Рис. 6.8. Редактирование радиального градиента

3. Для того чтобы изменить масштаб градиента, необходимо переместить круглый средний маркер на окружности. При этом курсор принимает вид кружка со стрелкой. При выполнении этого действия изменяется радиус градиента.
4. Для того чтобы повернуть градиент, необходимо переместить круглый крайний маркер. При этом курсор принимает вид круговых стрелок.
5. Для того чтобы выполнить скос градиента, необходимо переместить квадратный маркер на окружности. При этом курсор принимает вид двухнаправленной стрелки.
6. Для того чтобы перенести центр градиента, необходимо переместить центральный круглый маркер. При этом курсор принимает вид крестообраз-

ной стрелки. Выполнение этого действия приведет к смещению всего градиента внутри векторной формы.

Примечание

Для того чтобы изменить параметры градиента, добавленного в каталог цветов, можно прямо в палитре **Swatches** выделить его цветовой образец, щелкнув на нем, и переопределить параметры градиента при помощи панели **Color Mixer**.

Использование режима фиксации заливки *Lock Fill*

Инструменты **Paint Bucket** (Ведро Заливки) и **Brush** (Кисть) содержат модификатор **Lock Fill** (Фиксирование заливки) , который позволяет использовать общую заливку для нескольких векторных форм. Для использования данной возможности необходимо выбрать градиент из каталога цветов, активизировать один из указанных инструментов и, включив модификатор **Lock Fill**, последовательно создавать или окрашивать векторные формы. В результате одна и та же градиентная заливка будет распространена на несколько форм (рис. 6.9).

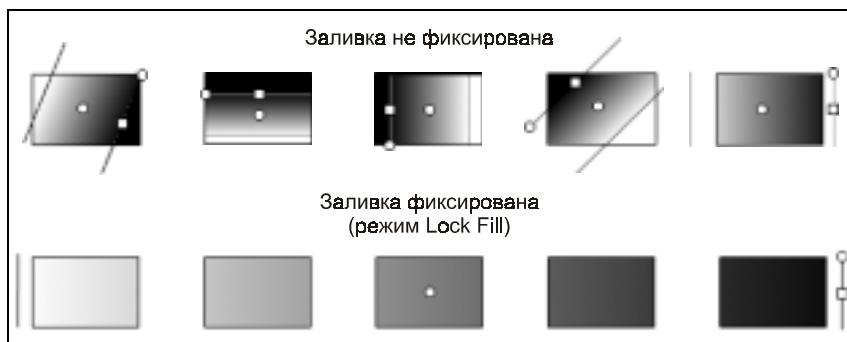


Рис. 6.9. Режим фиксации заливки **Lock Fill**

В случае, если окрашенные формы содержат только часть градиента, его нужно масштабировать при помощи инструмента **Fill Transform** (Трансформация заливки). Как правило, масштаб градиента достаточно велик и маркеры редактирования находятся за пределами рабочей области. Для того чтобы отредактировать градиент, нужно уменьшить масштаб просмотра, так чтобы на сцене появились маркеры редактирования (рис. 6.10).

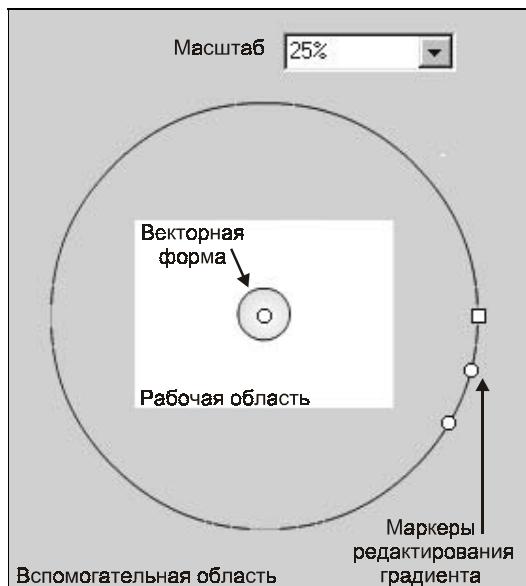
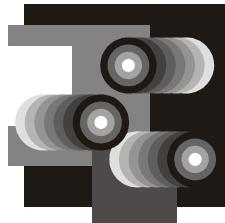


Рис. 6.10. Редактирование фиксированной заливки, выходящей за пределы рабочей области

При редактировании фиксированной заливки изменения отражаются на всех векторных формах, на которые она была распространена. Таким образом, фиксированная заливка трансформируется как единый объект.

◀▶ Примечание

Любая трансформация (перемещение, вращение, масштабирование и т. п.) объекта, входящего в состав объектов с фиксированной заливкой, приводит к тому, что связь с ними утрачивается. В дальнейшем заливка такого объекта трансформируется совершенно независимо от остальных.



Глава 7

Работа с растровой графикой. Импорт видео

Так мало времени и так много пикселов.

Кай Краузе

Несмотря на то, что Flash использует векторный формат представления графической информации, рабочая среда Flash предоставляет достаточно широкие возможности по работе с растровой графикой. Будучи импортированным в рабочую среду, растровое изображение интегрируется в содержимое документа и может быть подвергнуто различным преобразованиям.

Понятие о растровой графике

Растровое изображение представляет собой двумерную матрицу, состоящую из элементарных частей изображения — *пикселов* (pixel — picture element). Каждый пиксель характеризуется точными координатами размещения в расчетной сетке и соответствующим цветовым значением. В растровом изображении цвет превалирует над формой. Форма создается путем окрашивания пикселов в соответствующей области тем или иным цветом. Поскольку количество пикселов конечно, при увеличении масштаба просмотра на определенном этапе изображение "распадается" на отдельные элементы.

Количество пикселов, приходящихся на линейный дюйм изображения, называется *разрешением* изображения (Image resolution). Разрешение задается в единицах ppi — pixels per inch. Чем больше мелких деталей содержит изображение, тем выше должно быть его разрешение для точной передачи нюансов. Размер растрового изображения задается количеством пикселов по вертикали и по горизонтали.

Между размерами растрового изображения в пикселях и его разрешением имеется прямая связь. При увеличении разрешения плотность пикселов в изображении будет повышаться, что, в свою очередь, приведет к уменьшению его физических размеров. Так, изображение с разрешением 72 ppi при

размере 1×1 дюйм состоит из $72 \times 72 = 5184$ пиксела. Если увеличить разрешение до 300 ppi, то плотность размещения пикселов увеличится; соответственно, физический размер изображения составит $0,24 \times 0,24$ дюйма. При этом оно так же, как и раньше будет состоять из 5184 пикселов. Для того чтобы при изменении разрешения физический размер изображения оставался неизменным, необходимо увеличивать количество пикселов, входящих в его состав.

Монитор для отображения изображения использует пиксели или *точки* (dots) экрана. Разрешение экрана также оценивается как количество точек на единицу длины и обычно измеряется в точках на дюйм (имеется в виду логический дюйм) — dpi (dots per inch). Большинство современных мониторов используют разрешение от 72 до 96 dpi.

Часто размер растрового изображения на экране отличается от размера печатного изображения. Это вызвано тем, что пиксели изображения в точности переводятся в пиксели монитора. Поэтому в случае, если разрешение изображения больше, чем разрешение монитора, оно выглядит на экране крупнее, чем на бумаге. Так, для отображения одного линейного дюйма изображения с разрешением 192 ppi монитору с разрешением 96 dpi понадобится два дюйма.

Каждый пиксель растрового изображения имеет соответствующую цветовую характеристику. Количество цветов, которыми может быть окрашен пиксель, зависит от *глубины цвета* изображения (color depth, bit depth). Глубина цвета измеряется в битах (bits) и указывает, какое количество информации отводится для описания цвета каждого пикселя. Одним битом можно закодировать только два состояния или цвета (белый и черный). Например, глубина цвета 24 бита позволяет получить $2^{24} \approx 16,78$ миллионов цветов.

Объем файла, занимаемого растровым изображением и, соответственно, его качество определяются двумя основными параметрами: разрешением и глубиной цвета. Чем выше разрешение и глубина цвета, тем больше объем конечного файла. Растровые изображения, как правило, имеют достаточно большой объем, поэтому перед тем как импортировать их в рабочую среду Flash, необходимо провести оптимизацию в редакторе растровой графики, например Macromedia Fireworks или Adobe Photoshop.

Импорт растровой графики

Flash MX 2004 позволяет импортировать в рабочую среду растровые изображения большинства наиболее широко используемых форматов. К их числу относятся следующие форматы (в скобках указаны расширения файлов):

- GIF (gif);
- JPEG (jpg);
- PNG (png);

- BMP (bmp);
- TGA (tga);
- TIFF (tif, tiff) и некоторые другие.

Импортировать файл можно тремя способами:

- при помощи команды главного меню Flash **File>Import**. Команда **Import** предлагает следующий выбор:
 - **Import to Stage** (Импортировать на сцену) — импортированное изображение помещается на сцену, при этом оно автоматически заносится в библиотеку;
 - **Import to Library** (Импортировать в библиотеку) — импортированное изображение помещается в библиотеку, без размещения на сцене;
- через *буфер обмена*. Во внешнем растровом редакторе векторный объект копируется в буфер и затем вставляется в рабочей среде Flash при помощи команды меню **Edit>Paste** (<Ctrl>+<V>).
- При импорте растрового изображения оно автоматически заносится в библиотеку. Библиотека является хранилищем всех импортированных объектов, включая растровые изображения, видео, звук. Открыть окно библиотеки можно при помощи команды **Window>Library** (<Ctrl>+<L>). В библиотеке рядом с пиктограммой растрового изображения указывается его название. В окне предварительного просмотра выводится уменьшенное отображение (рис. 7.1). (*Подробная информация о работе с библиотекой содержится в гл. 11.*)

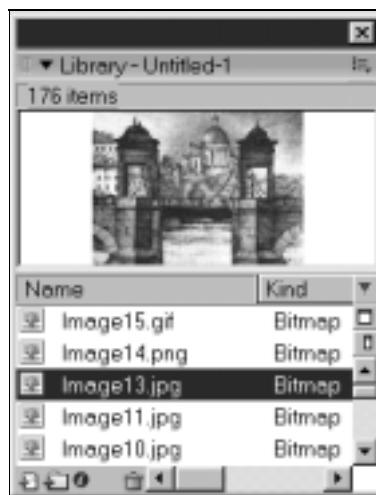


Рис. 7.1. Библиотека документа с импортированными растровыми изображениями

Использование библиотеки имеет характерную особенность. При импорте растрового изображения в библиотеку заносится его оригинал (**Master object**). На сцене размещается копия или ссылка на этот оригинал, которая в терминах Flash носит название **экземпляра** (Instance) изображения. Для того чтобы поместить экземпляр изображения на сцену, его нужно вытащить из библиотеки, взявшись курсором за название или за изображение в окне предварительного просмотра библиотеки. В момент перетаскивания рядом с курсором появляется пиктограмма рамки. В одном фильме можно использовать любое количество экземпляров изображения, при этом конечный объем файла практически не зависит от их числа. Например, если в библиотеке находится изображение, занимающее 40 килобайт, то при размещении на сцене пяти экземпляров этого изображения объем конечного файла превысит данное значение менее чем на 100 байт. Это позволяет многократно использовать одно и то же изображение без существенного увеличения размера файла.

При импорте файлов формата PNG можно задать ряд параметров, позволяющих интегрировать изображение в рабочую среду в виде набора векторных и растровых объектов.

□ **File Structure** (Структура файла).

- **Import as movie clip and retain layers** — импортирует файл как символ типа **Movie Clip**, сохраняя всю его структуру, включая слои и кадры монтажной линейки.
- **Import into new layer in current scene** — импортирует файл в текущий документ на новый слой, который автоматически создается над активным слоем.

□ **Objects** (Объекты).

- **Rasterize if Necessary to Maintain Appearance** — при импорте сохраняются заливки, обводки и эффекты, однако в случае, если их нельзя импортировать как векторные формы, производится растиривание.
- **Keep All Paths Editable** — все объекты импортируются в виде кривых, однако при этом возможна потеря некоторых элементов.

□ **Text** (Текст).

- **Rasterize if Necessary to Maintain Appearance** — при импорте сохраняются заливки, обводки и эффекты текста, однако в случае, если их нельзя импортировать как векторные формы, производится растиривание.
- **Keep All Paths Editable** — весь текст остается редактируемым, однако при этом возможна потеря некоторых элементов.

Кроме одиночных растровых изображений, можно импортировать и целые растровые последовательности. Последовательность представляет собой на-

бор файлов, имя которых заканчивается порядковым номером, например: sequence01.gif, sequence02.gif, sequence03.gif и т. д. При импорте такой последовательности в Flash составляющие ее изображения будут автоматически помещены в библиотеку, а экземпляры этих изображений распределены по соответствующим кадрам монтажной линейки, в результате чего получится готовый анимационный ролик. При попытке импорта одного из набора файлов последовательности Flash запросит уведомление о том, нужно ли импортировать всю последовательность.

Растровое изображение по умолчанию является объектом наложенного уровня и выделяется особой серой рамкой.

В качестве альтернативы внедрению растрового изображения в документ Flash существует возможность динамической загрузки изображений в формате JPEG из внешних файлов в процессе воспроизведения фильма. Для этого используются программные средства языка сценариев ActionScript (*подробнее об этих возможностях см. в гл. 21*).

Оптимизация растровых изображений

Растровые изображения, как правило, занимают больший объем памяти, чем векторные. В связи с этим, растровое изображение, включенное в фильм Flash, должно быть обязательно оптимизировано. Эта оптимизация должна включать в себя два этапа:

- оптимизация в специализированном редакторе растровой графики. Необходимо установить разрешение изображения, не превышающее разрешение экрана монитора (72 dpi или 96 dpi). Размер изображения должен быть уменьшен до величины, которая необходима при использовании изображения в Flash-проекте. Тоновая и цветовая коррекции также выполняются на этапе, предшествующем импорту;
- оптимизация средствами Flash непосредственно в рабочей среде программы, включающая выбор соответствующего алгоритма сжатия.

Для того чтобы оптимизировать растровое изображение в рабочей среде Flash, необходимо выполнить следующие действия:

1. Открыть окно библиотеки текущего документа при помощи команды **Window>Library** (<Ctrl>+<L>).
2. Выполнить одно из следующих действий:
 - два раза щелкнуть на пиктограмме рядом с названием изображения в библиотеке;
 - два раза щелкнуть на изображении в окне просмотра библиотеки;
 - щелкнуть на пиктограмме **Properties** (Свойства)  в нижней части окна библиотеки;

- щелкнуть правой кнопкой мыши на названии изображения в библиотеке и из появившегося контекстного меню выбрать команду **Properties** (Свойства);
 - из контекстного меню панели библиотеки выбрать команду **Properties** (Свойства).
3. В появившемся диалоговом окне **Bitmap Properties** (Свойства растрового изображения) установить флажок **Allow smoothing** (Включить сглаживание) для активизации режима сглаживания краев растрового изображения (рис. 7.2).

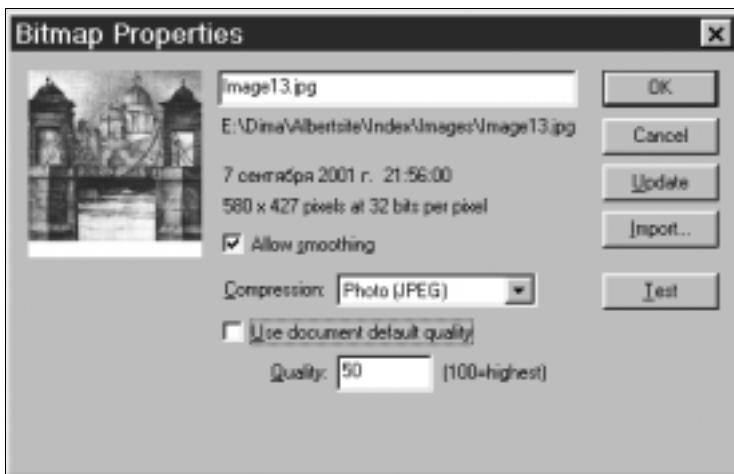


Рис. 7.2. Окно настройки параметров оптимизации растрового изображения **Bitmap Properties**

4. В списке **Compression** (Сжатие) выбрать один из алгоритмов сжатия растровых изображений:
- Photo (JPEG)**. Для использования установок качества по умолчанию установить флажок **Use document default quality**. В этом случае настройки качества задаются при публикации в окне **Publish Settings** (см. гл. 17). При снятии этого флажка появляется поле **Quality** (Качество), позволяющее задать значение коэффициента качества в диапазоне от 0 до 100, влияющее на степень сжатия растрового изображения. Чем выше значения коэффициента качества, тем выше качество изображения и, соответственно, тем больший объем оно будет занимать. Используйте этот алгоритм для фотографических изображений и изображений, содержащих градиенты и сложные цветовые переходы;

- **Lossless (PNG/GIF).** Данный алгоритм используется для сжатия без потерь, в результате которого не происходит утраты информации о цветовом содержании. Используйте этот алгоритм для изображений, содержащих простые формы и локальные (сплошные) цвета, например, схемы, технические рисунки.
5. Для просмотра результата нажмите кнопку **Test**. В результате этого действия в нижней части окна появится сообщение о первоначальном объеме изображения и о результатах оптимизации. В окне просмотра можно оценить внешний вид скатого изображения. Щелчок правой кнопкой мыши по окну просмотра приводит к появлению контекстного меню, в котором можно изменить масштаб отображения.
 6. Нажать кнопку **Update** (Обновить), для того чтобы обновить изображение, загрузив его текущую версию. Эта возможность имеет место в том случае, когда уже после импорта изображения в Flash оно было изменено во внешнем приложении.
 7. Нажать кнопку **Import** (Импортировать), для того чтобы заменить текущее изображение в библиотеке на другое. При этом все его экземпляры в фильме также будут заменены.

Примечание

После оптимизации экземпляр изображения на сцене останется без визуальных изменений, однако в конечном фильме внешний вид оптимизированного изображения может отличаться от исходного. Для того чтобы визуально оценить результаты оптимизации, необходимо перейти в среду тестирования (см. гл. 16).

Параметры оптимизации растрового изображения, заданные в библиотеке, будут применены ко всем его экземплярам, используемым в фильме.

Примечание

Уменьшение физических размеров экземпляра растрового изображения при помощи масштабирования не приведет к уменьшению размера конечного файла, поэтому в рабочую среду необходимо импортировать изображение с тем размером, с которым оно будет использовано в фильме.

Работа с растровыми изображениями

Экземпляр растрового изображения является объектом наложенного уровня. Соответственно, с ним можно выполнять все операции, доступные для объектов наложенного уровня. Однако с растровым изображением можно работать и как с особым объектом рабочего уровня. Для этого его необходимо подвергнуть *разбиению* (Break Apart).

Растровое изображение как объект наложенного уровня

Экземпляр растрового изображения выделяется особой серой рамкой, свидетельствующей о том, что это объект наложенного уровня. При этом в левом верхнем углу Инспектора свойств появляется пиктограмма растрового изображения с надписью **Bitmap**, а чуть правее указывается название изображения, экземпляр которого выделен в данный момент. С каждым экземпляром растрового изображения может быть выполнена одна из следующих операций:

- перемещение при помощи инструмента **Selection** или управляющих клавиш клавиатуры;
- копирование или дублирование. Выполнение этих операций эквивалентно вытаскиванию нового экземпляра изображения из библиотеки на сцену;
- все виды трансформаций, доступных на наложенном уровне: масштабирование, вращение/скос, зеркальное отражение (рис. 7.3.);

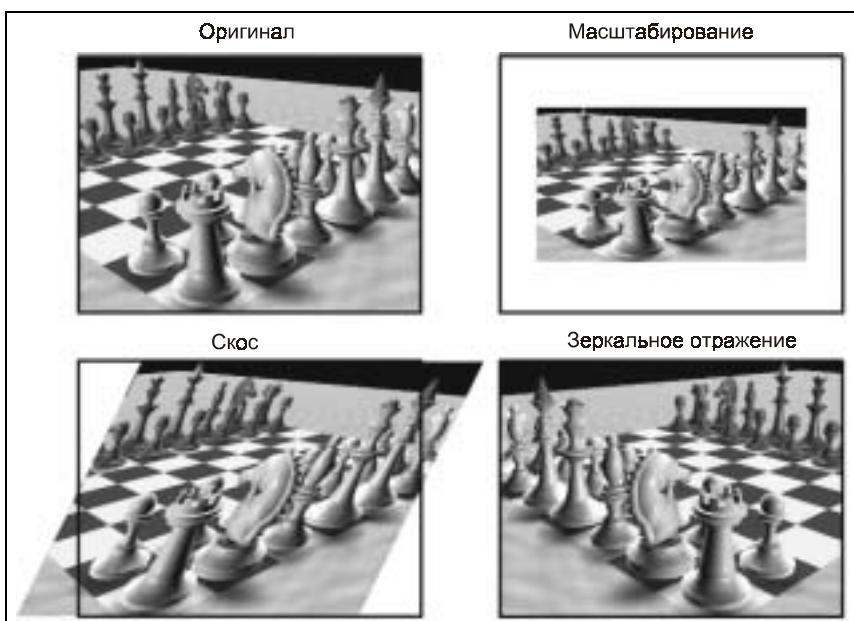


Рис. 7.3. Трансформация экземпляра растрового изображения

- изменение порядка следования в стопке объектов наложенного уровня (меню **Modify>Arrange**);
- группировка с другими объектами рабочего или наложенного уровня, а также с другими растровыми изображениями;

□ замена экземпляра одного изображения экземпляром другого изображения. В результате замены к новому экземпляру будут применены все параметры трансформации, заданные для заменяемого изображения. Эта возможность бывает очень удобна в ситуациях, когда необходимо заменить только один экземпляр другим изображением, поместив его в те же координаты с теми же значениями масштаба, угла поворота и т. д. Для выполнения этой процедуры можно выполнить одно из следующих действий:

- выполнить команду меню **Modify>Bitmap>Swap Bitmap**. В появившемся окне (рис. 7.4) выбрать изображение, экземпляр которого заменит текущий экземпляр;
- нажать кнопку **Swap** Инспектора свойств и далее поступить аналогичным образом.

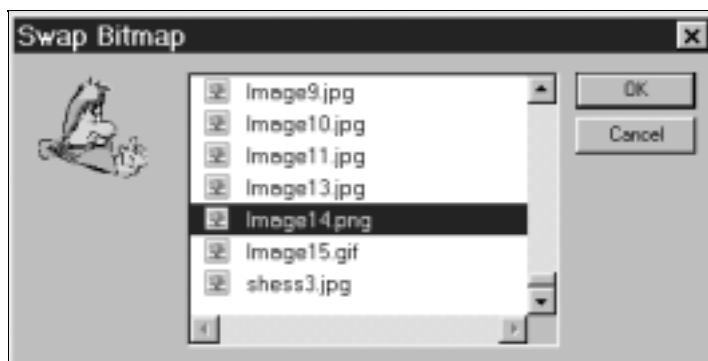


Рис. 7.4. Окно Swap Bitmap

Для того чтобы отредактировать изображение библиотеки во внешнем приложении, нужно, выделив его экземпляр на сцене, нажать кнопку **Edit** (Редактировать) Инспектора свойств или выполнить команду контекстного меню библиотеки **Edit with Fireworks** (Редактировать в Fireworks). Команда контекстного меню библиотеки **Edit with** (Редактировать в) позволит отредактировать изображение в любом внешнем приложении, указав путь к соответствующему исполняемому файлу. При использовании контекстного меню библиотеки название изображения необходимо выделить. Результат редактирования отразится на всех экземплярах данного изображения.

Растровое изображение как объект рабочего уровня

Для перевода экземпляра растрового изображения на рабочий уровень используется команда меню **Modify>Break Apart** или сочетание клавиш

<Ctrl>+. Для того чтобы разбить изображение, его предварительно необходимо выделить. В результате разбиения образуется векторная форма, содержащая раstralную заливку. При выделении она, как и любая векторная форма, отображается сплошной сеткой выделения. Разбитое раstralное изображение продолжает поддерживать связь с библиотекой. Все настройки, связанные с оптимизацией изображения в библиотеке, будут применены и к раstralной заливке. При удалении изображения из библиотеки все использующие его раstralные заливки будут заменены сплошными.

Редактирование разбитого раstralного изображения

При редактировании разбитого раstralного изображения можно использовать все возможности, доступные для объектов рабочего уровня. К их числу относятся:

- перемещение при помощи инструмента **Selection** или управляющих клавиш;
- копирование или дублирование;
- выделение частей раstralной заливки при помощи инструмента **Selection**, инструмента **Lasso** или его модификатора **Magic Wand** (Волшебная палочка);
- взаимодействие с другими объектами рабочего уровня, в том числе с другими разбитыми изображениями. Используя эту особенность, можно, например, "нарезать" раstralное изображение на части при помощи контуров или вырезать в изображении очертания другого предмета (рис. 7.5);

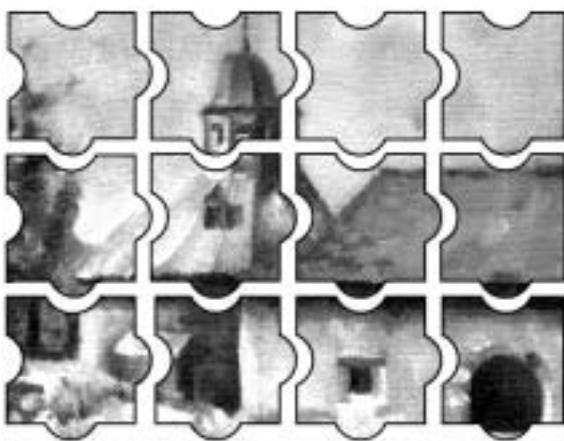


Рис. 7.5. Разбитое раstralное изображение можно "нарезать" на части

- изменение формы при помощи инструмента **Selection**. Данная операция не приводит к искажению изображения. Изменению подвергаются толь-

ко границы формы, изображение же будет либо срезаться, либо заполнять всю внутреннюю область наподобие мозаики (рис. 7.6);

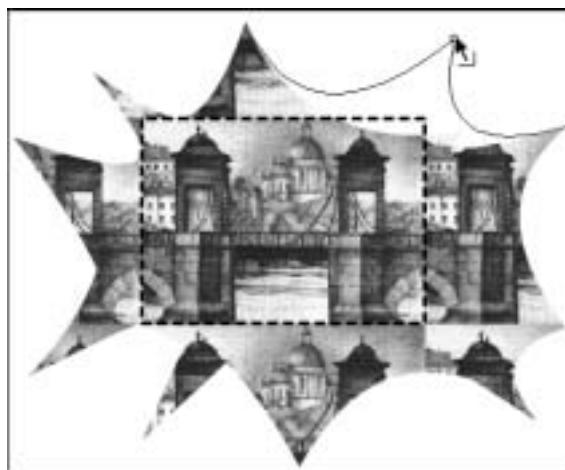


Рис. 7.6. Изменение формы растрового изображения при помощи инструмента **Selection**

- все виды трансформации — масштабирование, поворот, скос, зеркальное отражение, искажение, редактирование при помощи огибающих (рис. 7.7);

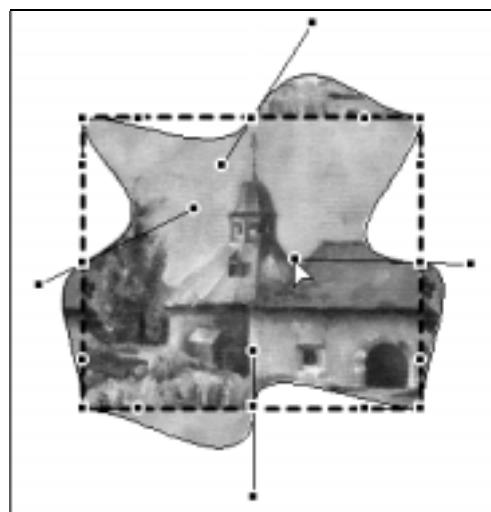


Рис. 7.7. Редактирование растрового изображения при помощи огибающих (**Envelope**)

- группировка с другими объектами рабочего или наложенного уровня. В результате данной операции образуется группа, содержащая векторную форму с раstralной заливкой, а не раstralное изображение наложенного уровня.

Работа с модификатором

Magic Wand

Модификатор инструмента **Lasso** (Лассо) **Magic Wand** (Волшебная палочка) заимствован из программ обработки раstralной графики и предназначен для работы с разбитыми раstralными изображениями (раstralными заливками). С его помощью можно выделять области изображения на основе близости их цветовых значений.

Для активизации волшебной палочки необходимо выделить инструмент **Lasso**. В разделе **Options** панели инструментов появляются два модификатора: **Magic Wand**  и **Magic Wand Properties** . Кнопка **Magic Wand Properties** позволяет задать параметры работы волшебной палочки. В результате ее нажатия появляется окно **Magic Wand Settings**, содержащее два элемента.

- Поле **Threshold** (Порог) позволяет задать числовое значение допуска, на основании которого в выделенную область будут попадать пиксель с большей или меньшей разницей цветовых значений. Если разница цветовых значений соседних пикселов *меньше* заданного порога, то они войдут в область выделения. Если разница цветовых значений соседних пикселов *больше* заданного порога, то они не будут включены в одну область выделения. Значение порога может быть задано в диапазоне от 0 до 200. Таким образом, при нулевом значении порога в выделенную область войдут соседние пиксель одного и того же цвета, а при максимальном значении порога будет выделен практически весь диапазон цветов.
- Список **Smoothing** (Сглаживание) позволяет задать, насколько сглаженными будут границы выделенной области.
- **Pixels** (Пиксели) — сглаживание отсутствует, область выделения с точностью до пикселя соответствует цветовой области. Граница области выделения имеет очень сложную форму и содержит множество ступеней.
 - **Rough** (Грубо) — сглаживание выполняется в незначительной степени.
 - **Normal** (Нормально) — нормальное сглаживание.
 - **Smooth** (Гладко) — границы выделенной области сильно сглажены.

Для применения волшебной палочки нужно включить модификатор **Magic Wand** и щелкнуть на предварительно разбитом изображении. Последовательно применяя этот инструмент к различным областям изображения, можно увеличивать область выделения. При использовании волшебной па-

лочки будут выделены только близкие по цветовым значениям смежные пиксели.

С помощью этого инструмента можно создавать интересные эффекты постерилизации (от англ. *poster* — плакат), т. е. замены сложных цветовых переходов локальными цветами. Для этого необходимо выделять соответствующие области изображения и назначать им в качестве заливки сплошной цвет (рис. 7.8).

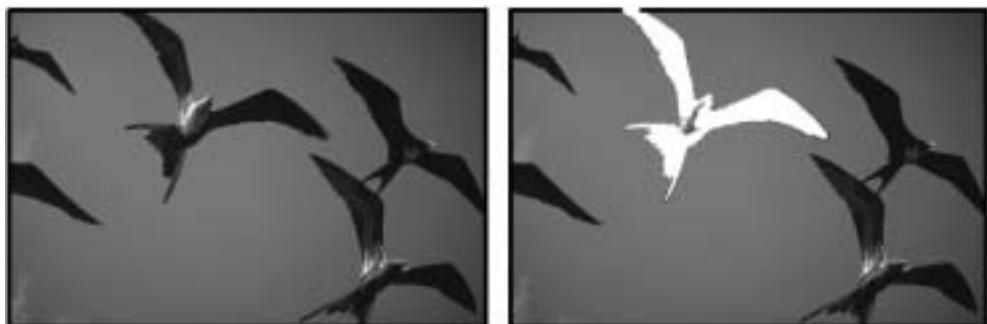


Рис. 7.8. Постерилизация растровых изображений

Выделяя фрагменты изображения и добавляя им обводки (контуры) при помощи инструмента **Ink Bottle** (Чернильница) можно добиться своеобразной имитации витража. Кроме того, при помощи инструмента **Magic Wand** (Волшебная палочка) можно удалить элементы изображения, например фон. Особенно хорошо эта процедура выполняется в случае однородного контрастного фона (рис. 7.9).



Рис. 7.9. Удаление фона растрового изображения при помощи инструмента **Magic Wand**

Использование и настройка растровой заливки

Для того чтобы переопределить существующую заливку или залить замкнутый контур растровой заливкой, применяют два способа. Первый заключается в использовании панели **Color Mixer**. Для заливки растром необходимо выполнить следующие действия:

1. Импортировать в библиотеку растровое изображение, которое предполагается использовать в качестве заливки (**File>Import>Import to Library**).
2. Открыть панель **Color Mixer** (**Window>Design Panels>Color Mixer**).
3. В поле **Fill style** выбрать тип заливки **Bitmap**. В нижней части панели **Color Mixer** будут отображены все имеющиеся в библиотеке растровые изображения (рис. 7.10).



Рис. 7.10. Панель **Color Mixer**
при выборе образца растровой заливки

4. На панели **Color Mixer** выбрать требуемое изображение. После этого в блоке управления цветом в качестве цвета заливки будет выведен образец данного изображения.
5. Активизировать любой инструмент, создающий векторные формы, и начать рисование или использовать инструмент **Paint Bucket** (Ведро Заливки) для заливки замкнутого контура. В результате этих действий заливка будет выполнена с узорным (Pattern) или мозаичным эффектом (рис. 7.11).

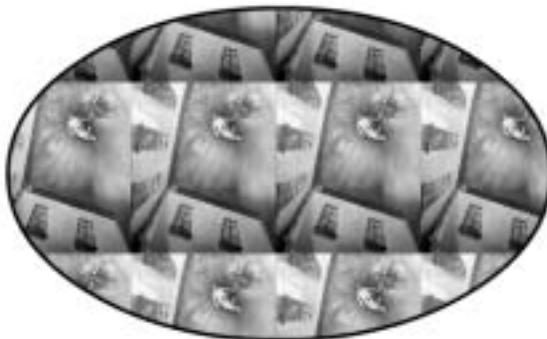


Рис. 7.11. Мозаичный эффект при использовании растровой заливки

Второй способ создания растровой заливки предполагает использование инструмента **Eyedropper** (Пипетка). В этом случае для заливки растром на сцене должно находиться разбитое растровое изображение, которое будет использовано в качестве образца заливки. При этом необходимо выполнить следующие действия:

1. Импортировать на сцену растровое изображение, которое предполагается использовать в качестве заливки (**File>Import>Import to Stage**).
2. Разбить экземпляр этого изображения (**Modify>Break Apart**).
3. Создать на сцене замкнутый контур.
4. Активизировать инструмент **Eyedropper** (Пипетка) и щелкнуть им по разбитому растровому изображению. При этом инструмент **Eyedropper** автоматически заменится на инструмент **Paint Bucket**.
5. Щелкнуть инструментом **Paint Bucket** внутри замкнутого контура или на векторной форме. Контур будет залит растром с мозаичным эффектом в режиме фиксации заливки (**Lock Fill**), поэтому размер ячейки изображения заливки будет совпадать с размером оригинала. При трансформации заливки, созданной в режиме фиксации, все действия будут также отражаться на разбитом оригинале, т. е. обе векторные формы будут залиты одной и той же заливкой. Связь между ними утрачивается при редактировании форм по отдельности.

Для настройки растровой заливки используется инструмент **Fill Transform** (Трансформация Заливки). С его помощью можно выполнять следующие операции редактирования:

- смещение заливки внутри формы;
- масштабирование (пропорциональное и непропорциональное);
- поворот;
- скос.

Для того чтобы трансформировать раstralную заливку, необходимо выполнить следующие действия:

1. Активизировать инструмент **Fill Transform** на панели инструментов.
2. Щелкнуть на векторной форме, содержащей раstralную заливку. Рядом с векторной формой появится ограничивающая рамка раstralной заливки, содержащая маркеры трансформации (рис. 7.12).

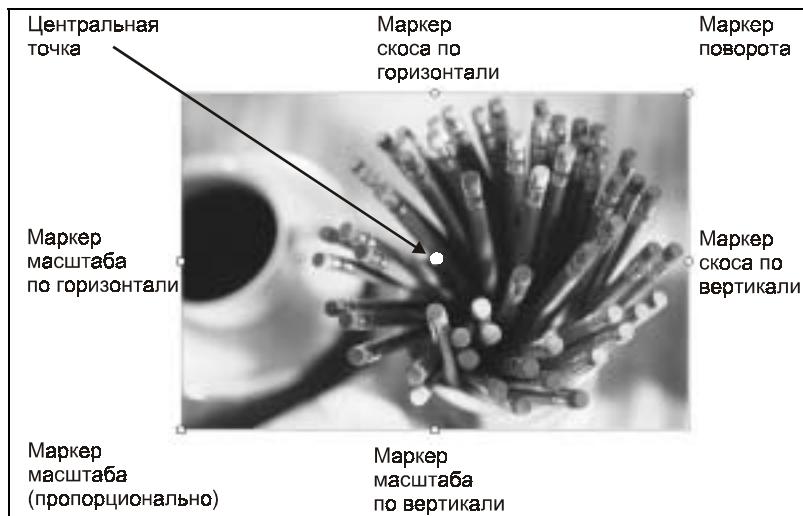


Рис. 7.12. Редактирование раstralной заливки

3. В случае, если имеет место мозаичный эффект, можно щелкнуть на любой ячейке изображения. Трансформация одной ячейки оказывает влияние на всю заливку. Если размеры ячеек очень малы, можно увеличить масштаб просмотра.
4. Для того чтобы сдвинуть заливку, необходимо переместить центральный круглый маркер ограничивающей рамки. При этом курсор принимает вид крестообразной стрелки.
5. Для пропорционального изменения масштаба используется угловой квадратный маркер. При наведении на него курсор принимает вид диагональной двунаправленной стрелки. Для выполнения непропорционального изменения масштаба необходимо воздействовать на квадратные маркеры, расположенные на серединах граней.
6. Для поворота раstralной заливки необходимо переместить круглый угловой маркер. При этом курсор принимает вид круговых стрелок.

7. Для выполнения скоса растровой заливки необходимо воздействовать на круглые боковые маркеры. При этом курсор принимает вид вертикальной или горизонтальной двунаправленной стрелки.

Автотрассировка и ручная трассировка растровых изображений

Трассировка (Tracing), или векторизация — это процедура, превращающая растровое изображение в набор векторных форм, с сохранением при этом в большей или меньшей степени его визуального содержания (рис. 7.13). Фактически трассировка — это действие, противоположное растирированию. Однако если растирование — это задача, имеющая точные технологические методы автоматической реализации, то трассировка далеко не всегда может дать удовлетворительный результат.



Рис. 7.13. Трассированное изображение

Лучше всего трассировке поддаются изображения, содержащие четкие формы, окрашенные локальными цветами. Идеально трассируются черно-белые (однобитные) изображения. Чем больше сложных цветовых переходов и мелких деталей содержит изображение, тем хуже оно подвергается трассировке.

Flash позволяет выполнять автоматическую трассировку растровых изображений в рабочей среде. Для того чтобы выполнить эту операцию, необходимо поместить на сцену экземпляр растрового изображения (не разбивая его), выделить его и выполнить команду меню **Modify>Bitmap>Trace Bitmap**. Появившееся окно **Trace Bitmap** предлагает задать следующие параметры трассировки (рис. 7.14).

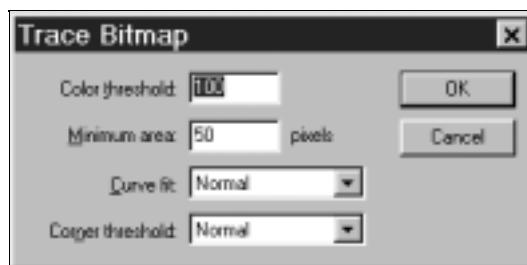


Рис. 7.14. Окно настройки параметров трассировки **Trace Bitmap**

- **Color threshold** (Цветовой порог) — данное значение указывает, какова должна быть максимальная разница цветовых значений соседних пикселов для того, чтобы в результирующем векторном изображении они были окрашены одним и тем же цветом и помещены в одну векторную форму. Если разница цветовых значений соседних пикселов *меньше* значения цветового порога, они будут окрашены одним цветом и помещены в одну векторную форму. Если разница цветовых значений соседних пикселов *больше* цветового порога, они будут окрашены разным цветом и, соответственно, помещены в разные векторные формы. Чем выше значение данного параметра, тем *меньше* цветов (и соответственно векторных форм) будет в результирующем изображении. Допустимые значения цветового порога лежат в диапазоне от 1 до 500. Для того чтобы получить векторное изображение, наиболее точно соответствующее растровому оригиналу, необходимо установить малое значение цветового порога. Следует также иметь в виду, что при задании небольшого значения данного параметра число векторных форм в результирующем изображении может быть очень велико, что приведет к существенному увеличению объема файла и времени выполнения трассировки (при неизменных других параметрах);

- Minimum area** (Минимальная площадь) — данный параметр указывает, какое количество смежных пикселов принимается во внимание при назначении цвета каждому пикселу. Данный параметр влияет на детализацию результирующего векторного изображения. Чем меньше значение **Minimum area**, тем более мелкими могут быть составляющие его векторные формы и тем, соответственно, выше степень детализации. Допустимые значения минимальной площади лежат в диапазоне от 1 до 1000.
- Curve fit** (Подгонка кривых) — данный параметр определяет, насколько гладкими или, наоборот, прямолинейными будут границы векторных форм в результирующем изображении. Список содержит следующие значения.
 - **Pixels** (Пиксели) — границы векторных форм соответствуют границам с точностью до 1 пикселя. В результате получаются ступенчатые границы с большим числом опорных точек. Данное значение позволяет получить наиболее точное соответствие оригиналу, однако приводит к наибольшему увеличению размера результирующего файла и времени выполнения трассировки (при неизменных прочих параметрах).
 - **Very tight** (Очень плотно) — векторные формы достаточно точно соответствуют границам объектов изображения.
 - **Tight** (Плотно) — векторные формы достаточно точно соответствуют границам объектов изображения, однако имеет место несущественное сглаживание.
 - **Normal** (Нормально) — компромиссное значение между точностью подгонки и сглаживанием кривых.
 - **Smooth** (Гладко) — сглаживание превалирует над точностью подгонки.
 - **Very smooth** (Очень гладко) — максимальная степень сглаживания. Результирующие векторные формы достаточно произвольно воспроизводят оригинал. При этом объем файла и время выполнения трассировки минимальны (при неизменных прочих параметрах).
- Corner threshold** (Угловой порог) — данный параметр определяет наличие углов в результирующих векторных формах. Список содержит следующие значения.
 - **Many corners** (Много углов) — результирующие векторные формы будут содержать большое количество углов. Это позволяет добиться большей детализации, делая границы форм более сложными, и, соответственно, увеличивает конечный объем файла.
 - **Normal** (Нормально) — компромиссное значение между большим количеством и сглаживанием углов.
 - **Few corners** (Небольшое число углов) — углы сглаживаются. Это приводит к потере детализации и, соответственно, к уменьшению конечного объема файла.

Как следует из вышесказанного, слишком высокие параметры детализации приводят к тому, что в результате трассировки образуется огромное количество векторных форм, в свою очередь содержащих множество опорных точек. Это приводит к тому, что такое изображение занимает очень большой объем, что делает его неприемлемым для размещения в Интернете. Кроме того, обработка такого изображения требует значительных ресурсов, что снижает общее быстродействие.

В некоторых случаях для получения качественного результата значительно эффективнее применять ручную трассировку. Ручная трассировка предполагает создание векторных форм, передающих содержание растрового оригинала, при помощи инструментов рисования (**Pen**, **Pencil**, **Line**, **Brush**). Это процесс значительно более трудоемкий и кропотливый, но усердие вознаграждается получением живого и интересного векторного рисунка.

Наиболее рациональный способ ручной трассировки в среде Flash связан с использованием палитры слоев (*подробная информация о работе со слоями содержится в гл. 9*) и включает в себя следующие этапы.

1. Два раза щелкнуть на названии текущего слоя в палитре слоев и переименовать его, присвоив имя "**Bitmap**".
2. На слой **Bitmap** поместить экземпляр растрового изображения, которое предполагается трассировать, вытащив его из библиотеки.
3. Два раза щелкнуть на пиктограмме слоя **Bitmap** и в появившемся диалоговом окне **Layer Properties** (Свойства слоя) выбрать тип **Guide** (Направляющий слой) для того, чтобы само растровое изображение не было видно в конечном ролике в процессе тестирования.
4. Заблокировать слой **Bitmap**, щелкнув в палитре слоев по расположенной на нем точке, соответствующей пиктограмме с изображением замка (на слое появится замок). Это позволит избежать случайного смещения изображения.
5. В палитре слоев создать еще один слой, щелкнув на пиктограмме, расположенной в нижней части палитры слоев слева .
6. Два раза щелкнуть на названии нового слоя в палитре слоев и переименовать его, присвоив имя "**Vector**".
7. При помощи инструментов **Pen** (Перо) или **Pencil** (Карандаш) контрастным цветом вручную последовательно обвести формы оригинала. Необходимо следить за тем, чтобы контуры, предназначенные для заливки, были замкнуты.
8. Для заливки контура цветом оригинала можно воспользоваться возможностью подобрать цвет с любой точки монитора, описанной в *прим. к разд. "Инструмент Eyedropper" гл. 4*.

9. Для того чтобы сделать невидимым растровое изображение, можно отключить отображение слоя **Bitmap**, щелкнув в палитре слоев по расположенной на нем точке, соответствующей пиктограмме с изображением глаза, в результате чего слой будет перечеркнут. Повторный щелчок вновь включит отображение слоя.
10. По окончании трассировки слой **Bitmap** можно удалить, выделив его и щелкнув на пиктограмме с изображением мусорной корзины в нижней части палитры слоев.
11. Вышеизложенная информация обобщена в табл. 7.1.

Таблица 7.1. Операции с растровой графикой

Операция	Способ
Наложенный уровень	Импорт Команда File>Import – выбрать один файл или последовательность
	Оптимизация Дважды щелкнуть на пиктограмме растрового символа в окне библиотеки
	Копирование 1. Вытащить картинку из библиотеки на сцену. 2. Edit>Duplicate . 3. <Ctrl>+<D> . 4. Тащить с <Ctrl> или команда <Alt> . 5. Через буфер обмена
	Удаление 1. Команда Edit>Clear . 2. <Delete> . 3. Удалить из библиотеки растровое изображение
	Трансформация Инструмент Free Transform (масштаб, поворот, скос, зеркальное отражение)
	Трассировка Команда Modify>Bitmap>Trace Bitmap создает векторные формы рабочего уровня
	Замена в фильме Команда Modify>Bitmap>Swap Bitmap – замена одной растровой картинки в фильме на другую
	Анимация Тип анимации движения Motion Tween
	Разбиение (переход к рабочему уровню) Команда Modify>Break Apart , <Ctrl>+
Рабочий уровень	Выделение <i>Целиком</i> – щелчок инструментом Selection , <i>частично</i> – рамкой инструмента Selection , <i>часть произвольной формы</i> – инструментом Lasso , <i>области одного цвета</i> – модификатор Magic Wand инструмента Lasso
	Изменение формы Инструменты Selection , Free Transform (модификаторы Envelope , Distort), Subselection , Pen , Eraser

Таблица 7.1 (окончание)

Операция	Способ
Рабочий уровень	Заливка растром
	Палитра Color Mixer , инструмент Eyedropper , инструмент Paint Bucket (для заливки всех форм одним растром включить режим Lock Fill), инструмент Brush
	Редактирование заливки
	Инструмент Fill Transform
Анимация	Тип анимации формы Shape Tween
Группировка	Команда Modify>Group — <Ctrl>+<G> — переход к наложенному уровню

Импорт и работа с видеофайлами

Видеофайл представляет собой последовательность статических растровых изображений, синхронизированных со звуковым сопровождением, которая воспроизводится с определенной скоростью. Flash MX 2004 позволяет импортировать в рабочую среду видеофайлы различных форматов. Кроме того, существует возможность связать видеофайл с Flash-фильмом, не внедряя его в документ. Язык сценариев ActionScript позволяет осуществлять динамическую загрузку видеофайла и управление его воспроизведением.

В рабочую среду Flash могут быть импортированы следующие форматы видеофайлов (в скобках указаны расширения файлов):

- Audio Video Interleaved (avi) — при наличии QuickTime 4 и выше или DirectX 7 и выше;
- Digital video (dv) — при наличии QuickTime 4 и выше;
- Motion Picture Experts Group (mpg, mpeg) — при наличии QuickTime 4 и выше или DirectX 7 и выше;
- QuickTime movie (mov) — при наличии QuickTime 4 и выше;
- Windows Media file (wmv, asf) — при наличии DirectX 7 и выше;
- Macromedia Flash Video (FLV) (flv).

Импорт и экспорт файлов осуществляется при помощи *кодека* (кодера-декодера) Sorenson-Spark codec, позволяющего внедрять видеофайлы непосредственно в Flash-документ. Декодер Sorenson-Spark встроен в Flash Player и позволяет просматривать содержимое сжатого видеофайла, встроенного в ролик.

Существуют два типа сжатия видеофайлов: *временной* (temporal) и *пространственный* (spatial).

При использовании временного сжатия обнаруживаются различия в кадрах и обрабатывается только та информация, которая изменяется от кадра к кадру. Области кадра, не содержащие изменений, просто повторяются. Таким образом, каждый последующий кадр описывается на основе его отличия от предыдущего.

При использовании пространственного сжатия, напротив, обрабатывается каждый кадр вне зависимости от остальных. Данный метод допускает *сжатие без потерь* (lossless), при котором вся информация о содержимом каждого кадра сохраняется, или *сжатие с потерями* (lossy), при котором часть информации отбрасывается.

Кодек Sorenson-Spark использует временное сжатие, в ходе которого всякий раз, как последующий кадр существенно отличается от предыдущего, создаются ключевые кадры (keyframes). Первый кадр последовательности всегда является ключевым.

Внедрение видеофайлов в Flash-документ

Импорт видеофайлов осуществляется при помощи команды главного меню **File>Import>Import to Stage** или **File>Import>Import to Library**. При импорте видеофайл помещается в библиотеку, а на сцене находится его экземпляр (instance).

Процедура импорта осуществляется при помощи Помощника (Wisard), позволяющего отредактировать видеоролик и задать параметры кодирования.

После выполнения команды импорта на экране появляется диалоговое окно, предлагающее следующий выбор действий:

- Import the Entire Video** (Импортировать весь ролик целиком) — в результате выбора данного пункта весь видеофайл будет целиком помещен в библиотеку;
- Edit the Video First** (Отредактировать ролик) — выбор данного пункта позволяет задать точку начала и конца ролика и разбить его на несколько отдельных клипов.

При импорте некоторых файлов Flash не поддерживает импорт звуковой дорожки, в этом случае выводится соответствующее предупреждение. Также в некоторых случаях кодек импортируемого файла не позволяет редактировать ролик. В этой ситуации опция **Edit the Video First** недоступна.

При выборе опции **Edit the Video First** появляется окно редактирования ролика (рис. 7.15). Здесь можно выполнить следующие операции.

- Playback** (Воспроизведение).

- Для начала воспроизведения можно нажать кнопку **Play from current position** или кнопку **Preview clip**. Кнопка **Stop playing** позволяет остановить воспроизведение.

- Для перемещения с дискретностью в один кадр можно воспользоваться клавишами **Step back one frame** или **Advance by one frame**.
 - Перемещая воспроизведяющую головку (желтый треугольный указатель) курсором, можно просмотреть содержимое ролика вручную.
- Для установки точек начала и окончания ролика можно выполнить одно из следующих действий:
- переместить маркеры начала (**in point**) и конца (**out point**) ролика вручную;
 - установить воспроизведяющую головку в начальную точку и нажать кнопку **Set in point to current position**, после чего переместить воспроизведяющую головку в конечную точку и нажать кнопку **Set out point to current position**.
- Создание клипов.
- Для того чтобы создать клип, нужно установить его начальную и конечную точки и нажать кнопку **Create clip**. Созданный таким образом клип отобразится в поле, расположенном в левой части окна. Для создания нескольких клипов необходимо повторять эту процедуру, каждый раз устанавливая точки начала и конца нового клипа. В результате импорта клипы будут помещены в библиотеку как самостоятельные объекты.
 - Для того чтобы переименовать клип, нужно два раза щелкнуть на его названии и ввести новое имя.
 - Для того чтобы удалить клип, нужно выделить его название и щелкнуть на пиктограмме мусорного бачка.
 - Для изменения порядка следования клипов можно воспользоваться стрелками **Move clip up** и **Move clip down**.
 - Для того чтобы отредактировать клип, необходимо выделить его, затем изменить положение точек начала и конца, после чего нажать кнопку **Update clip**.
 - Для того чтобы объединить все клипы в один клип, нужно установить флагок **Combine List of Clips into a Single Library Item After Import**. Клипы будут объединены в один клип в том порядке, в каком они расположены в окне. В результате этого действия в библиотеку будет помещен один объект.
- По окончании редактирования необходимо нажать кнопку **Next**. Появившаяся вслед за нажатием панель позволяет осуществить следующие настройки.
- В поле **Compression profile** можно выбрать один из стандартных профилей сжатия. В списке представлены различные варианты, учитывающие раз-

ную скорость соединения от 56 kbs с использованием модема до 786 kbs для DSL или волоконно-оптической связи.



Рис. 7.15. Окно редактирования видеоролика

□ Кнопка **Edit** позволяет задать параметры настройки.

- **Bandwidth** — примерная скорость загрузки видео в килобайтах в секунду (kbs). Чем ниже значение параметра **Bandwidth**, тем большая степень сжатия будет применена к изображению и, соответственно, тем хуже будет его качество. Качество отдельных кадров может колебаться, для того чтобы обеспечить заданную скорость потока.
- **Quality** — альтернатива параметра **Bandwidth**. Позволяет задать качество видеоролика в диапазоне от 0 до 100. Данный параметр задает степень сжатия для всех кадров, но игнорирует скорость потока.
- **Keyframes** — диапазон между ключевыми кадрами. Чем ниже это значение, тем меньше интервал между ключевыми кадрами и тем, соответственно, больше объем файла.

- **High quality keyframes** — обеспечивает качественное отображение ключевых кадров. Эта опция недоступна при использовании параметра **Quality**.
- **Quick compress** — позволяет ускорить процедуру сжатия.
- **Synchronize to Macromedia Flash document frame rate** — синхронизировать скорость видео со скоростью Flash-документа. Этот параметр позволяет избежать пропуска или двойного показа кадров в случае, когда скорость видео не совпадает с текущей скоростью монтажной линейки Flash. Так, если скорость видео меньше, чем скорость монтажной линейки, то для того, чтобы кадры видео не пропускались, эту опцию нужно отключить. Если в данной ситуации разрешить синхронизацию кадров, то часть кадров видеоролика будет пропускаться, чтобы "успевать" за монтажной линейкой.
- **Number of video frames to encode per number of Macromedia Flash frames** — соотношение между кадрами видео и кадрами монтажной линейки. Если, например, 2 кадра видео кодировать на каждые 3 кадра монтажной линейки (2:3), то в единицу времени будет показано меньшее количество кадров видео. Это приведет к прерывистому воспроизведению, т. е. движение будет происходить толчками.

Раскрывающийся список **Advanced settings** позволяет задать дополнительные настройки для видео. Для того чтобы это сделать, необходимо создать новый профиль, выполнив команду **Create new profile**. Появившаяся в результате выполнения команды панель представлена на рис. 7.16. В правой части окна имеется окно предварительного просмотра, где можно оценить все производимые изменения. Левая часть панели содержит следующие параметры.

- Color** — настройки цветовой и тоновой коррекции.
 - **Hue** — оттенок.
 - **Saturation** — насыщенность.
 - **Gamma** — гамма.
 - **Brightness** — яркость.
 - **Contrast** — контраст.
- Dimensions** — задание размеров и кадрирование.
 - **Scale** — масштаб в процентах.
 - **Width** — ширина в пикселях.
 - **Height** — высота в пикселях.
 - **Crop** — параметры кадрирования. При помощи четырех полей задаются размеры и координаты рамки кадрирования.



Рис. 7.16. Окно настройки дополнительных параметров редактирования видеоролика

□ **Track options** — параметры дорожек.

- **Import into** — параметры размещения импортированного видео.
 - ◊ **Current timeline** — видеоролик импортируется на монтажную линейку основного фильма. При этом если на ней недостаточно кадров, чтобы вместить весь диапазон, будет выведено сообщение с предложением автоматически добавить кадры.
 - ◊ **Movie clip** — видеоролик импортируется на монтажную линейку символа типа **Movie Clip**. Символ будет создан автоматически.
 - ◊ **Graphic symbol** — видеоролик импортируется на монтажную линейку символа типа **Graphic**. Символ будет создан автоматически.
- **Audio track** — звуковая дорожка.
 - ◊ **Separate** — как отдельный файл. В результате импорта в библиотеку будут добавлены видеоролик и звуковой файл.

- ◊ **Integrated** — включена в видеоролик.
- ◊ **None** — импортировать без звука.

После задания всех настроек необходимо нажать кнопку **Next**. В появившемся окне в поле **Name** можно ввести название созданного профиля, а в поле **Description** его краткое описание.

Далее необходимо снова нажать кнопку **Next** и следом кнопку **Finish**. Видеофайл будет импортирован и внедрен в рабочую среду Flash.

Файл Macromedia Flash Video (FLV) может быть импортирован и внедрен в документ без дополнительной настройки параметров импорта.

В результате импорта видеоролик помещается в библиотеку, а на сцене используются его экземпляры (instances). В одном фильме можно использовать несколько экземпляров видеоролика. Этот принцип аналогичен принципу использования экземпляров растрового изображения.

Экземпляр видеофайла привязывается к кадрам монтажной линейки, на которую он был помещен при импорте или вручную. Видеоролик будет виден только в тех кадрах, в которых он физически размещен. Диапазон кадров монтажной линейки, содержащих видеоролик, включает в себя один ключевой кадр (всегда первый) и необходимое количество простых кадров. Видеоролик размещен в первом ключевом кадре. Простые кадры сами по себе не содержат никакой новой информации, а используются для заполнения, т. е. для того, чтобы видеоролик был виден в течение необходимого промежутка времени (*подробная информация о кадрах монтажной линейки и работе с ними содержится в гл. 9*). Для того чтобы видеоролик был виден полностью, после содержащего его ключевого кадра необходимо добавить интервал простых кадров, по длительности равный времени воспроизведения ролика.

Примечание

Если при импорте или размещении экземпляра видеофайла при помощи библиотеки на монтажной линейке основного фильма или символа не имеется достаточно большого диапазона кадров, чтобы вместить весь видеоролик, Flash выводит предупреждающее сообщение и предлагает автоматически добавить необходимые кадры. Если кадры не будут добавлены, то на сцене и в конечном фильме можно будет увидеть только ту часть видеоролика, которая соответствует длительности диапазона кадров, в который он помещен.

В среде Flash внедренный видеофайл представляет собой единый контейнер, содержащий видеоролик и, в общем случае, звуковую дорожку. Содержимое импортированного видеофайла редактировать в рабочей среде невозможно.

Импорт связанных видеофайлов QuickTime movie

В качестве одной из альтернатив встраиванию видеофайлов в документ Flash можно использовать импортирование видеоролика в формате QuickTime

movie (с расширением mov) как самостоятельного файла, связав его с документом. Для того чтобы это сделать, необходимо выполнить команду главного меню **File>Import>Import to Stage** или **File>Import>Import to Library**. В появившемся вслед за этим окне Мастера импорта установить переключатель в положение **Link to external video file**. В результате в библиотеке появляется ссылка на внешний файл, а на сцене отображается экземпляр видеоролика.

При использовании связанных видеофайлов Flash-ролик должен быть опубликован только в формате QuickTime movie (*подробная информация о настройках публикации содержится в гл. 17*). На сегодняшний день формат QuickTime movie поддерживает только файлы формата SWF версии не выше Flash 4. Поэтому для того чтобы гарантировать корректную работу Flash-фильма со связанным видеороликом QuickTime, необходимо использовать только те возможности, которые доступны проигрывателю Flash Player 4.

В противном случае будет выведено сообщение о том, что текущая версия QuickTime не поддерживает данную версию Flash Player.

Работа с видеороликом как с объектом наложенного уровня

Экземпляр видеоролика является объектом наложенного уровня. К нему можно применять любые операции, доступные на наложенном уровне. К их числу относятся следующие операции.

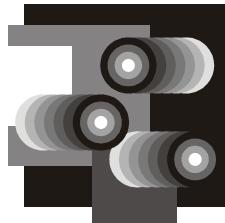
- Перемещение при помощи инструмента **Selection** или управляющих клавиш.
- Копирование или дублирование. Выполнение этих операций эквивалентно вытаскиванию нового экземпляра видеоролика из библиотеки на сцену.
- Все виды трансформаций, доступных на наложенном уровне (масштабирование, вращение, скос, зеркальное отражение).
- Изменение порядка следования в стопке объектов наложенного уровня (меню **Modify>Arrange**).
- Группировка с другими объектами рабочего или наложенного уровня, а также с другими видеороликами.
- Замена экземпляра одного видеоролика экземпляром другого. В результате замены к новому экземпляру будут применены все параметры трансформации, заданные для заменяемого видеоролика. Для выполнения этой процедуры нужно, выделив соответствующий экземпляр, нажать кнопку **Swap** в Инспекторе свойств и в появившемся окне **Swap Embedded Video** (если видеофайл внедрен в Flash-документ) или **Swap Linked Video** (если видеофайл связан с Flash-документом) выбрать видеоролик, экземпляр которого заменит текущий экземпляр. Экземпляр вне-

дренного видеоролика можно заменить только экземпляром другого внешнего видеоролика, а экземпляр связанного — только экземпляром связанного.

- Изменение свойств видеоролика при помощи библиотеки. Для выполнения этих действий необходимо выделить объект в библиотеке и из ее контекстного меню выбрать команду **Properties**. В качестве альтернативы можно воспользоваться одноименной кнопкой в нижней части окна библиотеки или два раза щелкнуть на пиктограмме рядом с названием видеоролика. Также можно два раза щелкнуть на изображении видеоролика в окне просмотра библиотеки. Любое из этих действий приведет к появлению диалогового окна **Embedded Video Properties**. Данное окно предлагает следующие возможности.
- **Изменение имени объекта библиотеки** — для этого необходимо ввести новое имя в поле ввода в верхней части окна.
 - **Update** — загружает последнюю версию видеофайла.
 - **Import** — данная команда позволяет заменить текущий видеофайл FLV-файлом, указав его местоположение на диске.
 - **Export** — данная команда позволяет экспорттировать текущий видеоролик в формате FLV.

При работе со связанным файлом Quick Time Movie диалоговое окно **Linked Video Properties** предлагает только возможность переименования объекта и указания пути к внешнему файлу при помощи клавиши **Set Path**.

- Задание имени экземпляра для программного управления видеороликом при помощи сценария ActionScript. Для того чтобы задать имя экземпляра, его необходимо выделить и ввести имя в поле **Instance Name** Инспектора свойств.



Глава 8

Работа с текстом

Знаки, которые они рассматривали в течение месяца,
вовсе не были планами строительных сооружений,
это были сильно увеличенные буквы древнейших азбук,
начиная с иероглифов и кончая греческой скорописью и глаголицей.

Милорад Павич, "Два студента из Ирака"

Текст в Flash

Текст является важнейшим элементом мультимедийного проекта. В Flash текст может использоваться для выполнения следующих функций:

- отображение статической текстовой информации в проекте;
- динамический вывод текстовой информации в ходе воспроизведения фильма;
- ввод текстовой информации пользователем в ходе воспроизведения фильма;
- организация взаимодействия с пользователем (вывод сообщений в качестве реакции на те или иные действия, обработка информации, введенной пользователем, и т. д.);
- использование текста как элемента дизайна (шрифтовые композиции, логотипы и т. д.);
- анимация текста.

Flash MX 2004 поддерживает стандарт кодирования текста Unicode. Данный стандарт, разработанный ассоциацией Юникод в 1991 году, позволяет охватить символы большинства мировых языков, включая знаки пунктуации, математические и технические символы и т. д. Проигрыватель Flash Player 7 поддерживает 8-битную и 16-битные (UTF-16 BE, Big Endian и UTF-16 LE, Little Endian) кодировки Unicode. Формат UTF-8 использует для кодирования символов последовательности байтов различной длины, которые преоб-

разуются в коды Unicode. UTF-16 — это 16-битная кодировка, которая кодирует каждый символ, используя 2-байтную последовательность. При разработке Flash-проектов рекомендуется использовать формат кодирования UTF-8. Формат Unicode позволяет отображать текст на различных языках, при условии, что в системе установлены соответствующие шрифты.

Создание и форматирование текста

В рабочей среде Flash создаваемый текст помещен в текстовый блок, представляющий собой прямоугольную рамку. Текстовый блок — это объект *написанного уровня*. Текст может быть использован для различных целей: от создания всевозможных заголовков, надписей и декоративных элементов до использования текстовых блоков в качестве полей ввода, обрабатываемых программно средствами языка сценариев ActionScript. В связи с этим Flash предлагает различные типы текста, обладающие разной степенью интерактивности.

Инструмент **Text**. Виды текстовых блоков

Для создания текстового блока используется инструмент **Text** (Текст)  . Текстовый блок, представляет собой контейнер, содержащий текст, который можно редактировать, изменять его тип и параметры форматирования. Текстовые блоки бывают двух видов:

- текстовый блок произвольной длины (текстовая метка, заголовочный текст);
- текстовый блок фиксированной длины.

Характерной особенностью текстового блока произвольной длины является то, что текст вводится в одну строку, длина которой практически не ограничена. Переход на новую строку происходит только в результате нажатия клавиши <Enter>. Для создания текстового блока произвольной длины необходимо выбрать инструмент **Text** и дважды щелкнуть им по сцене. Появившийся в результате текстовый блок будет иметь минимальную длину, однако по мере ввода в него текста длина текстового блока будет увеличиваться. В правом верхнем углу текстового блока произвольной длины находится *круглый маркер*.

Текстовый блок фиксированной длины, напротив, позволяет вводить текст только в строго ограниченную по длине область. Когда курсор достигает правой границы текстового блока, происходит автоматический перенос строки. Для создания текстового блока фиксированной длины необходимо активизировать инструмент **Text**, щелкнуть им по сцене и, не отпуская

кнопку мыши, протянуть курсор, задав необходимый размер при помощи появляющейся рамки. По мере ввода текста высота текстового блока может изменяться в результате переноса строки, однако длина остается фиксированной. В правом верхнем углу текстового блока произвольной длины находится *квадратный* маркер.

Рассмотренные типы текстовых блоков легко трансформируются один в другой. Для того чтобы превратить текстовый блок произвольной длины в текстовый блок фиксированной длины, необходимо взяться за круглый маркер в правом верхнем углу и протянуть его, установив необходимый размер. При этом будет выполнена автоматическая разбивка на строки. Для выполнения обратного преобразования нужно дважды щелкнуть на квадратном маркере в правом верхнем углу текстового блока фиксированной длины. В результате данной операции все автоматические переносы строки будут отменены, текст вытянется в одну строку. Разбивка на строки сохранится только в том случае, если при наборе текста использовалась клавиша <Enter>.

Текстовый блок, создаваемый в рабочей среде, не имеет фона и границы (за исключением динамического или пользовательского текста, который может быть снабжен белым фоном и черной рамкой), т. е. в конечном фильме зритель увидит только заключенный внутри него текст. Для того чтобы дополнительно выделить текстовый блок цветом или создать вокруг него рамку, можно разместить под ним цветной прямоугольник или любую другую векторную форму.

Характерные особенности текстовых блоков произвольной и фиксированной ширины обобщены в табл. 8.1.

Таблица 8.1. Виды текстовых блоков

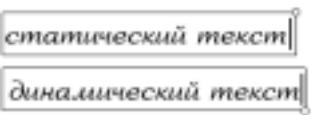
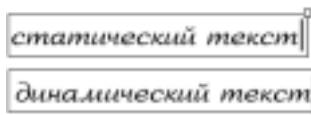
Параметры	Блок произвольной длины	Блок фиксированной длины
Маркеры	Круглой формы	Квадратной формы
		
Ввод текста	Инструментом Text щелкнуть в любом месте рабочей области и ввести текст	Инструментом Text нарисовать рамку нужной длины в любом месте рабочей области и ввести текст
Длина строки	Практически не ограничена, для перевода строки необходимо использовать клавишу <Enter>	Ограничена шириной рамки, при достижении границы рамки текст автоматически переходит на новую строку без разрыва абзаца

Таблица 8.1 (окончание)

Параметры	Блок произвольной длины	Блок фиксированной длины
Переход к следующему абзацу	По нажатию клавиши <Enter>	По нажатию клавиши <Enter>
Смена вида текстового блока	Перетащите круглый маркер блока в сторону	Два щелчка на квадратном маркере блока

Импорт текста в Flash

Текст может быть импортирован из внешнего приложения в рабочую среду Flash через буфер обмена. Для выполнения этой процедуры используются стандартные команды **Copy** (Копировать) и **Paste** (Вставить). При импорте текста, содержащего несколько строк, каждая строка помещается в отдельный текстовый блок. При импорте текста из внешнего текстового редактора (например, Microsoft Word 2000) каждая строка помещается в отдельный текстовый блок, который, в свою очередь, группируется. Некоторые параметры форматирования, не свойственные Flash, могут быть утрачены.

Типы текста и настройка его параметров

Во Flash существуют три различных типа текста, позволяющие решать задачи, связанные с художественным содержанием, информационным наполнением и обеспечением интерактивности проекта. К их числу относятся:

- Static** — статический текст (текстовые надписи);
- Dynamic** — динамический текст;
- Input** — пользовательский текст (текст ввода, редактируемый текст).

Текст любого типа размещается внутри текстового блока и наряду с общими для всех типов текста атрибутами содержит характерные только для него параметры редактирования.

Для указания типа текстового блока и настройки параметров его форматирования можно использовать Инспектор свойств или команды главного меню **Text**.

При выделении текстового блока инструментом **Selection** он заключается в прямоугольную рамку, а Инспектор свойств отображает все параметры настройки и форматирования соответствующего типа текста. В этом случае заданные изменения будут применены ко всему содержимому текстового блока.

Для того чтобы отформатировать только часть содержимого текстового блока, необходимо войти в режим его редактирования. Для этого можно либо

один раз щелкнуть на текстовом блоке инструментом **Text** либо дважды щелкнуть на нем инструментом **Selection**. При входе в режим редактирования текстового блока курсор примет вид вертикальной черты. Форматирование применяется только к выделенным фрагментам текста. Выделение может быть выполнено следующим образом.

- Для выделения произвольного фрагмента текста необходимо, удерживая кнопку мыши, протащить курсор через текстовый фрагмент. Для выделения диапазона текста нужно разместить курсор в начале диапазона, а затем щелкнуть в конце диапазона, удерживая клавишу <Shift>. Аналогичным образом можно снять выделение с части текстового фрагмента. Выделяя отдельные символы и форматируя их отдельно от остального текста, можно создать визуальные акценты внутри одного и того же текстового блока, например, используя различный шрифт, цветные буквы или варьируя их размер и начертание (рис. 8.1).
- Для того чтобы выделить одно слово, можно дважды щелкнуть на нем.
- Для выделения всего фрагмента можно выполнить команду **Edit>Select All** или воспользоваться сочетанием клавиш <Ctrl>+<A>.



Рис. 8.1. Пример форматированного текстового блока

Статический текст

Основная особенность данного типа текста состоит в том, что содержимое статического (**Static**) текстового блока не изменяется в процессе воспроизведения фильма. Информационное наполнение и форматирование статического текста осуществляется исключительно в рабочей среде Flash на стадии разработки проекта. Текст типа **Static** может быть использован для создания всевозможных надписей, вывода информации, которая не должна динамически обновляться или корректироваться, и, наконец, просто как декоративный элемент композиции. Статический текст является *графическим* элементом и допускает возможность анимации при помощи монтажной линейки.

Задать параметры форматирования текста можно либо до создания текстового блока, активизировав инструмент **Text**, либо отформатировать уже готовый текст.

Для того чтобы задать параметры форматирования шрифта и свойства статичного текста, необходимо выделить соответствующий текстовый блок любым из указанных выше способов. При этом Инспектор свойств принимает вид, представленный на рис. 8.2.

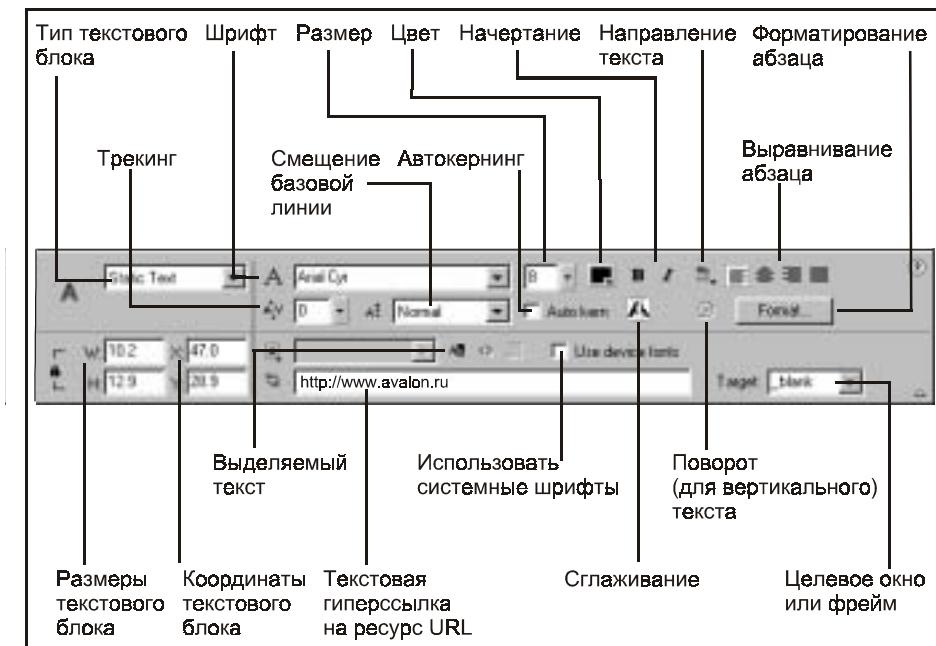


Рис. 8.2. Форматирование статического текста.
Инспектор свойств

Инспектор свойств позволяет задать следующие параметры:

- поле **Text Type** (Тип текста) — данный раскрывающийся список позволяет выбрать тип текста (статический, динамический или пользовательский);
- **Width, Height, X-location, Y-location** (Высота, Ширина, X-координата, Y-координата) — при помощи этих полей можно задать размеры и координаты текстового блока. Следует иметь в виду, что изменение размеров приводит к трансформации текстового блока как объекта наложенного уровня вместе с его содержимым. В результате масштабирования пропорции шрифта могут быть искажены. Масштабирование текстового блока не приводит к изменению значения параметра **Font Size** (Кегль), изменяется только размер текстового блока (вместе со шрифтом). Для того чтобы отменить все произведенные с текстовым блоком трансформации,

нужно выполнить команду **Modify>Transform>Remove transform** или нажать кнопку **Reset** панели **Transform**;

- **Font** (Шрифт) — раскрывающийся список, позволяющий выбрать любой из установленных в системе шрифтов;
- **Font Size** (Кегль) — размер символов шрифта, задаваемый в пунктах (один пункт равен 0,35 мм). Допустимые значения лежат в диапазоне от 1 до 2500 пунктов. В некоторых случаях шрифт определенного размера в Flash выглядит мельче, чем шрифт, набранный тем же кеглем в другом приложении;
- **Font Color** (Цвет текста) — щелчок на цветовом образце позволяет задать для текста любой цвет, содержащийся в текущем каталоге цветов. Текст может быть окрашен только сплошным цветом (*solid*). Этот цвет может содержать любую степень прозрачности (*см. разд. "Синтез сплошного цвета" гл. 6*). Цвет текста также можно задать как цвет заливки при помощи блока управления цветом, панели **Color Mixer** или каталога **Color Swatches**;
- **Bold style** (Жирное начертание) — использование полужирного начертания (выделенный текст);
- **Italic style** (Курсивное начертание) — использование курсивного начертания;
- **Orientation of text** (Направление текста) — щелчок на данной пиктограмме позволяет выбрать из появляющегося списка направление текста:
 - **Horizontal** (Горизонтальный) — используется по умолчанию (в случае если в меню **Edit>Preferences>Editing** не задано иное);
 - **Vertical Left to Right** (Вертикальный Слева Направо) — выбор данной опции изменяет направление расположения текстового блока. Вертикальный текст используется в некоторых азиатских языках. Существует возможность развернуть символы шрифта внутри текстового блока. Для этого необходимо щелкнуть на пиктограмме **Rotation** (Поворот), которая становится активна при выборе вертикальной ориентации текста;
 - **Vertical Right to Left** (Вертикальный Справа Налево);
- **Align** (Параметры выравнивания абзаца) — данные параметры идентичны соответствующим настройкам текстового редактора (например, Microsoft Word):
 - **Align Left** — выключка по левому краю;
 - **Align Center** — выключка по центру;
 - **Align Right** — выключка по правому краю;
 - **Justify** — выключка по формату;

- **Character spacing** (Трекинг) — межсимвольное расстояние. Данный параметр отвечает за плотность размещения символов в строке и может принимать значения в диапазоне от -60 до 60. При этом отрицательные значения приводят к тому, что символы, сближаясь, перекрывают друг друга, а положительные значения увеличивают интервал между символами, растягивая строку (в случае текстового блока фиксированной длины это может привести к переносу строки);
- **Character position** (Индекс) — раскрывающийся список, позволяющий задать положение символа. **Superscript** — верхний индекс, **Normal** — текст на базовой линии, **Subscript** — нижний индекс;
- **Auto kern** (Автоматический кернинг) — установка данного флажка позволяет использовать информацию об автоматическом кернинге, содержащуюся в шрифтовом файле. *Кернинг* — это регулировка расстояний между некоторыми парами символов шрифта, позволяющая добиться впечатления равномерного распределения всех элементов последовательности символов. Дело в том, что в силу специфики зрительного восприятия человека расстояние между некоторыми символами шрифта визуально кажется больше, чем между остальными (например, между буквами A и V, Г и А), и это создает впечатление неравномерности размещения символов. Автоматический кернинг учитывает эту специфику. Правда, в некоторых случаях автоматический кернинг не дает положительного результата и распределение символов необходимо задавать вручную, регулируя межсимвольное расстояние;
- **Alias text** (Отключение сглаживания символов) — эта кнопка отключает сглаживание текста (или отдельных выделенных символов) и применяется при работе с текстом маленького размера, делая его более четким и читабельным. В Flash сглаживание для шрифтов включено по умолчанию, и это приводит к тому, что текст небольшого размера выглядит размытым и плохо читается. При отключении сглаживания очертания шрифта выравниваются вдоль границ пикселов, что напоминает эффект растирования. Однако в результате отключения сглаживания некоторые символы шрифта могут утратить свою оригинальную форму, поэтому не стоит использовать слишком мелкий кегль. Стоит иметь в виду, что шрифты с засечками (serif) при маленьких размерах отображаются хуже, чем рубленые шрифты без засечек (sans serif), кроме того, курсивное или полужирное начертание также приводит к снижению читаемости текста;
- **Format** (Параметры форматирования абзаца) — нажатие на эту кнопку приводит к появлению диалогового окна **Format Options**, представленного на рис. 8.3. Здесь имеются следующие параметры:
 - **Indent** — отступ абзаца (красная строка);
 - **Line spacing** — интерлиньяж (межстрочное расстояние);

- **Left margin** — левое поле;
- **Right margin** — правое поле;

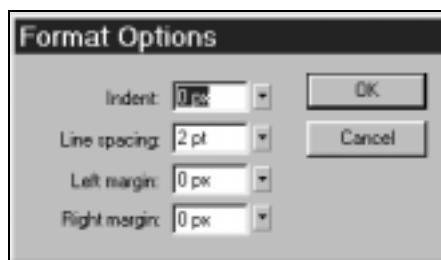


Рис. 8.3. Параметры форматирования абзаца

- **Selectable** (Выделяемый текст) — активизация данного режима позволит пользователю выделять текст в конечном фильме и, соответственно, копировать его в буфер при помощи команды **Copy**. Если режим **Selectable** отключен, пользователь не сможет получить доступ к содержимому соответствующего текстового блока;
- **Use device fonts** (Использовать машинонезависимые шрифты) — установка флагшка приводит к тому, что контуры текущего шрифта не будут встроены в конечный файл. Это позволит уменьшить размер результирующего файла. Данная опция может быть использована только со статическим горизонтальным текстом. При просмотре проекта на машине конечного пользователя в случае отсутствия у него в системе данного шрифта вместо него будет подставлен машинонезависимый шрифт (device font), наиболее близко соответствующий оригиналу. Применение данной опции отключает сглаживание символов шрифта. Текст, набранный с использованием машинонезависимого шрифта, не может быть трансформирован (поворнут или скосен), т. к. это приведет к его исчезновению в конечном фильме. Использование некоторых кириллических шрифтов с включенной опцией **Use device fonts** в случае их отсутствия в системе может привести к некорректному отображению (или исчезновению) символов шрифта в конечном фильме. Подробнее о системных шрифтах будет сказано в разд. "Использование системных шрифтов" настоящей главы.
- **URL link** (Гиперссылка) — данное поле позволяет назначить гиперссылку на все содержимое текстового блока или конкретный выделенный фрагмент текста. Для этого в поле **URL link** необходимо ввести URL с указанием протокола (например, <http://www.avalon.ru>). Для того чтобы создать ссылку на адрес электронной почты, необходимо использовать ключевое слово **mailto:** (например, mailto:book_flash@mail.ru). Текстовая гиперссыл-

ка в рабочей среде Flash подчеркивается пунктирной линией, но в конечном фильме, в отличие от традиционной HTML-ссылки, автоматически никак особо не выделяется. При наведении на текст, содержащий гиперссылку, в конечном фильме курсор принимает вид руки. Для того чтобы выделить гиперссылку на фоне обычного текста, можно окрасить данный фрагмент текста другим цветом и разместить позади текстового блока цветную векторную форму (например, цветной прямоугольник) или линию. Для того чтобы протестировать ссылку, необходимо перейти в среду тестирования при помощи команды меню **Control>Test Movie** (<Ctrl>+<Enter>). Подробная информация о тестировании фильма содержится в гл. 16.

□ **Target** (Окно) — данное поле становится активным при создании гиперссылки. При помощи раскрывающегося списка можно указать, в какое окно или фрейм (при фреймовой структуре) будет загружен вызываемый ресурс:

- **_blank** — в новое окно браузера;
- **_parent** — в родительский по отношению к данному фрейм;
- **_self** — в текущий фрейм текущего окна;
- **_top** — в самый верхний фрейм текущего окна.

Примечание

В некоторых случаях Flash не сможет экспорттировать очертания символов шрифта, встроив их в конечный фильм. В этом случае необходимо либо отказаться от использования данного шрифта, либо разбить его, конвертируя в графику (см. разд. "Разбиение текста" настоящей главы). Для того чтобы определить, будет ли шрифт экспорттироваться в конечный фильм, нужно в рабочей среде включить режим просмотра **View>Preview Mode>Antialias Text**. Если края текста в этом режиме не сглаживаются и имеют неровные зубчатые границы, это свидетельствует о том, что Flash не сможет экспорттировать очертания шрифта.

Динамический текст

Динамический текст (**Dynamic**) используется для вывода информации, которая изменяется в процессе воспроизведения Flash-фильма. Динамический текстовый блок обрабатывается программно при помощи языка сценариев ActionScript. Механизм использования динамического текста состоит в следующем. Каждому динамическому текстовому блоку может быть присвоен уникальный идентификатор, при помощи которого осуществляется обращение к данному текстовому блоку, динамический вывод в нем определенной информации и программное форматирование. В качестве такого идентификатора может использоваться имя экземпляра текстового блока или имя переменной, сопоставленной данному текстовому блоку.

Таким образом, задав в рабочей среде все необходимые параметры форматирования и снабдив текстовый блок идентификатором, можно оставить его пустым. Содержимое может быть добавлено уже в процессе воспроизведения программным образом.

Динамический текстовый блок может создаваться и форматироваться в рабочей среде документа либо непосредственно в процессе воспроизведения фильма программным способом. Динамический текст — это интерактивный элемент. Диапазон применения динамического текста чрезвычайно широк. Среди областей его применения можно отметить следующие:

- информация о ходе процесса загрузки из сети;
- всевозможные сообщения и подсказки пользователю как реакция на произведенные действия;
- отображение курсов валют, прогноза погоды, текущего времени;
- новостные группы;
- текстовая реклама;
- игры (например, отображение счета игры), тесты (например, вывод результатов тестирования);
- декоративные элементы оформления (например, сменяющиеся в случайном порядке буквы и цифры) и др.

Создание динамического блока осуществляется так же, как и создание статического, однако в отличие от последнего маркер динамического текстового блока располагается в правом *нижнем* углу, а невыделенный динамический текстовый блок отображается на сцене пунктирной рамкой. При создании динамического текстового блока можно строго задать не только длину строки, но и высоту блока. При этом если в процессе воспроизведения текст, помещаемый в данный блок сценарием ActionScript, превышает размер текстового блока, текст будет срезан его границами.

Если к динамическому текстовому блоку программно или при помощи инструментов рабочей среды применяются операции трансформации (поворот, скос, зеркальное отражение), необходимо встроить очертания символов, выводимых в нем. В противном случае текст будет отображаться некорректно или исчезать.

Для форматирования динамического текстового блока в рабочей среде его необходимо выделить и задать соответствующие параметры при помощи Инспектора свойств и команды главного меню **Text**. О динамическом создании и форматировании текста рассказывается в гл. 21. Внешний вид Инспектора свойств при редактировании динамического текстового блока представлен на рис. 8.4.

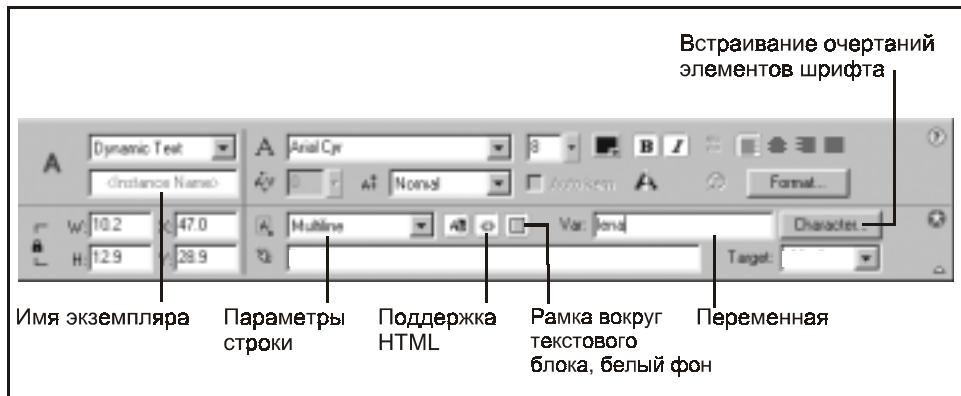


Рис. 8.4. Форматирование динамического текста. Инспектор свойств

Наряду с уже рассмотренными параметрами здесь имеются новые, характерные для данного типа текста. К ним относятся:

- **Instance Name** (Имя экземпляра) — данное поле используется для задания имени экземпляра текстового блока. Имя экземпляра является уникальным идентификатором текстового блока, позволяющим динамически выводить в него определенную информацию и выполнять форматирование средствами языка сценариев ActionScript. Используя имя экземпляра, можно программно манипулировать данным текстовым блоком, например, позиционировать его, масштабировать, вращать, включать или отключать отображение, задавать степень прозрачности текста, цвет фона и рамки и др. Имя экземпляра должно быть задано в соответствии с правилами именования переменных (см. гл. 20);
- **Line type** (Тип строки) — данное поле используется для указания параметров переноса строки:
 - **Single Line** — текст выводится в одну строку;
 - **Multiline** — перенос строки осуществляется автоматически при достижении границы тестового блока;
 - **Multiline no wrap** — перенос строки осуществляется только при использовании соответствующей команды;
- **Render text as HTML** (HTML-форматирование) — включение данного режима позволяет программно форматировать текст, используя некоторые теги HTML;
- **Show border around text** (Текст в рамке) — нажатие этой кнопки приводит к тому, что текст в текстовом блоке будет выводиться на белом фоне, а сам блок будет заключен в черную рамку. Изменить цвет фона блока и цвет рамки можно только программными средствами;

- **Var** (Переменная) — в это поле вводится имя переменной, сопоставленной данному блоку. Строковое значение этой переменной будет выводиться в качестве динамического текста. Переменная должна быть названа в соответствии с правилами именования переменных (см. гл. 20);
 - **Character** (Символы) — нажатие этой кнопки приводит к появлению диалогового окна **Character Options** (Параметры символов), которое позволяет указать, очертания каких символов должны быть встроены в конечный фильм (рис. 8.5). Дело в том, что если в системе у конечного пользователя не окажется шрифта, используемого для вывода текста в динамическое поле, информация будет отображена некорректно. Для того чтобы избежать этого, необходимо либо использовать шрифты, которые наверняка должны быть установлены у потенциальной аудитории, либо встраивать используемый в проекте шрифт (или несколько). Встраивание шрифтов приводит к увеличению объема конечного файла, но гарантирует корректное отображение текстовой информации.

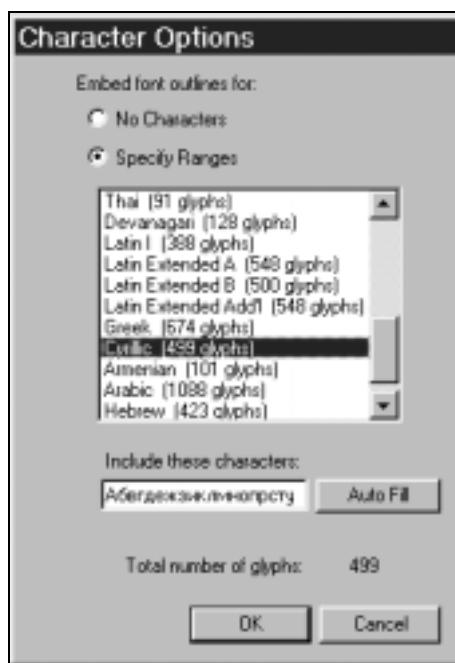


Рис. 8.5. Окно **Character Options**

Окно **Character Options** содержит следующие параметры:

- **No Characters** — не встраивать очертания шрифта. В этом случае объем конечного файла будет минимален, но в случае отсутствия использо-

зумого шрифта в системе пользователя текст будет отображен некорректно;

- **Specify Ranges** — данная опция позволяет выбрать любой диапазон символов (или несколько при помощи клавиш <Shift> или <Ctrl>) из представленных в расположеннном здесь же поле. Каждая строка содержит название диапазона, начальный и конечный символы (или слово **All**, означающее все символы) и общее число символов данного диапазона;
- **Include these characters** — в это поле можно вручную ввести произвольный набор символов, включая знаки пунктуации и некоторые специальные знаки, которые будут встроены в конечный фильм. Знаки вводятся подряд без разделителей. Если один и тот же набор символов используется в нескольких динамических или пользовательских текстовых блоках, его достаточно встроить один раз;
- **Auto Fill** — эта кнопка позволяет автоматически заполнить поле **Include these characters**, указав только те символы, которые набраны в соответствующем текстовом блоке;
- **Total number of glyphs** — здесь указывается общее количество встраиваемых символов.

Пользовательский текст

Принципиально пользовательский текст (**Input**) имеет много общего с динамическим текстом, однако его основная особенность состоит в том, что содержимое текстового блока типа **Input** вводится самим пользователем в процессе воспроизведения фильма. Так же как и динамический, пользовательский текст обрабатывается программно и является интерактивным элементом. Пользовательский текстовый блок может содержать уникальный идентификатор — имя экземпляра или имя переменной. При вводе данных в это поле соответствующая переменная принимает введенную строку в качестве своего значения. После чего это значение обрабатывается сценарием ActionScript.

Пользовательский текстовый блок, как и динамический, может создаваться и форматироваться в рабочей среде или непосредственно в ходе воспроизведения фильма программными средствами. Пользовательский текстовый блок может служить не только для ввода текста, но и выводить его. При этом пользователь может вносить в текст изменения.

Текстовый блок типа **Input** применяется в случаях, когда необходимо получить какую-либо информацию от конечного пользователя. Таким образом, он позволяет реализовать взаимодействие с аудиторией, обеспечивая возможность обратной связи. Диапазон его использования не менее широк,

чем у динамического текста. Среди областей его применения можно отметить следующие:

- Flash-формы (в том числе E-mail-формы);
- тесты, викторины, анкеты;
- гостевые книги;
- голосования;
- игры, развлекательные программы;
- авторизация доступа (проверка пароля).

Процесс создания и форматирования пользовательского текстового блока полностью аналогичен работе с динамическим текстом. Следует иметь в виду, что в конечном проекте поле ввода редактируемого текста должно быть визуально выделено, чтобы пользователь понял, куда именно он должен вводить те или иные данные. Для этого можно либо в рабочей среде при разработке дизайна интерфейса акцентировать его при помощи соответствующих графических элементов (цветной прямоугольник, рамка, линии и т. п.), либо задать цвет фона и рамки программно (*см. гл. 21*).

Все параметры Инспектора свойств, используемые для настройки динамического текста, могут быть применены и к пользовательскому. Имеется только два новых параметра:

- список **Line type** (Тип строки) содержит значение **Password** (Пароль). При выборе этого значения символы, вводимые пользователем в процессе воспроизведения фильма, будут заменены символом "*". Это прием, используемый при вводе паролей;
- поле **Maximum Characters** позволяет задать максимальное число символов, которые пользователь сможет ввести в данное поле. Например, если в редактируемое поле предлагается ввести год рождения, очевидно, что для этого нужно отвести не более четырех символов. Нулевое значение соответствует неограниченному количеству символов в текстовом поле.

Примечание

Следует иметь в виду, что для того, чтобы осуществить ввод, у пользователя должен быть установлен соответствующий шрифт. Для того чтобы избежать неприятностей, связанных с отсутствием шрифта, можно встроить набор символов, который должен быть использован для ввода. Например, для ввода года рождения необходимо встроить 10 символов цифр.

Если к пользовательскому текстовому блоку программно или при помощи инструментов рабочей среды применяются операции трансформации (поворот, скос, зеркальное отражение), необходимо встроить очертания символов, выводимых в нем. В противном случае текст в конечном фильме будет отображаться некорректно или не вводиться в редактируемое поле.

Для того чтобы протестировать работу пользовательского текстового блока, необходимо перейти в среду тестирования при помощи команды **Control>Test Movie** (<Ctrl>+<Enter>). Для выхода из среды тестирования нужно закрыть текущее окно. Подробная информация о тестировании фильма содержится в гл. 16.

Проверка орфографии

Flash MX 2004 содержит встроенные функции проверки орфографии. Однако в стандартный комплект поставляемых вместе с приложением словарей словарь русского языка не входит.

Для настройки параметров проверки орфографии используется команда меню **Text>Spelling Setup**. Ее выполнение приводит к появлению одноименного диалогового окна. Здесь имеются следующие основные разделы:

- Document options** — данный раздел позволяет указать, какие именно элементы будут подвергнуты проверке написания (текстовые поля, названия сцен и слоев, метки и комментарии кадров, строки в сценариях Action-Script, названия символов и растровых изображений), в какой части проекта будет выполняться проверка (только текущая сцена или весь проект). Кроме того, можно задать параметры проверки (отображение процесса проверки всех элементов иерархии фильма в Проводнике (**Movie Explorer**) и возможность редактирования текущего элемента на месте);
- Dictionaries** — перечень используемых при проверке словарей;
- Personal dictionary** — путь к личному словарю. Щелчок на кнопке **Edit Personal Dictionary** позволит отредактировать личный словарь;
- Checking options** — параметры проверки орфографии.

Для проверки орфографии нужно выполнить команду **Text>Check Spelling**. При проверке орфографии выводится окно, в котором указано местонахождение проверяемого элемента, само слово, содержащее ошибку, и предложения по исправлению. Можно либо принять исправления (**Change**, **Change All**), либо отклонить (**Ignore**, **Ignore All**), либо добавить слово в личный словарь (**Add to Personal**).

Использование машинонезависимых шрифтов

При работе со статическим текстом шрифт, используемый в наборе, встраивается в конечный файл проекта. Это, с одной стороны, обеспечивает его корректную передачу, с другой — увеличивает конечный объем файла.

В качестве альтернативы встраиванию очертаний шрифта при работе со статическим горизонтальным текстом можно использовать так называемые машинонезависимые шрифты (device fonts). В Flash имеются три таких

шрифта. В списке доступных шрифтов они располагаются вначале, их названия начинаются с символа нижнего подчеркивания "_":

- _sans;
- _serif;
- _typewriter.

Каждый из указанных шрифтов является псевдонимом для шрифта определенного начертания и соответствует одной из наиболее широко используемых гарнитур (семейств шрифта).

Шрифт _sans соответствует гарнитурам Arial или Helvetica, шрифт _serif соответствует гарнитуре Times New Roman, шрифт _typewriter соответствует гарнитуре Courier.

Применение машинонезависимых шрифтов позволяет избежать встраивания очертаний шрифтов, уменьшая тем самым конечный объем файла. При воспроизведении фильма проигрыватель Flash Player подставляет шрифт, наиболее близко соответствующий используемому машинонезависимому шрифту, выбирая его из набора установленных в системе пользователя шрифтов. Зачастую это приводит к тому, что шрифт, отображаемый в различных системах, выглядит несколько по-разному.

Машинонезависимые шрифты не слаживаются, поэтому при их использовании в конечном фильме символы имеют ступенчатые неровные границы. Отключение слаживания делает шрифт более разборчивым при небольших размерах (менее 10 пунктов).

При использовании системных шрифтов к тексту не могут быть применены операции трансформации (вращение, скос), поскольку это приведет к исчезновению текста в конечном фильме. При масштабировании системных шрифтов изменяется только размер символов, пропорции же остаются неизменными.

Системный шрифт не может быть маскирован в рабочей среде при помощи слоя маски. Для того чтобы использовать эффект маски применительно к текстовому блоку, использующему системный шрифт, необходимо создавать маску программно.

При выделении текстового блока в рабочей среде инструментом **Selection**, его содержимое, набранное с использованием системного шрифта по-русски, может отображаться некорректно, однако в конечном фильме текст будет выведен без искажений.

Текст как объект наложенного уровня

Текстовый блок по умолчанию является объектом наложенного уровня. В связи с этим к тексту можно применять все операции, доступные для объектов

наложенного уровня. Следует иметь в виду, что при трансформации текстового блока очертания выводимого в нем текста должны быть встроены в конечный фильм. Таким образом, если речь идет о трансформации статического текста, нужно отказаться от применения системных шрифтов, при трансформации же динамического или пользовательского текста необходимо встроить очертания используемых символов, указав соответствующий набор в окне **Character Options**.

К текстовому блоку могут быть применены следующие операции трансформации:

- масштабирование (пропорциональное и непропорциональное);
- поворот;
- скос;
- зеркальное отражение по вертикали или по горизонтали.

В результате трансформации текстового блока можно добиваться достаточно интересных визуальных текстовых эффектов. Однако следует иметь в виду, что чрезмерное искажение пропорций и внешнего вида шрифта может привести к снижению читаемости. К эффектам, реализуемым при помощи операций трансформации, можно отнести эффект тени (рис. 8.6) и эффект отражения (рис. 8.7).



Рис. 8.6. Эффект тени

Для создания текста с тенью можно выполнить следующие действия:

1. При помощи инструмента **Text** создать текстовый блок типа **Static**.
2. Ввести в текстовый блок необходимый текст.
3. Задать параметры форматирования (шрифт, кегль, цвет, выравнивание абзаца, трекинг и др.). Цвет должен быть достаточно темным, поскольку он будет выполнять роль цвета тени.
4. Скопировать текстовый блок, протянув его инструментом **Selection**, удерживая клавишу <Ctrl>.

5. Перекрасить цвет копии текстового блока, сделав его ярче и светлее, чем цвет оригинала.
6. Переместить скопированный текстовый блок поверх оригинала с небольшим смещением по вертикали и горизонтали, как показано на рис. 8.6 вверху.
7. Нижележащий текстовый блок можно немного скосить для придания эффекта объема (рис. 8.6 внизу).

Эффект тени можно также получить автоматически при помощи эффекта монтажной линейки **Drop Shadow** (*подробная информация об эффектах монтажной линейки содержится в гл. 13*).



Рис. 8.7. Эффект отражения

Для создания эффекта отражения можно выполнить следующие действия:

1. При помощи инструмента **Text** создать текстовый блок типа **Static**.
2. Ввести в текстовый блок необходимый текст.
3. Задать параметры форматирования (шрифт, кегль, цвет, выравнивание абзаца, трекинг и др.).
4. Скопировать текстовый блок, протянув его инструментом **Selection**, удерживая нажатой клавишу <Ctrl>.
5. Отразить скопированный текст по вертикали при помощи команды **Modify>Transform>Flip Vertical**.
6. Подвести скопированный текст к оригиналу и немного скосить его при помощи инструмента **Free Transform**, как показано на рис. 8.7.
7. Задать степень прозрачности цвета скопированного текста отличной от 100%.
8. При необходимости можно сжать отражение по вертикали.

Кроме эффектов трансформации, можно использовать такие средства визуального выделения текста, как цветные векторные формы или растровые изображения, размещенные под текстовыми блоками, выполняющие роль фона, рамки, выносные линии, создание нескольких блоков наподобие колонок текста и т. д.

Разбиение текста

Для того чтобы перевести текст на рабочий уровень, необходимо применить к нему операцию *разбиения* (*breaking apart*). В результате этой процедуры текст превращается в набор векторных форм, каждая из которых может редактироваться как *графический объект* рабочего уровня. Разбиение текстового блока приводит к тому, что его содержимое уже не может редактироваться и подвергаться форматированию в качестве текста. Например, после разбиения невозможно заменить один символ на другой, выбрать другой шрифт или изменить начертание. С другой стороны, к получившемуся в результате разбиения набору векторных форм можно применить любую процедуру, доступную на наложенном уровне (искажение формы, окраска любым типом цветового заполнения, добавление обводки и др.).

Разбить текстовый блок можно, выделив его и применив команду меню **Modify>Break Apart** или ее клавиатурный эквивалент <Ctrl>+. Причем первое применение данной команды приводит к тому, что текст распадается на совокупность составляющих его символов, каждый из которых помещен в отдельный текстовый блок. Таким образом, текст по-прежнему остается объектом наложенного уровня, допуская возможности редактирования и форматирования, но состоит из множества независимых элементов. Повторное применение к выделенным единичным блокам команды **Modify>Break Apart** позволяет окончательно разбить текст, переводя его на рабочий уровень с утратой возможности форматирования.

Основное назначение разбиения текста состоит в следующем.

- Разбиение дает возможность без искажений отобразить текстовую информацию на любой машине, даже если в системе пользователя не установлены соответствующие шрифты. При работе с редко используемыми экзотическими шрифтами в качестве альтернативы встраиванию (а также если шрифт не может быть экспортирован) можно использовать разбиение.
- Разбиение позволяет выполнять с текстом произвольные преобразования и трансформации, открывающие широкие возможности его декоративного применения.

Примечание

Следует иметь в виду, что разбиение большого количества текста может привести к появлению множества векторных форм и, как следствие, к существенному увеличению объема конечного файла.

На рабочем уровне с текстом, конвертированным в графику, можно выполнять следующие действия.

- Окраска градиентной и растровой заливкой (рис. 8.8, *а*).
- Добавление обводки (рис. 8.8, *б*).
- Взаимодействие с другими объектами рабочего уровня (рис. 8.8, *в*).
- Изменение формы символов инструментом **Selection** (рис. 8.8, *г*).
- Масштабирование, поворот, скос, зеркальное отражение.
- Искажение формы символов при помощи инструмента **Free Transform** (рис. 8.8, *д*).
- Редактирование формы символов при помощи огибающих **Envelope** (рис. 8.8, *е*).

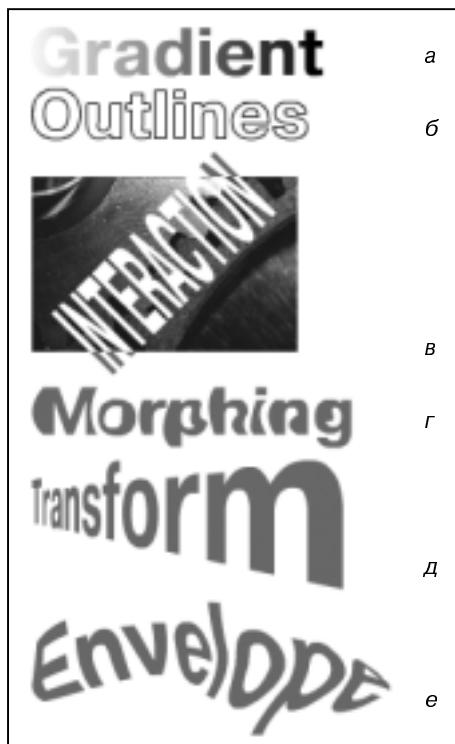


Рис. 8.8. Возможности работы с текстом как с объектом рабочего уровня

В табл. 8.2 обобщены и систематизированы возможности работы с текстом как с объектом наложенного и рабочего уровня.

Таблица 8.2. Операции с текстом

	Операция	Способ
Наложенный уровень	Копирование	1. Команда Edit>Duplicate , <Ctrl>+<D>. 2. Тащить с <Ctrl> или <Alt>. 3. Через буфер обмена
	Удаление	1. Команда Edit>Clear . 2. <Delete>
	Трансформация	Инструмент Free Transform (масштабирование, поворот, скос, зеркальное отражение)
	Заливка сплошным цветом	Блок управления цветом; панель Properties ; палитра Color Swatches ; палитра Color Mixer
	Анимация	Тип анимации движения Motion Tween
	Разбиение текста (переход к рабочему уровню)	Команда Modify>Break Apart , <Ctrl>+; команду применить дважды; после разбиения текста его форматирование невозможно, т. к. он становится набором векторных форм
Рабочий уровень	Изменение формы	Инструменты Selection ; Free Transform (включая модификаторы Envelope , Distort); Subselection ; Pen ; Eraser
	Добавление обводки	Инструмент Ink Bottle
	Заливка градиентом или растром	Инструмент Paint Bucket (для заливки всех форм одним градиентом включить режим Lock Fill); блок управления цветом Color ; панель Properties ; палитра Color Swatches ; палитра Color Mixer
	Анимация	Тип анимации формы Shape Tween
	Группировка	Команда Modify>Group ; <Ctrl>+<G> переход к наложенному уровню

Многоязыковая поддержка. Панель *Strings*

Flash MX 2004 обладает возможностями, позволяющими разрабатывать проекты на нескольких языках. При этом на компьютере конечного пользователя содержимое автоматически отображается на том языке, который по умолчанию используется в системе. Для отображения текста на нескольких языках можно использовать только динамический (**Dynamic**) или пользовательский (**Input**) текст.

Технология разработки многоязыковых проектов состоит в следующем. В исходном файле определяются языки, которые будут использованы в про-

екте. Создаются текстовые поля, каждому из которых сопоставляется уникальный идентификатор панели **Strings** и текст (строка — string) на "родном" языке, называемом *языком сцены* (Stage language). После этого выполняется перевод всех строк (strings) панели **Strings** с языка сцены на остальные используемые языки. Таким образом, одному текстовому блоку, имеющему уникальный идентификатор, соответствует несколько текстовых строк на разных языках. При публикации проекта Flash автоматически генерирует файлы XML, соответствующие этим языкам и содержащие все используемые строки. Например, при использовании русского и английского языков будут созданы два файла, содержащие информацию на английском и русском языках. Эти файлы должны быть размещены на сервере вместе с опубликованными файлами проекта. Когда проект загружается на компьютере конечного пользователя, выполняется проверка языковых настроек системы и Flash отображает текст на его родном языке, используя информацию "языковых" XML файлов. В случае если используемый системой язык не является одним из языков Flash-проекта, текст будет отображаться на языке, заданном в рабочей среде в качестве языка по умолчанию.

Существует несколько схем выполнения перевода. Перевод может выполняться непосредственно в рабочей среде проекта при помощи панели **Strings**, либо в XML-файлах, соответствующих переводимым языкам.

Для создания проекта, поддерживающего несколько языков, при помощи панели **Strings** можно выполнить следующие действия:

1. Открыть панель **Strings** при помощи команды **Window>Other Panels>Strings**. Данная панель представлена на рис. 8.9.

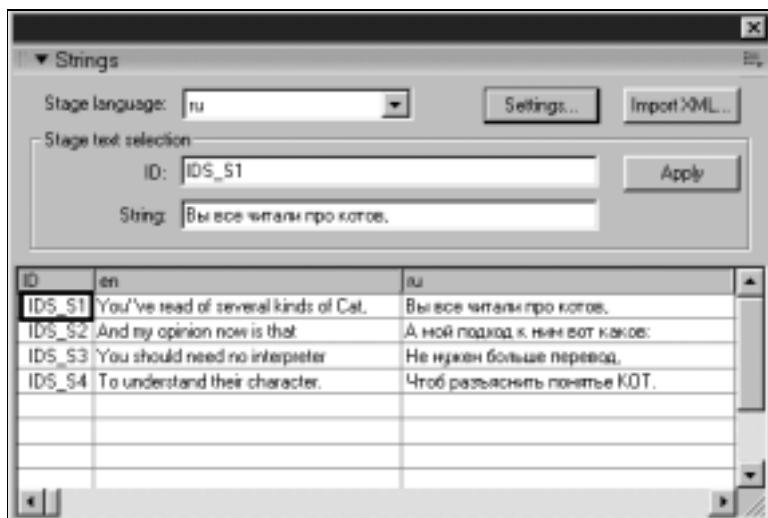


Рис. 8.9. Панель **Strings**

2. Выбрать языки, которые будут использованы в проекте. Для этого нужно нажать на кнопку **Settings** панели **Strings**. Одноименное окно представлено на рис. 8.10. В поле **Select languages** можно выбрать язык и добавить его в список используемых языков **Available languages** при помощи кнопки **Add**. Таким образом, можно добавить все необходимые языки. Если требуемый язык отсутствует в списке (например, русский), нужно ввести его код в поле, расположенное под списком **Select languages**, после чего добавить в список **Available languages** при помощи кнопки **Add**. Код состоит из двух частей и имеет формат `xx_XX`. Первые два символа задают язык, используя его обозначение в соответствии со стандартом ISO 639-1, вторые два символа являются необязательными и используются для указания страны в соответствии со стандартом ISO 3166-1. Для русского языка можно ввести "ru". Для того чтобы удалить язык из списка **Available languages**, необходимо выделить его и нажать кнопку **Remove**. В поле **Select Default language** нужно указать язык, используемый по умолчанию. В случае если используемый системой конечного пользователя язык не является одним из языков Flash-проекта, текст будет отображаться, используя язык по умолчанию. Поле **URL** используется для указания пути к файлу XML, содержащему перевод, в случае если его местоположение отлично от используемого по умолчанию. Для автоматического определения языка пользователя и вывода соответствующего текста необходимо установить флажок **Insert ActionScript for automatic language detection**. После выполнения всех указанных действий необходимо нажать кнопку **OK**. На панели **Strings** будут добавлены столбцы, соответствующие используемым языкам.

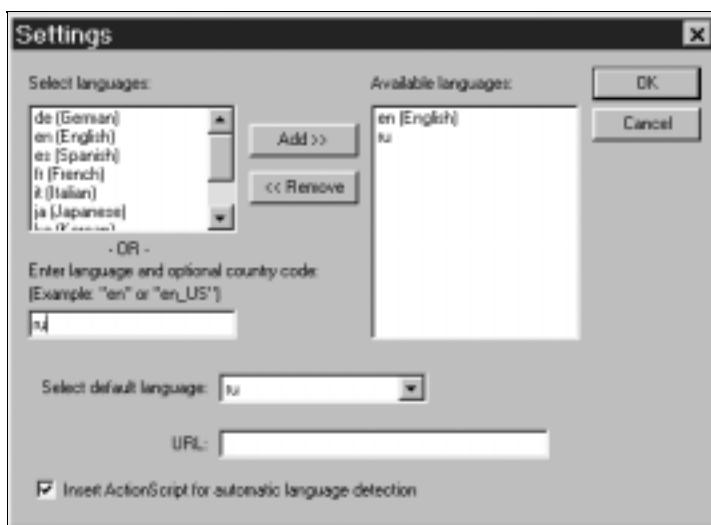


Рис. 8.10. Диалоговое окно языковых настроек **Settings**

3. Сопоставить уникальный идентификатор и строку динамическому или пользовательскому текстовому блоку. Это можно сделать различными способами. Можно либо сразу создать текстовый блок и сопоставить ему идентификатор и текстовую строку, либо сначала создать все идентификаторы и соответствующие строки, а потом последовательно сопоставить их создаваемым текстовым блокам.
 - Для того чтобы сразу создать текстовый блок и сопоставить ему идентификатор и текстовую строку, нужно выполнить следующие действия. При помощи инструмента **Text** создать на сцене динамический или пользовательский текстовый блок. Выделить его и на панели **Strings** в поле **ID** ввести идентификатор (используя латинские символы, без пробелов), например `s1`. Далее в поле **String** необходимо ввести текст, используя язык сцены, который устанавливается при помощи списка **Stage language**. После этого необходимо нажать кнопку **Apply** или клавишу `<Enter>`. В столбце, соответствующем языку сцены, будет отображен введенный текст, соответствующий идентификатору, указанному в самом левом столбце этой же строки.
 - Для того чтобы сначала создать все идентификаторы и соответствующие строки, а потом последовательно сопоставить их создаваемым текстовым блокам, необходимо поступить следующим образом. В поле **ID** ввести идентификатор, а в поле **String** — текст, используя язык сцены, который выбирается при помощи списка **Stage language**. После этого необходимо нажать кнопку **Apply** или клавишу `<Enter>`. В столбце, соответствующем языку сцены, будет отображен введенный текст, соответствующий идентификатору, указанному в самом левом столбце этой же строки. Повторять эти действия следует до тех пор, пока не будут созданы все необходимые строки. Далее нужно при помощи инструмента **Text** создать на сцене динамический или пользовательский текстовый блок, выделить его, а на панели **Strings** и в поле **ID** ввести идентификатор строки, которая должна выводиться в данном поле, заданный ранее. После этого необходимо нажать кнопку **Apply** или клавишу `<Enter>`. Стока на языке сцены будет выведена в данном текстовом блоке. Повторяя эти действия, можно последовательно сопоставить строки панели **Strings** соответствующим текстовым блокам, расположенным на сцене.
4. Выполнить перевод созданных строк с языка сцены на остальные используемые языки. Для этого можно либо воспользоваться панелью **Strings**, либо редактировать XML-файл, соответствующий языку, на который осуществляется перевод.
 - Для выполнения перевода непосредственно в панели **Strings** необходимо в столбце, соответствующем языку, на который осуществляется перевод, дважды щелкнув на ячейке, расположенной в одном ряду с

идентификатором переводимой строки, ввести текст перевода. После этого нужно щелкнуть на кнопке **Apply** или нажать клавишу <Enter>. Таким образом можно последовательно выполнить перевод всех строк. Для того чтобы в рабочей среде документа отображать содержимое текстовых блоков на различных языках, используемых в проекте, можно воспользоваться раскрывающимся списком **Stage Language** панели **Strings**.

- Каждый раз при сохранении или публикации документа, содержащего несколько языков, Flash автоматически генерирует файлы XML, соответствующие стандарту XLIFF 1.0 (XML Localization Interchange File Format), предназначенные для хранения языковой информации. Файл для каждого языка помещается в собственный каталог, названный в соответствии с кодом языка, указанным в окне **Settings**. Так, при использовании русского и английского языков при сохранении исходного документа автоматически будут созданы два каталога с названиями ru и en, содержащие XML-файлы, название которых будет соответствовать названию исходного документа Flash. Каталоги языковых файлов автоматически размещаются в каталоге, где был сохранен исходный документ.

Перевод можно выполнить непосредственно в XML-файле. Для этого его нужно открыть в любом текстовом редакторе, позволяющем работать с кодированным текстом (например, MS Word) или в HTML-редакторе (например, Macromedia Dreamweaver). Текст XML-файлов, загружаемых проигрывателем Flash Player 7, кодируется с использованием 8-битной кодировки Unicode — UTF-8. Примеры автоматически генерированных XML-кодов, соответствующих русской и английской версиям текстового содержимого, представлены в листингах 8.1 и 8.2 соответственно. Перевод задается при помощи элемента **source**, соответствующего идентификатору **ID**, заданному в панели **Strings**. Сохранять созданные вручную XML-документы надо, используя кодировку UTF-8.

Листинг 8.1. Пример XML-кода для русскоязычного содержимого

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xliff PUBLIC "-//XLIFF//DTD XLIFF//EN"
"http://www.oasis-open.org/committees/xliff/documents/xliff.dtd" >
<xliff version="1.0" xml:lang="ru">
  <file datatype="plaintext" original="multiLang_ru_en.swf" source
language="EN">
    <header></header>
```

```
<body>
    <trans-unit id="001" resname="IDS_S1">
        <source>Вы все читали про котов,</source>
    </trans-unit>
    <trans-unit id="002" resname="IDS_S2">
        <source>А мой подход к ним вот каков:</source>
    </trans-unit>
    <trans-unit id="003" resname="IDS_S3">
        <source>Не нужен больше перевод,</source>
    </trans-unit>
    <trans-unit id="004" resname="IDS_S4">
        <source>Чтоб разъяснить понятье КОТ.</source>
    </trans-unit>
</body>
</file>
```

Листинг 8.2. Пример XML-кода для англоязычного содержимого

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xliff PUBLIC "-//XLIFF//DTD XLIFF//EN"
"http://www.oasis-open.org/committees/xliff/documents/xliff.dtd" >
<xliff version="1.0" xml:lang="en">
    <file datatype="plaintext" original="multiLang_ru_en.swf" source
language="EN">
        <header></header>
        <body>
            <trans-unit id="001" resname="IDS_S1">
                <source>You've read of several kinds of Cat,</source>
            </trans-unit>
            <trans-unit id="002" resname="IDS_S2">
                <source>And my opinion now is that</source>
            </trans-unit>
            <trans-unit id="003" resname="IDS_S3">
                <source>You should need no interpreter</source>
            </trans-unit>
            <trans-unit id="004" resname="IDS_S4">
                <source>To understand their character.</source>
            </trans-unit>
```

```
</body>
</file>
</xliff>
```

Для того чтобы импортировать XML-файл, содержащий перевод в панель **Strings**, находясь в рабочей среде, нужно щелкнуть на клавише **Import XML** и в появившемся одноименном диалоговом окне выбрать язык, соответствующий файлу. В панель **Strings** можно импортировать только файлы, соответствующие выбранным языкам. При этом важно не перепутать файлы и не загрузить, скажем, английский перевод в столбец, предназначенный для русского языка.

◀ Примечание ▶

Возможна ситуация, когда в рабочей среде документа локальная загрузка XML-файлов не выполняется. Следствием этого является отсутствие отображения данных в панели **Strings** при новом открытии документа, а также невозможность осуществления импорта XML-файла при помощи клавиши **Import XML**. Однако, несмотря на это, при наличии соответствующих языковых XML-файлов в конечном фильме информация будет отображаться корректно и загрузка данных из XML-будет выполняться. В этой ситуации можно порекомендовать удалить из всех XML-файлов следующие строки:

```
<!DOCTYPE xliff PUBLIC "-//XLIFF//DTD XLIFF//EN"
"http://www.oasis-open.org/committees/xliff/documents/xliff.dtd" >.
```

После этого загрузка данных из файлов XML в панель **Strings** в рабочей среде будет выполняться корректно.

5. Сохранить и опубликовать документ Flash. Следует иметь в виду, что при сохранении, тестировании или публикации документа все файлы XML перезаписываются с использованием данных, содержащихся в панели **Strings**.

В результате сохранения, тестирования и публикации проекта, содержащего несколько языков, образуется структура файлов следующего вида:

/Projects/myPrj.fla — исходный файл;
/Projects/myPrj.swf — конечный Flash-фильм;
/Projects/myPrj.html — HTML-страница;
/Projects/ru/myPrj_ru.xml — XML с русскоязычным содержимым фильма;
/Projects/en/myPrj_en.xml — XML с англоязычным содержимым фильма.

Все файлы, за исключением исходного документа, должны быть помещены на web-сервер. Если при создании исходного документа была включена опция автоматического определения языковых настроек системы **Insert Action-Script for automatic language detection** в окне **Settings** панели **Strings**, то при

загрузке на компьютере пользователя текст автоматически будет выведен на его "родном" языке. Например, если языковые настройки (**Regional settings**), задаваемые в Панели управления (**Control panel**) Windows, определяют в качестве используемого языка русский, то текст будет выведен на русском языке. Если в качестве языка системы используется английский — то текст будет выведен на английском языке. Если система использует язык, отсутствующий в фильме, то информация будет выводиться на языке, заданном по умолчанию (**Default language**) в панели **Strings**.

Использование общих шрифтовых ресурсов

При работе над большими проектами, состоящими из нескольких разделов, представляющих собой отдельные файлы SWF, приходится многократно использовать один или несколько шрифтов, встраивая их в каждый файл. Это приводит к увеличению объема файлов и, соответственно, к увеличению времени их загрузки из сети. В качестве альтернативы встраиванию шрифта в каждый файл проекта Flash предлагает возможность создания общего шрифтового ресурса, помещенного в отдельный SWF-файл и загружаемого в соответствующий раздел непосредственно в процессе его воспроизведения. Таким образом, файл, содержащий общий шрифтовой ресурс (или несколько), должен быть загружен один раз, после чего к нему может быть осуществлен доступ со стороны всех остальных файлов проекта, использующих данный шрифт. При этом использование общего шрифта несколькими файлами практически не увеличивает их конечный объем. Общий шрифтовой ресурс может быть использован текстовым блоком любого типа (Static, Dynamic, Input).

Для использования общего шрифтового ресурса необходимо выполнить две процедуры:

1. Объявление общего шрифтового ресурса (или нескольких) путем создания документа-источника, содержащего общую библиотеку (**Shared library**).
2. Включение общего шрифтового ресурса (или нескольких) в документы-получатели.

Объявление общего шрифтового ресурса осуществляется в рабочей среде Flash путем создания *шрифтового символа* (Font symbol). Для этого необходимо выполнить следующие действия:

1. Создать новый документ Flash. **File>New**.
2. Открыть библиотеку нового документа (**Window>Library**) и из контекстного меню выбрать команду **New Font**.

3. В появившемся окне **Font Symbol Properties** (рис. 8.11) в поле **Name** ввести название шрифтового символа (например, Pragmatica_symb), в списке **Font** выбрать шрифт, используемый в качестве общего шрифтового ресурса (например, Pragmatica). Установка флажков **Bold** или **Italic** позволяет указать начертание шрифта. Флажок **Alias text** отключает сглаживание шрифта. Один шрифтовой символ может включать описание только одного начертания шрифта (regular, bold или italic) или сочетание начертаний (например, bold и italic одновременно). При необходимости использования нескольких начертаний одного шрифта (например, отдельно bold и отдельно italic) нужно для каждого из них создавать отдельные шрифтовые символы. Задав все параметры, необходимо нажать на клавишу **OK**. В результате выполненных действий в библиотеке появится новый элемент, помеченный пиктограммой .

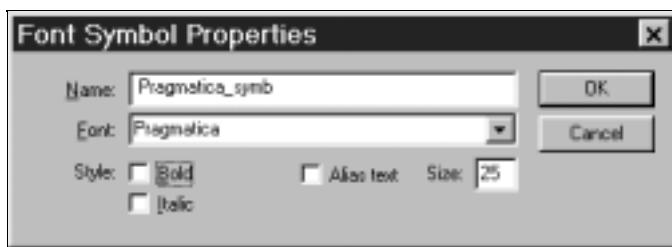


Рис. 8.11. Диалоговое окно создания шрифтового символа **Font Symbol Properties**

4. Задать параметры связи. Для этого щелкнуть правой кнопкой мыши по шрифтовому символу в библиотеке и из всплывающего контекстного меню выбрать команду **Linkage** или, выделив шрифтовой символ, выбрать эту команду из контекстного меню библиотеки. В появившемся диалоговом окне **Linkage Properties** (рис. 8.12) установить флажок **Export for runtime sharing**. В поле **Identifier** ввести имя, служащее уникальным идентификатором данного шрифта. Имя должно состоять из латинских символов и не может содержать пробелов (например, prgm_reg). В поле **URL** задается абсолютный или относительный путь к файлу SWF, содержащему библиотеку. Относительная ссылка разрешается относительно местоположения файла-получателя. Так, например, если файл, содержащий общую библиотеку, называется prgm_lib.swf и в конечном проекте будет размещен в одном каталоге с файлом, использующим общий шрифтовой ресурс, то в поле **URL** нужно ввести prgm_lib.swf. Если файл с общей библиотекой prgm_lib.swf будет помещен в отдельный каталог с названием fonts, который, в свою очередь находится в том же каталоге, что и файл, использующий шрифт, то путь будет выглядеть следующим образом: fonts/prgm_lib.swf.

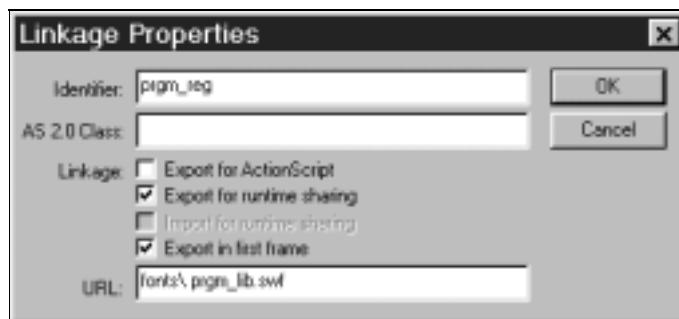


Рис. 8.12. Диалоговое окно настройки параметров связи шрифтового символа **Linkage Properties**

- Сохранить документ, содержащий шрифтовой символ, указав соответствующее имя (**File>Save As**). Если имя исходного документа с общей библиотекой, заданное при сохранении, будет совпадать с названием файла SWF, указанным в поле **URL** диалогового окна **Linkage Properties** (в нашем случае *prgm_lib.fla*), то при публикации файл SWF с соответствующим именем будет генерирован автоматически. В противном случае необходимо будет изменить имя результирующего SWF-файла в соответствии с именем, указанным в поле **URL** (см. гл. 17). Необходимо иметь в виду, что по умолчанию при публикации результирующий файл записывается в том же каталоге, где был сохранен исходный документ.
- Опубликовать исходный документ в формате SWF при помощи команды **File>Publish**.

Для включения общего шрифтового ресурса в документ необходимо выполнить следующие действия:

- Открыть (или создать) документ, в котором будет использован шрифт общей библиотеки **File>Open** (**File>New**).
- Открыть библиотеку документа, содержащего общую библиотеку **File>Import>Open External Library**. В рабочей среде откроется окно библиотеки внешнего файла.
- Импортировать шрифтовой символ, перетащив его курсором из внешней библиотеки на сцену или в библиотеку текущего документа. При этом параметры связи будут заданы автоматически. В окне **Linkage Properties** должен быть установлен флажок **Import for runtime sharing**, поле **Identifier** должно содержать идентификатор, заданный на шаге 4 предыдущей процедуры, а в поле **URL** должен быть указан путь к общей библиотеке относительно месторасположения файла, полученного в результате публикации текущего документа. После выполнения этих действий внешнюю библиотеку можно закрыть.

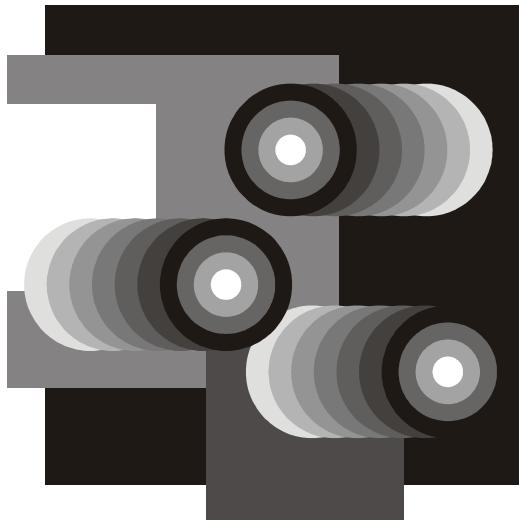
4. Создать текстовый блок, выделить его и в списке доступных шрифтов выбрать шрифт с названием, указанным в поле **Name** диалогового окна **Font Symbol Properties**. В списке это название помечено звездочкой (например, Pragmatica_symb*). Ввести текст, задав необходимые параметры форматирования.
5. Задать тип текстового блока (статический (Static), динамический (Dynamic), пользовательский (Input)).
6. Параметры импортированного шрифтового символа, заданные при его создании, указаны в окне **Font Symbol Properties**. Чтобы открыть это окно, необходимо либо дважды щелкнуть на пиктограмме шрифтового символа в библиотеке, либо воспользоваться командой контекстного меню самого символа или библиотеки. Если шрифт, используемый в качестве общего ресурса, имеет жирное (**Bold**) или курсивное (**Italic**) начертание, необходимо выполнить одно из следующих действий:
 - установить соответствующий флажок и отформатировать текстовый блок при помощи Инспектора свойств, задав ему полужирное или курсивное начертание. Вообще, если в окне **Font Symbol Properties** документа-получателя установлен тот или иной тип начертания, то соответствующие установки необходимо задать и для текстового блока при помощи Инспектора свойств. В конечном фильме шрифт будет отображен с использованием начертания, заданного в документе общей библиотеки;
 - снять соответствующий флажок в окне **Font Symbol Properties** документа-получателя и не задавать начертание шрифта при помощи Инспектора свойств. В конечном фильме шрифт будет отображен с использованием начертания, указанного в документе общей библиотеки.
7. Если используется обычное начертание (**Regular**) то все флашки в окне **Font Symbol Properties** документа-получателя должны быть сняты. Начертание шрифта на сцене также не задается.
8. Если в документе-получателе задать тип начертания, отличный от определенного в общей библиотеке, это приведет к тому, что вместо импорта из общей библиотеки шрифт будет встроен в конечный файл, увеличивая его размер. При этом связь с общей библиотекой будет разорвана.
9. Сохранить документ под любым именем. Необходимо иметь в виду, что по умолчанию при публикации результирующий файл записывается в тот же каталог, где был сохранен исходный документ, при этом их названия совпадают за исключением расширений.
10. Опубликовать документ **File>Publish**. Результирующий файл SWF должен быть размещен в соответствующем каталоге таким образом, чтобы ссылка, заданная в параметрах связи **Linkage Properties**, разрешалась относительно его текущего местоположения.

 **Примечание** 

При использовании общего шрифтового ресурса следует отказаться от применения кириллических символов, поскольку в конечном фильме они отображаются некорректно или исчезают.

При размещении проекта в сети Интернет необходимо поместить на сервер файл SWF, содержащий общую библиотеку, и файлы, использующие общий шрифт. Исходные документы (с расширением fla) на сервер не отправляются.

После создания общей библиотеки и включения шрифтового символа во все файлы проекта достаточно изменить шрифт в файле общей библиотеки и опубликовать его, после чего соответствующий шрифт будет изменен во всех файлах-получателях.

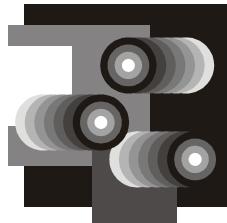


ЧАСТЬ II

АНИМАЦИЯ

Flash позволяет использовать различные способы создания анимации, каждый из которых обладает своими характерными особенностями и имеет определенную область применения. Определение анимации может быть сформулировано следующим образом. *Анимация* — это искусственное представление движения путем отображения последовательности рисунков или кадров с частотой, обеспечивающей целостное зрительное восприятие образов.

Как следует из данного определения, анимация только имитирует движение, последовательно отображая его фазы в виде статичных изображений. В силу инертности зрительного восприятия человека, дискретные фрагменты сливаются, создавая впечатление движения. Чем больше число промежуточных фаз и чем точнее они отображают состояние движущегося объекта, тем убедительнее выглядит анимация.



Глава 9

Монтажная линейка *Timeline*

Если кажется, что работу сделать легко, это непременно будет трудно.

Если на вид она трудна, выполнить ее абсолютно невозможно.

Теорема Стокмайера

Монтажная линейка **Timeline** (шкала времени, хронометрическая линейка, времененная диаграмма) — это важнейший компонент рабочей среды, предназначенный для организации структуры фильма и создания анимации.

Структура монтажной линейки

Монтажная линейка состоит из двух частей, совмещенная в себе палитру слоев, позволяющих структурировать документ, и последовательности кадров, обеспечивающих изменение содержимого сцены во времени.

Слои (Layers) можно сравнить с прозрачными листами пленки или стекла, на которых могут быть размещены любые объекты рабочей среды. Слои предоставляют возможность изолировать объекты друг от друга, исключая взаимодействие между ними и формируя структуру сцены. Слои Flash прозрачны и размещаются друг над другом в стопке. Сквозь области вышележащих слоев, свободные от объектов, можно видеть содержимое слоев, расположенных под ними. Каждый слой содержит свой рабочий и наложенный уровни, в пределах которых действуют все правила, изложенные в гл. 4. Таким образом, размещение объектов на разных слоях придает сцене глубину. Документ Flash может содержать практически неограниченное число слоев, причем их количество не влияет на размер конечного файла SWF (рис. 9.1).

Кадры (Frames) представляют собой место фактического размещения любого объекта, включенного в фильм в рабочей среде. Кадры последовательно размещаются на монтажной линейке и могут содержать графические объекты, звуковые файлы или сценарии ActionScript. Число кадров в документе практически неограничено. Следует различать два понятия: *кадр фильма* и *кадр монтажной линейки*.

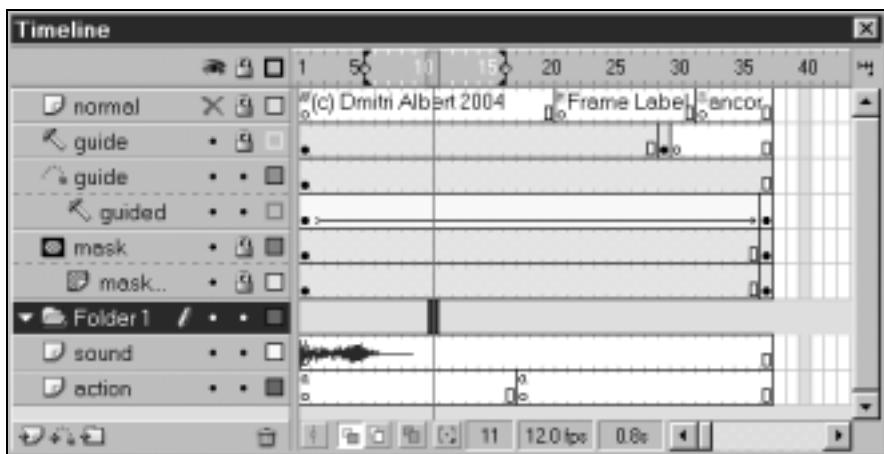


Рис. 9.1. Монтажная линейка **Timeline**

Кадр фильма (кадр сцены) — это все объекты сцены, размещенные на различных слоях в определенной временной позиции. Каждый кадр сцены имеет уникальный номер, соответствующий его временной позиции. Номер кадра указан в шкале, расположенной в верхней части монтажной линейки. Здесь же находится указатель текущего кадра, представляющий собой розовый прямоугольник. Перемещая его вдоль монтажной линейки, можно переходить с одного кадра на другой. Длительность экспонирования (просмотра) каждого кадра определяется скоростью воспроизведения монтажной линейки. Так, если скорость воспроизведения монтажной линейки составляет 12 кадров в секунду (frames per second — fps), то при воспроизведении каждый кадр фильма будет виден в течение $1/12$ (≈ 0.08) секунды. Десятисекундный фильм при данной скорости должен содержать 120 кадров. Содержимое кадра фильма может быть распределено по различным слоям.

Кадр монтажной линейки — элемент кадра фильма, расположенный в определенной временной позиции в пределах одного слоя. На каждом слое может находиться практически неограниченное число кадров монтажной линейки. Кадры монтажной линейки, имеющие один и тот же номер (т. е. расположенные в одной и той же временной позиции), формируют кадр фильма. Таким образом, кадр фильма представляет собой срез всех слоев в элементарный промежуток времени, определяемый скоростью воспроизведения монтажной линейки (рис. 9.2).

Для того чтобы лучше разобраться в этих понятиях, можно условно ввести систему четырех координат, описывающих структуру документа Flash (рис. 9.3). Координаты *X* и *Y* определяют размещение объекта на сцене, т. е. в пределах рабочей или вспомогательной области. Палитре слоев может быть сопоставлена ось *Z*, условно соответствующая глубине документа. Изменение содержимого сцены от кадра к кадру обуславливает развитие во

времени, дающее четвертую, временную координату, которая откладывается по оси t , направленной слева направо вдоль монтажной линейки.

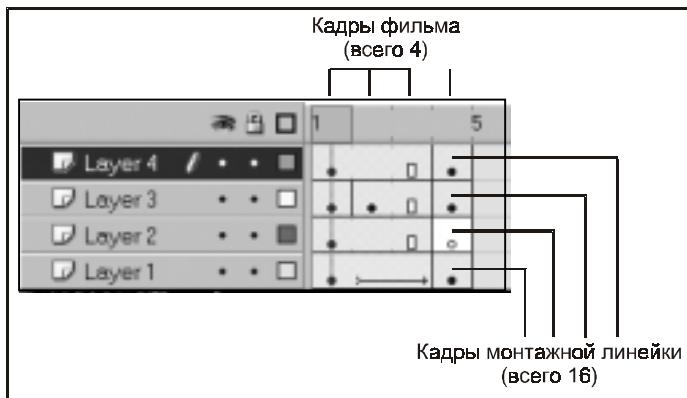


Рис. 9.2. Кадры фильма и кадры монтажной линейки

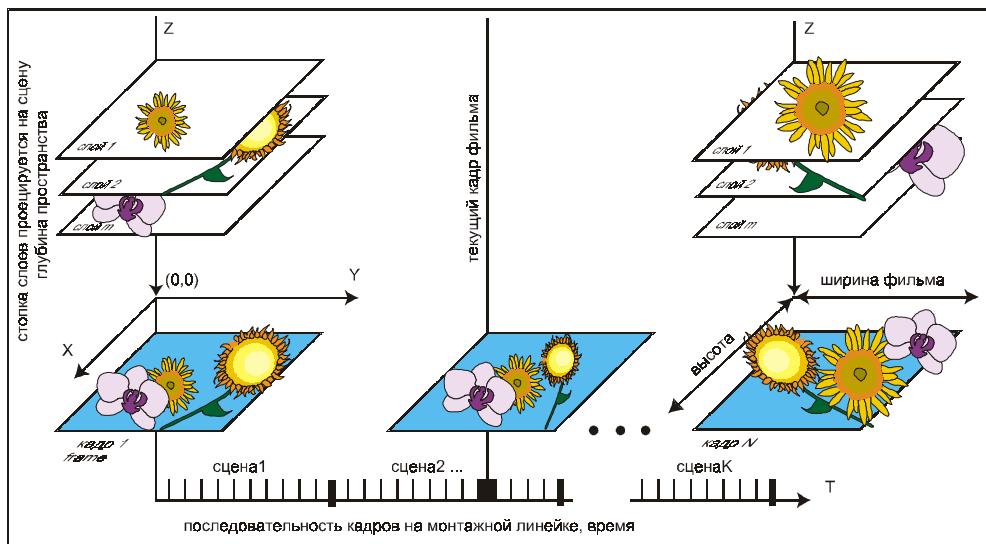


Рис. 9.3. Структура монтажной линейки Flash

Работа со слоями

При создании нового документа он содержит только один слой. Находясь в рабочей среде Flash, можно создавать новые слои, выполнять с ними раз-

личные манипуляции и использовать разные режимы отображения и редактирования слоев.

Создание слоев

При создании нового слоя он будет добавлен в палитру слоев и размещен над текущим слоем. Текущий слой в палитре слоев отмечен пиктограммой карандаша. Для того чтобы создать новый слой, можно выполнить одно из следующих действий:

- выполнить команду главного меню **Insert>Timeline>Layer**;
- правой кнопкой мыши щелкнуть на слою, над которым должен быть создан новый слой, и выполнить команду контекстного меню слоя **Insert Layer**;
- щелкнуть на пиктограмме  **Insert Layer** в нижней части палитры слоев;
- выполнить команду **Modify>Timeline>Distribute to Layers** ($<\text{Ctrl}>+<\text{Shift}>+<\text{D}>$). Эта команда позволяет автоматически распределить все выделенные объекты сцены по новым слоям. При этом необходимое число слоев будет создано автоматически. В результате выполнения этой команды новые слои помещаются под текущий слой. Если данная команда применяется для распределения по слоям экземпляров символов (instances), то новые слои получат имена в соответствии с именами символов. Если данные экземпляры имеют уникальные идентификаторы — имена экземпляров (instance names), то новые слои будут названы в соответствии с именами экземпляров. Подробная информация о работе с символами и экземплярами содержится в гл. 10.

Новые слои также автоматически добавляются в палитру слоев при вставке кадров монтажной линейки, принадлежащих разным слоям, из буфера обмена. В этом случае новые слои получают имена в соответствии с названиями слоев, откуда были скопированы кадры. О работе с кадрами см. далее в этой главе.

Переименование слоев

Имя слоя может состоять из любых символов, включая кириллические символы и знаки пунктуации. Имя слоя должно быть лаконичным и отражать содержание слоя (например, слои, содержащие передний и задний планы фильма, могут иметь названия "foregrnd" и "backgrnd" соответственно). Для того чтобы переименовать слой, можно выполнить одно из следующих действий:

- дважды щелкнуть на имени слоя и ввести новое название;
- выделить нужный слой и воспользоваться командой главного меню **Modify>Timeline>Layer Properties**. В появившемся диалоговом окне **Layer Properties** (рис. 9.4) в поле **Name** ввести новое имя;

- дважды щелкнуть на пиктограмме слоя и в появившемся диалоговом окне **Layer Properties** в поле **Name** ввести новое имя;
- щелкнуть на слое правой кнопкой мыши и из контекстного меню слоя выбрать команду **Properties**.

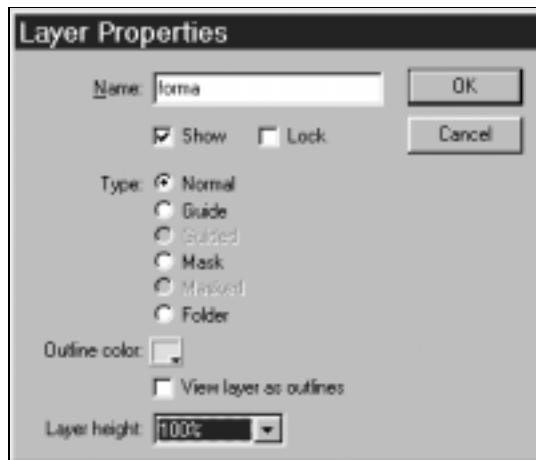


Рис. 9.4. Диалоговое окно **Layer Properties**

Выделение слоев

При выделении слоя выделяются все кадры монтажной линейки и, соответственно, все графические объекты, расположенные в пределах данного слоя. Одновременно можно выделить любое количество слоев. Для того чтобы выделить слой (или несколько), можно выполнить одно из следующих действий:

- щелкнуть на слое в палитре слоев;
- щелкнуть на кадре монтажной линейки, расположенном в пределах данного слоя;
- для того чтобы выделить несколько слоев, можно воспользоваться клавишей **<Shift>** для выделения диапазона слоев или клавишей **<Ctrl>** для выделения слоев вразбивку;
- щелчок инструментом **Selection** на любом объекте сцены, расположенному в пределах данного слоя, делает слой текущим.

Копирование слоев

В Flash не предусмотрено специальной команды для этой операции. Выполнить копирование слоя можно, скопировав расположенные на нем кадры

монтажной линейки и вставив их в новый слой. Для этого необходимо выполнить следующие действия:

1. Выделить все кадры слоя, щелкнув на нем.
2. Скопировать кадры слоя **Edit>Timeline>Copy Frames** или одноименной команды контекстного меню кадра.
3. Создать новый слой.
4. Выделить новый слой, щелкнув на нем (кадры этого слоя должны выделяться).
5. Вставить скопированные кадры при помощи команды главного меню **Edit>Timeline>Paste Frames** или одноименной команды контекстного меню кадра.

Изменение порядка следования слоев

Изменяя порядок следования слоев, можно управлять структурой размещения объектов на сцене, формируя планы фильма (передний, средний, задний и т. д.). Для того чтобы изменить порядок следования слоев в палитре слоев, можно выполнить одно из следующих действий:

- взяться за слой и перенести его в другое положение на палитре слоев;
- для переноса нескольких слоев необходимо выделить их и, взявшись за любой из выделенных слоев, перенести их в другое положение на палитре слоев.

Использование режимов отображения и редактирования слоев

В верхней части палитры слоев справа расположены три пиктограммы. Каждая из них отвечает за определенный режим слоя. Щелчок на одной из них приводит к тому, что соответствующий режим включается для *всех* слоев в палитре слоев. В то же время щелчок в точке или на цветном квадрате, расположенных на каждом слое, активизирует данный режим для данного слоя. Повторный щелчок на пиктограмме приводит к отключению режима. Палитра слоев позволяет применять следующие режимы при работе со слоями.

- Режим отображения **Show/Hide**  — позволяет показать/скрыть все объекты слоя, включив/отключив его отображение. Если отображение слоя отключено, то он перечеркивается красным крестом.
- Режим блокировки **Lock/Unlock**  — использование данного режима исключает возможность работы с объектами на сцене. Попытка создать объект на заблокированном слое приведет к появлению сообщения,

предлагающего разблокировать его. Эта возможность бывает полезна при необходимости исключить возможность редактирования объектов слоя. Следует иметь в виду, что кадры заблокированного слоя можно очистить или удалить, работая с монтажной линейкой.

- Режим отображения содержимого слоя в контурах **Show Layers as Outlines**  — использование данного режима приводит к тому, что все объекты данного слоя отображаются в виде цветных контуров. В этом режиме видны только границы векторных форм или рамки растровых изображений. Текст, однако, отображается без изменений, но окрашивается цветом, соответствующим цвету слоя. Каждому слою сопоставляется определенный цвет, который используется в качестве цвета контура. Эта особенность позволяет осуществлять идентификацию объектов на слое. В качестве цвета слоя может быть использован любой цвет из текущего каталога цветов. Для задания или изменения цвета слоя можно воспользоваться цветовым образцом диалогового окна **Layer Properties** (см. рис. 9.4). Для вывода этого окна можно дважды щелкнуть на пиктограмме слоя.

Для того чтобы включить любой из указанных режимов для всех слоев, кроме текущего, можно щелкнуть на соответствующей пиктограмме на слое, удерживая клавишу **<Alt>**. Для того чтобы включить любой из указанных режимов для всех слоев одновременно, можно щелкнуть на соответствующей пиктограмме на слое, удерживая клавишу **<Ctrl>**.

Включить любой из указанных режимов также можно при помощи диалогового окна, установив соответствующий флажок.

Диалоговое окно **Layer Properties** также содержит раскрывающийся список **Layer height**, позволяющий задать высоту отображения слоя в процентах.

Объекты наложенного уровня выделяются рамками, в качестве цвета которых можно использовать цвет слоя или фиксированный произвольный цвет. Данные установки могут быть заданы на вкладке **General** в меню **Edit> Preferences** (см. разд. "Настройки программы" гл. 3).

Удаление слоев

В результате удаления слоя удаляются все расположенные на нем кадры монтажной линейки и, соответственно, находящиеся в них элементы фильма. Для того чтобы удалить слой, можно выполнить одно из двух действий:

- выделить слой (или несколько) и щелкнуть на пиктограмме мусорной корзины  в нижней части палитры слоев;
- выполнить команду **Delete Layer** контекстного меню слоя.

В табл. 9.1 обобщены описанные возможности работы со слоями.

Таблица 9.1. Операции со слоями

Операция	Способ
Создать слой	<ol style="list-style-type: none"> Команда Insert>Timeline>Layer Команда Insert Layer контекстного меню слоя Щелкнуть на пиктограмме  Insert Layer в нижней части палитры слоев Команда Modify>Timeline>Distribute to Layers — автоматическое размещение выделенных объектов по слоям
Переименовать	<ol style="list-style-type: none"> Команда Modify>Timeline>Layer Properties Дважды щелкнуть на имени слоя
Выделить слой	<ol style="list-style-type: none"> Щелкнуть на слое в палитре слоев (все кадры слоя выделяются), с <Shift> можно выделить диапазон слоев, с <Ctrl> несколько слоев вразброс Выделить объект на слое Щелкнуть мышью на кадре
Копировать слой	Команда Edit>Timeline>Copy Frames , затем создать новый слой, вставить кадры командой Edit>Timeline>Paste Frames
Изменить порядок расположения слоев	Перетащить слой за его пиктограмму в палитре слоев
Задать свойства слоя	<ol style="list-style-type: none"> Команда Modify>Timeline>Layer Properties Дважды щелкнуть на пиктограмме слоя Команда Properties контекстного меню слоя. В свойствах задаются тип слоя, имя, режимы отображения и редактирования
Удалить слой	<ol style="list-style-type: none"> Выделить слой и щелкнуть на пиктограмме  мусорной корзины палитры слоев Команда Delete Layer контекстного меню слоя

Типы слоев

Документ Flash может содержать слои различных типов. Каждый тип слоя обладает своими характерными особенностями и применяется для выполнения определенных задач. В палитре слоев каждый тип слоя имеет свою пиктограмму, таким образом, по внешнему виду слоя можно определить его принадлежность к тому или иному типу. Для задания типа слоя применяется диалоговое окно **Layer Properties** (см. рис. 9.4), которое можно открыть при

помощи команды **Properties** контекстного меню слоя или двойного щелчка на его пиктограмме. Далее, в табл. 9.2, представлены пиктограммы всех типов слоев и обобщены их основные особенности.

Тип **Normal**

В пределах слоя типа **Normal** (Нормальный) могут быть размещены любые объекты рабочей среды: графика, текст, анимация, видео, звук, сценарии ActionScript. При публикации все содержимое слоя типа **Normal** транслируется в конечный фильм. По умолчанию вновь создаваемому слою автоматически присваивается нормальный тип. Слой типа **Normal** служит отправной точкой для создания других типов слоев. Для того чтобы преобразовать любой другой тип в нормальный тип, необходимо воспользоваться диалоговым окном свойств слоя **Layer Properties** и установить переключатель в положение **Normal**.

Тип **Guide**

Основной особенностью данного типа слоя является то, что его содержимое не транслируется в конечный фильм. Все объекты, размещенные на слое типа **Guide** (Направляющий), видны только в рабочей среде документа. Направляющий слой используется для создания всевозможных вспомогательных элементов, линий разметки, шаблонов и т. д., которые применяются на стадии разработки для точного позиционирования элементов композиции и при создании содержимого сцены. Удобство применения направляющего слоя состоит в том, что перед публикацией не нужно заботиться об удалении всех вспомогательных элементов, которые не должны войти в конечный фильм. Например, в шаблонах Mobile Devices, о которых упоминалось в разд. "Использование стартовой страницы *Start Page*" гл. 3, изображение мобильного устройства помещено на направляющий слой и в конечном фильме видно не будет. Кроме того, при необходимости отключения публикации в конечный фильм тех или иных элементов документа на стадии тестирования проекта можно назначить соответствующим слоям тип **Guide**.

Для создания направляющего слоя можно выполнить одно из двух действий:

- в контекстном меню слоя типа выбрать команду **Guide**;
- выделить слой типа **Normal**, дважды щелкнуть на его пиктограмме в палитре слоев и в появившемся диалоговом окне **Layer Properties** установить переключатель в положение **Guide**.

Типы **Motion Guide** и **Guided**

Слои типа **Motion Guide** (Слой пути) и **Guided** (Ведомый) используются при создании анимации движения по заданной траектории (анимация по маршруту). Эти слои всегда используются в сочетании друг с другом, образуя

группу слоев. Сгруппированные слои в палитре слоев разделяются пунктирной линией. Каждый слой в этой группе выполняет свою определенную функцию. Слой пути (**Motion Guide**) всегда размещается над ведомым и содержит саму траекторию движения. Ведомый слой (**Guided**) всегда размещается под слоем пути и содержит автоматическую анимацию движения (**Motion tween**), реализующую линейное перемещение объекта из начального положения в конечное (*подробнее об автоматической анимации см. в гл. 13*). Пиктограммы ведомых слоев отображаются с отступом. При установке соответствующих параметров анимации применение группы "слой пути — ведомый слой" приводит к тому, что объект, расположенный на ведомом слое, перемещается по траектории, находящейся на слое пути. Используя данный прием, можно реализовать автоматическое перемещение объекта по криволинейной траектории любой сложности, затратив на это минимум усилий (*процесс создания анимации по маршруту описывается в гл. 13*).

Для создания группы слоев, состоящей из слоя пути и ведомого слоя, можно воспользоваться одним из перечисленных ниже способов.

- Выделить слой типа **Normal** и щелкнуть на пиктограмме **Add Motion Guide**  в нижней части палитры слоев. При этом в палитре автоматически будет добавлен слой пути, а текущему слою будет назначен тип **Guided**.
- Выделить слой типа **Normal** и выполнить команду главного меню **Insert>Timeline>Motion Guide**.
- Создать два слоя типа **Normal**, расположив их друг над другом. Назначить верхнему слою тип **Guide**, после чего подгруппировать к нему нижележащий слой типа **Normal**. Подгруппировать слой можно одним из следующих способов.
 - Дважды щелкнуть на пиктограмме слоя и в диалоговом окне **Layer Properties** установить переключатель в положение **Guided**.
 - Взяться за пиктограмму нижележащего слоя и протянуть ее слегка вверх и вправо (при этом пиктограмма вышележащего слоя типа **Guide** затеняется). В результате группировки верхнему слою автоматически назначается тип **Motion Guide**, а текущему нижележащему слою — тип **Guided**. Не следует протягивать слой слишком высоко — это приведет к его перемещению в палитре слоев.

Для того чтобы разгруппировать слои, можно либо щелкнуть на слое пути правой кнопкой мыши и в контекстном меню снять флагок **Guide**, либо взяться за пиктограмму ведомого слоя и слегка потянуть вниз и влево.

Примечание

К одному слою пути можно подгруппировать любое количество ведомых слоев. На практике это означает, что по одной и той же траектории может перемещаться любое количество объектов, причем в различных направлениях.

Типы **Mask** и **Masked**

Как и предыдущие, данные слои также всегда используются в сочетании друг с другом, образуя группу слоев. В этой группе верхний слой всегда имеет тип **Mask** (Слой-маска), а нижний (или несколько) — тип **Masked** (Маскируемый). Такая группа позволяет реализовать эффект маски, который заключается в том, что содержимое маскируемого слоя (или нескольких слоев) видно только сквозь векторные формы, размещенные на слое типа **Mask**.

Слой-маску можно сравнить со стеклом, закрашенным черной непрозрачной краской. Создание на этом слое векторной формы эквивалентно стиранию краски на соответствующей области, в результате чего образуется прорезь, сквозь которую можно видеть содержимое нижележащих маскируемых слоев. Следует отметить, что действие слоя-маски распространяется только на сгруппированные с ним маскируемые слои и не оказывает влияния на другие слои монтажной линейки.

При создании слоя-маски **Mask** следует иметь в виду следующие особенности:

- на слое-маске должны быть размещены векторные формы, т. е. области. Контуры не производят никакого действия, поэтому от их применения следует отказаться;
- на слое-маске можно разместить либо *любое* число объектов рабочего уровня, либо *только один* объект наложенного уровня (группу, текстовый блок, экземпляр символа). При использовании объектов рабочего и наложенного уровней одновременно приоритет отдается объектам рабочего уровня;
- цвет и степень прозрачности объектов на слое-маске не играет роли, поскольку на результат действия маски влияет только ее форма. В связи с этим в качестве цвета маски нужно использовать сплошной (solid) цвет;
- маска не может быть использована в символах типа **Button**;
- для включения действия маски в рабочей среде Flash необходимо заблокировать слой-маску и маскируемый слой. В конечном фильме маска будет действовать вне зависимости от того, были ли заблокированы слои при публикации или нет.

Для создания группы слоев, состоящих из слоя-маски и маскируемого слоя, можно выполнить одно из следующих действий.

- Создать два слоя типа **Normal**, расположив их друг над другом. Щелкнуть на верхнем слое правой кнопкой мыши и из контекстного меню выбрать команду **Mask**. Слои автоматически заблокируются.
- Создать два слоя типа **Normal**. При помощи диалогового окна **Layer Properties** последовательно назначить им типы **Mask** и **Masked**.
- Создать два слоя типа **Normal**. При помощи диалогового окна **Layer Properties** назначить верхнему из них тип **Mask**, после чего вручную

подгруппировать к нему нижележащий слой так же, как это было сделано со слоем пути и ведомым слоем.

Для того чтобы разгруппировать слои, можно либо щелкнуть на слое-маске правой кнопкой мыши и в контекстном меню снять флажок **Mask**, либо взяться за пиктограмму маскируемого слоя и слегка потянуть вниз и влево.

Flash не поддерживает использование прозрачных масок, но этот эффект может быть симитирован при помощи обычной маски и дополнительного слоя, содержащего форму с полупрозрачной градиентной заливкой (рис. 9.5). В качестве примера можно рассмотреть создание эффекта, имитирующего растушевку краев растрового изображения.

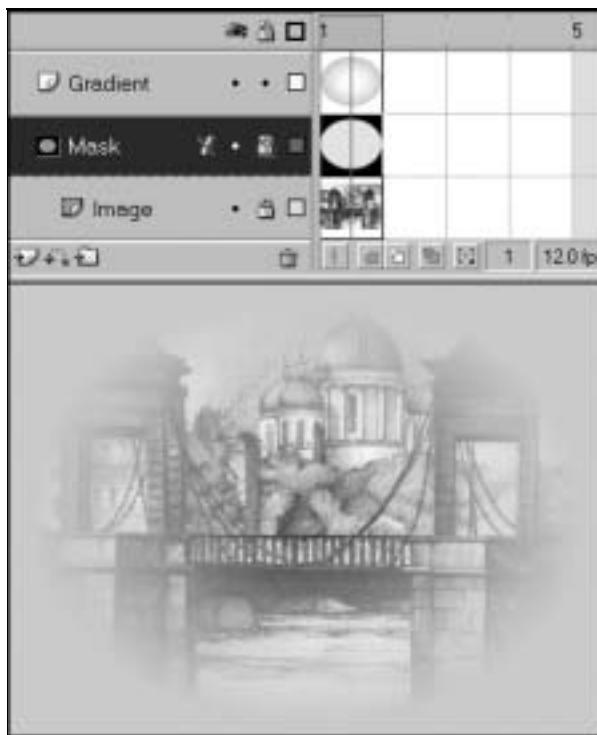


Рис. 9.5. Имитация прозрачной маски

Для его реализации необходимо выполнить следующие действия:

1. Создать два нормальных слоя.
2. Верхнему слою назначить тип **Mask**, нижнему — тип **Masked** и назвать их **Mask** и **Image** соответственно. Если слои заблокированы, снять блокировку.

3. На слой **Image** импортировать растровое изображение (**File>Import>Import to Stage**).
4. На слое **Mask** нарисовать круг или овал (без контура), используя сплошной цвет в качестве заливки.
5. Создать новый нормальный слой (**Insert>New Layer**), назвать его **Gradient** и сделать самым верхним в палитре слоев.
6. Скопировать векторную форму со слоя **Mask** (**Edit>Copy**) и вставить ее в точку с теми же координатами на слой **Gradient** (**Edit>Paste in Place**).
7. Открыть палитру **Color Mixer** (**Window>Color Mixer**) и создать радиальный градиент, состоящий из двух цветовых порогов, цвет которых совпадает с цветом рабочей области. Необходимо сделать цвет левого порога градиента (центр градиента) полностью прозрачным (**Alpha = 0**). Добавить получившийся градиент в каталог цветов (**Add Swatch**).
8. Залить круг на слое **Gradient** синтезированным градиентом.
9. Заблокировать слои **Mask** и **Image**.

Тип **Folder**

Тип **Folder** (Слой-папка) не содержит кадров и используется для более компактной организации слоев в палитре. Для того чтобы создать слой-папку, можно выполнить одно из следующих действий:

- щелкнуть на кнопке  **Insert Layer Folder** в нижней части палитры слоев;
- в контекстном меню слоя типа выбрать команду **Insert Folder**;
- выделить слой типа **Normal**, дважды щелкнуть на его пиктограмме в палитре слоев, и в диалоговом окне **Layer Properties** установить переключатель в положение **Folder**;
- выполнить команду главного меню **Insert>Timeline>Layer Folder**.

Выделив один или несколько слоев, их можно переместить в папку. Для того чтобы развернуть или свернуть папку, необходимо щелкнуть на пиктограмме белого треугольника, расположенной слева от пиктограммы слоя-папки. Слой-папки могут вкладываться друг в друга. При удалении слой-папки автоматически удаляются все находящиеся в нем слои.

Таблица 9.2. Типы слоев и их назначение

Тип и пиктограмма слоя	Назначение
Нормальный Normal 	Может содержать любые объекты (статичную графику, анимацию, звук, сценарии, интерактивные элементы), которые будут транслированы в конечный фильм

Таблица 9.2 (окончание)

Тип и пиктограмма слоя	Назначение
Направляющий Guide Layer 	Используется для создания вспомогательных элементов, произвольных линий разметки, макетов, шаблонов, позволяющих позиционировать или создавать объекты на других слоях; содержимое не публикуется в конечный фильм
Слой пути Motion Guide 	Содержит траекторию движения анимированного объекта (только непрерывный, незамкнутый контур), содержимое не видно в конечном фильме, связан с одним или несколькими ведомыми слоями
Ведомый слой Guided 	Сгруппирован со слоем пути, содержит автоматическую анимацию движения (Motion Tween), реализующую линейное перемещение объекта из начального положения в конечное. Ведомых слоев может быть несколько
Слой-маска Mask Layer 	Содержит трафарет (векторная форма, текст окрашенные сплошным цветом), через который видны нижние подгруппированные маскированные слои. Может содержать анимацию
Маскируемый слой Masked 	Содержимое маскируемого слоя видно только сквозь векторные формы (области), расположенные на слое-маске. Может содержать анимацию. Маскируемых слоев может быть несколько
Слой-папка Folder 	Служебный слой, используется для компактной организации слоев в палитре. Может содержать вложенные слой-папки

Работа с кадрами

При создании нового документа Flash его монтажная линейка содержит только один кадр фильма. Остальная область монтажной линейки представляет собой пустые ячейки, в которые могут быть помещены кадры. В результате добавления новых кадров фильма на монтажную линейку при его воспроизведении воспроизводящая головка будет автоматически перемещаться из кадра в кадр со скоростью, заданной в окне настройки глобальных параметров фильма **Document Properties (Modify>Document)**.

Контекстное меню монтажной линейки позволяет задать различные режимы отображения кадров. Для этого необходимо щелкнуть на пиктограмме , расположенной над вертикальной полосой прокрутки монтажной линейки. Контекстное меню содержит следующие команды:

- Tiny** (Очень маленький), **Small** (Маленький), **Normal** (Нормальный), **Medium** (Средний), **Large** (Большой) — задание размеров отображения

кадров монтажной линейки. При необходимости работать с большим диапазоном кадров удобно уменьшить их размер, чтобы одновременно видеть максимальное количество кадров;

- Short** (Невысокий) — данная команда позволяет уменьшить высоту слоев и, соответственно, кадров, что приводит к их более компактному размещению в пределах монтажной линейки;
- Tinted Frames** (Подсветка кадров) — при включении данного режима кадры монтажной линейки окрашиваются в соответствии с их типом. Это облегчает задачу идентификации типа кадра и позволяет лучше ориентироваться на монтажной линейке;
- Preview** (Просмотр), **Preview in Context** (Просмотр в контексте) — режимы просмотра содержимого кадров непосредственно на монтажной линейке. В режиме **Preview** в кадре монтажной линейки отображаются только графические объекты, содержащиеся в нем, а в режиме **Preview in Context** — графические объекты, содержащиеся в кадре вместе с рабочей областью. Для того чтобы отменить режим просмотра, нужно выбрать любой из описанных выше размеров кадра в данном меню.

Виды кадров

Монтажная линейка может содержать три основных типа кадров, имеющих различное назначение и область применения. К их числу относятся:

- ключевые кадры — **Keyframes**;
- простые (обычные) кадры — **Frames**;
- кадры трансформации — **Tweened frames**.

Ключевые кадры

Ключевые кадры используются при необходимости подвергнуть содержимое сцены любым изменениям. Ключевые кадры содержат информацию обо всех объектах, расположенных на сцене и об изменении их параметров: положения, цветовых характеристик, трансформации и т. д. Введение в фильм нового элемента, любое изменение его характеристик и удаление элемента из фильма требуют применения ключевых кадров. Ключевой кадр также необходим для размещения звука или сценария ActionScript на монтажной линейке. Первый кадр фильма всегда является ключевым. Ключевые кадры оказывают существенное влияние на объем конечного фильма, поскольку они, как правило, содержат уникальную информацию.

Содержимое каждого ключевого кадра может редактироваться изолированно от остальных ключевых кадров.

Ключевой кадр может быть *заполненным* или *пустым*. Заполненным является ключевой кадр, содержащий графические объекты, на монтажной линейке в

режиме Tinted Frames он отображается в виде серого прямоугольника с за-полненным кружком . Пустым ключевым кадром является ключевой кадр, не содержащий графических объектов. На монтажной линейке он отображается в виде белого прямоугольника с пустым кружком . Пустой ключевой кадр может содержать метку, звук или сценарий ActionScript.

Для того чтобы создать заполненный ключевой кадр, необходимо выделить ячейку монтажной линейки, в которую он должен быть помещен, и выполнить одно из следующих действий:

- щелкнуть на ячейке правой кнопкой мыши и из контекстного меню кадра выбрать команду **Insert Keyframe**;
- выполнить команду главного меню **Insert>Timeline>Keyframe**;
- щелкнуть на ячейке правой кнопкой мыши и из контекстного меню кадра выбрать команду **Convert to Keyframes**;
- выполнить команду главного меню **Modify>Timeline>Convert to Keyframes**;
- нажать клавишу <F6>.

В результате выполнения любого из этих действий будет создан ключевой кадр и в него автоматически будет скопировано графическое содержимое предыдущего ключевого кадра. Если данному ключевому кадру предшествует пустой ключевой, то созданный кадр также будет пустым. Если выделить несколько ячеек монтажной линейки, то при помощи любой из трех последних команд можно создать несколько ключевых кадров.

Для того чтобы создать пустой ключевой кадр, необходимо выделить ячейку монтажной линейки, в которую он должен быть помещен, и выполнить одно из следующих действий:

- щелкнуть на ячейке правой кнопкой мыши и из контекстного меню кадра выбрать команду **Insert Blank Keyframe**;
- выполнить команду главного меню **Insert>Timeline>Blank Keyframe**;
- щелкнуть на ячейке правой кнопкой мыши и из контекстного меню кадра выбрать команду **Convert to Blank Keyframes**;
- выполнить команду главного меню **Modify>Timeline>Convert to Blank Keyframes**;
- нажать клавишу <F7>.

Если выделить несколько ячеек монтажной линейки, то при помощи любой из трех последних команд можно создать несколько пустых ключевых кадров.

Простые кадры

В отличие от ключевых, простые кадры не содержат никакой новой информации, а только отображают содержимое предшествующего им ключевого кадра.

Простые кадры используются для того, чтобы продлить видимость предшествующего ключевого кадра. Простые кадры не оказывают влияние на объем конечного фильма, поскольку они не содержат новой информации, а используются для заполнения. Например, если ключевой кадр содержит фоновое изображение, то оно должно быть видно на протяжении всего фильма. Между тем при скорости 12 fps каждый кадр экспонируется в течение 1/12 секунды. К решению этой задачи можно подойти двумя способами. Первый предполагает использование ряда последовательно расположенных ключевых кадров, содержащих одно и то же фоновое изображение. Такой способ является *крайне* неэффективным, поскольку один и тот же элемент без изменений перерисовывается из кадра в кадр, что приводит к неоправданному увеличению конечного объема файла и трудоемкости работы. Второй способ основывается на применении простых кадров, заполняющих весь диапазон, на протяжении которого необходимо отображать фоновое изображение, расположенное в ключевом кадре. В этом случае фон будет использован только один раз, что позволит уменьшить объем конечного файла. Очевидно, что второй способ является значительно более рациональным и эффективным. Таким образом, для того, чтобы продлить время видимости и действия содержимого ключевого кадра, *необходимо* разместить после него диапазон простых кадров, соответствующий этому времени.

Поскольку простые кадры не содержат уникальной информации, редактировать их без изменения содержимого предшествующего ключевого кадра невозможно.

Как и ключевые, простые кадры могут быть *заполненными* или *пустыми*. Простой кадр является заполненным, если он следует за заполненным ключевым кадром; на монтажной линейке в режиме Tinted Frames он отображается в виде серого прямоугольника. Простой кадр является пустым, если он следует за пустым ключевым кадром; на монтажной линейке он отображается в виде белого прямоугольника. Последний кадр в диапазоне простых кадров называется *концевым* и на монтажной линейке содержит изображение прямоугольника .

При создании простого кадра ему обязательно должен предшествовать ключевой. Если между выделенной пустой ячейкой монтажной линейки, куда должен быть помещен простой кадр, и предшествующим ему ключевым кадром имеется интервал, то при создании простого кадра этот интервал автоматически заполнится простыми кадрами. Таким образом, для создания диапазона простых кадров достаточно создать простой кадр, поместив его в положение, соответствующее концу этого диапазона. Чтобы создать один или несколько простых кадров, необходимо выделить ячейку монтажной линейки, соответствующую положению последнего простого кадра диапазона, и выполнить одно из следующих действий:

- щелкнуть на ячейке правой кнопкой мыши и из контекстного меню кадра выбрать команду **Insert Frame**;

- выполнить команду главного меню **Insert>Timeline>Frame**;
- нажать клавишу <F5>;
- если ключевые кадры создаются с определенным интервалом (например, с номерами 1, 12, 24, 36 и т. д.), то промежутки между ними автоматически заполняются простыми кадрами.

Кадры трансформации

Кадры трансформации представляют собой единую последовательность кадров, применяемую при использовании автоматической анимации. Кадры трансформации формируются и обрабатываются автоматически и описывают переход анимируемого объекта из начального состояния в конечное. Начальное и конечное состояния задаются в соответствующих ключевых кадрах диапазона автоматической анимации. На монтажной линейке в режиме Tinted Frames кадры трансформации окрашиваются зеленым или голубым цветами (в зависимости от типа автоматической анимации) и содержат стрелку. Подробная информация о создании и работе с автоматической анимацией содержится в гл. 13.

Информация о типах кадров, их назначении и обозначении на монтажной линейке сведена в табл. 9.3.

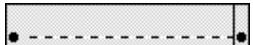
Таблица 9.3. Типы кадров

Тип	Вид кадра на монтажной линейке	Назначение и содержание кадра	Способ создания
Ключевой кадр Keyframe		Используется для любого изменения содержимого сцены — добавление, удаление объектов из кадра, изменения в анимации	Команда Insert>Timeline>Keyframe ; <F6>, контекстное меню кадра
Пустой ключевой кадр Blank Keyframe		Ключевой кадр без графического содержимого	Команда Insert>Timeline>Blank Keyframe ; <F7>, контекстное меню кадра
Простой заполненный кадр Frame	Светло-серый прямоугольник	Следует после ключевого кадра, отображает содержимое предшествующего ключевого кадра; используется для увеличения времени его экспонирования (просмотра)	Команда Insert>Timeline>Frame ; <F5>, контекстное меню кадра

Таблица 9.3 (продолжение)

Тип	Вид кадра на монтажной линейке	Назначение и содержание кадра	Способ создания
Простой пустой кадр Frame	Белый прямоугольник	Следует после пустого ключевого кадра	Команда Edit>Clear – очистка содержимого предшествующего ключевого кадра
Простой концевой кадр		Простой кадр, используется для регулирования длины диапазона простых кадров	Последний в диапазоне простых кадров
Ключевой кадр, содержащий код AS		Содержит код ActionScript, простые пустые кадры после него также содержат данный сценарий	Команда Window>Development Panels>Actions , <F9>
Метка label , комментарий comment , или якорь anchor	Красный флажок, с подписью, текстовая строка комментария или значок якоря с названием в заполненном или пустом кадре 	Содержит метку (имя) кадра, используемую для обращения к нему, комментарий автора фильма или якорь, применяемый при перемещении между разделами конечного проекта при помощи кнопок браузера "Вперед", "Назад"	Инспектор свойств, поле Frame , тип Label Type
Кадры трансформации Motion tween , анимация движения	Ключевые кадры отмечены черной точкой, кадры трансформации – черная стрелка на синем фоне 	Кадры трансформации, единая последовательность кадров. Два ключевых кадра в начале и в конце диапазона анимации задают начальное и конечное состояние объекта, между ними промежуточные кадры, отображающие трансформацию объекта	1. Инспектор свойств, список Tween , тип Motion . 2. Команда Insert>Timeline>Create Motion Tween – создание анимации движения

Таблица 9.3 (окончание)

Тип	Вид кадра на монтажной линейке	Назначение и содержание кадра	Способ создания
Кадры трансформации Shape tween автоматического изменения формы	Ключевые кадры отмечены черной точкой, кадры трансформации — черная стрелка на зеленом фоне		Инспектор свойств, список Tween , тип Shape
Ошибка в автоматической анимации	Пунктирная линия		

Операции с кадрами

Выделение кадров

Рабочая среда Flash предоставляет два различных режима выделения кадров: *покадровый* и *интервальный*. Покадровый режим выделения используется по умолчанию. Для того чтобы переключиться на интервальный режим выделения, необходимо установить флажок **Span based selection** на вкладке **General** диалогового окна **Preferences** (**Edit>Preferences**). Снятие данного флажка снова активизирует покадровый режим выделения кадров.

Покадровое выделение

Для выделения кадра монтажной линейки достаточно просто щелкнуть на нем левой кнопкой мыши. Выделенный кадр подсвечивается черным цветом. Щелчок на кадре правой кнопкой мыши также позволяет выделить его, при этом вызывая контекстное меню кадра.

Для того чтобы выделить сразу несколько последовательно расположенных кадров монтажной линейки, нужно щелкнуть на первом кадре последовательности и, не отпуская кнопку мыши, протянуть курсор через весь требуемый диапазон. Таким образом можно выделить диапазон кадров на нескольких соседних слоях.

Выделить диапазон кадров можно, дважды щелкнув на любом кадре, принадлежащем данному диапазону. Диапазон последовательно расположенных кадров также можно выделить, используя клавишу **<Shift>**. Для этого необ-

ходимо выделить первый кадр диапазона и затем щелкнуть на последнем кадре диапазона, удерживая клавишу **<Shift>**.

Для того чтобы выделить несколько кадров вразбивку или несколько непоследовательно расположенных диапазонов кадров, необходимо использовать клавишу **<Ctrl>**.

Интервальное выделение

Использование данного режима выделения позволяет одним щелчком выделять целый диапазон кадров. Так, щелчок на простом кадре выделяет весь диапазон простых кадров вместе с предшествующим им ключевым вплоть до следующего ключевого кадра. Щелчок на кадре трансформации выделяет весь диапазон кадров автоматической анимации, включая начальный и конечный ключевые кадры данного диапазона. Однако щелчок на ключевом кадре выделяет только данный ключевой кадр.

Для того чтобы выделить единичные простые кадры или кадры трансформации вразбивку, необходимо использовать клавишу **<Ctrl>**. Для того чтобы выделить единичный кадр, можно также щелкнуть по нему правой кнопкой мыши. Это приведет к выделению кадра и вызову контекстного меню. Для выделения произвольных диапазонов кадров нужно щелкнуть на первом кадре первого диапазона, удерживая клавишу **<Ctrl>** и, не отпуская кнопку мыши, протянуть курсор, после чего повторить эту процедуру для следующего диапазона.

Использование сочетания клавиш **<Ctrl>** и **<Shift>** позволяет выделить произвольный диапазон последовательно расположенных кадров. Для этого необходимо щелкнуть с **<Ctrl>** на первом кадре (или протянуть вдоль нескольких, расположенных на соседних слоях кадрах) и затем щелкнуть на последнем кадре (или протянуть вдоль нескольких, расположенных на соседних слоях кадрах) диапазона, удерживая **<Ctrl>** и **<Shift>** одновременно.

Кроме того, вне зависимости от используемого режима выделения, выделить все кадры монтажной линейки можно при помощи команды главного меню **Edit>Timeline>Select all Frames** (**<Ctrl>+<Alt>+<A>**) или одноименной команды контекстного меню кадра.

Щелчок на слое в палитре слоев выделяет все расположенные на нем кадры монтажной линейки.

При выделении единичного ключевого кадра на сцене автоматически выделяются все расположенные в нем графические объекты.

Перемещение кадров

Для того чтобы переместить кадр (или несколько кадров), его необходимо предварительно выделить. Взявшись курсором за любой кадр выделенного диапазона, можно переместить весь диапазон. При перемещении кадров ря-

дом с курсором появляется пиктограмма рамки , а перемещаемые кадры отображаются серой рамкой. Выделенные кадры можно перемещать с одного слоя на другой.

Перемещая концевой кадр, можно регулировать длину диапазона простых кадров, изменяя интервал времени, в течение которого экспонируется предшествующий им ключевой кадр. При работе в покадровом режиме выделения для выполнения данной процедуры необходимо перемещать концевой кадр, удерживая клавишу <Ctrl>. Курсор при этом принимает вид горизонтальной двунаправленной стрелки .

При работе в интервальном режиме выделения кадров можно просто перемещать ключевой или концевой кадры без их предварительного выделения.

Копирование/Вставка кадров

При копировании кадров они помещаются в буфер обмена вместе со всем содержимым. Вставить кадры из буфера обмена можно либо на ту же монтажную линейку, откуда они были скопированы (или вырезаны), либо на монтажную линейку символа (см. гл. 10).

Копирование (Copy)/Вырезание (Cut) в буфер

Для того чтобы скопировать кадр (или несколько кадров) в буфер, его необходимо предварительно выделить. Скопировать выделенный кадр или диапазон кадров можно, выполнив одно из следующих действий:

- выполнить команду главного меню **Edit>Timeline>Copy Frames** или применить ее клавиатурный эквивалент <Ctrl>+<Alt>+<C>;
- выполнить команду контекстного меню кадра **Copy Frames**.

Примечание

Следует понимать разницу между командами **Edit>Copy** (<Ctrl>+<C>) и **Edit>Timeline>Copy Frames** (<Ctrl>+<Alt>+<C>). Первая копирует в буфер только выделенные объекты сцены, вторая копирует в буфер кадр целиком со всеми расположенными в нем объектами.

Для того чтобы вырезать кадр (или несколько кадров) в буфер, его также необходимо предварительно выделить. Вырезать выделенный кадр или диапазон кадров в буфер можно, выполнив одно из следующих действий:

- выполнить команду главного меню **Edit>Timeline>Cut Frames** или применить ее клавиатурный эквивалент <Ctrl>+<Alt>+<X>;
- выполнить команду контекстного меню кадра **Cut Frames**.

При вырезании в буфер простого кадра или кадра трансформации на его месте создается пустой ключевой кадр.

Вставка (Paste) из буфера

Перед выполнением вставки кадров из буфера обмена необходимо выделить кадр или пустую ячейку монтажной линейки, куда будет вставлен кадр или диапазон кадров. При вставке диапазона номер выделенного кадра или ячейки монтажной линейки будет соответствовать первому кадру вставляемого диапазона. Если перед вставкой выделить несколько кадров монтажной линейки, то в результате вставки они будут заменены. Для вставки кадров необходимо выполнить одно из следующих действий:

- выполнить команду главного меню **Edit>Timeline>Paste Frames** или применить ее клавиатурный эквивалент **<Ctrl>+<Alt>+<V>**;
- выполнить команду контекстного меню кадра **Paste Frames**.

Для того чтобы быстро скопировать и вставить кадры, можно выделить требуемый диапазон и перетащить его в другой участок монтажной линейки, удерживая при этом клавишу **<Alt>**. Рядом с курсором при этом появляется пиктограмма плюса.

Преобразования типов кадров

Для того чтобы преобразовать ключевой кадр в простой, необходимо выделить его и выполнить одно из следующих действий:

- выполнить команду главного меню **Modify>Timeline>Clear Keyframe** или применить ее клавиатурный эквивалент **<Shift>+<F6>**;
- выполнить команду контекстного меню кадра **Clear Keyframe**.

Простой кадр может быть преобразован в пустой или в заполненный ключевой кадр. Для того чтобы преобразовать простой кадр в пустой ключевой, необходимо выделить его и выполнить одно из следующих действий:

- выполнить команду главного меню **Modify>Timeline>Convert to Blank Keyframes** или применить ее клавиатурный эквивалент **<F7>**;
- выполнить команду контекстного меню кадра **Convert to Blank Keyframes**.

Для того чтобы преобразовать простой кадр в заполненный ключевой, необходимо выделить его и выполнить одно из следующих действий:

- выполнить команду главного меню **Modify>Timeline>Convert to Keyframes** или применить ее клавиатурный эквивалент **<F6>**;
- выполнить команду контекстного меню кадра **Convert to Keyframes**.

В результате выполнения этой процедуры в созданный ключевой кадр будут скопированы графические объекты из предыдущего ключевого кадра.

Изменение порядка следования кадров

Flash позволяет автоматически изменить порядок следования кадров на обратный. Этую процедуру можно применить к выделенному диапазону кадров, содержащему как минимум два ключевых кадра.

Для того чтобы автоматически разместить кадры в обратном порядке, изменив, таким образом, последовательность отображения их содержимого, можно выполнить одно из следующих действий:

- выполнить команду главного меню **Modify>Timeline>Reverse Frames**;
- выполнить команду контекстного меню кадра **Reverse Frames**.

Данная возможность может быть использована при создании анимации откапного движения (см. гл. 12). Для анимации любого симметричного движения, таким образом, достаточно отрисовать только половину полного цикла, вторая половина может быть получена в результате копирования диапазона кадров и изменения порядка их следования. Например, анимация одного цикла движения маятника требует отрисовки его перемещения только из одного крайнего положения в другое в одном направлении. Перемещение в обратном направлении получается в результате копирования и изменения порядка следования уже отрисованных кадров.

Очистка содержимого и удаление кадров

Для того чтобы очистить кадр (или несколько кадров), удалив из него содержимое (включая сценарии и метки), необходимо выделить его и выполнить одно из следующих действий:

- выполнить команду главного меню **Edit>Timeline>Clear Frames** или применить ее клавиатурный эквивалент **<Alt>+<Backspace>**;
- выполнить команду контекстного меню кадра **Clear Frames**.

Применение данной команды к простому кадру превращает его в пустой ключевой кадр. Удалить графическое содержимое из кадра можно также, выделив его и нажав клавишу **<Delete>** или **<Backspace>**.

Для того чтобы удалить кадр или диапазон кадров с монтажной линейки, необходимо выделить удаляемые кадры и выполнить одно из следующих действий:

- выполнить команду главного меню **Edit>Timeline>Remove Frames** или применить ее клавиатурный эквивалент **<Shift>+<F5>**;
- выполнить команду контекстного меню кадра **Remove Frames**;
- вырезать выделенные кадры с монтажной линейки (**Edit>Timeline>Cut Frames**).

При удалении кадров с монтажной линейки все их содержимое (включая графику, звук, сценарии, метки) будет удалено.

Описание операций с кадрами и способов их выполнения обобщено в табл. 9.4.

Таблица 9.4. Операции с кадрами монтажной линейки

Операция	Способ
Создание заполненного ключевого кадра	Команда Insert>Timeline>Keyframe , <F6> (автоматически в новый ключевой копируется содержимое предыдущего ключевого); команда контекстного меню кадра Insert Keyframe
Создание пустого ключевого кадра	Команда Insert>Timeline>Blank Keyframe , <F7>; команда контекстного меню кадра Insert Blank Keyframe
Создание простого кадра	Команда Insert>Timeline>Frame , <F5>; команда контекстного меню кадра Insert Blank Keyframe ; если ключевые кадры создаются с определенным интервалом, то промежутки между ними автоматически заполняются простыми кадрами
Выделение в покадровом режиме	кадра Щелчок на кадре монтажной линейки; с <Ctrl> — выделение кадров вразброс
	диапазона кадров Выделить первый кадр диапазона, затем с <Shift> щелкнуть на последнем кадре диапазона; 2 щелчка по любому кадру диапазона
	всех кадров Edit>Timeline>Select All Frames , <Ctrl>+<Alt>+<A>, команда контекстного меню кадра Select All Frames ; щелчок по слову выделяет все кадры слоя
Выделение в интервальном режиме (Span based selection)	кадра Щелчок на кадре монтажной линейки с <Ctrl>
	диапазона кадров Щелчок по любому кадру диапазона; сочетание клавиш <Ctrl> и <Shift> позволяет выделить произвольный диапазон последовательно расположенных кадров
	всех кадров Edit>Timeline>Select All Frames , <Ctrl>+<Alt>+<A>, команда контекстного меню кадра Select All Frames ; щелчок на слове выделяет все кадры слоя
Перемещение кадра	Выделить кадр или несколько кадров и перетащить мышью в требуемое положение
Изменение длины диапазона кадров	В покадровом режиме с <Ctrl> переместить концевой или ключевой кадр, изменения соответствующий диапазон. В интервальном режиме просто переместить концевой или ключевой кадр
Копирование кадров	Выделить кадры и выполнить команду Edit>Timeline>Copy Frames , <Ctrl>+<Alt>+<C> или команду контекстного меню кадра Copy Frames

Таблица 9.4 (окончание)

Операция	Способ
Вырезание кадров	Выделить кадры и выполнить команду Edit>Timeline>Cut Frames , <Ctrl>+<Alt>+<X> или команду контекстного меню кадра Cut Frames
Вставка скопированных кадров	Выделить кадр, несколько кадров или пустую ячейку монтажной линейки, соответствующих началу вставляемого диапазона и выполнить команду меню Edit>Timeline>Paste Frames , <Ctrl>+<Alt>+<V> или команду контекстного меню кадра Paste Frames
Преобразование ключевого кадра в простой	Выделить ключевые кадры и выполнить команду Modify>Timeline>Clear Keyframe , <Shift>+<F6> или команду контекстного меню кадра Clear Keyframe
Преобразование простого кадра в заполненный ключевой	Выделить простые кадры и выполнить команду Modify>Timeline>Convert to Keyframes , <F6> или команду контекстного меню кадра Convert to Keyframes
Преобразование простого кадра в пустой ключевой	Выделить простые кадры и выполнить команду Modify>Timeline> Convert to Blank Keyframes , <F7> или команду контекстного меню кадра Convert to Blank Keyframes
Изменение направления анимации	Выделить последовательность кадров, включающую несколько ключевых кадров, и выполнить команду Modify>Timeline>Reverse Frames или команду контекстного меню кадра Reverse Frames
Очистка содержимого кадра	Выделить кадры и выполнить команду Edit>Timeline>Clear Frames , <Alt>+<Backspace> или команду контекстного меню кадра Clear Frames
Удаление кадров	Выделить кадры и выполнить команду Edit>Timeline>Remove Frames , <Shift>+<F5> или команду контекстного меню кадра Remove Frames
Редактирование свойств кадра	Щелкнуть на кадре и открыть Инспектор свойств (Window>Properties , <Ctrl>+<F3>)
Режим калькирования Onion Skin и режим множественного редактирования кадров Edit Multiple Frames	Использовать пиктограммы в нижней части монтажной линейки  . Диапазон калькирования или множественного редактирования устанавливается при помощи маркеров в верхней части монтажной линейки

Метки, комментарии и указатели

Кроме графических объектов, звука и сценария, кадр может содержать метку (Label), комментарий (Comment) или указатель (якорь) (Named anchor).

Метка (Label)

Метка представляет собой имя кадра и используется в качестве его идентификатора. Сценарий ActionScript может обращаться к кадру по номеру или по метке. Использование меток является предпочтительным, поскольку они позволяют избежать жесткой привязки кадра к его положению на монтажной линейке. Так, если в сценарии имеются ссылки на кадр, использующие его номер, то при перемещении кадра потребуется корректировка соответствующих фрагментов кода. Использование метки позволяет избежать этой проблемы, поскольку метка перемещается вместе с кадром. Рекомендуется размещать метки в отдельном слое. Для того чтобы создать метку, необходимо щелкнуть на ключевом кадре монтажной линейки и в поле **Frame** Инспектора свойств ввести текст метки. В списке **Label type** нужно выбрать значение **Name**.

На монтажной линейке метка отображается в соответствующем кадре рядом с пиктограммой красного флагшка (см. табл. 9.3).

Комментарий (Comment)

Комментарий может содержать произвольный текст (строку) и предназначается для создания пометок, помогающих лучше ориентироваться на монтажной линейке при наличии большого числа кадров, либо для других вспомогательных отметок. Комментарий используется только в рабочей среде документа. Для того чтобы создать комментарий, необходимо щелкнуть на ключевом кадре монтажной линейки и в поле **Frame** Инспектора свойств ввести соответствующий текст. В списке **Label type** нужно выбрать значение **Comment**. Текст комментария начинается со знака "//", это обозначение принято в ActionScript для создания комментариев. Комментарии, как и метки, рекомендуется размещать в отдельном слое.

На монтажной линейке комментарий выводится в соответствующем кадре монтажной линейки рядом с пиктограммой зеленых кавычек (см. табл. 9.3).

Указатель (Named anchor)

Указатели или якоря — это особый тип идентификатора кадра, который применяется для того, чтобы при просмотре результирующего фильма в браузере можно было бы перемещаться между кадрами, содержащими якоря при помощи кнопок браузера **Forward** (Вперед) и **Back** (Назад). Якорь также

может использоваться для адресации кадра в сценарии ActionScript. В названии якоря нужно избегать использования символов кириллического алфавита. Для того чтобы якоря функционировали в конечном фильме, необходимо при публикации использовать шаблон **Flash with Named Anchors** (см. гл. 17). Для того чтобы создать указатель, необходимо щелкнуть на ключевом кадре монтажной линейки и в поле **Frame** Инспектора свойств ввести название. В списке **Label type** нужно выбрать значение **Anchor**. Якоря, как и метки, рекомендуется размещать в отдельном слое.

На монтажной линейке указатель выводится в соответствующем кадре монтажной линейки рядом с пиктограммой золотого якоря (см. табл. 9.3).

Покадровая анимация

Покадровая (Frame by frame animation), или *ручная* анимация предполагает использование ключевых кадров для размещения в них отдельных фаз движения. Если та или иная фаза должна экспонироваться в течение некоторого промежутка времени, более длительного, чем время просмотра ключевого кадра, то после данного ключевого кадра размещается необходимое число простых кадров. Таким образом, покадровая анимация представляет собой последовательность ключевых и простых кадров. Ключевые кадры содержат основные фазы движения, а простые служат для продления времени их просмотра.

При создании покадровой анимации необходимо придерживаться следующих правил.

- **Разделение статики и динамики** — статичные объекты, не подвергающиеся анимации, должны размещаться на отдельных слоях. Так, например, фоновое изображение, остающееся неподвижным в течение всего фильма, должно быть помещено на отдельный слой, в противном случае его придется перерисовывать из кадра в кадр, что в итоге приведет к запутанности структуры фильма и увеличению объема конечного файла.
- **Разделение одновременных анимационных действий** — если в фильме одновременно используется несколько анимационных процессов, то каждый из них должен быть размещен отдельно на своем собственном слое. Это правило позволит правильно организовать структуру фильма и гибко манипулировать объектами в процессе работы. Так, например, если на сцене одновременно движутся два персонажа, то анимировать каждого из них нужно в отдельном слое, в противном случае можно столкнуться с серьезными трудностями при редактировании анимации, изменениями порядка следования планов фильма или одновременном выполнении преобразований содержимого нескольких кадров.

- **Разделение планов фильма** — объекты, принадлежащие разным планам (передний, средний, задний и т. д.), должны быть размещены в разных слоях. Такая организация позволит успешно реализовать впечатление пространственной глубины и легко управлять структурой фильма.

Покадровая анимация используется для художественной передачи сложных движений и по принципу реализации близка к классической анимации. Применение покадровой анимации предполагает значительные трудозатраты, поскольку каждый кадр фильма отрисовывается (или формируется) вручную. Детальная передача движения требует большого числа ключевых кадров и, соответственно, приводит к получению конечного файла большего размера. Однако использование покадровой анимации позволяет получить живое и убедительное движение, свободное от механистичности, свойственной автоматической анимации.

В качестве примера покадровой анимации можно рассмотреть процесс создания анимации горящей на ветру свечи. Он включает следующие этапы.

- Определение длительности фильма. Допустим, что при скорости 12 кадров в секунду длительность фильма составит 0,5 секунды, таким образом, для его создания потребуется 6 кадров.
- Разработка визуального решения, определение статичных и движущихся элементов, разделение планов. В нашем случае язычок пламени будет подвижным элементом, а сама свеча и фон — статичными. Таким образом, для создания данного ролика потребуются три слоя.
- Техническая реализация.

Техническая реализация анимационного эффекта горящей свечи включает следующую последовательность действий:

1. Открыть новый документ (**File>New**).
2. Создать три слоя (**Insert>Timeline>Layer**) и назвать их, начиная с верхнего, **Flame**, **Candle**, **Background**.
3. В первом ключевом кадре слоя **Background** создать изображение, используемое в качестве фона. Поскольку ролик будет длиться 0,5 с (6 кадров), необходимо, чтобы на протяжении этого времени фон находился на сцене. Для этого нужно выделить шестой кадр слоя **Background** и создать в нем простой кадр (**Insert Frame**). По окончании слой можно заблокировать.
4. В первом ключевом кадре слоя **Candle** нарисовать свечу с фитилем. Добавить простой кадр в шестой кадр данного слоя. По окончании слой можно заблокировать.
5. Слой **Flame** будет содержать покадровую анимацию пламени свечи. В первом ключевом кадре необходимо нарисовать начальное состояние пламени.

6. Создать второй ключевой кадр на слое **Flame**, используя команду **Insert Keyframe** или **Insert Blank Keyframe**. При создании заполненного ключевого кадра во втором кадре достаточно будет только изменить внешний вид пламени по сравнению с содержимым первого кадра (например, используя трансформацию). При создании пустого ключевого кадра во втором кадре необходимо будет снова нарисовать пламя, изменив его по сравнению с предыдущей фазой.
7. Последовательно создавая ключевые кадры на слое **Flame** и изменения внешний вид пламени, необходимо отрисовать все шесть фаз движения (рис. 9.6).

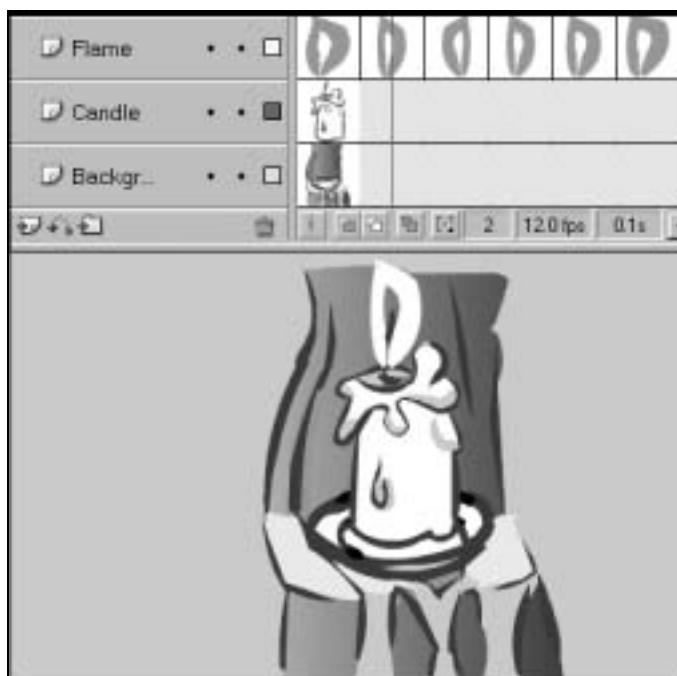


Рис. 9.6. Пример покадровой анимации свечи

8. Сохранить документ, присвоив ему соответствующее имя (**File>Save As**).
9. Протестировать фильм можно непосредственно в рабочей среде (**Control>Play**) или в среде тестирования (**Control>Test Movie**), открывющейся в собственном окне. Можно также воспользоваться контроллером (**Window>Toolbars>Controller**). Для того чтобы зациклить воспроизведение монтажной линейки документа в рабочей среде, нужно включить режим **Control>Loop Playback**, а в среде тестирования — **Control>Loop**.

Следует иметь в виду, что просмотреть анимацию экземпляров символа типа **Graphic** можно непосредственно в рабочей среде, выполнив команду главного меню **Control>Play** (<Enter>), или воспользовавшись контроллером. Для того чтобы зациклить воспроизведение основной монтажной линейки, можно использовать команду меню **Control>Loop Playback**.

Калькирование.

Множественное редактирование кадров

При создании покадровой анимации, разрабатывая новую фазу движения, для внесения требуемых изменений необходимо видеть предыдущую фазу движения. Кроме того, зачастую возникает потребность в изменении содержимого сразу нескольких ключевых кадров. Для решения этих задач Flash предлагает использовать режимы калькирования (режимы шлейфа) и множественного редактирования кадров.

Режим калькирования — Onion Skin — позволяет одновременно видеть на сцене содержимое произвольного числа кадров, причем содержимое кадров, входящих в диапазон калькирования, отображается с определенной степенью прозрачности.

Режим множественного редактирования кадров — Edit Multiple Frames — позволяет одновременно выполнять определенные манипуляции над содержимым выделенных ключевых кадров, входящих в диапазон редактирования.

Для того чтобы активизировать режим калькирования, необходимо щелкнуть на пиктограмме **Onion Skin** , расположенной в нижней части монтажной линейки. При этом на шкале монтажной линейки появятся маркеры, позволяющие задать длительность диапазона калькирования. Все кадры, находящиеся между этими маркерами, входят в диапазон калькирования, и их содержимое будет отображаться на сцене одновременно (рис. 9.7). Перемещая маркеры, можно регулировать длину диапазона калькирования.

Щелчок на пиктограмме  позволяет включить режим калькирования в контурах **Onion Skin Outlines**. При этом содержимое всех кадров, входящих в диапазон калькирования, отображается в виде контуров.

Настройка режима калькирования может быть выполнена при помощи пиктограммы **Modify Onion Markers** , щелчок на которой приводит к появлению всплывающего меню, содержащего следующие команды:

- Always Show Markers** — отображение маркеров калькирования даже при отключении режима калькирования;
- Anchor Onion** — зафиксировать диапазон калькирования. Данная команда приводит к тому, что маркеры диапазона калькирования не следуют за указателем текущего кадра, а всегда находятся в том положении, куда они были помещены;

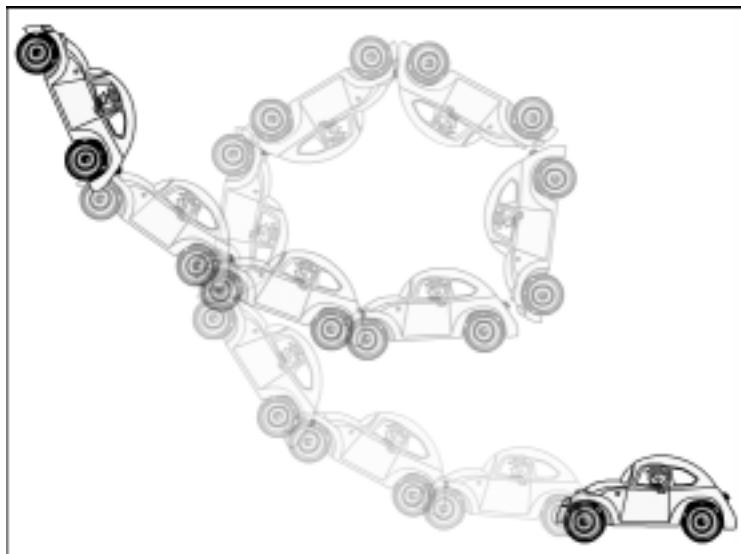
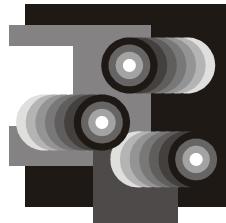


Рис. 9.7. Отображение содержимого кадров в режиме калькирования **Onion Skin**

- **Onion 2** — в режиме калькирования отображаются два кадра слева и два справа от указателя текущего кадра;
- **Onion 5** — в режиме калькирования отображаются пять кадров слева и пять справа от указателя текущего кадра;
- **Onion All** — диапазон калькирования автоматически устанавливается равным всему диапазону кадров монтажной линейки.

Для включения режима множественного редактирования кадров необходимо щелкнуть на пиктограмме **Edit Multiple Frames** . Одновременно можно редактировать содержимое только выделенных ключевых кадров, входящих в диапазон калькирования, задаваемый при помощи маркеров, расположенных на шкале монтажной линейки. Используя этот режим, можно, например, одновременно отцентрировать содержимое всех кадров при помощи палитры **Align**. При редактировании большого числа кадров, содержащих сложные изображения, возможно существенное снижение скорости выполнения операций. Для того чтобы ускорить обработку, можно включить режим отображения в контурах (**View>Preview Mode>Outlines**) для всей сцены или режим отображения в контурах содержимого текущего слоя.

Щелчок на пиктограмме позволяет отцентрировать указатель текущего кадра, прокрутив монтажную линейку. Эта команда выполняется при наличии на монтажной линейке такого количества кадров, что они не вмещаются в видимую область.



Глава 10

Символы Flash

Символ — это абстрактная реальность, воплощенная в конкретный знак, способная передать сложнейшие логические понятия, идеи, мистические явления и состояния.

Словарь символов

Разработка проекта, предназначенного для размещения в Интернете, требует наиболее эффективной организации его структуры с целью уменьшения объема конечного файла. Используемая в Flash технология символов позволяет существенно оптимизировать рабочий процесс и минимизировать объем конечного фильма. Символы являются мощнейшими инструментами, которые позволяют многократно использовать элементы фильма без значительного увеличения его объема.

Понятие символа

Символ (Symbol) — это объект рабочей среды, который может быть неоднократно использован в фильме, при этом его многократное применение в очень несущественной степени влияет на объем конечного файла.

В рабочей среде Flash символ содержится в библиотеке (Library), представляющей собой хранилище импортированных объектов (растровой графики, видеофайлов, звуковых файлов) и символов. Для того чтобы открыть окно библиотеки, необходимо выполнить команду главного меню **Window>Library** ($<\text{Ctrl}>+<\text{L}>$). В результате создания символ автоматически помещается в библиотеку. Таким образом, в библиотеке находится эталон, или, в терминах Flash, — **Master object** — оригинал объекта. В фильме используется **экземпляр** — **Instance** — символа, являющийся ссылкой на оригинал. На сцене одновременно может находиться неограниченное количество экземпляров одного и того же символа, причем число этих экземпляров практически не влияет на объем конечного файла. Так, если на сцене находятся десять экземпляров символа, добавляющего к объему конечного файла 1 Кбайт, то

конечный объем будет увеличен менее чем на 100 байт. Причем это значение не зависит от объема, занимаемого самим символом. Очевидно, что данная особенность открывает чрезвычайно широкие возможности оптимизации размера конечного фильма.

Другой замечательной особенностью применения символов является возможность изменения каждого экземпляра символа независимо от остальных. Используя операции трансформации и *цветовые эффекты экземпляров*, можно изменить пропорции и цветовые характеристики экземпляров, добиваясь визуального разнообразия. Таким образом можно, например, "посадить" на сцене целый лес деревьев, используя множество экземпляров одного и того же символа. При этом конечный объем файла будет зависеть главным образом от объема изображения дерева, помещенного в символ (рис. 10.1).

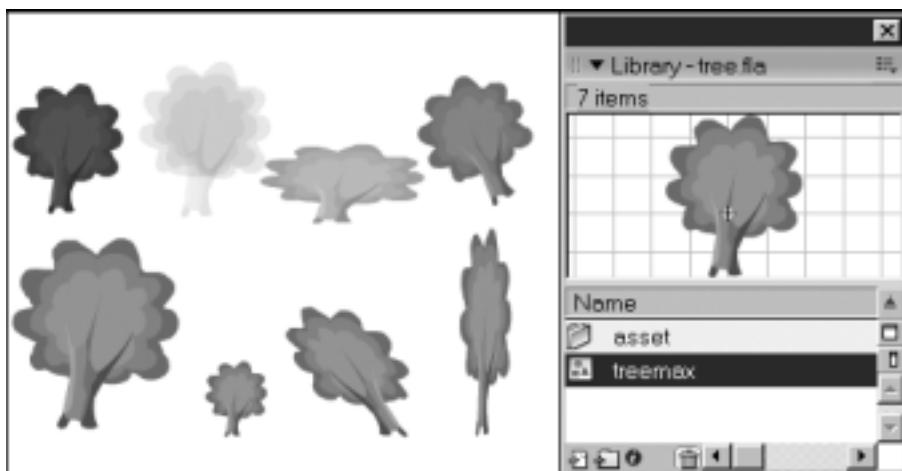


Рис. 10.1. Символ (Symbol) и его экземпляры (Instances)

Следует иметь в виду, что между символом и его экземплярами существует тесная связь. Любое изменение символа в библиотеке приведет к изменению всех его экземпляров, используемых в фильме. В то же время невозможно изменить базовые свойства экземпляра, не изменяя символа. В связи с этим, к экземплярам символа не могут быть применены операции, изменяющие базовые свойства элементов векторных объектов.

Символы могут содержать не только статичные графические объекты, но также анимацию, звук и интерактивность (сценарии ActionScript), это дает возможность многократного использования в фильме анимированных фрагментов или интерактивных элементов. Любой символ (за исключением шрифтового) содержит свою собственную монтажную линейку, работа с которой осуществляется точно так же как и с монтажной линейкой основного фильма. Зависи-

мость монтажной линейки символа от монтажной линейки, на которой размещается его экземпляр, определяется поведением экземпляра.

Кроме того, некоторые типы символов, являясь объектами ActionScript, допускают возможность программного управления и выполнения различных действий, позволяющую реализовать широчайшую интерактивность.

Типы символов

В Flash применяются три типа символов, которые имеют определенное назначение и используются для выполнения различных задач. Тип символа характеризуется его *поведением* — **Behavior**, которое задается при создании символа и может быть в дальнейшем переопределено.

Тип **Graphic** (Графический) — как правило, содержит статичные графические изображения, однако может также включать анимацию. В символ типа **Graphic** могут быть помещены экземпляры любых других символов (включая графический), но кадры его монтажной линейки не могут содержать звук и сценарии ActionScript. Монтажная линейка графического символа зависит от родительской монтажной линейки (т. е. линейки, в кадр которой помещен его экземпляр). Вследствие этого анимация графического символа может быть увидена целиком только в том случае, если его экземпляр присутствует на сцене в течение промежутка времени, соответствующего длительности этой анимации. Так, если на монтажной линейке графического символа содержится десятисекундная анимация, занимающая 120 кадров, то при размещении экземпляра такого символа в ключевом кадре монтажной линейки основного фильма после него необходимо будет добавить 119 простых кадров. В противном случае можно будет увидеть только тот фрагмент анимации символа, длительность которого соответствует диапазону кадров основной монтажной линейки, содержащих экземпляр данного символа. Экземпляр графического символа исключает возможность программной обработки, поскольку не является объектом ActionScript. Символ типа **Graphic**, как правило, используется для хранения статических графических изображений, которые используются в фильме более одного раза.

Тип **Movie Clip** (Муви-клип, клип) — наиболее мощный и гибкий инструмент Flash. Символ типа муви-клип может содержать на своей монтажной линейке любые графические объекты, анимацию, экземпляры любых других символов (включая муви-клип), звук и сценарии ActionScript. Замечательной особенностью муви-клипа является независимость его монтажной линейки от монтажной линейки, на которую помещен его экземпляр. Вследствие этого вся функциональность муви-клипа будет выполняться до тех пор, пока его экземпляр присутствует на сцене. Так, десятисекундная анимация, занимающая 120 кадров монтажной линейки символа типа **Movie Clip**, будет показана целиком, даже если его экземпляр помещен в единственный кадр основной монтажной линейки. Экземпляру муви-клипа может быть присво-

ен идентификатор и назначен сценарий ActionScript, что позволяет реализовывать его программную обработку. Диапазон применения символа типа чрезвычайно широк. С его помощью можно структурировать проект, создавая контейнеры, содержащие широкую функциональность и интерактивность, которые, по сути, представляют самостоятельные мини-фильмы. Из таких универсальных блоков может быть "собран" конечный фильм. На практике достаточно часто используется такой подход к организации фильма, при котором монтажная линейка основного фильма содержит один единственный кадр, содержащий экземпляр муви-клипа, который, в свою очередь, содержит весь проект в виде набора вложенных символов, анимации, звука и сценариев.

Тип **Button** (Кнопка) — интерактивный элемент, внешний вид которого может изменяться при воздействии на него курсором мыши. При создании символа типа **Button**, на его монтажной линейке задаются три состояния, определяющие внешний вид кнопки при отсутствии взаимодействия, при наведении на нее курсора мыши и в момент нажатия. Кроме того, в отдельном кадре задается активная область, в которой кнопка реагирует на данные действия. Кадры монтажной линейки кнопки не воспроизводятся последовательно, как это происходит с монтажными линейками других символов. Вместо этого, при осуществлении определенного воздействия на экземпляр кнопки отображается соответствующий кадр ее монтажной линейки. Символ типа **Button** может содержать экземпляры любых других символов, за исключением кнопки, а также звук. В кнопке нельзя использовать слой-маску. Так же как и у муви-клипа, монтажная линейка кнопки не зависит от родительской монтажной линейки. Экземпляру символа типа **Button** может быть присвоен идентификатор и назначен сценарий ActionScript, выполняемый в результате наступления определенного события, связанного с кнопкой или клавиатурой. Применение кнопок всегда связано с созданием сценария, поскольку кнопка является интерактивным элементом. Воздействие на кнопку, кроме изменения ее внешнего вида, должно приводить к выполнению определенных действий, определяемых в сценарии кнопки. О программировании кнопок речь пойдет в гл. 19. Кнопки служат для инициирования определенных действий и обеспечивают интерактивное взаимодействие с пользователем.

Примечание

В официальной терминологии существует еще один тип символа — шрифтовой символ (**Font symbol**). Однако этот символ, будучи членом общей библиотеки (*Shared library*), не имеет явного визуального выражения и поэтому в данной главе не упоминается. Подробная информация об использовании шрифтовых символов содержится в разд. "Использование общих шрифтовых ресурсов" гл. 8.

В табл. 10.1 обобщены сведения о типах символов, их назначении и характерных особенностях.

Таблица 10.1. Типы символов

Тип символа	Назначение	Содержание				
		Векторная графика, растр, текст	Анимация	Экземпляры символов	Звук	Сценарий AS
Graphic графика 	Служит для хранения статичных изображений или анимированных объектов. Монтажная линейка зависит от родительской монтажной линейки	+	+	Графика, Муви-клип, Кнопка	-	-
Movie Clip муви-клип 	Служит для создания анимированных интерактивных роликов, многократно используемых в фильме и/или при необходимости программного управления объектом. Монтажная линейка не зависит от родительской монтажной линейки. Обрабатывается программно	+	+	Графика, Муви-клип, Кнопка	+	+
Button кнопка 	Элементы интерактивности, служат для инициирования действий, осуществления навигации по фильму и т. д. Монтажная линейка имеет 4 кадра, не зависит от родительской монтажной линейки. Могут быть пустыми (без графики). Обрабатываются программно	+ (маска не работает)	+	Графика, Муви-клип	+	+(только на экземпляре кнопки, а не на монтажной линейке)

Создание и редактирование символов

В результате создания символ помещается в библиотеку (Library), на сцене используются его экземпляры, являющиеся объектами наложенного уровня. Экземпляры представляют собой контейнеры, содержимое которых изолировано от взаимодействия с другими объектами и не может редактироваться без изменения содержимого родительского символа библиотеки.

Для изменения содержимого самого символа, находящегося в библиотеке, необходимо войти в режим его редактирования, представляющий собой изолированную от остальных объектов сцены среду, обеспечивающую доступ к монтажной линейке символа и расположенным на ней объектам. В среде редактирования символа, так же как и на сцене, существуют слои, каждый из которых содержит рабочий и наложенный уровни, и действуют те же принципы и приемы работы, что и на монтажной линейке основного фильма. Однако в отличие от сцены, начало координат которой находится в левом верхнем углу рабочей области, в среде редактирования символа имеется свое собственное начало координат, находящееся в *точке регистрации символа*. Координаты всех объектов, находящихся внутри символа, регистрируются относительно данной точки.

Создание символа

Существуют два способа создания символа. Первый способ позволяет преобразовать в символ любые выделенные графические объекты, второй предполагает создание пустого символа и наполнение его содержимым "изнутри" в среде редактирования.

Для того чтобы создать символ на основе выделенных объектов сцены, необходимо выполнить следующие действия:

1. Нарисовать или импортировать любой графический объект (векторную форму, контур, группу, растровое изображение, текстовый блок, видеоролик).
2. Выделить графический объект (или несколько).
3. Выполнить команду главного меню **Modify>Convert to Symbol (<F8>)** или просто перетащить выделенный объект в библиотеку курсором мыши.
4. В появившемся диалоговом окне **Convert to Symbol**, представленном на рис. 10.2, содержаться следующие элементы:
 - **Name** — данное поле предназначено для задания имени символа. Указанное здесь имя будет отображено в библиотеке рядом с пиктограммой символа. В качестве имени можно использовать любую последовательность букв, цифр и знаков пунктуации, в том числе кириллицу. Данное имя используется только в рабочей среде для идентификации объекта библиотеки. Имя должно быть осмысленным и отражать содержание символа.
 - **Behavior** — при помощи данного переключателя назначается поведение символа, определяющее его тип.
 - **Registration** — данный элемент позволяет задать положение объекта в среде редактирования символа относительно точки регистрации (начала координат символа). Выделяя соответствующий маркер пиктограм-

мы **Registration**, можно указать положение ограничивающей рамки объекта относительно точки регистрации создаваемого символа. Так, если геометрический центр объекта должен совпадать с точкой регистрации содержащего его символа, необходимо выделить центральный маркер ; если в точке регистрации символа должен находиться левый верхний угол ограничивающей рамки объекта, необходимо выделить точку , расположенную в левом верхнем углу, и т. д.

- **Advanced/Basic** — данная кнопка позволяет показать/скрыть дополнительные настройки, задающие параметры связи (Linkage) символа, которые используются при работе с общими библиотеками (см. гл. 11) и при программном создании (присоединении) экземпляров символов средствами ActionScript (см. гл. 21).

5. Нажать кнопку **OK**.

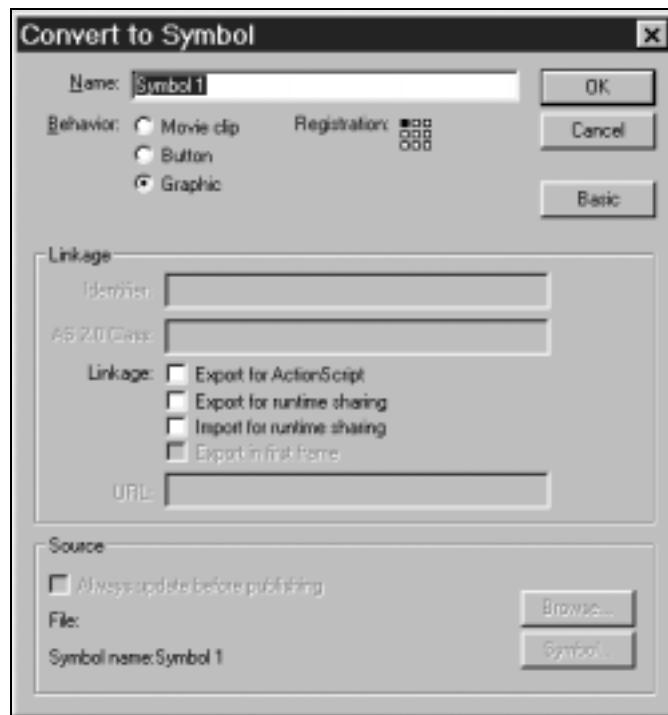


Рис. 10.2. Диалоговое окно **Convert to Symbol**

В результате выполнения этих действий в библиотеке будет создан новый символ, а его экземпляр автоматически будет размещен на сцене.

Для того чтобы создать пустой символ, необходимо выполнить следующие действия:

1. Убедиться, что на сцене нет выделенных объектов.
2. Выполнить одно из следующих действий:
 - команду главного меню **Insert>New Symbol** (<Ctrl>+<F8>);
 - команду контекстного меню библиотеки **New Symbol**;
 - щелчок на пиктограмме  в нижней части библиотеки (Library).
3. В появившемся диалоговом окне **Create New Symbol** задать параметры, аналогичные (за исключением элемента **Registration**) описанным в п. 4 предыдущего способа.
4. Нажать кнопку **OK**, после чего произойдет переход в среду редактирования символа, где можно разместить любые объекты в соответствии с требованиями, предъявляемыми к содержанию данного типа символа.
5. Для того чтобы выйти из среды редактирования символа, можно выполнить одно из следующих действий:
 - щелкнуть на названии текущей сцены, расположенном слева в полосе редактирования рядом с пиктограммой  (по умолчанию — **Scene 1**);
 - щелкнуть на стрелке , расположенной слева в полосе редактирования;
 - выполнить команду главного меню **Edit>Document** (<Ctrl>+<E>).

В результате выполнения этих действий в библиотеке будет создан новый символ, однако его экземпляр не будет автоматически помещен на сцену. Для того чтобы поместить экземпляр на сцену, нужно вручную вытащить его из библиотеки (*подробнее о создании экземпляров см. в разд. "Создание экземпляров" настоящей главы*).

Редактирование символа

Для того чтобы создать содержимое или добавить объекты в символ, необходимо войти в среду его редактирования. При создании нового символа среда редактирования открывается автоматически. Разработчики Flash предусмотрели множество различных способов, обеспечивающих переход в среду редактирования символа. Для того чтобы отредактировать уже готовый символ, можно выполнить одно из следующих действий:

- выделить требуемый символ в библиотеке, щелкнуть на пиктограмме, расположенной в правом верхнем углу панели **Library**, и в появившемся контекстном меню библиотеки выполнить команду **Edit**;
- щелкнуть на названии символа в библиотеке правой кнопкой мыши и в появившемся контекстном меню выполнить команду **Edit**;

- дважды щелкнуть на пиктограмме требуемого символа в библиотеке;
- выделить требуемый символ в библиотеке и дважды щелкнуть на его изображении в окне просмотра;
- щелкнуть на экземпляре символа на сцене правой кнопкой мыши и из появившегося контекстного меню выбрать одну из следующих команд:
 - **Edit** — перемещает в среду редактирования символа, при этом остальное содержимое сцены не отображается на экране;
 - **Edit in Place** — перемещает в среду редактирования символа, однако при этом все остальное содержимое сцены отображается, но отсутствует возможность его редактирования. Это напоминает работу в режиме редактирования группы;
 - **Edit in New Window** — среда редактирования открывается в новом окне Flash;
- выделить экземпляр требуемого символа и выполнить любую из команд главного меню:
 - **Edit>Edit Symbols (<Ctrl>+<E>);**
 - **Edit>Edit Selected';**
 - **Edit>Edit in Place';**
- дважды щелкнуть на экземпляре требуемого символа на сцене;
- щелчок на пиктограмме , расположенной в полосе редактирования слева от поля ввода масштаба просмотра, приводит к появлению перечня всех имеющихся в библиотеке текущего документа символов. Щелкнув на имени любого из них, можно переместиться в среду его редактирования.

При нахождении в среде редактирования символа в полосе редактирования (Edit bar) рядом с именем текущей сцены появляется пиктограмма с названием редактируемого символа. Если символ содержит экземпляры других символов, можно последовательно входить в среду редактирования вложенных символов, опускаясь на более низкие уровни иерархии вложенности. При этом на полосе редактирования будут появляться пиктограммы с названиями редактируемых символов, расположенные в соответствии с иерархией уровней их вложенности (рис. 10.3).



Рис. 10.3. Полоса редактирования отражает текущее местонахождение в иерархии вложенности объектов

Для того чтобы выйти из среды редактирования символа, можно воспользоваться различными способами.

- Если при нахождении в среде редактирования символа остальное содержимое сцены скрыто, можно выполнить одно из следующих действий:
 - щелкнуть на названии текущей сцены в полосе редактирования;
 - щелчок на стрелке  расположенной слева в полосе редактирования, позволяет подняться на один уровень вверх при редактировании вложенных символов. Эта функция напоминает использование кнопки браузера **Back**. При редактировании самого верхнего в иерархии символа щелчок на стрелке позволит выйти из среды редактирования символа;
 - выполнить команду главного меню **Edit>Document** (<Ctrl>+<E>).
- Если при нахождении в среде редактирования символа остальное содержимое сцены отображается на экране, можно выполнить любое из приведенных выше действий или просто дважды щелкнуть на любом участке, свободном от объектов, принадлежащих данному символу.
- Если среда редактирования символа открывается в новом окне Flash, необходимо просто закрыть (или свернуть) окно или выполнить команду главного меню **Edit>Document** (<Ctrl>+<E>).

Символы *Graphic* и *Movie Clip*

Процесс создания (редактирования) символов типа **Graphic** и **Movie Clip** технически аналогичен процессу работы на монтажной линейке основного фильма (сцены). Различия обусловлены только ограничениями, накладываемыми на содержимое символов соответствующего типа (см. выше в данной главе). Каждый из данных символов может содержать неограниченное количество слоев и кадров монтажной линейки. При работе в среде редактирования символа точка регистрации, являющаяся началом координат, отображается в виде крестика. Для того чтобы при позиционировании объектов выполнить выравнивание или распределение объектов относительно точки регистрации символа, можно воспользоваться режимом **To stage** панели **Align**.

Если при создании символа один и тот же объект используется многократно, необходимо объявлять его символом и использовать как вложенный символ.

Следует иметь в виду, что просмотреть анимацию экземпляров символа типа **Graphic** можно непосредственно в рабочей среде, выполнив команду главного меню **Control>Play** (<Enter>) или воспользовавшись контроллером (**Window>Toolbars>Controller**). Для того чтобы зациклить воспроизведение основной монтажной линейки, можно использовать команду меню **Control>**

Loop Playback. Анимация экземпляров символа типа **Movie Clip** может быть увидена только в среде тестирования, которая открывается в отдельном окне в результате выполнения команд меню **Control>Test Movie** (<Ctrl>+<Enter>) или **Control>Test Scene** (<Ctrl>+<Alt>+<Enter>).

Символ Button

Процесс создания (редактирования) символа типа **Button** заключается в создании четырех кадров, три из которых задают внешний вид кнопки в соответствующем состоянии, а четвертый используется для задания области реагирования кнопки. Монтажная линейка кнопки содержит четыре именованных кадра.

- **Up** — кадр содержит изображение кнопки в неактивном состоянии, т. е. когда курсор мыши находится в области реагирования кнопки.
- **Over** — кадр содержит изображение кнопки при наведении на нее курсора мыши, т. е. это активное состояние, при котором курсор находится в области реагирования, но кнопка мыши не нажата.
- **Down** — кадр содержит изображение кнопки в момент нажатия левой кнопки мыши.
- **Hit** — этот кадр является служебным и содержит область реагирования кнопки. В конечном фильме содержимое кадра **Hit** не отображается. В качестве области реагирования кнопки должна быть использована векторная область (объект рабочего или наложенного уровня, являющийся формой). Ее цвет не имеет значения, поэтому в целях минимизации конечного объема файла в качестве цветового заполнения нужно использовать сплошной (solid) цвет. По этой же причине желательно избегать использования в кадре **Hit** контуров (обводок). Форма области реагирования может быть совершенно произвольной и либо совпадать с формой самой кнопки, либо выходить за ее пределы, или, наоборот, находиться внутри области, занимаемой кнопкой. Область реагирования также может быть размещена отдельно от кнопки или представлять собой несколько несвязанных областей (рис. 10.4). Следует, однако, помнить, что кнопка реагирует на действия только при воздействии курсора мыши на ее активную область.

Примечание

Если кадр **Hit** не будет создан, его роль возьмет на себя кадр **Up**, т. е. расположенные в нем области будут использованы в качестве области реагирования кнопки. Однако если кадр **Hit** будет создан, но останется пустым, то кнопка работать не будет.

Кнопка может содержать неограниченное число слоев.

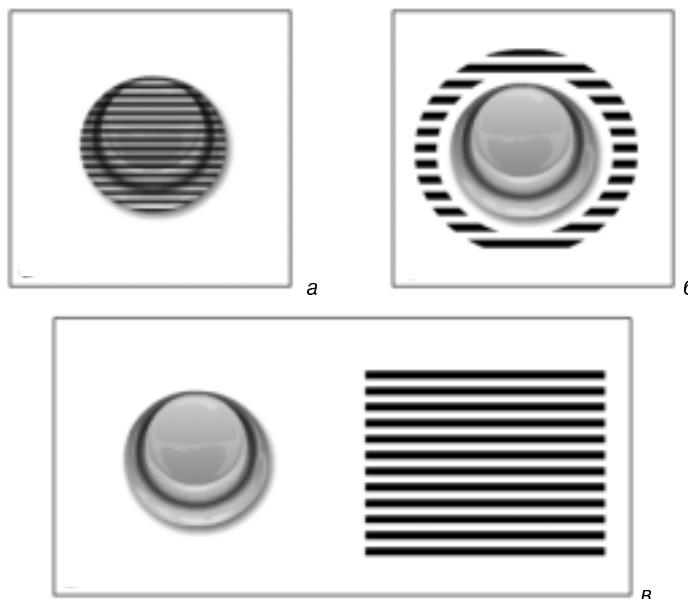


Рис. 10.4. Области реагирования кнопки:
а — область реагирования совпадает с кнопкой;
б — область реагирования ограничивает кнопку;
в — область реагирования находится вне кнопки

Примечание

Большинство команд контекстного меню монтажной линейки, задающих виды отображения кадров, не работают в среде редактирования кнопки. Тем не менее, при необходимости рассмотреть содержимое кадров непосредственно на монтажной линейке в режиме **Preview**, можно скопировать их и разместить на основной монтажной линейке или на линейке символа.

Для создания кнопки нужно выполнить следующие действия:

1. Создать новый символ типа **Button** (**Modify**>**Convert to Symbol** или **Insert**>**New Symbol**).
2. Войти в среду его редактирования.
3. Создать необходимое число слоев, присвоив им соответствующие названия (**Insert**>**Timeline**>**Layer**). Следует иметь в виду, что в кнопках нельзя использовать слой-маску, однако можно поместить в кнопку экземпляр мультиклипа, содержащего маску.
4. Создать изображение кнопки в неактивном состоянии, соответствующем положению **Up** (отжатая кнопка), размещая разные элементы изображения в предназначенных для них слоях.

5. Создать новый ключевой кадр в положении **Over** (кнопка при наведении) (**Insert Keyframe** или **Insert Blank Keyframe**) и поместить в него изображение кнопки в соответствующем состоянии (или изменить скопированное из предыдущего кадра изображение).
6. Повторяя эти действия, создать два следующих кадра; в кадре **Down** задать вид нажатой кнопки, а в кадре **Hit** — область реагирования.
7. Выйти из среды редактирования и поместить на сцену экземпляр созданной кнопки.
8. Не все кадры монтажной линейки кнопки обязательно должны быть ключевыми. Если внешний вид кнопки или ее отдельного элемента, размещенного в одном из слоев, не изменяется в различных состояниях, достаточно использовать один-единственный ключевой кадр и поместить после него простые кадры, соответствующие тем состояниям кнопки, где не происходит никаких изменений.
9. Если в различных кадрах монтажной линейки кнопки используется один и тот же объект, необходимо объявлять его символом и использовать как вложенный символ типа **Graphic** или **Movie Clip**. Поскольку монтажная линейка муви-клипа не зависит от родительской монтажной линейки, в любом из трех первых кадров монтажной линейки кнопки может быть помещен экземпляр анимированного символа типа **Movie Clip**. В результате анимация вложенного символа будет воспроизводиться, когда кнопка находится в соответствующем состоянии.
10. На рис. 10.5 представлены некоторые варианты создания кнопок. Кроме того, большой набор кнопок содержится в стандартной библиотеке Flash, которую можно открыть при помощи команды **Window>Other Panels>Common Libraries>Buttons**.

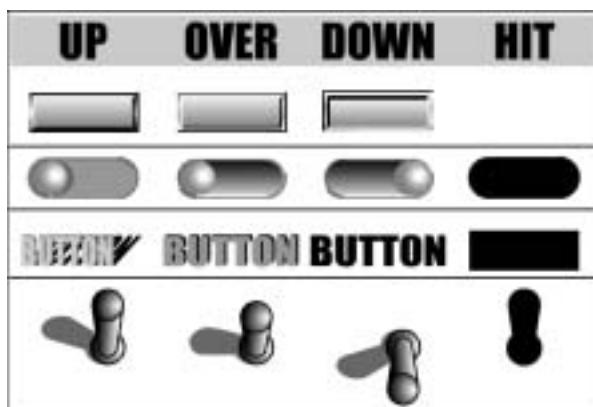


Рис. 10.5. Различные типы кнопок

Для того чтобы протестировать кнопку, необходимо либо перейти в среду тестирования, выполнив команду главного меню **Control>Test Movie** (<Ctrl>+<Enter>), либо сделать это прямо в рабочей среде, включив режим **Control>Enable Simple Buttons** (<Ctrl>+<Alt>+).

Особый тип кнопки представляет собой *пустая кнопка*. Пустая кнопка не содержит графических объектов и, соответственно, не видна в конечном фильме. Такая кнопка позволяет создать прозрачную область произвольной формы, выполняющую функции кнопки. Используя этот прием, можно заставить любой статический объект (например, растровое изображение) вести себя как кнопка, "накрыв" его сверху пустой кнопкой.

Для того чтобы создать пустую кнопку, необходимо оставить первые три кадра ее монтажной линейки пустыми, а в четвертый кадр **Hit** поместить форму (объект рабочего или наложенного уровня), используемую в качестве области реагирования. В рабочей среде такая кнопка отображается прозрачным бирюзовым цветом, но в конечном фильме она будет не видна.

В табл. 10.2 обобщены различные операции с символами и способы их выполнения.

Таблица 10.2. Операции с символами

Операция	Способ
Создание символа	1. Команда Insert>New Symbol <Ctrl>+<F8> — создать новый пустой символ. 2. Команда Modify>Convert to Symbol , <F8> — создать символ из выделенных объектов сцены или среды редактирования другого символа. 3. Пиктограмма в библиотеке. 4. Команда New Symbol в контекстном меню библиотеки. 5. Перетащить выделенный объект в окно библиотеки
Редактирование символа	1. Команда Edit в контекстном меню библиотеки или объекта библиотеки. 2. Дважды щелкнуть на пиктограмме символа в библиотеке или по его изображению в окне просмотра. 3. Дважды щелкнуть на копии символа на сцене. 4. Команды Edit , Edit in Place , Edit in New Window контекстного меню экземпляра. 5. Выделить экземпляр символа и выполнить команду Edit>Edit Symbols , <Ctrl>+<E>. 6. Выделить экземпляр символа и выполнить команду Edit>Edit Selected . 7. Выбрать требуемый символ из общего списка символов при помощи пиктограммы в полосе редактирования
Выделение в библиотеке	Щелкнуть на пиктограмме символа в окне библиотеки фильма, с <Shift> — выделение диапазона, с <Ctrl> — вразбивку
Переименование	1. Команда Rename в контекстном меню библиотеки. 2. Дважды щелкнуть на имени символа в библиотеке. 3. Команда Rename в контекстном меню объекта библиотеки, <F2>

Таблица 10.2 (окончание)

Операция	Способ
Копирование	1. Команда Duplicate в меню библиотеки. 2. Команда Duplicate в контекстном меню окна библиотеки. 3. Выделить экземпляр символа на сцене, команда Duplicate Symbol в контекстном меню экземпляра
Изменение типа символа	1. Команда Properties или команда Type в контекстном меню окна библиотеки. 2. команда Properties в меню библиотеки. 3. Пиктограмма  в библиотеке
Удаление	1. Команда Delete в меню библиотеки. 2. Щелчок на пиктограмме мусорной корзины в библиотеке. 3. Команда Delete в контекстном меню библиотеки

Экземпляр символа и его свойства

Экземпляр (Instance) является ссылкой на родительский символ, находящийся в библиотеке. Фильм Flash может включать любое число экземпляров одного и того же символа, которые могут размещаться на сцене или входить в состав других символов, образуя вложенные символы.

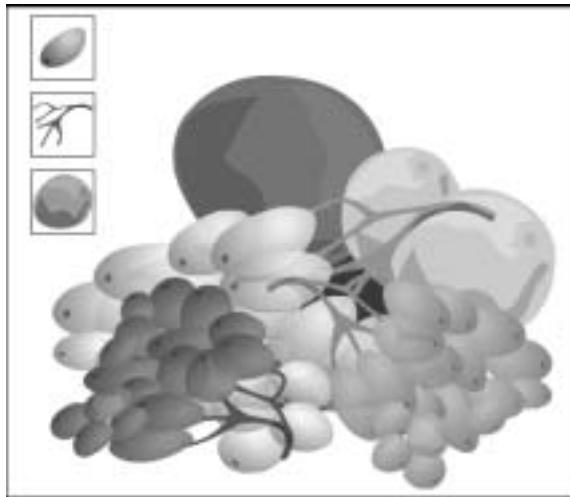


Рис. 10.6. Применение трансформации и цветовых эффектов экземпляров позволяет добиться разнообразия, используя лишь несколько символов

Применение символов позволяет существенно оптимизировать объем конечного фильма, поскольку число вхождений экземпляров практически не

отражается на размере результирующего файла. В связи с этим при необходимости использования в проекте одинаковых (или похожих) объектов применение символов следует взять за правило.

Flash позволяет изменять внешний вид, параметры и поведение экземпляров одного и того же символа, добиваясь их непохожести друг на друга. Это позволяет получить большое разнообразие, используя по сути ссылки на один и тот же объект (рис. 10.6).

Создание экземпляров

При создании символа из выделенного объекта (**Modify>Convert to Symbol**) его экземпляр автоматически помещается на сцену или монтажную линейку символа, содержащего выделенный объект.

Для того чтобы создать новый экземпляр символа, можно выполнить одно из следующих действий:

- взяться за название символа в библиотеке и вытащить его из библиотеки на сцену или в среду редактирования другого символа;
- взяться за изображение символа в окне просмотра библиотеки и вытащить его на сцену или в среду редактирования другого символа;
- скопировать и вставить имеющийся экземпляр символа на сцене или в среде редактирования другого символа.

При выделении экземпляра Инспектор свойств указывает тип его поведения (по умолчанию совпадающий с типом поведения родительского символа) в списке **Symbol behavior** и имя родительского символа. Таким образом, при выделении объекта можно сразу определить его тип (форма, группа, растр, видео, текст или экземпляр символа).

Трансформация экземпляров

Экземпляр символа является объектом наложенного уровня. В связи с этим к экземплярам могут быть применены только операции трансформации, не изменяющие базовых свойств объектов. К их числу относятся:

- масштабирование (пропорциональное и непропорциональное);
- поворот/скос;
- зеркальное отражение.

Трансформация экземпляров символов выполняется относительно их точки трансформации (Transformation point). По умолчанию точка трансформации экземпляра совпадает с точкой регистрации символа. Трансформация экземпляров приводит к трансформации их содержимого. Так, например, если на сцене имеется экземпляр муви-клипа, содержащий анимацию перемещения автомобиля слева направо, то его зеркальное отражение по горизонтали

приведет к тому, что движение автомобиля будет выполняться в противоположном направлении, т. е. справа налево. Масштабирование экземпляра этого символа приведет к изменению расстояния перемещения автомобиля, а вращение — к изменению направления движения.

Применяя операции трансформации к экземплярам символов, можно воздействовать на пропорции их содержимого, что позволяет достаточно сильно видоизменить экземпляры, добиваясь их несходства друг с другом (см. рис. 10.1, 10.6).

Цветовые эффекты экземпляров

Помимо операций трансформации, к экземплярам символов любого типа могут быть применены *цветовые эффекты экземпляров* (Color effects), позволяющие изменить их цветовые характеристики. В связи с тем, что экземпляр представляет собой контейнер, исключающий возможность редактирования содержимого, воздействие цветового эффекта в равной степени распространяется на все входящие в его состав графические объекты.

Для того чтобы применить к экземпляру символа (или сразу к нескольким экземплярам) цветовой эффект, необходимо выделить этот экземпляр, сделав его текущим объектом при помощи инструмента **Selection** и в списке **Color** Инспектора свойств выбрать требуемый эффект (рис. 10.7).

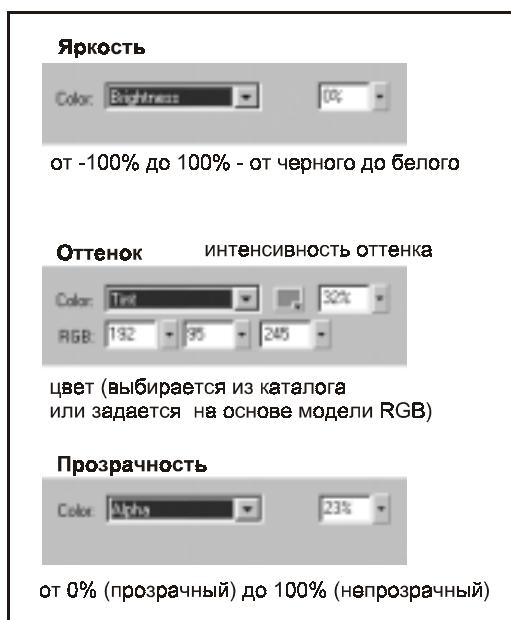


Рис. 10.7. Настройка параметров цветовых эффектов экземпляров

Список **Color** содержит следующие эффекты.

- **Brightness** (Яркость) — данный эффект позволяет задать степень яркости экземпляра в процентах в диапазоне от -100% до $+100\%$. Значение яркости может быть задано вручную или установлено при помощи вертикального слайдера, расположенного справа от поля **Brightness Amount**. Отрицательные значения приводят к затемнению экземпляра, положительные — к его освещению. Значение -100% окрашивает содержимое экземпляра сплошным черным цветом, значение $+100\%$ — сплошным белым.
- **Tint** (Оттенок) — данный эффект позволяет равномерно окрасить экземпляр заданным цветом с определенной степенью интенсивности. Цвет окрашивания можно выбрать из текущего каталога цветов, щелкнув на цветовом образце или задать числовые значения его составляющих на основе модели RGB. Поле **Tint Amount** позволяет задать степень интенсивности окрашивания в диапазоне от 0% до 100% . Стотысячная интенсивность приведет к тому, что экземпляр будет равномерно окрашен заданным сплошным цветом.
- **Alpha** (Прозрачность) — данный эффект позволяет задать степень яркости экземпляра в процентах в диапазоне от 0% до $+100\%$. Значение прозрачности может быть задано вручную или установлено при помощи вертикального слайдера, расположенного справа от поля **Alpha Amount**. Нулевое значение делает экземпляр полностью прозрачным, максимальное значение соответствует отсутствию прозрачности.
- **Advanced** (Комбинированный) — данный эффект позволяет одновременно задать цвет и степень окрашивания экземпляра, а также сообщить ему определенную степень прозрачности. Нажатие кнопки **Settings** приводит к появлению диалогового окна **Advanced Effect**, представленного на рис. 10.8.

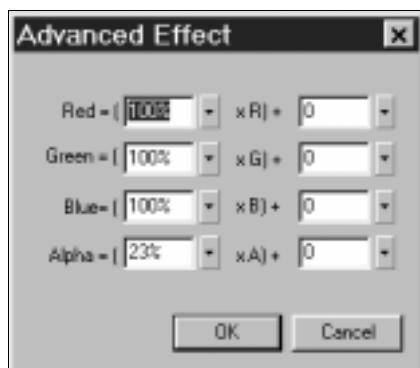


Рис. 10.8. Диалоговое окно **Advanced Effect**

Окно **Advanced Effect** содержит два столбца полей ввода. В каждой строке содержатся поля, отвечающие за соответствующие составляющие цвета и прозрачности экземпляра. Левый столбец предназначен для задания *относительных* значений составляющих в процентах (в диапазоне от -100% до 100%), которые рассчитываются как доля от текущих значений. Так например, если текущее значение красной составляющей содержимого экземпляра равно 255 (максимально), то задав в поле **Red** левого столбца значение 50%, можно уменьшить его до 127. Правый столбец предназначен для задания *абсолютных* значений составляющих (в диапазоне от -255 до +255). Прозрачность цвета на палитре **Color Mixer** задается в диапазоне от 0% до 100%. Этому диапазону соответствует диапазон абсолютных значений прозрачности от 0 до 255. Результатирующий эффект определяется значениями, заданными в полях левого и правого столбцов, и рассчитывается как сумма по формуле: *(заданное относительное значение как доля × текущее значение) + заданное абсолютное значение*. Так, например, пусть экземпляр содержит объект, имеющий следующие цветовые значения: **Red** = 255, **Green** = 0, **Blue** = 0, **Alpha** = 50 (красный полупрозрачный цвет). Для того чтобы сделать данный объект экземпляра белым и непрозрачным, т. е. присвоить его цветовым составляющим значения 255, 255, 255, 100 соответственно, необходимо в правый столбец ввести следующие значения: **Green** = 255, **Blue** = 255, **Alpha** = 127 (большие значения параметра **Alpha** в данном случае не будут оказывать влияния на внешний вид, поскольку максимальное значение прозрачности равняется 100%, что соответствует абсолютному значению 255). Задавать относительное и абсолютное значения для одной и той же составляющей одновременно нецелесообразно, поскольку рассчитать конечный результат в этом случае достаточно сложно.

Примечание

Для того чтобы быстро установить цвет окрашивания при использовании эффекта **Advanced**, можно сначала применить эффект **Tint**, синтез цвета в котором является более интуитивно понятным, а затем вызвать окно **Advanced Effect**, где автоматически будут установлены необходимые цветовые значения.

Свойства экземпляра типа **Graphic**

При одновременном использовании нескольких экземпляров анимированного символа типа **Graphic** их анимация выполняется абсолютно синхронно. В ряде случаев это приводит к механистичному, неестественному движению. Для того чтобы избежать такого эффекта, можно при помощи Инспектора свойств установить параметры воспроизведения для каждого экземпляра.

Для этого нужно выделить экземпляр символа и в Инспекторе свойств задать необходимые настройки.

- **Options for graphics** — данный список позволяет задать параметры воспроизведения монтажной линейки для каждого экземпляра графического символа в отдельности. Список содержит следующие значения (рис. 10.9).
 - **Loop** — зацикливание монтажной линейки символа. При достижении последнего кадра монтажной линейки символа воспроизводящая головка вновь переходит к первому кадру, и воспроизведение возобновляется.
 - **Play Once** — монтажная линейка символа будет воспроизведена один раз, после чего воспроизводящая головка останется в ее последнем кадре.
 - **Single Frame** — воспроизведение монтажной линейки символа отключено. Отображается только один ее кадр, номер которого указан в поле **First**, расположенном справа от списка **Options for graphics**.
- **First** — данное поле используется для задания номера кадра, с которого будет начато воспроизведение монтажной линейки символа для каждого экземпляра в отдельности. Если разным экземплярам графического символа задать различные начальные кадры, то их анимация перестанет быть синхронной.

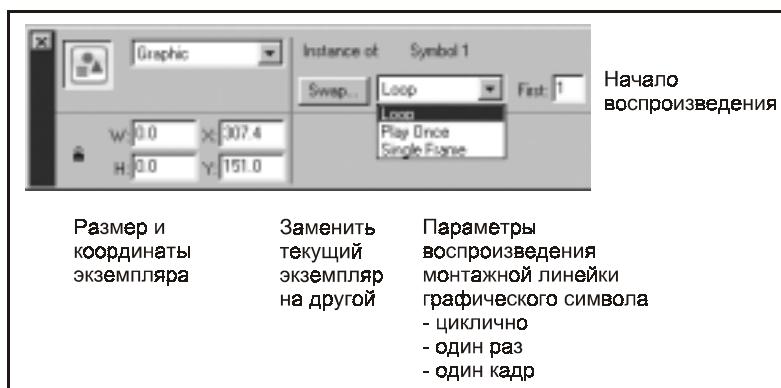


Рис. 10.9. Параметры настройки анимации экземпляров графических символов

Примечание

Следует иметь в виду, что режим зацикливания основной монтажной линейки фильма включен по умолчанию. В связи с этим при достижении ее последнего

кадра воспроизводящая головка вновь возвращается к первому кадру. При этом фильм будет запущен заново, и анимация всех экземпляров графического символа будет возобновлена.

В качестве примера использования данных настроек можно рассмотреть процесс создания анимации падающего снега с использованием анимированного графического символа. Он состоит из следующих этапов.

1. Для того чтобы решить эту задачу наиболее рациональным образом, вначале необходимо создать графический символ (**Graphic**), содержащий изображение снежинки, поскольку оно будет многократно использоваться в фильме. Этот символ можно назвать *snowflake*.
2. После этого нужно создать графический символ, на монтажной линейке которого будет создана анимация (покадровая или автоматическая) падающей снежинки — экземпляра *snowFlake*. Для расчета длины пробега снежинки при создании этой анимации нужно ориентироваться на размер окна конечного фильма, как правило, определяемый размером рабочей области. Если задумано, что снежинка при падении пролетает всю сцену, то в начале диапазона анимации она должна находиться над рабочей областью, а в конце — под ней. Для того чтобы в среде редактирования символа видеть границы рабочей области, нужно использовать режим редактирования **Edit in Place**. Скорость движения снежинки не должна быть высокой — это позволит получить более естественное движение. Например, при длине пробега снежинки 400 пикселов (высота рабочей области по умолчанию) для ее анимации можно использовать диапазон из 48 кадров (4 с). Автоматическая анимация по маршруту (см. гл. 13) позволит получить прекрасный результат при минимуме трудозатрат. Графический символ, содержащий анимацию падающей снежинки, можно назвать *snowFall*.
3. Далее необходимое число экземпляров символа *snowFall* может быть помещено в первый кадр монтажной линейки основного фильма. Следует убедиться, что данные экземпляры присутствуют на сцене в течение промежутка времени, соответствующего длительности их анимации. Для этого после ключевого кадра, содержащего экземпляры анимированных символов, нужно разместить соответствующее количество (в нашем примере 47) простых кадров. Для того чтобы снежинки двигались асинхронно, необходимо для каждого экземпляра задать свой начальный кадр. Сдвиг начального кадра анимации желательно задавать в шахматном порядке, чтобы разрядить экземпляры, добиваясь более естественного, напоминающего случайное, движения. Кроме того, к экземплярам можно применить зеркальное отражение по горизонтали и цветовые эффекты. Вращение экземпляра приведет к изменению направления движения снежинки, а масштабирование — к изменению длины ее пробега.

Свойства экземпляра типа *Movie Clip*

Любому экземпляру символа типа муви-клип может быть присвоен идентификатор, который используется для обращения к этому экземпляру при помощи сценария ActionScript. Эта возможность используется для программной обработки и управления экземпляром муви-клипа. Чтобы задать имя, необходимо выделить соответствующий экземпляр и ввести его в поле **Instance Name** Инспектора свойств (рис. 10.10). Имя должно быть уникально, т. е. одно и то же имя не должно использоваться для нескольких экземпляров. Требования, предъявляемые к именам экземпляров, идентичны требованиям к именам переменных ActionScript (см. гл. 20).

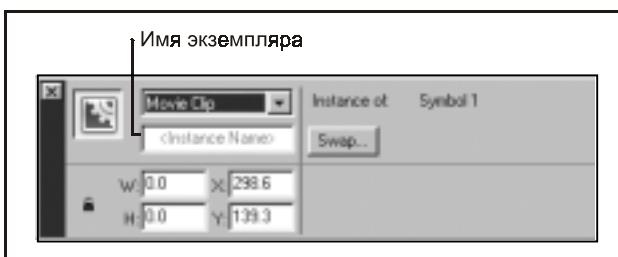


Рис. 10.10. Параметры экземпляров символа типа **Movie Clip**

Свойства экземпляра типа *Button*

Кнопка реагирует на курсор мыши, только когда он находится в области реагирования кнопки. Раскрывающийся список **Options for Buttons** позволяет определить, будет ли экземпляр кнопки реагировать на наведение курсора, если кнопка мыши была нажата еще до входления в активную область кнопки. Список содержит следующие значения:

- Track as Button** — если кнопка мыши нажата до входления в активную область кнопки, то кнопка не изменит состояние при наведении курсора, т. е. не прореагирует;
- Track as Menu Item** — состояние кнопки изменится при наведении курсора с нажатой кнопкой мыши, даже если нажатие кнопки мыши произошло до входления в активную область кнопки, т. е. кнопка прореагирует.

Так же как и экземпляру символа типа муви-клип, экземпляру кнопки может быть присвоен идентификатор, который используется для обращения к ней из сценария ActionScript. Чтобы это сделать, необходимо выделить соответствующий экземпляр и ввести имя в поле **Instance Name** Инспектора свойств (рис. 10.11).

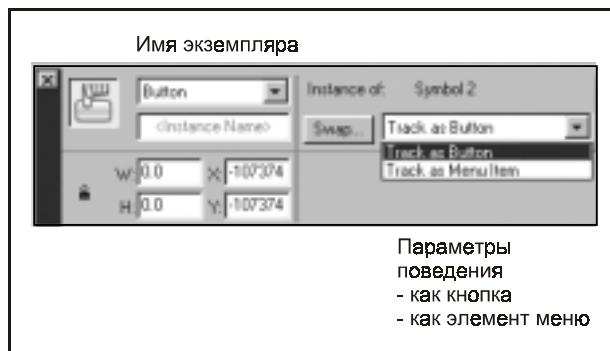


Рис. 10.11. Параметры экземпляров символа типа **Button**

Изменение типа поведения экземпляра

При создании экземпляра символа ему автоматически назначается тип поведения (behavior) родительского символа. Однако в дальнейшем тип поведения экземпляра может быть переопределен. Для того чтобы переопределить тип поведения экземпляра символа, необходимо выделить экземпляр и в списке **Symbol Behavior** Инспектора свойств выбрать требуемый тип. В результате переопределения поведения экземпляр символа, имеющего один тип поведения, приобретает все характерные особенности,ственные другому типу поведения. При этом связь между экземпляром и родительским символом сохраняется.

Табл. 10.3 иллюстрирует последствия переопределения типа поведения экземпляров.

Таблица 10.3. Результаты переопределения типа поведения экземпляров

Поведение как у родительского символа	Новое поведение		
	Graphic	Movie Clip	Button
Graphic	Можно задавать параметры воспроизведения при помощи Инспектора свойств. Монтажная линейка зависит от родительской. Не может содержать звуки и сценарии AS	Нельзя задавать параметры воспроизведения при помощи Инспектора свойств. Монтажная линейка не зависит от родительской. Может содержать сценарий и обрабатываться программно	Нельзя задавать параметры воспроизведения при помощи Инспектора свойств. Монтажная линейка не зависит от родительской. Ключевые кадры, расположенные в первых 4-х кадрах монтажной линейки, выполняют роль кадров Up, Over, Down, Hit . Остальные — игнорируются

Таблица 10.3 (окончание)

Поведение как у родительского символа	Новое поведение		
	Graphic	Movie Clip	Button
Movie Clip	Можно задавать параметры воспроизведения при помощи Инспектора свойств. Монтажная линейка зависит от родительской. Потеря сценария AS	Монтажная линейка не зависит от родительской. Может содержать любые типы символов, звуки, сценарий AS и обрабатываться программно. Может быть присвоен уникальный идентификатор	Ключевые кадры, расположенные в первых 4-х кадрах монтажной линейки выполняют роль кадров Up, Over, Down, Hit. Остальные — игнорируются. Потеря сценария AS
Button	Утрата способности реагирования на действие курсора. Можно задавать параметры воспроизведения при помощи Инспектора свойств. Монтажная линейка зависит от родительской. Кадры монтажной линейки последовательно воспроизводятся, создавая анимацию. Потеря сценария AS	Утрата способности реагирования на действие курсора. Кадры монтажной линейки последовательно воспроизводятся, создавая анимацию. Потеря сценария AS	Интерактивный элемент. Монтажная линейка не зависит от родительской. Может содержать экземпляры других символов, за исключением кнопки, звуки, сценарий и обрабатываться программно. Может быть присвоен уникальный идентификатор

Замена экземпляров

Для того чтобы автоматически заменить экземпляр одного символа на экземпляр другого символа, сохраняя все параметры трансформации и цветовые эффекты, можно выполнить одно из следующих действий:

- выделить экземпляр и щелкнуть на кнопке **Swap** Инспектора свойств. Затем в появившемся окне **Swap Symbol** (рис. 10.12) выбрать любой из имеющихся в библиотеке текущего документа символов и нажать **OK**. Кнопка **Duplicate Symbol** в нижней части окна позволяет создать копию символа в библиотеке;
- выделить экземпляр и выполнить команду главного меню **Modify>Symbol>Swap Symbol**;
- щелкнуть на экземпляре правой кнопкой мыши и из контекстного меню выбрать команду **Swap Symbol**;

- заменить один экземпляр символа другим можно также при помощи команды **Edit>Find and Replace** (см. разд. "Команды Find/Replace" гл. 15).

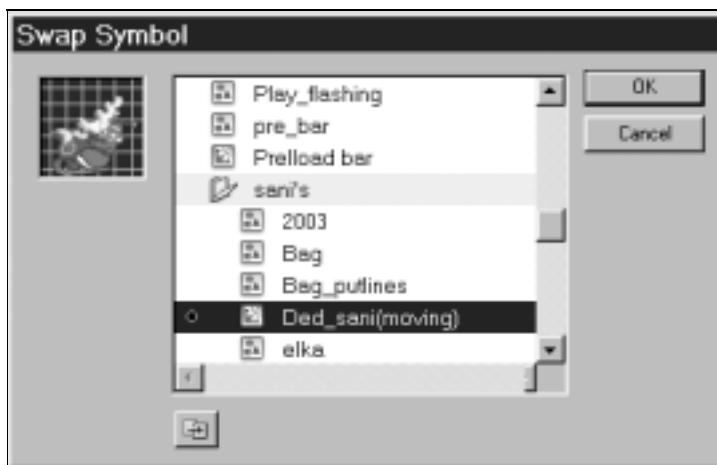


Рис. 10.12. Диалоговое окно **Swap Symbol**

В табл. 10.4 обобщены различные операции с экземплярами символов и способы их выполнения.

Таблица 10.4. Операции с экземплярами символов

Операция	Способ
Создание	Перетащить из библиотеки на сцену
Выделение	Щелкнуть инструментом Selection или обвести рамкой. Добавить к выделенному с <Shift>
Копирование	1. <Ctrl>+<D>. 2. Перетащить инструментом Selection с <Alt> или <Ctrl>. 3. Через буфер обмена
Изменение типа поведения Behavior	Выбрать другой тип поведения в поле Symbol Behavior Инспектора свойств
Трансформация	Инструмент Free Transform , возможны все операции трансформации, доступные на наложенном уровне, — масштабирование, поворот, скос, зеркальное отражение
Цветовые эффекты	Назначить эффекты цвета и прозрачность в поле Color Инспектора свойств

Таблица 9.4 (окончание)

Операция	Способ
Замена экземпляра на другой	1. Нажать кнопку Swap Инспектора свойств и выбрать в диалоговом окне Swap Symbol другой символ из списка. 2. Команда Modify>Symbol>Swap Symbol . 3. Команда Swap Symbol контекстного меню экземпляра
Конвертирование в новый символ	Команда Convert to Symbol контекстного меню экземпляра
Разделение	1. Команда Modify>Break Apart ; <Ctrl>+ приводит к разрыву связи экземпляра и символа, экземпляр становится набором векторных форм рабочего уровня. 2. Команда Break Apart контекстного меню экземпляра
Анимация	Тип анимации Motion Tween
Удаление	1. Клавиша <Delete>. 2. Команда меню Edit>Clear . 3. Удалить символ, по которому создан экземпляр, из библиотеки

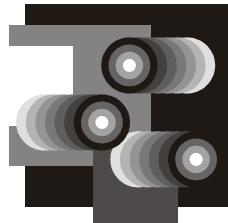
Импорт символов из других документов Flash

При работе над большим проектом, состоящим из нескольких документов, можно обмениваться символами и другими элементами библиотек между документами. Имеется несколько способов поместить в фильм объект библиотеки внешнего документа.

Для того чтобы в текущем документе открыть библиотеку любого внешнего документа Flash (с расширением fla), нужно выполнить команду главного меню **File>Import>Open External Library**. Библиотека внешнего документа имеет серый фон, находящиеся в ней объекты не могут быть отредактированы в текущем документе. Поместить объект в текущий документ можно, перетащив его в библиотеку текущего документа или прямо на сцену.

При создании нового символа можно импортировать его из внешнего документа. Для этого нужно выполнить команду **Insert>New Symbol** и в появившемся окне **Create New Symbol** щелкнуть на кнопке **Advanced**, для того чтобы получить доступ к разделу **Source**. После этого необходимо щелкнуть на кнопке **Browse** и найти документ, из которого должен быть импортирован символ. В диалоговом окне **Select Source Symbol** необходимо выбрать требуемый символ и нажать **OK**. Установка флашка **Always Update Before Publishing** приведет к тому, что импортированный символ будет обновляться всякий раз перед публикацией фильма.

Для того чтобы заменить символ в библиотеке текущего документа на другой символ из внешней библиотеки, необходимо выделить его в окне библиотеки и, щелкнув правой кнопкой мыши, из контекстного меню выбрать команду **Properties**. После этого в окне **Symbol Properties** необходимо щелкнуть на кнопке **Browse** (если символ должен быть импортирован из другого внешнего документа) или на кнопке **Symbol** (если текущий символ является импортированным и должен быть заменен на другой символ из той внешней библиотеки, которой он принадлежал). Далее, в зависимости от выполненного действия, нужно либо найти документ, из которого должен быть импортирован символ, либо в диалоговом окне **Select Source Symbol** выбрать символ, который должен заменить текущий, и нажать **OK**. Установка флажка **Always Update Before Publishing** приведет к тому, что импортированный символ будет автоматически обновляться всякий раз перед публикацией фильма.



Глава 11

Работа с библиотекой Flash

Verba volant, scripta manent.¹

Каждый документ Flash содержит свою библиотеку — **Library**, в которой хранятся импортированные объекты (растровые изображения, видеоролики, звуки) и символы. При помощи команд библиотеки можно задавать различные параметры содержащихся в ней объектов и упорядочивать их размещение.

Интерфейс библиотеки Flash

Для того чтобы открыть панель библиотеки, нужно выполнить команду главного меню **Window>Library** или воспользоваться ее клавиатурным эквивалентом <Ctrl>+<L>. Панель **Library** представлена на рис. 11.1. Окно библиотеки состоит из двух отделов: снизу расположены перечень всех объектов библиотеки, сверху находится окно просмотра выделенного объекта.

Панель библиотеки может отображаться в двух состояниях: развернутом и компактном. Для того чтобы развернуть окно библиотеки, можно щелкнуть на пиктограмме **Wide Library View** , расположенной над вертикальной полосой прокрутки библиотеки, или просто растянуть панель по горизонтали вручную. Для того чтобы привести окно библиотеки к более компактному виду, нужно щелкнуть на пиктограмме **Narrow Library View**  или изменить ширину окна вручную.

Каждый тип объектов библиотеки имеет свою пиктограмму, при помощи которой его можно легко идентифицировать (рис. 11.1). Каждый объект библиотеки также имеет уникальное имя, указанное в столбце **Name**. При развернутом окне библиотеки становится доступной следующая информация:

- Kind** — тип объекта;
- Use Count** — число использованных в фильме экземпляров данного объекта. Для отображения этой информации необходимо обновить счетчик экземпляров (см. далее в этой главе);

¹ Слова улетают, написанное остается (лат.).

- Linkage** — параметры связи, используемые при работе с общими библиотеками (Shared Libraries);
- Date Modified** — дата последнего редактирования объекта библиотеки.

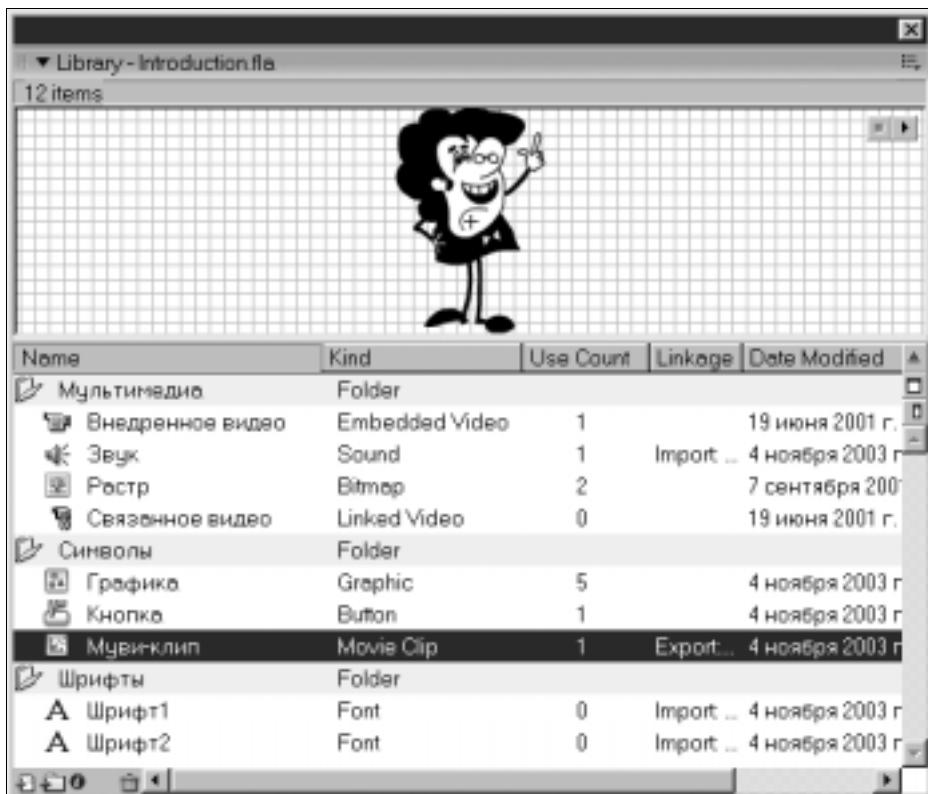


Рис. 11.1. Окно библиотеки документа

При выделении объекта библиотеки он отображается в окне просмотра. Воспроизведение монтажных линеек символов и прослушивание звуковых файлов, находящихся в библиотеке, может быть выполнено при помощи кнопки воспроизведения, появляющейся в правом верхнем углу окна просмотра . Щелчок правой кнопкой мыши в окне просмотра приводит к появлению контекстного меню, содержащего следующие команды:

- Movie's Background** — использовать цвет фона фильма в качестве фона в окне просмотра;
- White Background** — использовать белый цвет в качестве фона в окне просмотра;

- **Show Grid** — включить отображение сетки в окне просмотра (эта опция доступна только для символов). Сетка позволяет визуально оценить степень прозрачности символа.

Операции с объектами библиотеки

Рассмотрим операции, которые можно производить над объектами библиотеки.

- **Выделение.** Для того чтобы выделить объект библиотеки, нужно щелкнуть в перечне объектов по любому месту соответствующей строки. Для выделения нескольких объектов, следующих друг за другом, нужно выделить первый объект диапазона, а затем щелкнуть на последнем, удерживая клавишу <Shift>. При помощи клавиши <Ctrl> можно выделить объекты вразбивку.
- **Создание новых объектов.** Для того чтобы создать новый символ, можно воспользоваться пиктограммой **New Symbol**  в нижней части окна библиотеки или одноименной командой контекстного меню библиотеки. Для создания нового шрифтового символа (Font Symbol) или внедренного видео необходимо воспользоваться командами контекстного меню библиотеки **New Font** и **New Video** соответственно.
- **Переименование.** В качестве имени объекта библиотеки может быть использована любая последовательность символов (буквенные, числовые или знаки пунктуации); при этом допустимо использование кириллицы. Имя одного объекта библиотеки не может быть использовано в качестве имени другого ее объекта. Переименовать объект можно одним из следующих способов.
- Дважды щелкнуть на существующем имени и задать другое имя.
 - Выделить объект и, нажав клавишу <F2>, задать другое имя.
 - Щелкнуть правой кнопкой мыши по соответствующей строке, и из контекстного меню объекта выбрать команду **Rename**.
 - Выделить объект и, воспользовавшись контекстным меню библиотеки, выполнить команду **Rename**.
- **Создание копии.** Чтобы скопировать объект, можно выполнить одно из следующих действий.
- Щелкнуть правой кнопкой мыши по соответствующей строке и из контекстного меню объекта выбрать команду **Duplicate**.
 - Выделить объект и, воспользовавшись контекстным меню библиотеки, выполнить команду **Duplicate**.
- **Сортировка.** Изменить порядок размещения объектов в библиотеке можно при помощи пиктограммы **Toggle Sorting Order** , расположенной над вертикальной полосой прокрутки библиотеки. Для того чтобы отсор-

тировать объекты библиотеки по порядку на основе любого из свойств, указанных в заголовках столбцов библиотеки (**Name**, **Type** и т. д.), необходимо щелкнуть на соответствующем заголовке и затем воспользоваться пиктограммой **Toggle Sorting Order**.

- **Организация при помощи папок.** Папки выполняют в библиотеке примерно такую же роль, как и каталоги в файловой системе. Они служат для более компактной и упорядоченной организации элементов. Чтобы создать папку, можно либо щелкнуть на пиктограмме **New Folder**  в нижней части окна библиотеки, либо выполнить одноименную команду ее контекстного меню. Вновь созданная папка не содержит объектов и отображается пиктограммой . Для того чтобы поместить туда объекты, их нужно выделить и перетащить, "бросив" в требуемую папку. Допускается использование вложенных папок. Папка, наполненная содержимым, отображается пиктограммой . Чтобы раскрыть папку для доступа к ее содержимому, нужно дважды щелкнуть на ее пиктограмме или выполнить команду контекстного меню **Expand Folder**. Раскрытая папка отображается пиктограммой . Объекты, находящиеся внутри открытой папки, отображаются с отступом. Для закрытия папки нужно дважды щелкнуть на ее пиктограмме или выполнить команду контекстного меню **Collapse Folder**. Чтобы вытащить объект (или несколько) из папки, необходимо выделить его и переместить за пределы папки.

Для того чтобы поместить выделенные объекты в автоматически созданную папку, можно выполнить команду **Move to New Folder** контекстного меню объектов или библиотеки. В появившемся диалоговом окне **New Folder** нужно будет задать имя новой папки.

При работе с большим количеством символов (или импортированных объектов) имеет смысл размещать тематически близкие объекты библиотеки в отдельных папках, присваивая им лаконичные осмысленные имена.

- **Размещение в фильме.** Для того чтобы поместить экземпляр объекта библиотеки в фильм, необходимо перетащить его курсором из списка объектов или окна просмотра библиотеки на сцену или в среду редактирования символа.
- **Изменение свойств объекта.** Для того чтобы изменить свойства объекта библиотеки (изменить тип символа, задать параметры компрессии растровых изображений и звуков и т. д.), можно щелкнуть на пиктограмме **Properties**  либо выполнить одноименную команду контекстного меню библиотеки или объекта.
- **Удаление.** Для того чтобы удалить один или несколько объектов библиотеки, необходимо выделить их и либо щелкнуть на пиктограмме **Delete** , либо выполнить одноименную команду контекстного меню объекта или библиотеки.

Контекстное меню библиотеки

Библиотека, как и любая другая панель Flash, имеет свое контекстное меню. Для того чтобы его открыть, нужно щелкнуть на пиктограмме  в правом верхнем углу панели **Library**. Контекстное меню библиотеки содержит следующие команды:

- New Symbol** — создать новый символ. Выполнение данной команды приводит к появлению диалогового окна **Create New Symbol** (см. гл. 10);
- New Folder** — создать новую папку в библиотеке;
- New Font** — создать новый шрифтовой символ (*см. разд. "Использование общих шрифтовых ресурсов" гл. 8*);
- New Video** — создать пустой видео-объект;
- Rename** — переименовать выделенный объект библиотеки;
- Move to New Folder** — поместить выделенные объекты в новую папку;
- Duplicate** — создать копию выделенного символа;
- Delete** — удалить выделенный объект (или несколько объектов);
- Edit (with Application)** — редактировать символ или импортированный объект в приложении, используемом по умолчанию для редактирования файлов данного типа (например, Fireworks для растровых изображений);
- Edit with** — редактировать во внешнем приложении. После выполнения этой команды необходимо указать путь к исполняемому файлу, запускающему соответствующее приложение;
- Properties** — вызов окна настройки свойств объекта;
- Linkage** — задание параметров связи объекта. Данные установки используются при работе с общими библиотеками (*Shared Libraries*) и при необходимости создания экземпляра символа программным образом;
- Component Definition** — вызов одноименного диалогового окна, позволяющего создать компонент (*Component*) на основе выделенного символа типа муви-клип;
- Select Unused Items** — выделить все объекты библиотеки, экземпляры которых не используются в фильме;
- Update** — обновление выделенных растровых изображений и звуковых файлов библиотеки. Выполнение данной команды приводит к появлению диалогового окна **Update Library Items**, в котором галочкой нужно пометить те элементы, которые должны быть обновлены. Нажатие кнопки **Update** приведет к тому, что в библиотеку будут загружены более новые версии данных файлов. Если обновление не может быть выполнено (например, указан некорректный путь), то в столбце **Label** окна **Update Library Items** появится вопросительный знак;

- Play** — воспроизвести монтажную линейку символа или звуковой файл;
- Expand Folder** — раскрыть выделенную папку;
- Collapse Folder** — закрыть выделенную папку;
- Expand All Folders** — раскрыть все папки;
- Collapse All Folders** — закрыть все папки;
- Shared Library Properties** — задать путь, по которому будет размещена общая библиотека относительно файла-получателя (*информацию об общих библиотеках см. далее в этой главе*);
- Keep Use Count Updated** — постоянно обновлять счетчик числа использованных экземпляров объектов библиотеки. Активизация данного режима приведет к тому, что данные в столбце **Use Count** библиотеки будут автоматически обновляться при добавлении или удалении экземпляров;
- Update Use Count Now** — обновление счетчика числа использованных экземпляров по требованию.

Каждый объект библиотеки содержит собственное контекстное меню, которое можно вызвать, щелкнув на нем правой кнопкой мыши. Большинство команд контекстного меню объекта дублируют команды контекстного меню библиотеки, однако контекстное меню символов содержит некоторые уникальные команды:

- Type** — при помощи данной команды можно изменить тип символа в библиотеке. Изменение типа символа не приводит к автоматическому изменению поведения уже имеющихся экземпляров этого символа. Однако все экземпляры, созданные после переопределения типа символа, будут иметь такое же поведение, как и родительский объект библиотеки;
- к символам типа **Movie Clip** среди прочих можно также применить следующие действия:
 - **Export Flash Movie** — создание файла SWF на основе муви-клипа. Данная команда позволяет опубликовать муви-клип как отдельный фильм в формате SWF и сохранить его в файловой системе;
 - **Export SWC File** — создание файла SWC на основе муви-клипа. Формат SWC используется в Flash MX 2004 для хранения и распространения компонентов — типовых интерактивных, настраиваемых элементов, используемых при разработке Flash-проектов;
 - **Convert to Compiled Clip** — компиляция клипа в рабочей среде. Эта команда позволяет откомпилировать клип непосредственно в документе. Как правило, эту процедуру имеет смысл применять к сложному клипу, содержащему много объектов и большой фрагмент кода ActionScript, который уже не будет изменяться. Процесс публикации фильма в этом случае займет меньше времени, поскольку некоторые его элементы уже откомпилированы. Компилированные клипы не

могут редактироваться, однако с их экземплярами на сцене можно работать так же как и с компонентами, задавая значения свойств при помощи Инспектора свойств.

Стандартные библиотеки Flash

Flash содержит встроенные библиотеки, непосредственный доступ к которым можно получить из любого документа. Такие библиотеки называются стандартными (*Common Libraries*). Для того чтобы открыть стандартную библиотеку, необходимо выполнить команду главного меню **Window>Other Panels>Common Libraries** и далее выбрать требуемую библиотеку. Например, библиотека кнопок **Buttons** содержит большой набор стандартных кнопок, включающих декоративные кнопки, кнопки воспроизведения, кнопки-компоненты с настраиваемыми параметрами, ползунки и регуляторы и т. д.

Существует возможность создать собственные общие библиотеки. Для этого необходимо выполнить следующие действия:

1. Создать документ Flash и поместить в его библиотеку любые объекты, которые должны войти в общую библиотеку.
2. Сохранить документ на диске в папке *Libraries* каталога приложения Flash. Путь к этому каталогу может выглядеть следующим образом: C:/Program Files/Macromedia/Flash MX 2004/en/First Run/Libraries.
3. Для того чтобы при выполнении команды главного меню **Window>Other Panels>Common Libraries** в списке общих библиотек появилась вновь созданная библиотека, необходимо перезапустить приложение. Доступ к новой библиотеке может быть получен из любого документа Flash.

Работа с общими библиотеками (*Shared Libraries*)

При работе с проектом, включающим в себя несколько файлов SWF, содержащих различные разделы данного проекта, зачастую возникает необходимость использовать одинаковые элементы, включая их в каждый раздел. Например, можно использовать один и тот же анимированный логотип, поместив его в каждый раздел проекта. Многие типовые элементы дизайна используются в различных частях конечного проекта, обеспечивая единообразие оформления. Однако внедрение таких элементов в каждый SWF-файл проекта приведет к существенному увеличению общего потока загрузки, поскольку описывающая их информация будет вновь включаться в каждый отдельный файл. Для того чтобы многократно не дублировать одну и ту же информацию, можно прибегнуть к использованию общей библиотеки — *Shared Library*. Общая библиотека содержит ресурсы (символы, растревые изображения, звуки), которые могут быть использованы любыми файлами

проекта. При необходимости использования объекта вместо его встраивания в файл устанавливается связь (Linkage) с соответствующим элементом общей библиотеки. При этом файл, содержащий общую библиотеку, должен быть загружен один раз, после чего его содержимое может многократно использоваться другими файлами, причем применение элементов общей библиотеки практически не увеличивает объем файла-получателя. Допускается также использование нескольких общих библиотек. Общая библиотека может содержаться в одном из разделов проекта, т. е. файл SWF, представляющий определенный раздел, одновременно может содержать элементы, используемые в качестве общей библиотеки другими файлами.

Использование общих библиотек предполагает выполнение следующих этапов:

1. Определение элементов общей библиотеки.
2. Включение элементов общей библиотеки в другие файлы проекта.
3. Размещение файла SWF, содержащего общую библиотеку, и остальных файлов проекта на сервере.

Для создания общей библиотеки необходимо выполнить следующие действия:

1. Создать новый документ Flash или открыть имеющийся.
2. Поместить в документ элементы, используемые в качестве общих ресурсов. В качестве таких элементов могут использоваться символы (графический, муви-клип или кнопка), растровые изображения, звуки, шрифтовые символы (*см. гл. 8*).
3. Для каждого элемента задать параметры связи. Для этого необходимо выделить соответствующий элемент библиотеки документа и, щелкнув на нем правой кнопкой мыши, выбрать из появившегося контекстного меню команду **Linkage**. Далее в появившемся диалоговом окне **Linkage Properties** (рис. 11.2) задать следующие параметры связи:

- **Identifier** — уникальное имя, используемое для идентификации связанного элемента общей библиотеки. Данное имя должно содержать только символы латинского алфавита и не может включать пробелы.
- **Export for runtime sharing** — установка этого флагка указывает на то, что данный объект будет экспортироваться в качестве элемента общей библиотеки.
- **URL** — в это поле необходимо ввести путь к файлу, содержащему общую библиотеку относительно местоположения файла-получателя. Так, если в корневом каталоге находится файл с общей библиотекой Library.swf и папка Movie, содержащая файл Movie.swf, использующий элемент общей библиотеки, то путь, указанный в настройках связи элемента общей библиотеки, будет иметь следующий вид: ../Library.swf.

Адрес, по которому будет размещен файл, содержащий общую библиотеку относительно документа-получателя, может также быть задан при по-

мощи команды контекстного меню библиотеки **Shared Library Properties**. В результате выполнения этой команды появляется одноименное диалоговое окно, в котором можно задать требуемый путь. Путь, указанный в окне **Shared Library Properties**, автоматически прописывается в поле **URL** окна **Linkage Properties**. Верно и обратное — при изменении значения поля **URL** в окне **Linkage Properties** изменяется путь, заданный в окне **Shared Library Properties**. Следует иметь в виду, что, если файл-получатель размещается на HTML-странице, то в поле **URL** должен быть указан путь к файлу, содержащему общую библиотеку, относительно местоположения HTML-страницы файла-получателя.

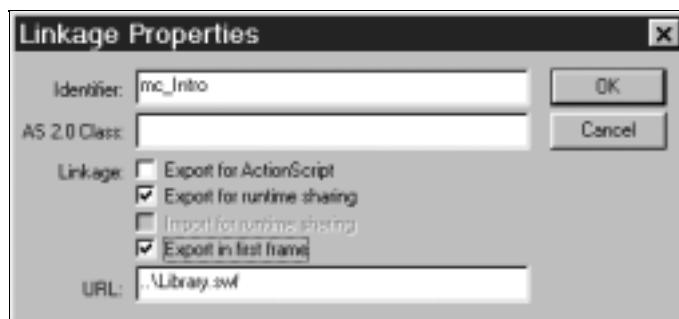


Рис. 11.2. Диалоговое окно **Linkage Properties**

◀▶ Примечание

Флажок **Export in first frame**, устанавливаемый по умолчанию, приводит к тому, что при загрузке файла, содержащего объекты, экспортруемые в качестве объектов общей библиотеки, они будут загружены до загрузки всего остального содержимого данного файла, т. е. перед содержимым его первого кадра. В случае если данный файл содержит несколько кадров (т. е. в свою очередь является разделом проекта), это может привести к задержке начала его воспроизведения. Если флажок **Export in first frame** снят, то объекты, экспортруемые в качестве общих ресурсов, будут загружены при их первом вхождении в фильм, содержащий общую библиотеку. В этом случае их экземпляры должны быть непосредственно помещены на монтажную линейку документа, содержащего общую библиотеку. Если этого не сделать, элементы общей библиотеки не будут экспортированы в конечный файл SWF. Таким образом, если документ используется исключительно как общая библиотека и не должен отображать какую-либо информацию, флажок **Export in first frame** необходимо установить. Если же документ, содержащий элементы общей библиотеки, выполняет еще и роль определенного раздела проекта и содержит какие-то объекты на своей монтажной линейке, нужно исходить из конкретной ситуации и либо установить флажок (это может привести к задержке воспроизведения), либо снять его (в этом случае элементы общей библиотеки должны быть помещены на монтажную линейку данного документа).

4. Сохранить документ (**File>Save As**) и опубликовать его (**File>Publish**), присвоив имя в соответствии с именем, указанным в поле **URL** окна **Linkage Properties** (*информация о публикации документа содержится в гл. 17*).

Чтобы включить элемент общей библиотеки в другой документ Flash, необходимо выполнить следующие действия:

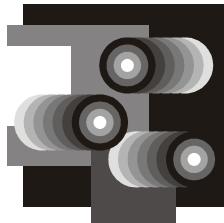
1. Открыть документ, в который должен быть помещен элемент общей библиотеки.
2. Открыть библиотеку документа, содержащего элементы общей библиотеки (**File>Import>Open External Library**).
3. Перетащить требуемые элементы из внешней библиотеки на сцену или в библиотеку текущего документа. В результате этих действий импортированным объектам в окне **Linkage Properties** автоматически будут заданы следующие параметры связи:
 - **Identifier** — уникальный идентификатор импортированного элемента общей библиотеки. Данное имя должно совпадать с именем, заданным при определении этого объекта в качестве элемента общей библиотеки;
 - **Import for Runtime Sharing** — установка этого флашка указывает на то, что данный объект будет импортирован из общей библиотеки в текущий документ;
 - **URL** — адрес SWF-файла, содержащего общую библиотеку, указанный относительно местоположения текущего документа. Если файл-получатель (текущий документ) размещается на HTML-странице, то в поле **URL** должен быть указан путь к файлу, содержащему общую библиотеку, относительно данной страницы.

Примечание

Необходимо убедиться в том, что для каждого импортируемого объекта параметры связи заданы верно.

4. Сохранить документ (**File>Save As**) и опубликовать его (**File>Publish**), разместив конечный файл таким образом, чтобы его местоположение относительно файла, содержащего общую библиотеку, соответствовало пути, указанному в поле **URL** окна **Linkage Properties**.

При размещении конечных файлов на сервере необходимо отправить и все SWF-файлы проекта и файл SWF, содержащий общую библиотеку. При этом важно разместить их друг по отношению к другу таким образом, чтобы все ссылки на общую библиотеку разрешались корректно (взаимное размещение файла-получателя и файла с общей библиотекой должно соответствовать путям, указанному в поле **URL** окна **Linkage Properties**).



Глава 12

Основы классической анимации

Хорошие указания приносят не меньшую пользу,
чем хорошие примеры.

Сенека Луций Анней Младший

В данной главе представлены некоторые сведения по созданию анимации, призванные облегчить этот нелегкий процесс для начинающих аниматоров. Компьютерная анимация имеет ряд специфических особенностей, в первую очередь обусловленных чисто технологической стороной вопроса. Сегодня один человек, по крайней мере, формально может выполнять работу, для которой еще вчера необходимо было содержать целый штат сотрудников. Flash-дизайнер может в одном лице совмещать в себе автора сценария, режиссера, художника-аниматора, оператора и звукорежиссера. Конечно, когда речь идет о сложных проектах, как правило, над их реализацией трудится целый коллектив, однако сегодня компьютер берет на себя множество технических обязанностей, делая процесс создания фильма менее рутинным.

Планирование фильма

Работа над анимационным фильмом начинается с разработки его концепции и идеи. В основе фильма может лежать литературное произведение или специально разработанный сюжет. В любом случае ядром работы является сценарий. Сценарий должен последовательно отражать все этапы развития сюжета, при этом характеризуя персонажей, выявляя их наиболее яркие стороны. Профессионально написанный сценарий не содержит лишних сведений. Каждое слово в нем направлено на то, чтобы ярче раскрыть индивидуальность героев, описать их действия, эмоции, а также создать общее

представлении о визуальной компоновке ключевых, наиболее значительных моментов.

Разработка образов персонажей

Герои мульти фильма, как правило, непохожи на реальных существ, встречающихся в жизни. В этом и состоит прелест анимационного кино. Анимационный персонаж всегда содержит некоторую условность, однако в то же время он должен быть совершенно убедительным (рис. 12.1).



Рис. 12.1. Анимационный персонаж, как правило, нереалистичен, но убедителен

Внешний вид персонажа должен полностью соответствовать его роли в фильме. Таким образом, его психологический портрет должен быть явно выражен графическими средствами (рис. 12.2).

Для начала нужно постараться сформировать персонаж из нескольких простых геометрических форм, исходя из отведенной ему роли. Любая геометрическая форма имеет свой собственный характер и в нашем восприятии

формирует представление о том или ином качестве. Так, формы, близкие к кругу, указывают на некоторую мягкость, иногда комичность, квадратные — на мощность, основательность, прямоту. Наличие острых углов свидетельствует о направленной силе, жесткости, возможно, агрессии. Таким образом, общие очертания персонажа уже дают о нем некоторое представление. Для того чтобы сделать персонаж более читаемым и выделить его на общем фоне, как правило, используется резкий контрастный контур.



Рис. 12.2. Внешний вид персонажа
должен отражать его характер

При создании анимируемого персонажа не существует каких-то определенных канонов пропорциональности, как в классическом рисунке. Здесь используется карикатурный подход, когда для выявления индивидуальности персонажа часто прибегают к гротескному преувеличению или приуменьшению отдельных частей тела. Однако при этом необходимо добиться убедительности и естественности фигуры.

Другое мощное средство разработки образа — цвет. Так же как и форма, цвет имеет свое индивидуальное звучание, которое, правда, может изменяться или дополняться при взаимодействии с другим цветом. Так, желтый цвет вызывает беспокойство; это острый, агрессивный цвет. Синий порождает ощущение глубины, покоя, отрешенности. Зеленый — это пассивный цвет, гармоничный и уравновешенный. Красный — теплый цвет, он производит впечатление мощи, энергии, целеустремленности. Конечно, на практике не стоит механически использовать цвета, сводя все к простым формальным схемам, поскольку взаимодействие цветов очень многогранно.

Одной из причин успеха фильма является *привлекательность* его героев (вспомните Винни-Пуха или Карлсона). Даже отрицательный персонаж должен быть привлекательным, иметь собственное обаяние, пусть и зловещее. Привлекательный персонаж — это персонаж, обладающий индивидуальностью. Избегайте стандартов и шаблонов. Не стоит стремиться к точной передаче деталей — главное, чтобы персонаж был ярким и живым.

Контраст часто помогает лучше выявить индивидуальность героя. Чтобы подчеркнуть высокий рост и худобу героя, его компаньоном можно сделать толстого коротышку (рис. 12.3), а красоту очаровательной героини можно еще больше усилить, заставив добиваться ее сердца неказистого ухажера.

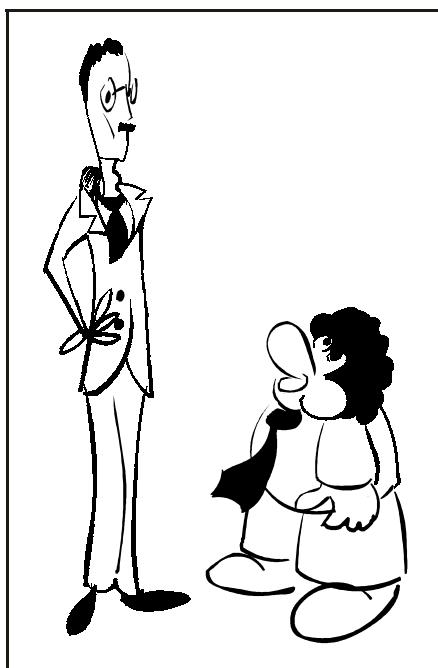


Рис. 12.3. Контраст помогает выявить индивидуальность

При разработке персонажа также следует помнить, что чересчур сложный рисунок с большим количеством деталей неизбежно вызовет трудности в процессе анимации. Многочисленные подробности придется перерисовывать из кадра в кадр, что приведет к увеличению трудоемкости рабочего процесса и повлияет на конечный размер файла. Необходимо снабдить пер-

сонажа только теми деталями, которые действительно влияют на формирование его образа, на создание его психологического портрета.

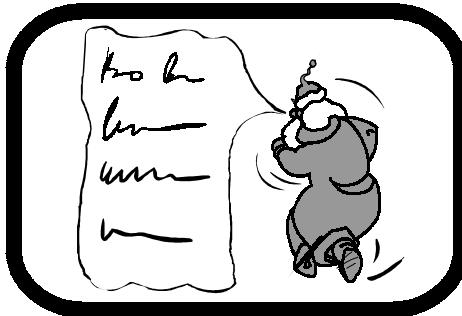
Понятие тайминга.

Раскадровка

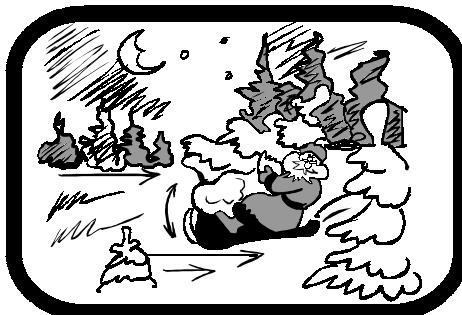
Когда образы персонажей созданы, т. е. актеры готовы приступить к игре, возникает следующая задача: как сделать их движения живыми, ясными и убедительными. Здесь ключевым является понятие *тайминга* (Timing). Слово "тайминг" в переводе с английского означает "распределение во времени", или "хронометрирование". Применительно к анимации определение этого термина можно сформулировать следующим образом. *Тайминг* — расчет распределения движения во времени.

Скорость воспроизведения ролика — это всегда фиксированная величина. Flash-анимация чаще всего использует значение 12 кадров в секунду (или 24 кадра в секунду). Длительность фильма, как правило, бывает известна заранее, и таким образом, в задачу режиссера входит распределение всего действия на протяжении этого времени. Весь фильм при этом разделяется на отдельные эпизоды, содержащие однородные, тематически близкие фрагменты сюжета, которые, в свою очередь, дробятся далее на отдельные сцены. На этом этапе очень важно принять верные решения о том, на протяжении какого времени будет длиться то или иное действие, какую паузу необходимо выдержать перед решающим действием, чтобы добиться максимального воздействия на зрителя, в течение какого времени выражается реакция героя и т. д. Все эти вопросы разрешаются методом проб и ошибок. Очень полезным бывает анализ профессиональных анимационных фильмов, где авторы виртуозно обращаются с субстанцией времени.

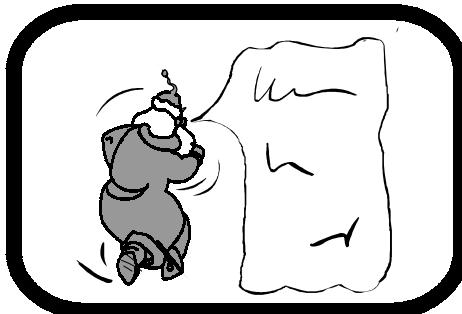
Важнейшим инструментом, выполняющим в дальнейшем роль плана при создании анимации, является раскадровка. *Раскадровка* (Storyboard) представляет собой набор рисунков, отражающих все ключевые моменты развития сюжета, а также психологическое состояние героев фильма. Раскадровка должна давать убедительное визуальное представление о развитии сюжета, поведении героев и общей композиции кадра, определяемой размером рабочей области. Раскадровочные рисунки снабжаются пояснительным текстом, содержащим реплики героев, описывающим характер действия, а также дающим указания оператору (наезд, панорамирование, съемка с обратной точки и т. д.). На рис. 12.4 представлена раскадровка короткого рекламного ролика для сети Интернет, представляющая собой визуальную последовательность действий.



1. Текстовка: "Во все времена Дед Мороз отправлялся в путь, чтобы успеть к Новому Году"
Диктор: "Вот как это было 200 лет назад..."



2. Панорамирование и постепенный наезд на лицо Деда Мороза (санки прыгают по кочкам)
Звуковые эффекты: свист вьюги, скрип санок, возгласы Деда Мороза.

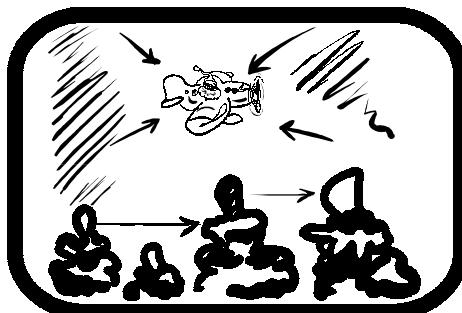


3. Текстовка: "Наступил XX век"
Диктор: "Да... Все летит, все меняется..."

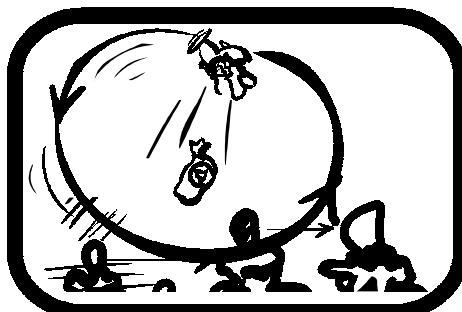


4. Переброс камеры. Лицо Деда Мороза крупно. Медленный отъезд камеры.
Звуковые эффекты: шум двигателя самолета, возгласы Деда Мороза.

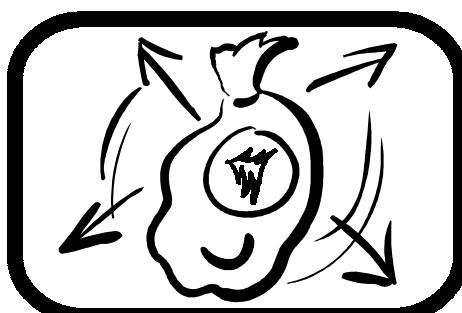
Рис. 12.4. Раскадровка рекламного интернет-ролика (начало)



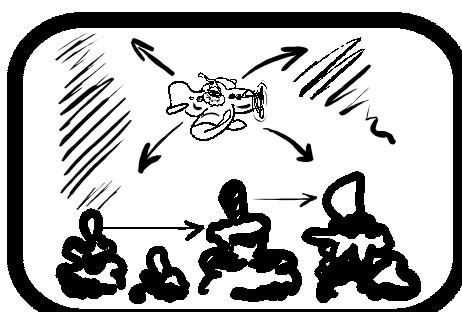
5. Панорамирование. Медленный отъезд камеры. (внизу проносятся елки) Звуковые эффекты: шум двигателя самолета, возгласы Деда Мороза отдаляются.



6. Самолет делает мертвую петлю, из него выпадает мешок с подарками и летит на зрителя.



7. Мешок с подарками крупно. Звуковые эффекты: свист падающего мешка.
Повтор сцены три раза.



8. Наезд на лицо Деда Мороза.
Звуковые эффекты: возгласы.

Рис. 12.4. Раскадровка рекламного интернет-ролика (продолжение)



9. Лицо крупно. Дед Мороз подмигивает. Медленный отъезд камеры.
Звуковые эффекты: шум мотора автобуса.

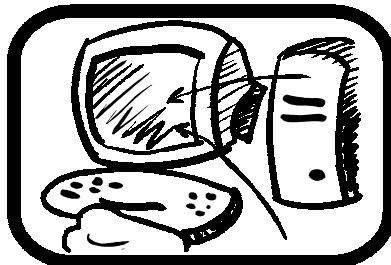


10. Панorama. Автобус подпрыгивает, сзади движется пейзаж города, мимо проносятся фонари.
Звуковые эффекты: шум мотора автобуса, городской шум.

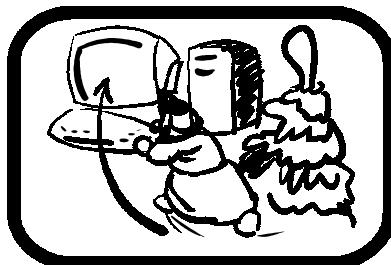


11. Текстовка: "Пришло третье
тысячелетие"
Диктор: "Железный конь приходит на
смену..."

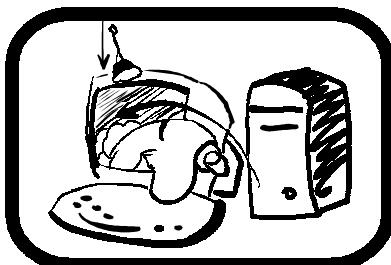
Рис. 12.4. Раскадровка рекламного интернет-ролика (продолжение)



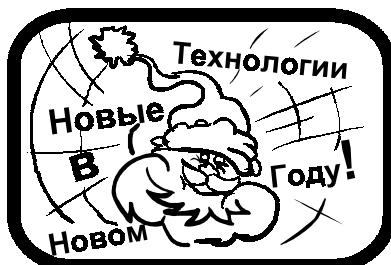
12. Компьютер крупно.
Отъезд на общий план.
Звуковые эффекты:
компьютерные звуки.



13. Дед Мороз вбегает в
кадр...
Звуковые эффекты: пыхтение,
скрип сапог.



14. ...Впрыгивает в монитор,
теряя шапку, которая,
повисев в воздухе,
устремляется вниз за ним.
Возгласы: "Ух-ты, вот так
штука!!!"



15. Летит по кабелю. Лицо
крупно.
Текстовка: "Новые технологии
в Новом году" (эффект typing).

Рис. 12.4. Раскадровка рекламного интернет-ролика (окончание)

Основные принципы анимации

При создании анимации мы имеем дело не с реальным движением, а с набором статичных рисунков, которые должны вызывать у зрителя иллюзию движения. Для того чтобы придать движению убедительность, довести до зрителя весь закладываемый в него смысл, необходимо, чтобы в рисунках отражались закономерности механики, действующие в реальной жизни. В то же время, чтобы сделать анимационное движение понятным для зрителя, оно всегда гиперболизируется, а иногда доводится до гротеска. При передаче движения задача состоит не в том, чтобы сделать движение натуральным, а в том, чтобы сообщить ему естественность, ясность и лаконичность (рис. 12.5).

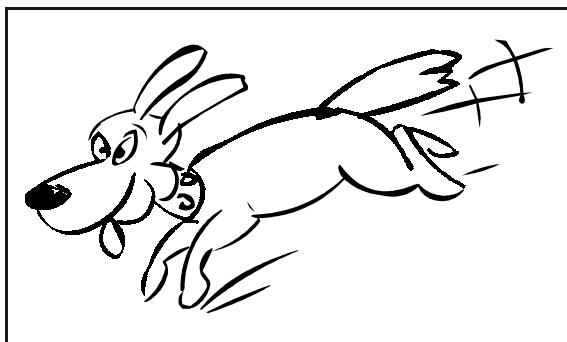


Рис. 12.5. Анимационное движение должно быть выразительным и лаконичным

Технология создания анимации включает в себя разработку компоновок и последующее фазование движения. *Компоновки* — это ключевые моменты движения. Они отрисовываются с максимальной тщательностью и являются костяком анимации. В кадре не должно быть нейтральных предметов и персонажей — каждый элемент композиции необходимо снабдить выразительными деталями характера, внешнего вида, поведения.

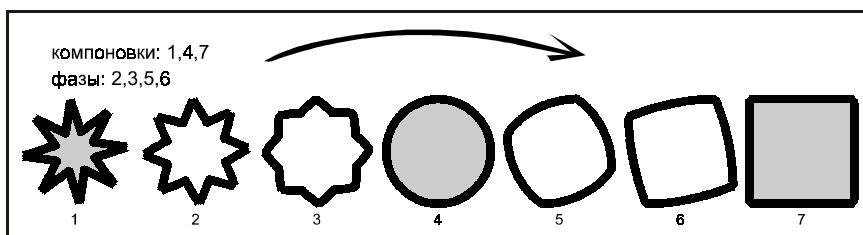


Рис. 12.6. Автоматическое фазование трансформации формы

Фазование представляет собой сглаживание перехода между соседними компоновками. Обычно художник-аниматор занимается разработкой компоновок, а за фазование отвечает фазовщик. В некоторых случаях, когда речь идет о достаточно простых переходах (несложные преобразования формы, перемещение объекта, вращение, переход цвета и т. д.), рутинную работу, связанную с фазованием, можно переложить на Flash, используя возможности автоматической анимации (tweening) формы или движения (рис. 12.6).

Вес

Все предметы имеют *вес*. Это необходимо помнить при расчете движения. Чем больше масса предмета, тем большей инерцией он будет обладать, тем сильнее будет его воздействие на окружающие предметы. Соответственно, для того чтобы привести в движение предмет, обладающий значительной массой, необходимо приложить достаточно большое усилие, и вряд ли такой предмет мгновенно наберет высокую скорость. Вес также определяет характер движения. Клочки газеты или перья падают на землю не так, как пудовая гиря. Здесь необходимо иметь в виду два аспекта: силу притяжения и сопротивление воздуха (рис. 12.7).

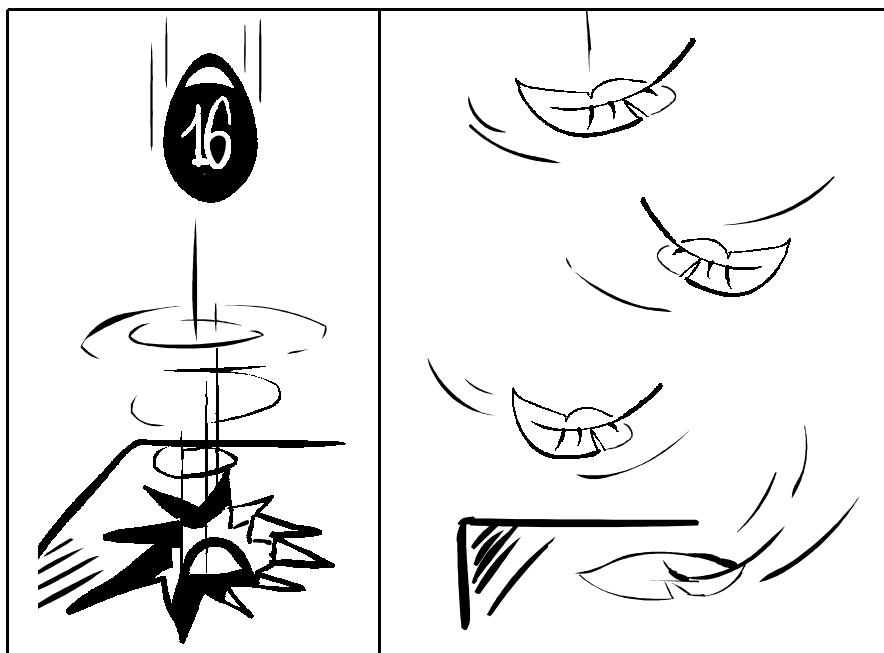


Рис. 12.7. Различия в характере движения пудовой гирь и перьев обусловлены разницей в весе и формой

Форма

Форма также является постоянным атрибутом любого движущегося объекта. Чтобы подчеркнуть характер движения и придать ему выразительность, используется так называемый эффект дышащего тела, когда движущаяся форма подвергается определенным деформациям. Однако при этом важно помнить о сохранении внутреннего объема (рис. 12.8).

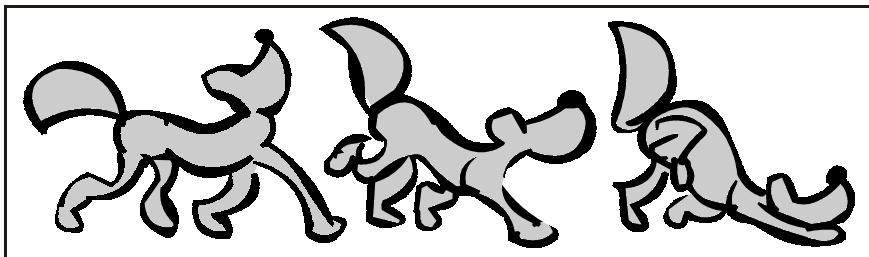


Рис. 12.8. При перемещении предметы движутся как полупустые мешки с мукоj, сохраняя внутренний объем



Рис. 12.9. Остаточное движение обусловлено инерцией

Тело персонажа — это сложная система, состоящая из различных элементов: костей скелета, мышц, прически и др. Длительность *перехода от статики к движению* зависит от массы тела и его энергии. Тело не может мгновенно набрать высокую скорость или сразу резко остановиться. Если машина резко затормозит на большой скорости, то она еще проедет некоторое расстояние по инерции, отклонившись назад, потом резко качнется вперед и, наконец, замрет на месте. Это называется *остаточным движением* (рис. 12.9).

При движении из-за инерционности различные части фигуры перемещаются по-разному. Так, например, при подъеме руки движение через суставы постепенно передается от плеча к кисти, причем кисть запаздывает во время подъема и далее, достигнув верхней точки, при опускании руки вновь отстает от предплечья (рис. 12.10).

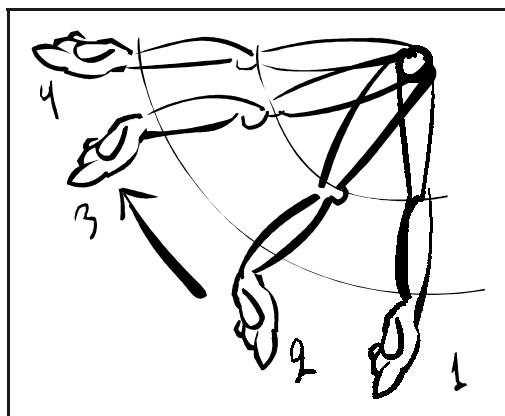


Рис. 12.10. Остаточное движение кисти руки

Точно так же мягкие части продолжают двигаться после остановки скелета. Когда собака остановилась, ее хвосту потребуется еще какое-то время на то, чтобы прийти в состояние покоя. Необходимо тщательно проанализировать механику движения, верно определить источник, направление силы и эффект, оказываемый силой на отдельные элементы тела персонажа.

Отказное движение (замах)

Отказное движение (take) позволит сделать ударный акцент и лучше довести до зрителя смысл, закладываемый в последующее действие. Например, удару всегда предшествует замах, подготавливающий зрителя к последующему быстрому действию (рис. 12.11).

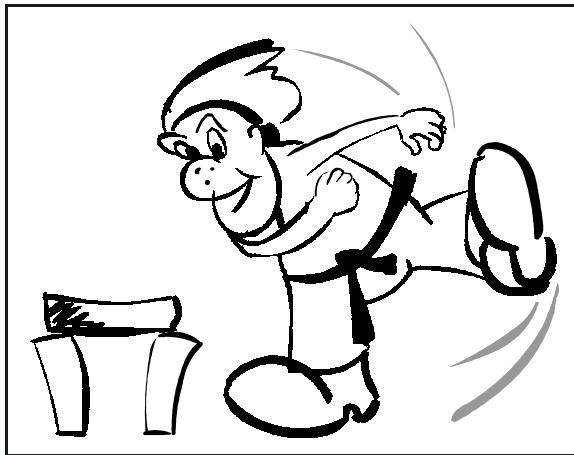


Рис. 12.11. Замах позволяет лучше прочитать следующее за ним действие

Отказное движение должно заострить внимание зрителя, заставить его ожидать последующего эффекта, и таким образом еще до выполнения самого движения зритель уже будет предвосхищать его. В этом случае само движение будет хорошо читаться и создаст необходимый драматический эффект.

Масштаб движения

Длительность движения зависит от относительного размера персонажа. Крупному и грузному герою необходимо больше времени для выполнения какого-либо действия, чем коротышке. В сцене прогулки человека с таксой на один цикл походки человека приходится три полных цикла походки собаки (рис. 12.12). Однако результат тайминга определяется конкретной ситуацией и в другом случае может быть совершенно иным (скажем, если огромный сенбернар тащит за собой маленькую хозяйку).

Для того чтобы передать масштабность происходящего действия, необходимо добиться эффекта замедленного движения. Так, при анимации землетрясения разрушение зданий должно происходить очень медленно, чтобы подчеркнуть глобальный характер происходящего.

Чем медленнее движение, тем больше фаз придется использовать для его передачи. Сложное медленное движение требует большого количества рисунков. Для передачи медленного движения можно увеличить длительность просмотра компоновок, разместив несколько простых кадров после каждого ключевого кадра, содержащего определенную фазу движения. В этом случае необходимо особенно тщательно отрисовать компоновки.

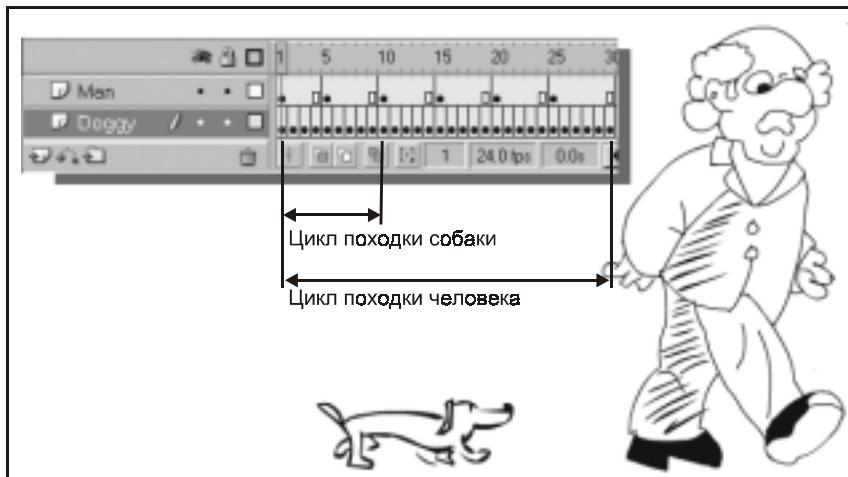


Рис. 12.12. Длительность цикла движения зависит от относительного размера персонажа

Быстрое движение, напротив, не требует большого числа кадров. Здесь можно подготовить зрителя к действию, используя замах, и затем ошеломить мгновенным поступком, добившись ясности прочтения, используя остаточное движение (рис. 12.13).

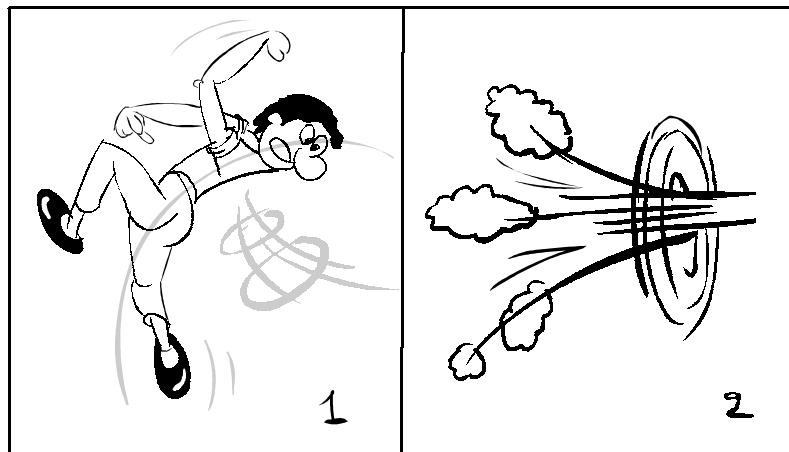


Рис. 12.13. Для очень быстрого движения достаточно минимального количества компоновок

При последовательном движении с промежуточными остановками (например, прыгающая лягушка) тело выходит из статики, постепенно приобретая ускорение, затем, достигнув максимальной скорости, начинает замедляться, вновь входя в статику, после чего весь цикл повторяется снова.

Анимация походок

При ходьбе тело перемещается, попеременно опираясь то на одну, то на другую ногу, постоянно теряя и восстанавливая равновесие. Фактически процесс ходьбы заключается в переменном перемещении и поддержании центра тяжести (рис. 12.14).



Рис. 12.14. Цикл походки человека (I)

Наиболее устойчивой фазой цикла походки является момент, когда одна нога выносится вперед с опорой на каблук, и центр тяжести находится посередине площади опоры (первая фаза). Далее туловище наклоняется, теряя равновесие, однако падения не происходит из-за согнутой в колене ноги (вторая фаза). После этого согнутая нога принимает опору, туловище выпрямляется, центр тяжести переносится назад и вторая нога заносится для нового шага (третья фаза). Затем весь процесс зеркально повторяется для другой ноги. Кроме поступательного движения, тело совершает еще целый ряд движений, которые необходимо иметь в виду, чтобы добиться естественности при передаче походки.

На каждом шаге туловище поднимается и опускается, производя *вертикальные колебания*. В результате этого макушка головы движется по синусоидальной траектории.

При ходьбе туловище поднимается и опускается, одновременно с этим наклоняясь то в одну, то в другую сторону. Это движение является *поперечным*

и *горизонтальным колебанием* и вызывается тем, что центр тяжести переносится с одной ноги на другую для восстановления равновесия.

Амплитуда этих колебаний зависит от индивидуальных особенностей человека (длинноногий человек при ходьбе раскачивается сильнее, чем коротконогий).

Кроме указанных движений, при ходьбе имеют место *вращательные движения*, состоящие в том, что бедро поворачивается в направлении шагающей вперед ноги, а при следующем шаге — в противоположную сторону (рис. 12.15).

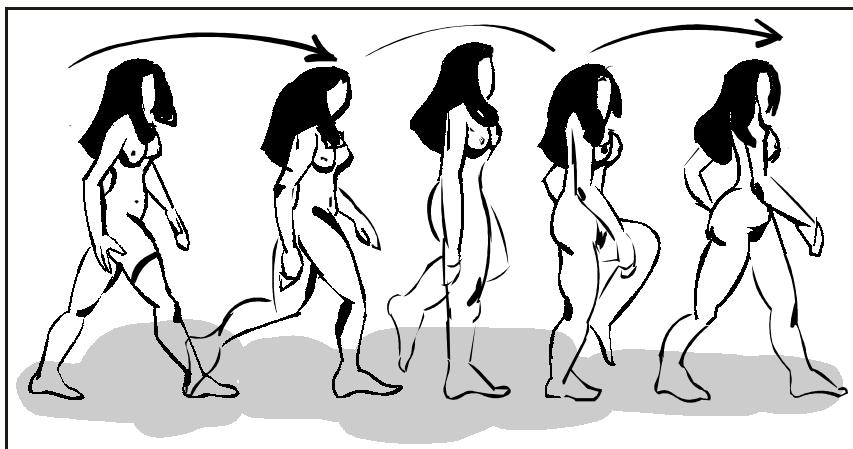


Рис. 12.15. Цикл походки человека (II)

При восхождении по наклонной плоскости центр тяжести тела на каждом шаге поднимается вверх и перемещается вперед, а при схождении — соответственно, опускается (рис. 12.16).

Бег представляет собой более динамичный и стремительный цикл, чем ходьба. Характерным для бега является момент, когда либо ни одна нога не касается земли, либо касается на очень короткий момент для толчка. Динамизм бега подчеркивается наклоном туловища и широким размахом ног (рис. 12.17).

При анимации походки четвероногих важно скоординировать движения конечностей (рис. 12.18).

Длительность полного цикла походки зависит от размера. Так, например, такса успеет совершить десять-двенадцать шагов за один шаг слона.

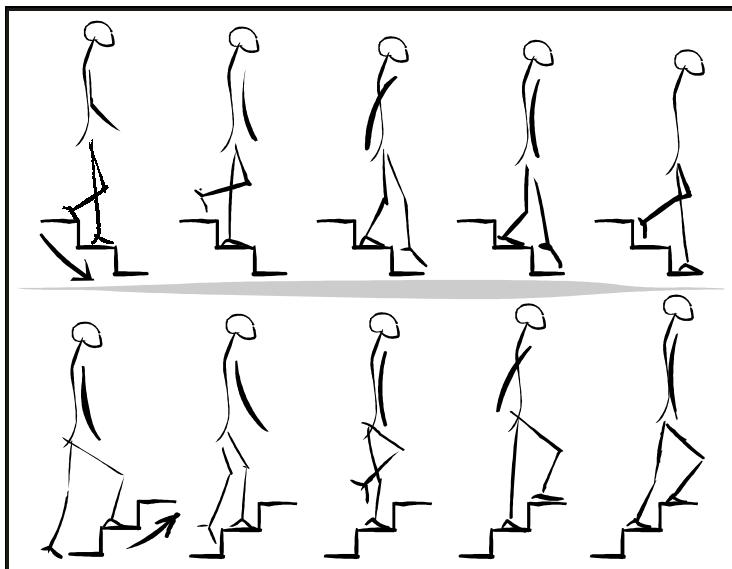


Рис. 12.16. Движение по наклонной плоскости

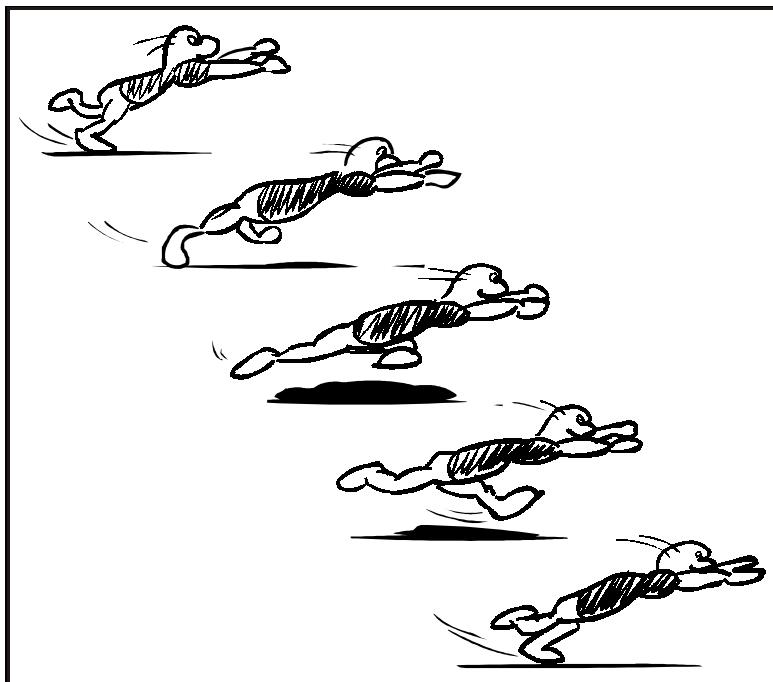


Рис. 12.17. Цикл бега

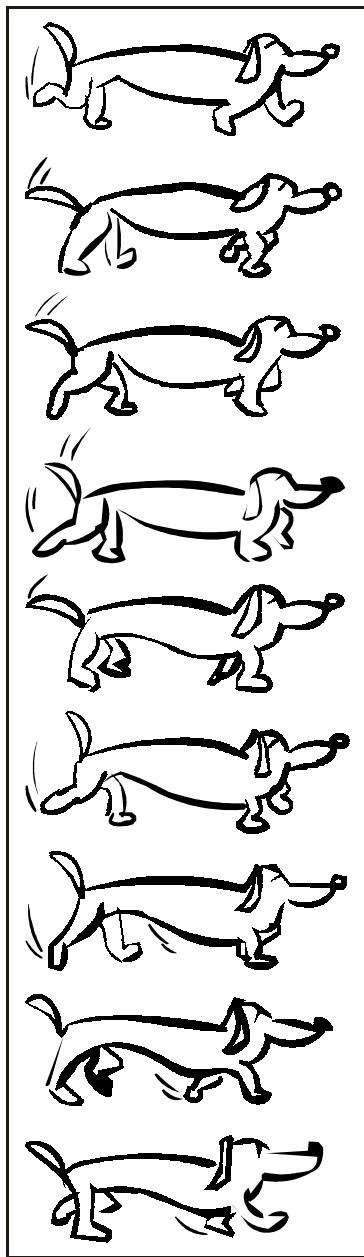


Рис. 12.18. Цикл походки таксы

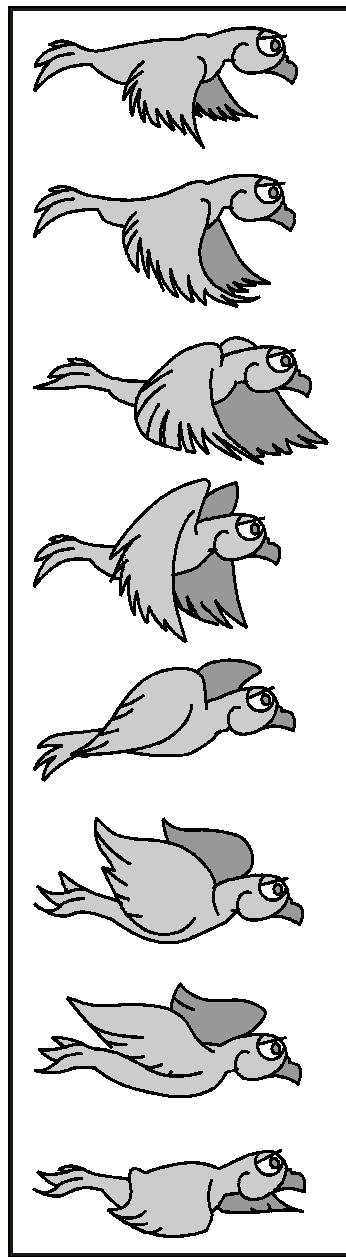


Рис. 12.19. Цикл движения птицы

Анимируя полет птицы, нужно учитывать несколько особенностей. Птица должна преодолевать сопротивление воздуха — этим объясняется сложная

механика ее полета. При взмахе крыльями вверх птица отталкивается от воздуха, причем ее туловище несколько опускается, а спина прогибается вниз. Когда крыло опускается, перья пропускают воздух, туловище приподнимается и спина снова выгибается вверх (рис. 12.19).

Анимационные эффекты

Огонь

Огонь представляет собой раскаленные потоки газов. Движение пламени обусловлено потоками воздуха, попадающими в зону огня. Приток холодного воздуха вызывает турбулентные завихрения. Теплые потоки воздуха устремляются вверх и вместе с искрами и продуктами горения выносятся из зоны пламени.

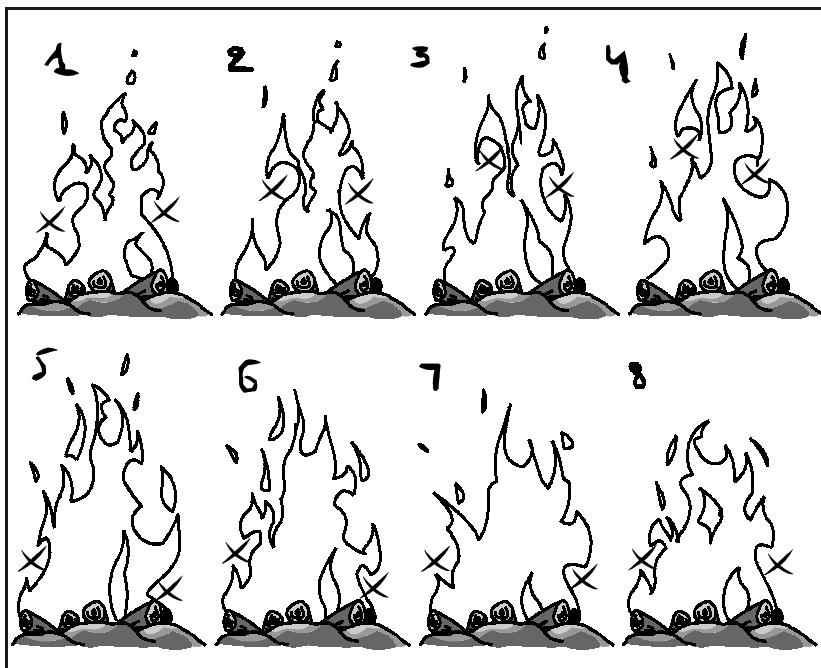


Рис. 12.20. Движение пламени

Пламя — это очень подвижная субстанция, но в казалось бы хаотичном движении языков пламени, тем не менее, имеется определенная закономерность. Чтобы движение пламени было естественным, необходимо передать его восходящий характер, а для этого определенные участки пламени, поднимаясь вверх, в незначительной степени трансформируются — по ним глаз

может проследить за общим движением (рис. 12.20). Кроме совершения восходящего движения, пламя может колебаться, изменяясь в размерах и попутно вытягиваясь вверх и вновь оседая.

Вода

Вода также чрезвычайно подвижная субстанция. При взаимодействии с водой создаются многочисленные брызги, которые движутся по своим собственным траекториям (как правило, параболическим). Скорость движения капли зависит от ее массы; легкие капли держатся в воздухе дольше тяжелых. На рис. 12.21 представлены основные фазы анимации всплеска, вызванного брошенным в воду камнем.

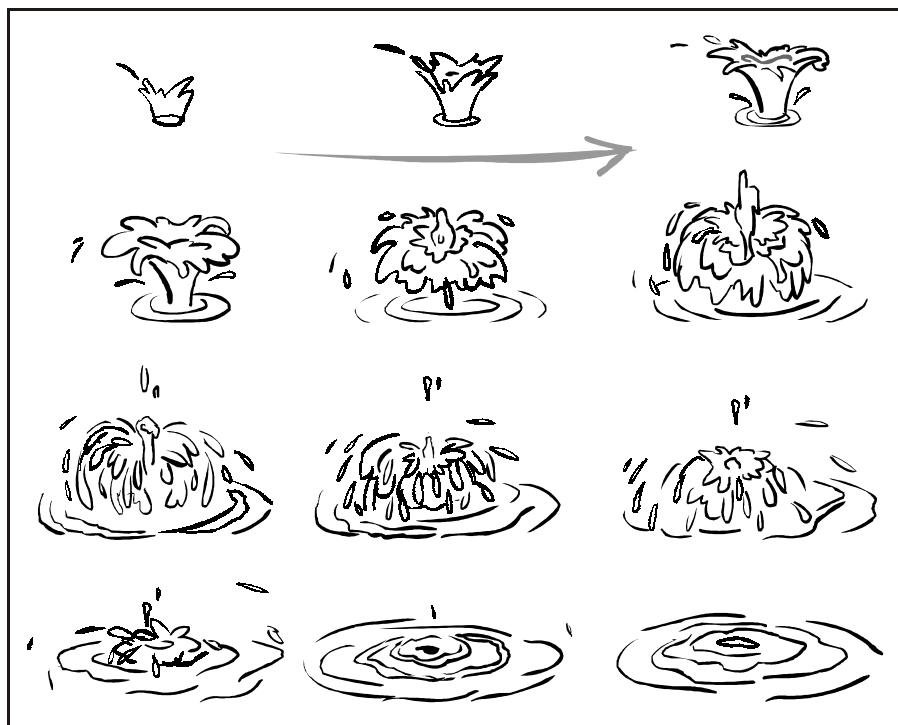


Рис. 12.21. Основные фазы анимации всплеска

Вначале (первые четыре фазы) в результате вытеснения воды формируется конический всплеск, похожий на ступу, далее в центре ступы вырастает фонтанчик (следующие три фазы), после чего масса воды распадается на отдельные брызги, всплеск затухает, по поверхности воды расходятся кольца, которые постепенно гаснут, разрываясь (последние три фазы).

Ветер

Действие ветра проявляется в поведении окружающих предметов, на которые он воздействует. Сам ветер невидим, однако его движение и сила явно выражаются характером перемещения предметов: верхушек деревьев, листьев, пыли, обрывков газет и т. д. (рис. 12.22).



Рис. 12.22. Направление и сила ветра проявляются в воздействии на окружающие предметы

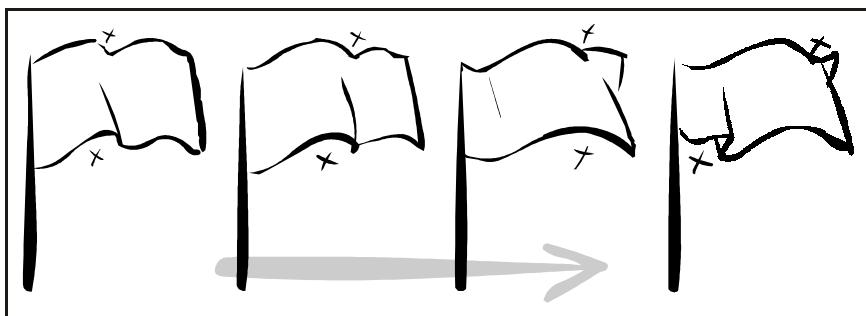


Рис. 12.23. Анимация флага

Ветер вызывает завихрение воздуха. Полоскание флага на ветру представляет собой непрерывный процесс формирования волн у основания и перемещение их к краю полотнища. Здесь, так же как и при анимации огня, необходимо выбрать определенные участки и перемещать их в соответствии с

направлением ветра. Конечная фаза должна плавно перетекать в начальную (рис. 12.23).

Взрыв

Взрыв представляет собой стремительное расширение материи. Движение вовне можно реализовать с использованием увеличивающейся в размерах вспышки, разлетающихся осколков. Анимация взрывов требует быстрой смены изображения. Часто прибегают к контрастной смене цвета фона или мельканию самой вспышки (рис. 12.24).

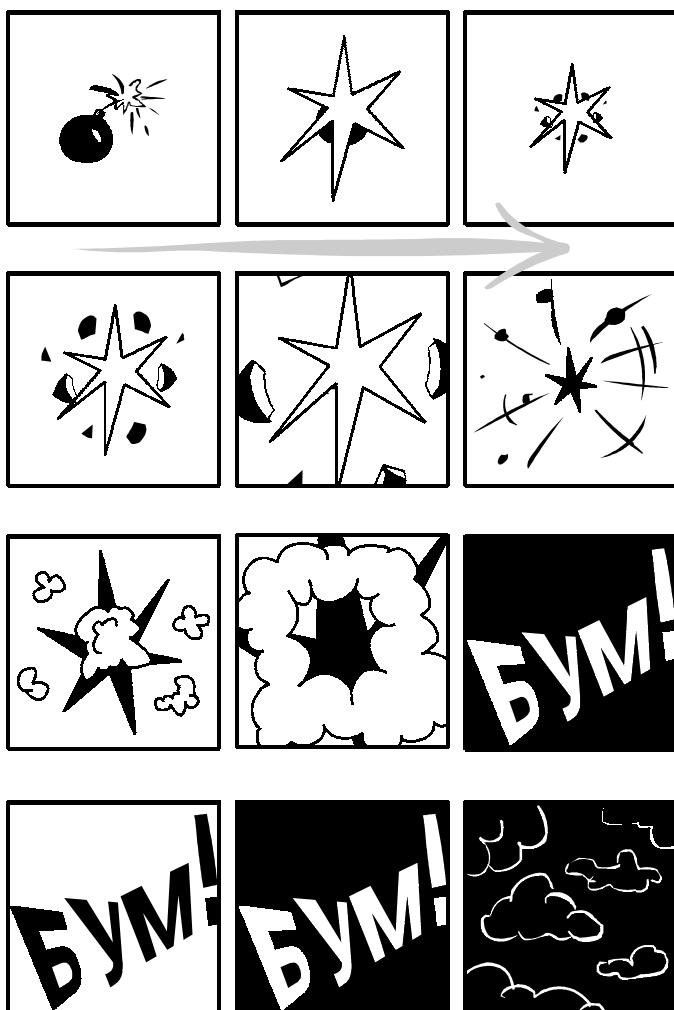


Рис. 12.24. Анимация взрыва

Speedlines

Чтобы придать движению динамизм, сделать его более драматичным и читаемым, можно использовать *спидлайны* (speedline). Спидлайн — это след, оставляемый быстро движущимся предметом, который подчеркивает траекторию движения и придает ему стремительность. Спидлайны могут следовать за объектом, повторяя его путь, или время от времени появляться, например, в местах смены направления движения (рис. 12.25). Спидлайн помогает акцентировать движение и используется в тех случаях, когда необходимо добиться передачи движения с очень большой скоростью.

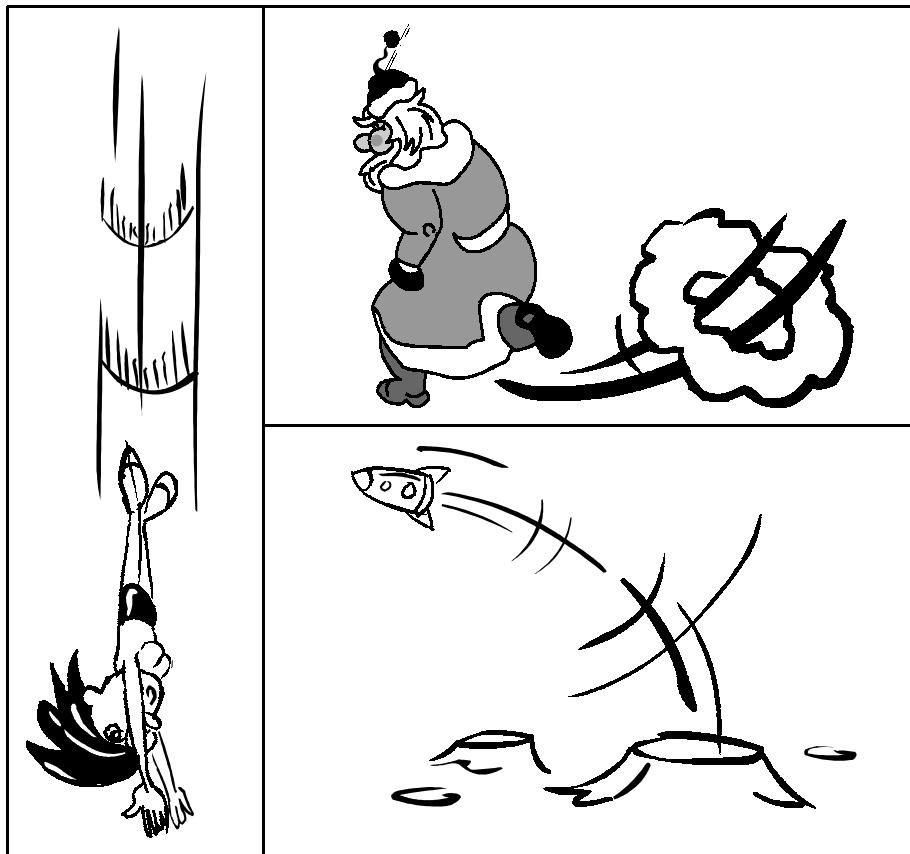
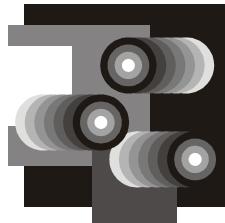


Рис. 12.25. Применение спидлайнов



Глава 13

Автоматическая анимация

Ошибка дилетанта: желание связать непосредственно фантазию и технику.

Geme

Flash предоставляет различные способы создания анимации. К их числу можно отнести:

- покадровую анимацию;
- автоматическую анимацию;
- программную анимацию.

О *покадровой* (Frame by frame), или *ручной* анимации речь шла в гл. 9. Покадровая анимация является наиболее трудоемкой, но в то же время позволяет добиться точной передачи нюансов движения и, как правило, используется для создания художественных, детально проработанных эффектов (например, анимация пламени).

Программная анимация (Program animation) создается при помощи языка сценариев ActionScript и позволяет реализовать разнообразные динамические эффекты, связанные с перемещением, изменением свойств, появлением и исчезновением объектов. Программная анимация обладает широкими возможностями и применяется для разработки декоративных эффектов анимации, связанных со сложным характером движения и/или нециклическим поведением объектов (например, эффект старого кино, имитирующий мерцание царапин на пленке). (*О создании программной анимации будет сказано в гл. 19 и 21.*)

Автоматическая анимация (Tweened animation) создается в рабочей среде на монтажной линейке основного фильма или символа (графического типа или типа мви-клип). Автоматическая анимация, в отличие от покадровой, не требует ручной прорисовки фаз. Достаточно задать начальное и конечное состояние анимируемого объекта, а все промежуточные состояния будут рассчитаны автоматически, в связи с этим автоматическую анимацию также называют *расчетной*. Диапазон применения автоматической анимации очень

широк. С ее помощью можно создавать эффекты, связанные с изменением формы, цвета и прозрачности, перемещением, вращением и масштабированием объектов. Процесс создания автоматической анимации значительно менее трудоемкий, чем покадровой, и во многих случаях позволяет добиться превосходных результатов.

Зачастую в фильме используется сочетание всех трех способов, что позволяет решать широкий спектр задач, поставленных перед аниматором.

Создание автоматической анимации

Существуют два типа автоматической анимации:

- анимация формы (Shape tweening), или морфинг;
- анимация движения (Motion tweening).

Наличие разных типов автоматической анимации обусловлено иерархией графических объектов, существующей в Flash, т. е. разделением на рабочий и наложенный уровни. Каждый тип анимации может быть применен только к объектам, принадлежащим соответствующему уровню иерархии. Это, в свою очередь, определяет возможности, которыми обладает анимация формы и анимация движения.

Анимация формы (трансформация формы) может быть применена только к объектам *рабочего* уровня. Соответственно, при помощи данного типа анимации можно создавать эффекты, связанные с изменением базовых свойств объектов (форма, цвет).

Анимация движения может быть применена только к объектам *наложенного* уровня. В связи с этим в рамках анимации движения можно трансформировать объект, не изменяя его базовых свойств, и при работе с экземплярами символов применять цветовые эффекты.

Диапазон кадров автоматической анимации всегда включает в себя два ключевых кадра (keyframes), находящиеся в начале и в конце диапазона, и расположенные между ними кадры трансформации (tweened frames). Начальный и конечный ключевые кадры описывают соответственно начальное и конечное состояния анимируемого объекта, а кадры трансформации описывают переход из начального состояния в конечное. Кадры трансформации создаются автоматически и представляют собой единую последовательность кадров. Содержимое кадров трансформации не может быть изменено. Все изменения вносятся только в начальный или конечный ключевые кадры.

Анимация формы (морфинг)

Анимация формы (Shape tweening) может быть применена исключительно к объектам *рабочего* уровня.

С ее помощью можно выполнить следующие виды трансформаций:

- изменение формы;
- изменение цвета и прозрачности;
- перемещение;
- изменение масштаба, скос (наклон).

Для того чтобы создать автоматическую анимацию формы, необходимо выполнить следующие действия:

1. Создать новый слой (**Insert>Timeline>Layer**).
2. В начальном ключевом кадре создать объект *рабочего* уровня (векторную форму и/или контур), например, красный квадрат.
3. Исходя из скорости воспроизведения монтажной линейки, рассчитать необходимое для выполнения анимации число кадров и создать конечный ключевой кадр (**Insert Keyframe** или **Insert Blank Keyframe**). Если анимация должна выполняться за 2 секунды, то при скорости воспроизведения 12 кадров в секунду новый ключевой кадр должен иметь номер 24.
4. Поместить измененный объект *рабочего* уровня в пустой ключевой кадр или изменить скопированный из начального кадра объект. В 24-м кадре можно в качестве примера создать синюю звезду. Если координаты объектов в начальном и конечном ключевых кадрах различны, то в результате анимации будет выполняться перемещение.
5. Задать параметры автоматической анимации. Для этого необходимо выделить *первый* ключевой кадр диапазона и в Инспекторе свойств установить следующие настройки (рис. 13.1).

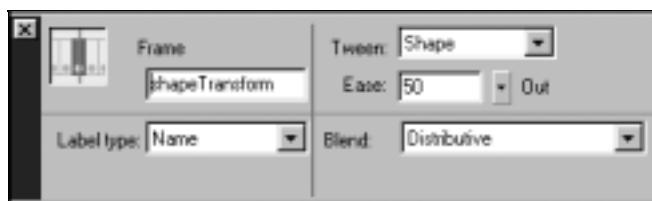


Рис. 13.1. Настройка параметров анимации формы (Shape tweening)

- В раскрывающемся списке **Tween** выбрать тип анимации **Shape**. При этом начальный и конечный ключевые кадры будут соединены прямой неразрывной линией со стрелкой. Если режим **Tinted Frames** контекстного меню монтажной линейки активен, то диапазон кадров автоматической анимации формы будет выделен зеленоватым цветом (см. табл. 13.1). В случае если в анимации допущена ошибка, на мон-

тажной линейке появляется пунктирная линия, а при выделении соответствующего кадра в Инспекторе свойств появляется кнопка с пиктограммой восклицательного знака , свидетельствующая об ошибке в анимации.

- Поле **Ease** используется для задания относительного замедления анимации в начале или в конце диапазона. Значение, принимаемое параметром **Ease**, изменяется в диапазоне от -100 до $+100$ и может быть непосредственно введено вручную либо задано при помощи вертикального слайдера. Отрицательные значения приводят к замедлению анимации в начале (**Easing In**) и, соответственно, к ускорению в конце. Положительные значения приводят к замедлению анимации в конце (**Easing Out**) и к ускорению в начале. При задании отрицательного значения рядом с полем **Ease** появляется предлог **In**, а при задании положительного значения — предлог **Out**. Замедление анимации достигается не за счет замедления скорости воспроизведения монтажной линейки, а за счет более плотного размещения фаз анимируемого объекта в начале или в конце диапазона кадров трансформации (рис. 13.2).

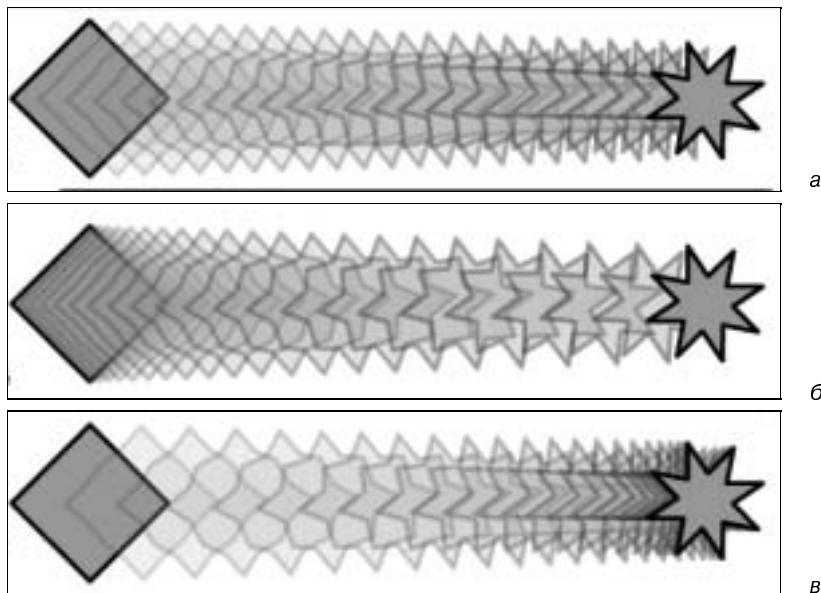


Рис. 13.2. Применение параметра **Easing**:

- а — замедление отсутствует (**Ease** = 0);
 б — замедление в начале (**Ease** = -100 , **In**);
 в — замедление в конце (**Ease** = 100 , **Out**)

- Параметр **Blend** позволяет задать тип перехода между начальной и конечной формами. Значение **Angular** приводит к тому, что при переходе у промежуточных форм сохраняются углы и прямолинейные сегменты. Значение **Distributive**, наоборот, сглаживает промежуточные формы. Переход типа **Angular** может быть применен только к формам, содержащим углы и прямолинейные сегменты, в противном случае переход выполняется с использованием значения **Distributive**.
6. Просмотреть результат в рабочей среде (**Control>Play**) или в среде тестирования (**Control>Test Movie**).

Для того чтобы удалить автоматическую анимацию, нужно выделить любой кадр диапазона, за исключением конечного ключевого и либо выбрать значение **None** в списке **Tween** Инспектора свойств, либо, щелкнув правой кнопкой мыши по кадру из контекстного меню, выбрать команду **Remove Tween**.

Следует иметь в виду, что в одном ключевом кадре одновременно могут находиться несколько объектов рабочего уровня, при этом каждый из них будет подвергнут анимации формы. Однако в этом случае не всегда можно получить удовлетворительный переход, поскольку одновременная трансформация нескольких сложных форм может привести к непредсказуемому результату. Для того чтобы трансформация формы отрабатывалась корректно, рекомендуется размещать различные, но одновременные анимационные процессы на *разных* слоях.

Примечание

Один и тот же ключевой кадр монтажной линейки может одновременно быть и конечным кадром предыдущего диапазона анимации формы, и начальным кадром следующего. Таким образом, можно создавать несколько последовательных трансформаций, не дублируя при этом ключевые кадры.

Содержимое кадров трансформации, используемых для реализации автоматической трансформации формы, не может быть отредактировано. Однако любой кадр трансформации можно сделать ключевым при помощи команд контекстного меню кадра **Convert to Keyframes** или **Convert to Blank Keyframes**. Таким образом, один диапазон трансформации может быть разбит на несколько, начальные и конечные кадры которых могут редактироваться независимо друг от друга.

Пример.

Эффект пробегания светового блика по тексту

Используя автоматическую анимацию формы, можно создавать цветовые переходы, в частности, с использованием градиентной заливки. В качестве примера такого эффекта рассмотрим анимацию радиального градиента, имитирующую движение светового блика по тексту (рис. 13.3).



Рис. 13.3. Движение градиента по тексту при помощи анимации формы

Для реализации данного эффекта можно выполнить следующие действия:

1. В первом ключевом кадре нового документа создать текстовый блок типа **Static** и ввести в него произвольный текст. Размер символов должен быть достаточно большим, чтобы эффект хорошо читался.
2. Поскольку анимация формы применяется только к объектам рабочего уровня, текстовый блок необходимо разбить, выделив его и дважды выполнив команду главного меню **Modify>Break Apart** (первое выполнение данной команды приведет к тому, что текстовый блок будет разделен на несколько текстовых блоков, повторное выполнение превратит текст в векторные формы).
3. Открыть панель **Color Mixer (Window>Design Panels>Color Mixer)** и в качестве цвета заливки выбрать в списке **Fill style** тип **Radial**. Синтезировать радиальный градиент от светлого к темному, имитирующий световой блик. Добавить этот градиент в текущий каталог цветов (команда **Add Swatch** контекстного меню палитры **Color Mixer**).
4. Выделить все векторные формы, активизировать инструмент **Paint Bucket**, установить в качестве цвета заливки синтезированный на шаге 3 градиент и, включив режим фиксации заливки **Lock Fill**, щелкнуть на выделенной области. В результате этих действий все формы букв будут залиты одним и тем же градиентным заполнением.
5. Активизировать инструмент **Fill Transform** и отредактировать градиент таким образом, чтобы его граница находилась слева от слова, как показано на рис. 13.4, а. Если маркеры редактирования не видны, необходимо уменьшить масштаб просмотра (см. гл. 6).
6. Создать конечный ключевой кадр (**Insert Keyframe**), исходя из времени, отведенного на выполнение анимации. Содержимое начального ключевого кадра должно быть автоматически скопировано в конечный ключевой кадр.
7. В конечном ключевом кадре при помощи инструмента **Fill Transform** сместить градиент вправо за пределы слова, как показано на рис. 13.4, б.
8. Выделить любой кадр диапазона, за исключением последнего ключевого, а затем при помощи Инспектора свойств выбрать в списке **Tween** тип анимации **Shape** и задать необходимые параметры анимации формы.



Рис. 13.4. Положение градиента в начальном (а) и конечном (б) ключевых кадрах

9. Просмотреть результат можно в рабочей среде (**Control>Play**) или в среде тестирования (**Control>Test Movie**).
10. Для того чтобы блик, дойдя до правой границы, возвращался обратно, нужно скопировать первый кадр (начальное состояние) и вставить его в новый кадр монтажной линейки, создав новый диапазон анимации. При этом кадр, созданный на шаге 6, будет одновременно и конечным кадром предыдущего диапазона анимации формы, и начальным кадром следующего.

Применение меток подсказки

Использование анимации формы для преобразования сложных форм зачастую приводит к непредсказуемому результату. Для того чтобы сделать анимацию более контролируемой, Flash предоставляет возможность использования *меток подсказки* (Shape hints).

Метки подсказки расставляются попарно в начальном и в конечном кадрах диапазона анимации формы. Метки промаркованы буквами латинского алфавита от a до z, в связи с чем их число не может превысить 26. Метка, установленной в начальном ключевом кадре, соответствует одноименная метка в конечном ключевом кадре. Механизм работы меток подсказки заключается в том, что точка объекта, помеченная меткой в начальном ключевом кадре, в ходе анимации перемещается в точку объекта, размещенного в конечном ключевом кадре, помеченную одноименной меткой. Таким образом, использование меток подсказки позволяет управлять ходом выполнения анимации формы.

При расстановке меток подсказки следует придерживаться следующих правил и рекомендаций:

- метки должны находиться на границе формы или на контуре;
- метки следует расставлять против часовой стрелки, начиная с левого верхнего угла объекта;
- метки следует расставлять попарно в начальном и конечном кадрах, т. е. сначала устанавливаются метки а—а и только потом создаются метки б—б.

Для того чтобы создать метки подсказки, нужно выделить начальный ключевой кадр диапазона анимации формы и выполнить команду главного меню **Modify>Shape>Add Shape Hints** или воспользоваться ее клавиатурным эквивалентом (**<Ctrl>+<Shift>+<H>**). В результате выполнения этой команды метки, помеченные буквой "а", добавятся в начальный и конечный кадры.

Для отключения (или включения) отображения меток подсказки нужно выполнить команду **View>>Show Shape Hints** (**<Ctrl>+<Alt>+<H>**). Отображение меток подсказки автоматически отключается при активизации любого инструмента, кроме инструмента **Selection**.

При расстановке меток подсказки они притягиваются к границам форм и к контурам. После того как одноименные метки установлены в обеих кадрах, они должны изменить цвет. Метка начального кадра окрашивается желтым, а метка конечного — зеленым цветом. Если цвет метки не изменился, необходимо скорректировать ее положение, в противном случае метка не будет обрабатываться. Рис. 13.5 иллюстрирует принцип применения меток подсказки.

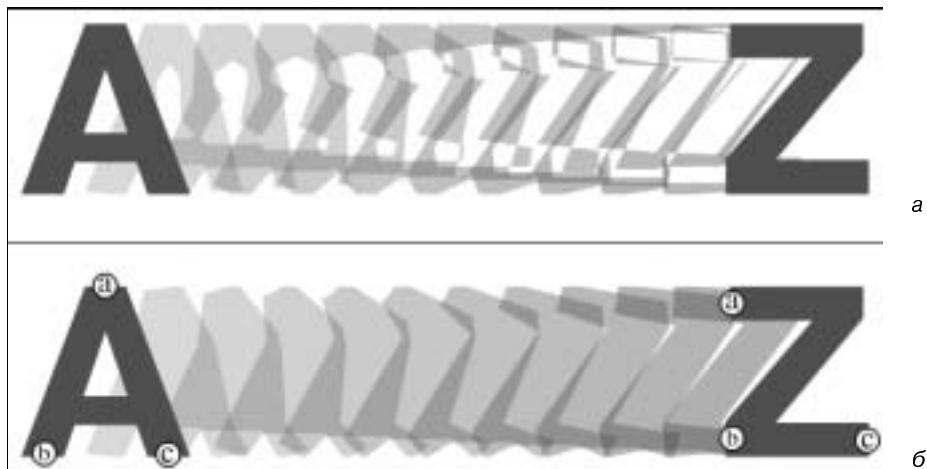


Рис. 13.5. Применение меток подсказки:
а — анимация без меток подсказки;
б — анимация с применением меток подсказки

При трансформации сложных, значительно отличающихся форм или контуров необходимо использовать несколько последовательных диапазонов анимации формы, на каждом из которых могут быть применены метки подсказки. При этом начальная форма трансформируется в промежуточную и лишь затем в конечную. Чем сложнее формы, тем больше промежуточных преобразований и, соответственно, диапазонов анимации может потребоваться для получения качественной трансформации.

Анимация движения

Анимация движения (Motion tweening) может быть применена исключительно к объектам *наложенного* уровня. С ее помощью можно выполнить следующие виды трансформаций:

- перемещение объектов по линейной и произвольной траектории;
- изменение масштаба, скос (наклон), вращение;
- изменение яркости, цвета и прозрачности экземпляров символов при помощи цветовых эффектов (**Color Effect**).

Технология создания автоматической анимации движения аналогична описанной выше процедуре создания анимации формы, однако данный тип анимации имеет ряд характерных только для него параметров настройки. Для создания анимации движения необходимо выполнить следующие действия:

1. В начальный ключевой кадр нового слоя поместить объект *наложенного* уровня (группу, растровое изображение, видеоролик, текстовый блок, экземпляр символа).
2. В конечный кадр диапазона вставить ключевой кадр (**Insert Keyframe**), поместив в него тот же объект, который содержится в начальном кадре.
3. В последнем кадре диапазона задать конечное состояние объекта, изменив его свойства как объекта наложенного уровня. Для этого можно изменить его координаты, отмасштабировать, повернуть или скосить и, если речь идет об экземпляре символа, применить любой цветовой эффект.
4. Задать параметры автоматической анимации. Для этого необходимо выделить любой кадр диапазона, кроме последнего ключевого, и в Инспекторе свойств установить следующие настройки (рис. 13.6).
 - В раскрывающемся списке **Tween** выбрать тип анимации **Motion**. При этом начальный и конечный ключевые кадры будут соединены прямой неразрывной линией со стрелкой. Если режим **Tinted Frames** контекстного меню монтажной линейки активен, то диапазон кадров автоматической анимации формы будет выделен голубоватым цветом (см. табл. 13.1 в конце раздела). В случае если в анимации допущена ошибка, на монтажной линейке появляется пунктирная линия, а при

выделении соответствующего кадра в Инспекторе свойств появляется кнопка с пиктограммой восклицательного знака , свидетельствующая об ошибке в анимации.

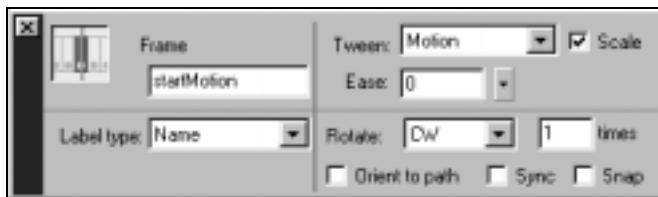


Рис. 13.6. Настройка параметров анимации движения (Motion tweening)

- Флажок **Scale** — устанавливается при необходимости плавного изменения масштаба объекта на протяжении диапазона анимации. Если размеры объекта в начальном и конечном кадрах отличаются, то автоматическое масштабирование будет выполняться только при установке данного флажка.
- Поле **Ease** используется для задания относительного замедления анимации в начале или в конце диапазона (*см. разд. "Анимация формы" данной главы*).
- Список **Rotate** позволяет задать параметры автоматического вращения объекта на протяжении диапазона кадров анимации. Вращение выполняется относительно точки трансформации объекта; смещая точку трансформации, можно изменить характер вращения объекта (рис. 13.7). Для корректного выполнения анимации точки трансформации у объектов в начальном и конечном кадрах должны находиться в одном и том же положении. Ниже перечислены параметры вращения объекта:
 - ◊ **None** — вращение отключено;
 - ◊ **Auto** — автоматическое вращение от угла поворота, заданного в начальном кадре, до угла, заданного в конечном кадре. Вращение выполняется по кратчайшему пути. Так, если в начальном кадре угол поворота составляет 0 градусов, а в конечном –45 градусов, то поворот будет выполняться на 45 градусов против часовой стрелки;
 - ◊ **CW** — автоматическое вращение по часовой стрелке (*clockwise*). Число полных оборотов, выполняемых за один диапазон анимации, задается в поле **Times**;
 - ◊ **CCW** — автоматическое вращение против часовой стрелки (*counter-clockwise*).

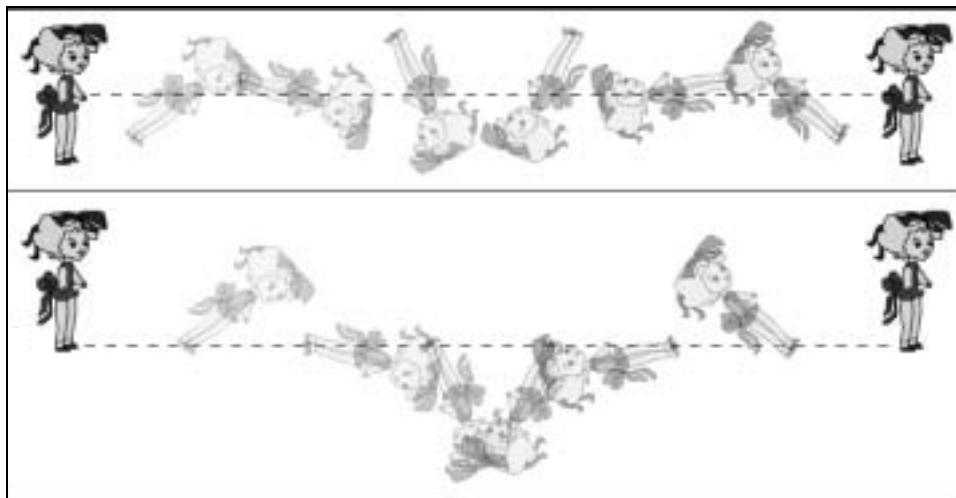


Рис. 13.7. Смещение точки трансформации.
Положение точки трансформации указано пунктирной линией

- Флажок **Sync** — применяется в случае, если начальный и конечный ключевые кадры диапазона анимации содержат разные экземпляры одного и того же анимированного символа типа **Graphic**. Установка данного флажка позволяет синхронизировать первые кадры монтажных линеек данных экземпляров, учитывая выделенный им диапазон кадров трансформации родительской монтажной линейки. Данный режим можно также установить, выделив весь диапазон кадров анимации движения и выполнив команду **Modify>Timeline>Synchronize Symbols** или соответствующую команду контекстного меню кадра.
 - Параметры **Orient to path** и **Snap** используются при создании анимации по заданному маршруту. О них будет сказано в следующем разделе данной главы.
5. Просмотреть результат в рабочей среде (**Control>Play**) или в среде тестирования (**Control>Test Movie**).

Анимацию движения можно также создать следующим образом:

1. В начальный ключевой кадр нового слоя поместить любой объект рабочего или наложенного уровня.
2. Выделить начальный кадр и выполнить команду главного меню **Insert>Timeline>Create Motion Tween** или соответствующую команду контекстного меню кадра. Если в кадре находился объект рабочего уровня, то в результате выполнения данной команды он преобразуется в графи-

ческий символ, которому в библиотеке будет присвоено стандартное имя Tween n, где n — номер символа, созданного подобным образом.

3. Создать конечный ключевой кадр диапазона при помощи команды **Insert Keyframe** и выполнить в нем все необходимые преобразования.

Для того чтобы удалить автоматическую анимацию, нужно выделить любой кадр диапазона, за исключением конечного ключевого и либо выбрать значение **None** в списке **Tween** Инспектора свойств, либо, щелкнув правой кнопкой мыши на кадре из контекстного меню, выбрать команду **Remove Tween**.

Необходимо иметь в виду, что в пределах одного диапазона анимации движения может находиться *только один* объект. Добавление в начальный или конечный ключевые кадры других объектов наложенного или рабочего уровня приведет к некорректной анимации и выводу сообщения об ошибке. В связи с этим, при необходимости одновременного выполнения нескольких трансформаций с использованием анимации движения *необходимо* каждый анимируемый объект размещать на *отдельном* слое.

Содержимое кадров трансформации, используемых для реализации анимации движения, не может быть отредактировано. Однако любой кадр трансформации можно сделать ключевым при помощи команд контекстного меню кадра **Convert to Keyframes** или **Convert to Blank Keyframes**. Таким образом, один диапазон трансформации может быть разбит на несколько диапазонов, начальные и конечные кадры которых могут редактироваться независимо друг от друга. Любая попытка изменить содержимое кадра трансформации также приведет к созданию ключевого кадра в соответствующей позиции.

Анимация по маршруту

При перемещении объектов с использованием автоматической анимации движение осуществляется из начального положения в конечное по наикратчайшему пути, т. е. вдоль прямой линии. Анимация движения позволяет реализовать автоматическое перемещение объектов наложенного уровня по произвольной траектории (маршруту). Для выполнения этой задачи необходимо применение группы слоев, включающей слой пути (**Motion Guide**), содержащий произвольную траекторию, и ведомый слой (**Guided**), содержащий автоматическую анимацию движения объекта из начального положения в конечное (см. разд. "Типы Motion Guide и Guided" гл. 9).

Для того чтобы создать анимацию движения объекта по заданной траектории, необходимо выполнить следующие действия:

1. Создать группу из двух слоев, верхний из которых имеет тип **Motion Guide** (Слой пути), а нижний — **Guided** (Ведомый) (см. гл. 9).

2. В первом ключевом кадре слоя пути при помощи любого инструмента, создающего контуры, нарисовать траекторию, отвечающую следующим требованиям (рис. 13.8):

- траектория должна быть контуром;
- траектория не должна быть замкнута (однако допустимы петли);
- траектория не может состоять из нескольких несвязанных сегментов или содержать разветвления.

Цвет, толщина и стиль линии траектории не имеют значения, поэтому рекомендуется использовать стиль **Solid** или **Hairline**.

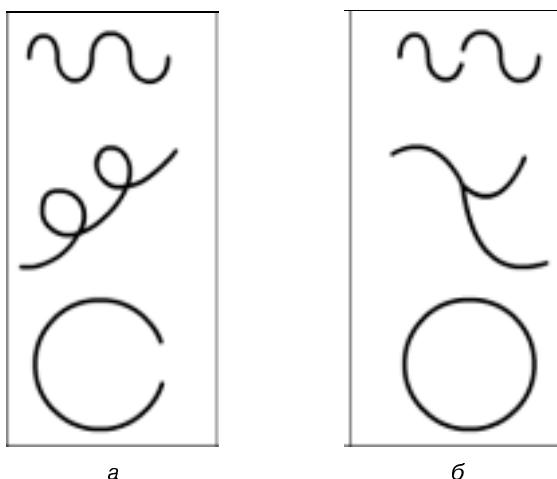


Рис. 13.8. Допустимые траектории (а);
недопустимые траектории (б)

3. На слой пути добавить столько простых кадров, сколько требуется для того, чтобы траектория присутствовала на монтажной линейке в течение всего времени выполнения анимации. Для этого на слое пути выделить ячейку монтажной линейки, соответствующую последнему кадру будущей анимации, и выполнить команду контекстного меню кадра **Insert Frame**. Затем слой пути, содержащий траекторию, можно заблокировать, чтобы случайно ее не изменить.
4. В первом ключевом кадре ведомого (**Guided**) слоя создать (или поместить) любой объект наложенного уровня, задав его начальное положение относительно нарисованной траектории, а также свойства (масштаб, угол поворота, цветовой эффект для экземпляра символа).

5. Создать конечный ключевой кадр анимации при помощи команды **Insert Keyframe** контекстного меню кадра; в нем задать конечное положение анимируемого объекта относительно нарисованной траектории и изменить значения его свойств.
6. Выделить первый ключевой кадр ведомого слоя и в списке **Tween** Инспектора свойств выбрать тип анимации **Motion**, создав анимацию линейного перемещения объекта.
7. Для того чтобы объект двигался по заданному пути, его необходимо притянуть (snap) к соответствующим участкам траектории в начальном и конечном ключевых кадрах ведомого слоя. Это можно сделать одним из следующих способов.
 - Выделить любой кадр диапазона анимации, кроме конечного, и в Инспекторе свойств установить флажок **Snap**. При этом в первом ключевом кадре точка трансформации объекта притягивается к траектории. После этого необходимо вручную притянуть объект к траектории. Для выполнения притягивания удобно воспользоваться режимом **View>Snapping>Snap to Objects**.
 - В режиме **View>Snapping>Snap to Objects** вручную притянуть объект к соответствующим участкам траектории так, чтобы его точка трансформации лежала на траектории.
 - Выделить любой кадр диапазона анимации, кроме конечного, и в Инспекторе свойств установить флажок **Snap**. Выделить конечный кадр диапазона анимации и в списке **Tween** Инспектора свойств выбрать тип анимации **Motion** и установить флажок **Snap**. Объект автоматически притягивается к траектории в последнем кадре. После этого можно, выделив конечный ключевой кадр, удалить из него автоматическую анимацию (поскольку она некорректна), выбрав в списке **Tween** Инспектора свойств значение **None**. Объект останется притянутым к траектории.

Примечание

При использовании режима **Snap** корректировать положение объекта на траектории можно при помощи курсоров клавиатуры. В результате объект будет перемещаться вдоль траектории, не открепляясь от нее.

8. Для того чтобы объект при перемещении автоматически ориентировался по направлению траектории, необходимо выделить любой кадр диапазона анимации ведомого слоя, за исключением конечного, и в Инспекторе свойств установить флажок **Orient to path**. Для получения естественного движения необходимо вручную сориентировать объект в начальном и конечном кадрах таким образом, чтобы его базовая линия (линия, проходящая через точку трансформации объекта и совпадающая с направлени-

ем его перемещения) являлась касательной к траектории в точке притягивания. Все промежуточные значения угла поворота базовой линии будут рассчитаны автоматически (рис. 13.9).

9. Просмотреть результат в рабочей среде (**Control>Play**) или в среде тестирования (**Control>Test Movie**).

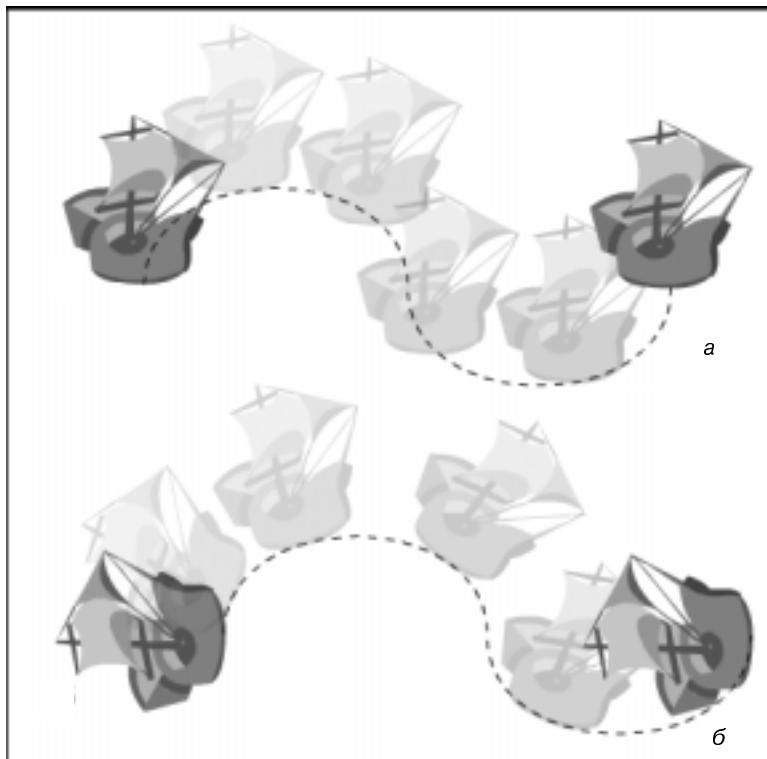


Рис. 13.9. Применение параметра **Orient to path**:
а — ориентирование по траектории отключено;
б — ориентирование по траектории включено

Достаточно часто при работе над фильмом возникает задача создания анимации движения по замкнутой траектории. Однако если в слой пути поместить замкнутую траекторию, анимация выполниться не будет. Для того чтобы обойти это ограничение, можно поступить следующим образом.

1. В слое пути создать замкнутую траекторию (например, окружность).
2. При помощи инструментов **Eraser** или **Selection** создать небольшой зазор, чтобы контур перестал быть замкнутым.

3. В начальном ключевом кадре ведомого слоя притянуть анимируемый объект к траектории с одной стороны разрыва, а в конечном — с другой. Для того чтобы движение было равномерным, необходимо задать начальное и конечное положения объекта таким образом, чтобы расстояние между ними соответствовало расстояниям между остальными фазами (проверить это можно, воспользовавшись режимом шлейфа — **Onion Skin**) (рис. 13.10).

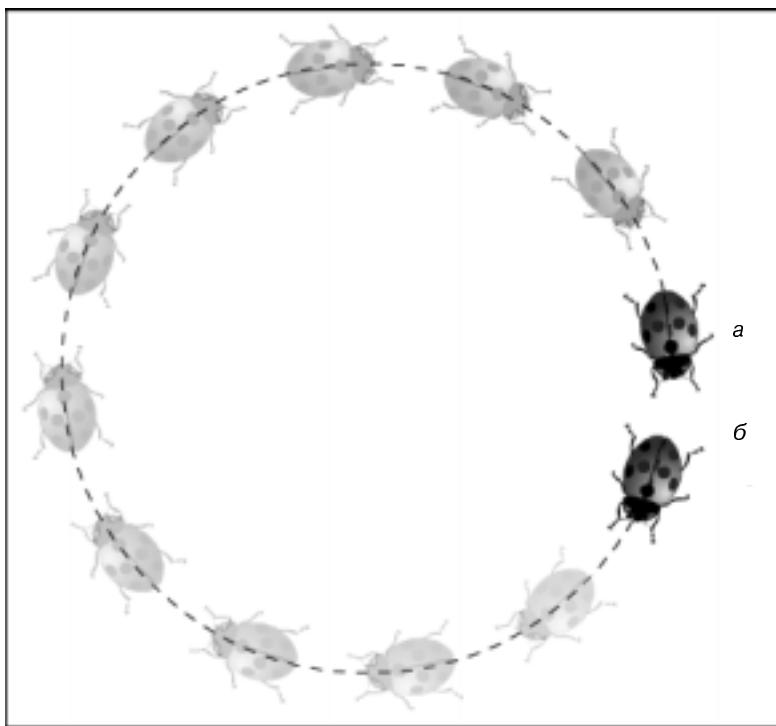


Рис. 13.10. Имитация движения по замкнутой траектории:
а — начальный кадр анимации;
б — конечный кадр анимации;

между точками а и б траектория терпит разрыв

◀ Примечание ▶

Для того чтобы "запустить" несколько объектов по одной и той же траектории, необходимо к соответствующему пути подгруппировать требуемое количество ведомых слоев, содержащих анимацию движения объектов. По одной траектории может одновременно в обоих направлениях с разными параметрами перемещаться неограниченное число объектов.

Таблица 13.1. Сравнительные характеристики типов автоматической анимации

Сравнительные характеристики	Тип автоматической анимации	
	Анимация формы Shape Tweening	Анимация движения Motion Tweening
Способ создания	Инспектор свойств Properties , список Tween , тип Shape	1. Инспектор свойств Properties , список Tween тип Motion . 2. Команда Insert>Timeline>CreateMotionTween . 3. Команда контекстного меню кадра Create Motion Tween
Вид на монтажной линейке	 зеленый фон	 голубой фон
Уровень иерархии объектов анимации	Рабочий уровень	Наложенный уровень
Типы объектов анимации	Векторные формы, контуры, разбитые текст и растровая графика	Группы, растровая графика, видеоролики, текстовые блоки, экземпляры символов
Количество одновременно анимируемых объектов в одном слое	Не ограничено	Только один объект
Свойства объектов, изменяемые при анимации	Форма, цвет, прозрачность, размер, положение в кадре, поворот, скос	Положение в кадре, размер, поворот, скос, цветовые эффекты экземпляров символов Color Effects (яркость, цвет и прозрачность)
Возможности	Автоматическое ускорение/замедление анимации	Автоматическое ускорение/замедление анимации, автоматическое вращение объекта
Особенности	Использование меток подсказки Shape Hints	Анимация по маршруту

Рис. 13.11 иллюстрирует алгоритм создания автоматической анимации.

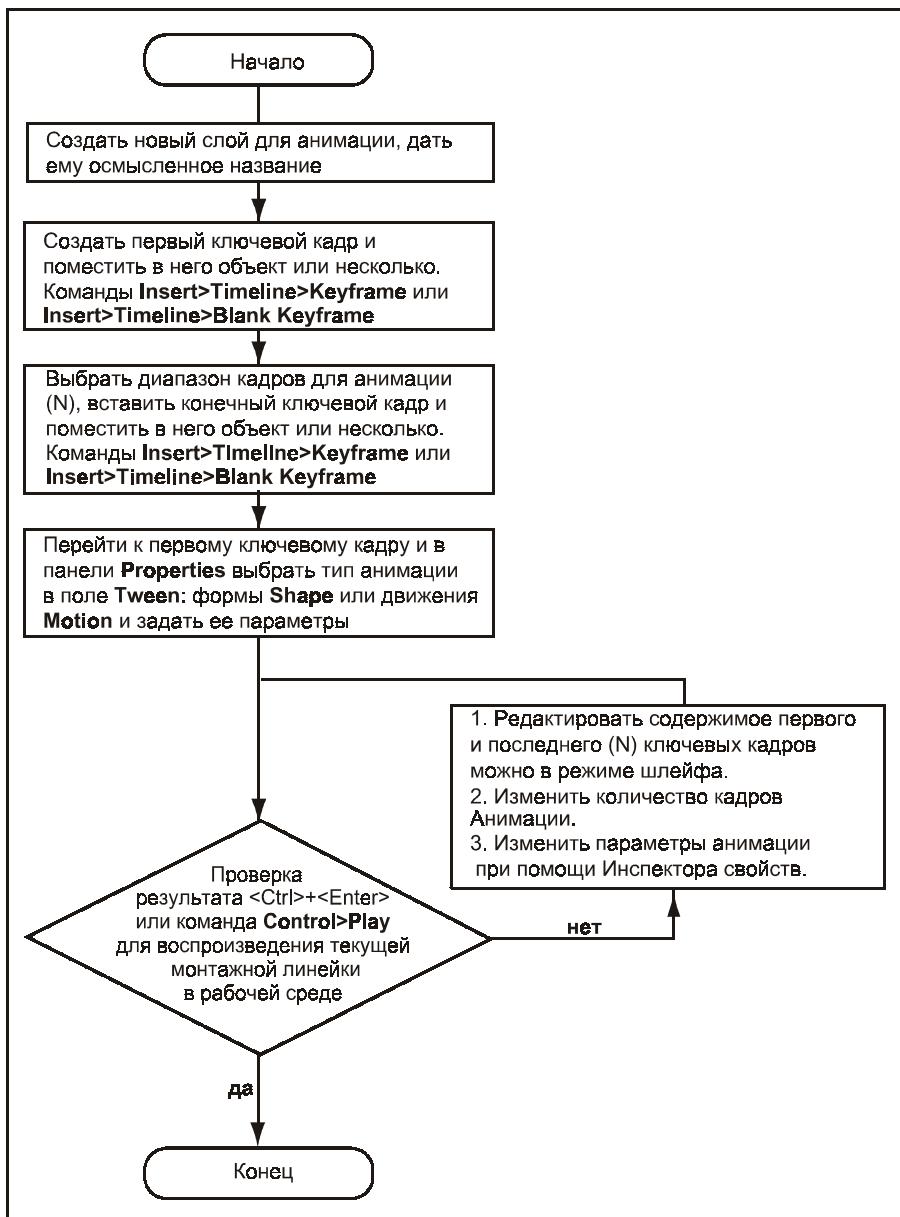


Рис. 13.11. Алгоритм создания автоматической анимации

Разработка анимационных эффектов на основе автоматической анимации

Циклическое движение

В реальной жизни многие предметы движутся циклически, т. е. совершая множество однотипных повторяющихся движений. Движение стрелок часов, вращение колеса автомобиля, колебание маятника, перемещение спутника по орбите — это все примеры циклических движений. Автоматическая анимация предоставляет великолепные возможности реализации циклических движений.

Пример. Бесконечное вращение

Простейший пример циклического движения — бесконечное вращение объекта, например, колеса автомобиля. Данную задачу удобнее всего реализовать при помощи автоматической анимации движения. Для этого нужно выполнить следующие действия:

1. Создать графический символ, содержащий изображение колеса. Для того чтобы движение колеса хорошо читалось, необходимо сделать его асимметричным, снабдив какой-нибудь характерной деталью, например, световыми бликами на ободе и у края (рис. 13.12). Движение колеса будет прослеживаться по перемещению данной детали, и, кроме того, это позволит избежать нежелательного стробоскопического эффекта. Символу можно присвоить имя *wheel*.



Рис. 13.12. Блик придаст колесу асимметричность и позволит лучше прочитать движение

2. Поместить экземпляр *wheel* в начальный ключевой кадр монтажной линейки, после чего создать конечный ключевой кадр анимации, автоматически скопировав в него содержимое первого кадра (**Insert Keyframe**). Таким образом, начальный и конечный кадры должны содержать абсолютно одинаковое изображение колеса в одном и том же положении.

3. Задать параметры автоматической анимации движения. В списке **Tween** выбрать тип анимации **Motion**. В списке **Rotate** задать направление вращения **CW** (по часовой стрелке) или **CCW** (против часовой стрелки) и количество полных оборотов (поле **times**). Следует иметь в виду, что при фиксированной скорости воспроизведения монтажной линейки и длине диапазона анимации увеличение числа оборотов приводит к увеличению относительного угла поворота колеса на соседних фазах. Это может привести к прерывистому движению, а при достаточно большом числе оборотов — к стробоскопическому эффекту, в результате которого движение будет выполняться в обратном направлении.

Если внимательно просмотреть получившуюся анимацию, зациклив воспроизведение в рабочей среде (**Control>Loop Playback+Control>Play**) или в среде тестирования (**Control>Test Movie**), то можно обнаружить небольшой скачок в конце каждого цикла поворота. Этот скачок обусловлен тем, что в начальном и конечном кадре анимации положение колеса абсолютно идентично, т. е. данная фаза фактически экспонируется вдвое дольше, чем все остальные. Для того чтобы ликвидировать этот эффект, необходимо изменить положение объекта в последнем кадре, так чтобы ни одна фаза не повторялась дважды. Наиболее простой способ выполнения данной процедуры состоит в следующем. Необходимо выделить кадр трансформации, предшествующий конечному ключевому кадру, и сделать его ключевым при помощи команды контекстного меню кадра **Convert to Keyframes** (<F6>). Новый ключевой кадр будет содержать фазу вращения, предшествующую исходному положению колеса в следующем цикле вращения (рис. 13.13). Данный ключевой кадр должен быть сделан последним кадром диапазона анимации, поэтому нужно выделить и удалить конечный ключевой кадр, созданный на шаге 2 (**Remove Frames**). Для того чтобы время выполнения цикла вращения осталось прежним, надо переместить конечный ключевой кадр на одну позицию вправо, увеличив таким образом диапазон кадров трансформации на один кадр. В результате получится идеальное циклическое вращение.



Рис. 13.13. Для получения идеального циклического вращения последний кадр трансформации преобразуется в ключевой

Пример.

Зацикливание фонового изображения

На практике нередко возникает задача создания иллюзии бесконечного движения, при котором изображение движется по принципу бесконечной

ленты, периодически повторяясь. Данный прием широко используется для создания движущихся фоновых изображений, панорам и для реализации анимационных эффектов, основанных на этом принципе.

В качестве примера приведем использование автоматической анимации движения для реализации бесконечного движения фона фильма. Движение будет выполняться слева направо.

1. Создать графический символ, содержащий изображение, которое будет использовано в качестве движущегося фона. Чем длиннее и разнообразнее будет фон, тем менее монотонным окажется результирующий эффект. Изображение должно быть нарисовано таким образом, чтобы его левая и правая части хорошо стыковались друг с другом. В принципе, ширина фона может быть любой. Символу можно присвоить имя `back_img`.
2. Поместить экземпляр `back_img` в первый ключевой кадр монтажной линейки и выровнять таким образом, чтобы правая грань изображения совпала с правой гранью рабочей области (панель **Align**). После этого создать вертикальную направляющую и совместить ее с левой гранью экземпляра `back_img`.
3. Скопировать данный экземпляр и разместить копию на сцене так, чтобы ее правая грань вплотную стыковалась с левой гранью оригинала (для этого можно воспользоваться созданной направляющей или режимом **Snap Align**). Между экземплярами не должно быть зазоров. После этого при помощи панели **Align** необходимо выровнять экземпляры, так чтобы их нижние грани совпадали (рис. 13.14).
4. Поскольку в пределах одного слоя при помощи анимации движения одновременно может быть анимирован только один объект наложенного уровня, необходимо сгруппировать оба экземпляра **Modify>Group** (или превратить их в новый символ **Insert Convert to Symbol**).

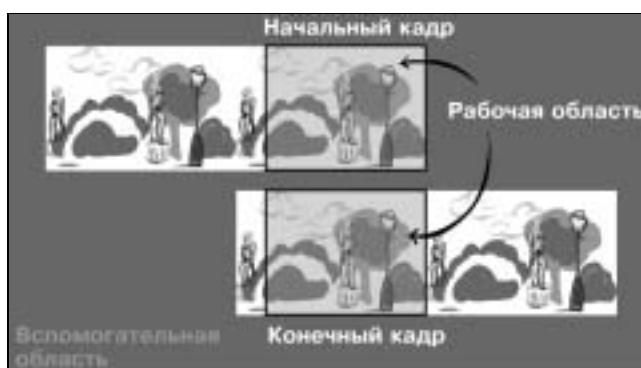


Рис. 13.14. Зацикливание фонового изображения

5. Создать конечный ключевой кадр, автоматически скопировав в него содержимое первого кадра (**Insert Keyframe**). В конечном ключевом кадре необходимо сместить группу (или экземпляр), содержащую две копии back_img вправо, таким образом, чтобы в пределах рабочей области находилась та же часть изображения, что и в начальном ключевом кадре (см. рис. 13.14). Это можно выполнить, совместив левую грань группы с направляющей, размещенной на шаге 2.
6. Выделить любой кадр диапазона, за исключением конечного, и в списке **Tween** Инспектора свойств выбрать тип анимации **Motion**. В пределах рабочей области будет выполняться бесконечное движение.
7. Для того чтобы скачок, происходящий на вспомогательной области, был не виден, можно скрыть ее, применив слой-маску. Для этого необходимо создать новый слой и поместить его над слоем, содержащим анимацию. Новый слой можно назвать **Mask**.
8. В первом ключевом кадре слоя **Mask** создать прямоугольник без контура, окрашенный сплошным цветом. При помощи панели **Align** привести его размеры к размерам рабочей области и отцентрировать по ней. Убедиться в том, что прямоугольник присутствует на слое **Mask** в течение всей анимации фона.
9. Щелкнуть на слое **Mask** правой кнопкой мыши и из контекстного меню выбрать команду **Mask**. Слои автоматически заблокируются, маска будет включена.
10. В данном случае снова имеет место эффект скачка, связанный с тем, что зритель в начальном и конечном кадрах видит одинаковое изображение. *О ликвидации этого эффекта см. выше в примере "Бесконечное вращение".*

Использование вложенных символов

Достаточно часто возникает необходимость создания сложного движения, включающего в себя несколько различных и одновременных анимационных процессов. Например, при анимации движения жука по произвольной траектории необходимо реализовать циклическое движение лап, одновременно перемещая его по сцене. В подобных ситуациях можно прибегнуть к использованию вложенных друг в друга символов.

В рабочей среде Flash существует понятие *множественности монтажных линеек*. Экземпляры статичных символов могут быть использованы для создания анимации на монтажных линейках других символов. В свою очередь, анимированные символы могут также входить в состав других символов для создания контейнеров, содержащих анимацию (а также звук и интерактивность) и представляющих собой самостоятельные мини-фильмы. В принципе все элементы конечного фильма могут быть помещены на монтажную линейку символа типа муви-клип (поскольку он может обладать широкой

функциональностью), экземпляр которого помещается в один-единственный кадр основной монтажной линейки. Такой подход достаточно часто используется на практике, поскольку позволяет легко выполнять преобразования (перемещение, масштабирование, трансформацию, цветовые эффекты экземпляров) сразу над всем содержимым ролика.

При реализации замысла в первую очередь необходимо проанализировать поставленную задачу и мысленно разбить общее движение на элементарные составляющие. Важно сразу определить статичные и динамические (подвижные) элементы. Те элементы, которые в неизменном виде многократно используются по ходу действия, должны быть объявлены символами.

Примечание

Следует иметь в виду, что анимация, содержащаяся на монтажных линейках символов типа **Movie Clip**, не работает в рабочей среде документа и может быть увидена только в среде тестирования фильма (**Control>Test Movie**).

Пример.

Сложное движение

с использованием вложенных символов

Рассмотрим приведенный в начале данного раздела пример с перемещением жука по произвольной траектории. Здесь можно выделить два составляющих общее движение анимационных процесса:

- движение частей тела жука;
- поступательное движение жука по траектории.

Исходя из этой схемы, нужно сначала создать символ типа **Movie Clip**, содержащий анимацию частей тела жука на *своей монтажной линейке*, а затем на *основной монтажной линейке* фильма запустить его экземпляр по произвольной траектории. Выполнение данной задачи предполагает следующую последовательность действий:

1. Поскольку все лапы жука будут выглядеть одинаково, необходимо создать графический символ, содержащий изображение лапы (**Insert>New Symbol**). Его можно назвать *leg* (рис. 13.15, *а*).
2. Поскольку перемещение всех лап жука будет выполняться одинаковым образом, необходимо создать новый графический символ с именем *leg_anim* и на его монтажной линейке создать анимацию движения одной лапы, используя экземпляр символа *leg* (рис. 13.15, *б*). Анимация включает в себя отведение и приведение лапы и требует создания двух последовательных диапазонов анимации движения.
3. Создать графический символ, содержащий изображение туловища жука, назвав его *body* (рис. 13.15, *в*).

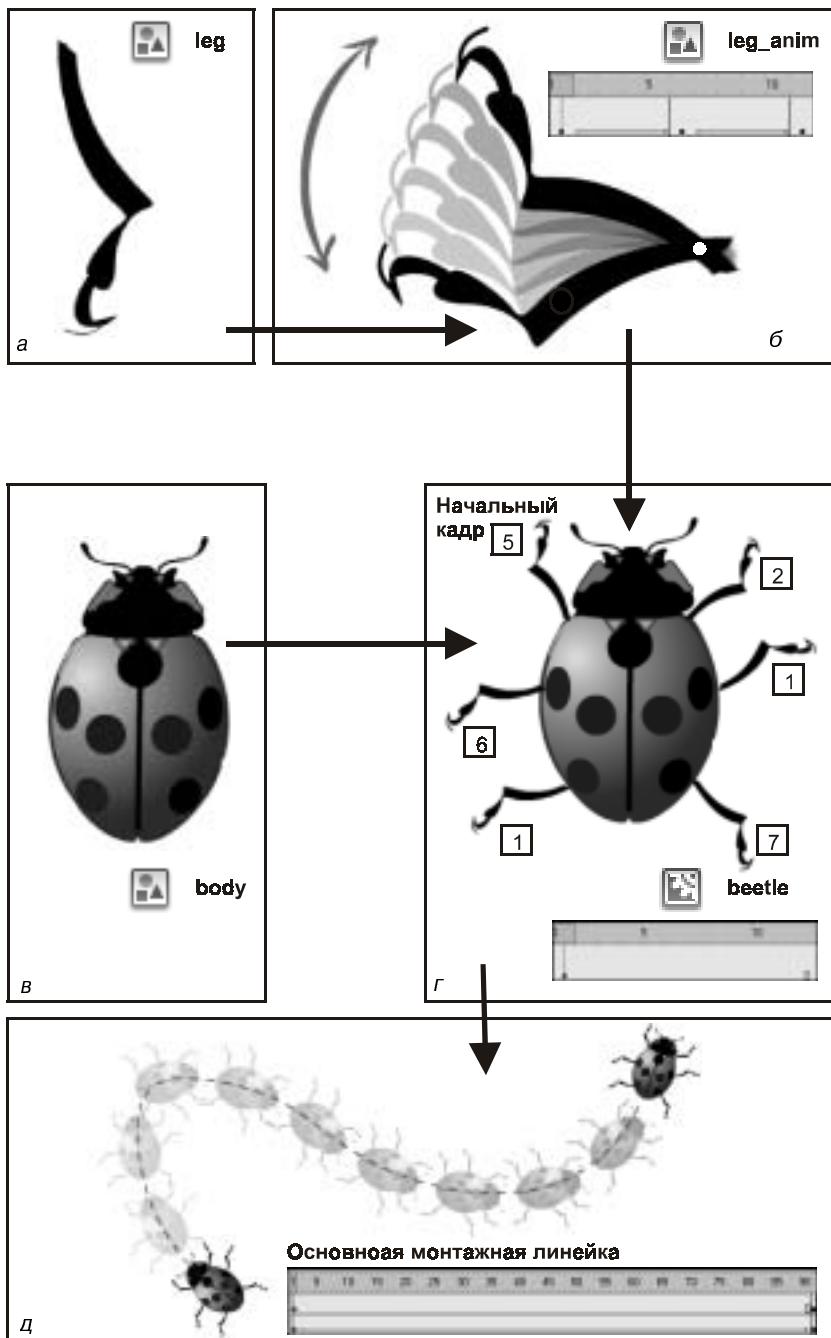


Рис. 13.15. Сложное движение с использованием вложенных символов

4. Далее необходимо создать новый символ типа **Movie Clip** с именем beetle, в котором нужно сформировать изображение жука, "собрав" его из экземпляров анимированного символа leg_anim и экземпляра статичного символа body (рис. 13.15, г). Различные элементы жука можно разместить на разных слоях монтажной линейки муви-клипа для лучшей организации и удобства работы.
5. Поскольку leg_anim — это символ типа **Graphic**, его монтажная линейка зависит от родительской, т. е. от линейки символа beetle. В связи с этим на монтажную линейку символа beetle необходимо добавить столько простых кадров, сколько длится анимация символа leg_anim. Следует убедиться, что все элементы жука присутствуют на монтажной линейке на протяжении всего диапазона содержащихся в ней кадров. Для того чтобы анимация лап выполнялась асинхронно, можно для каждого экземпляра leg_anim задать свой начальный кадр (см. разд. "Свойства экземпляра типа Graphic" гл. 10).
6. В результате символ body представляет собой контейнер, содержащий анимированное изображение жука. Монтажная линейка муви-клипа не зависит от родительской монтажной линейки. В связи с этим анимация экземпляра body будет выполняться в течение всего того времени, пока он присутствует на сцене. Таким образом, в итоге необходимо просто поместить экземпляр символа body на основную монтажную линейку и создать анимацию движения по произвольному маршруту (см. разд. "Анимация по маршруту" данной главы) (рис. 13.15, д).

Пример.

Эффекты наезда камеры и панорамирования

Использование вложенных символов позволяет добиваться впечатляющих эффектов, таких как наезд, отъезд, панорамирование и т. д., имитирующих манипуляции с кинематографической камерой. Так, если все содержимое фильма, включая статичную графику и анимацию, на монтажных линейках символов поместить в символ типа **Movie Clip**, то в дальнейшем, масштабируя экземпляр этого клипа на основной монтажной линейке, можно выполнить анимацию, имитирующую приближение или удаление. При этом часть объектов фильма будет исчезать за кадром при увеличении или, наоборот, уменьшаясь, отображаться в пределах рабочей области (рис. 13.16). Для того чтобы содержимое вспомогательной области нельзя было увидеть при воспроизведении в автономном проигрывателе Flash Player, можно либо воспользоваться маской, либо накрыть вспомогательную область непрозрачной прямоугольной формой с прорезью по форме рабочей области.

Пример.

Создание сложных сцен

При создании сложных сцен, содержащих несколько различных анимированных объектов, каждый из них должен быть помещен на отдельный слой.

Движение объекта может быть реализовано либо на основной монтажной линейке, либо на монтажной линейке символа, либо как комбинация того и другого.

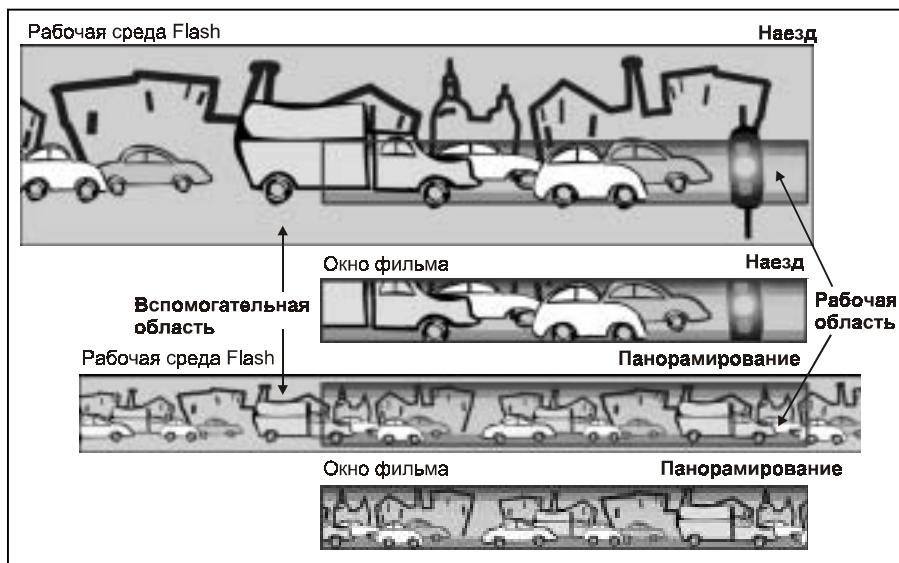


Рис. 13.16. Эффекты наезда камеры и панорамирования

В сцене прогулки человека с таксой, представленной на рис. 13.17, выполняются следующие движения:

- анимация походки человека (слой man);
- анимация походки собаки (слой doggy);
- анимация тени человека (слой man_shadow);
- анимация тени собаки (слой dog_shadow);
- движение заднего плана с фонарями, статуями, кустами и деревьями (слой background).

В данном примере покадровая анимация походки человека выполняется на монтажной линейке муви-клипа, экземпляр которого как статичный объект помещен на основную монтажную линейку фильма. Аналогичная ситуация имеет место и с собакой. Таким образом, эти персонажи фактически стоят на месте, впечатление движения возникает за счет перемещения фона.

Задний план, включающий изображения фонарей, статуй, кустов и деревьев, представляет собой набор статичных графических символов, объединенных в группу и анимированных на основной монтажной линейке, как это было описано в разд. "Зацикливание фонового изображения" данной главы.

Тени представляют собой экземпляры тех же анимированных символов, которые содержат походку человека и собаки соответственно. Это позволяет синхронизировать движение теней с движением персонажей. Для того чтобы сделать тени однородными по тону, к ним применен цветовой эффект **Tint** с максимальной интенсивностью окрашивания. В зависимости от положения фонарей тени становятся более интенсивными или, наоборот, бледнеют (это достигается за счет изменения тона окрашивания от светлого к темному и наоборот). На основной монтажной линейке тени анимируются с применением автоматической анимации движения. С ее помощью выполняется скос, масштабирование и изменение тона теней (цветовой эффект **Tint**) в зависимости от положения фонарей.



Рис. 13.17. Создание сложных сцен

Маска используется для того, чтобы скрыть вспомогательную область в конечном фильме.

Анимация растровой графики и текста

Растровые изображения, текстовые блоки и видеоролики, будучи объектами наложенного уровня, могут быть анимированы с использованием автоматической анимации движения (Motion tweening). К ним применимы все возможности анимации движения, включая автоматическое вращение и анимацию по маршруту. Для того чтобы получить возможность применения цветовых эффектов экземпляров (Color effects) к данным объектам, их необходимо конвертировать в символ (как правило, типа **Graphic** или **Movie Clip**). Так, например, поместив растровое изображение на монтажную линейку графического символа, можно анимировать экземпляр этого символа, изменяя степень его прозрачности при помощи цветового эффекта **Alpha**. В начальном ключевом кадре диапазона значение **Alpha** = 0, в конечном — значение **Alpha** = 100. Используя анимацию движения, можно получить эффект плавного проявления растрового изображения. Аналогичным образом можно использовать и другие цветовые эффекты, например при помощи эффекта **Tint** изменять колорит растрового изображения, помещенного в символ, или создавать множество разноцветных копий одного и того же текстового блока, конвертированного в символ.

При анимации текста следует иметь в виду следующие рекомендации.

- Избегайте анимации больших текстовых блоков. Чем меньше анимируемый текстовый блок, тем меньше нагрузка на процессор клиента при воспроизведении фильма.
- Анимируя текст, минимизируйте анимацию остальных объектов.
- Короткий текст, но с экзотическим шрифтом следует разбить, превратив в векторные формы.
- Ограничтесь малым количеством шрифтов в фильме.
- Из-за сглаживания шрифт маленького размера может читаться плохо, поэтому при необходимости использования мелкого текста работайте с машинонезависимыми шрифтами (без сглаживания) — `_sans`, `_serif`, `_typewriter` или отключайте сглаживание шрифта при помощи Инспектора свойств (Alias text).

Можно указать следующие возможности анимации текста.

- Сохранение целостности текстовой строки (работа с текстовым блоком) — анимация движения (Motion tweening), покадровая анимация (Frame by frame).
- Разбиение текста на отдельные формы — анимация формы (Shape tweening), покадровая анимация (Frame by frame).
- Помещение текста в символ, работа с его экземплярами — цветовые эффекты (Color effects), трансформация — анимация движения (Motion

tweening), покадровая анимация (Frame by frame), программное управление объектами.

- Программная анимация текста (текстовому блоку необходимо сопоставить переменную при помощи Инспектора свойств в поле **Instance name** (только для полей типа **Input** и **Dynamic**)).

В качестве типовых анимационных текстовых эффектов можно указать следующие:

- переход из буквы в букву — анимация формы;
- проявление надписи постепенно по букве с эффектом прозрачности и масштабирования;
- пробегание волны по тексту;
- маскирование — использование текста в качестве маски и/или в качестве содержимого маскируемого слоя;
- изменение цвета заливки текста, в том числе анимация градиента.

Анимация с применением маски

Использование маски позволяет получить множество разнообразных интересных анимационных эффектов. Все они основаны на том, что содержимое маскируемого слоя (или нескольких) видно только сквозь формы, размещенные на слое-маске. При этом действие маски распространяется исключительно на маскируемые слои, а все объекты, находящиеся на нижележащих нормальных слоях, видны, если они не перекрыты видимым содержимым маскируемого слоя. Анимация может находиться как на маскируемых слоях, так и на слое-маске. Таким образом, возможны следующие схемы анимации с применением маски.

- Анимация на маскируемом слое. Мaska неподвижна. В этом случае анимация содержимого маскируемого слоя видна только внутри области, помещенной на слой-маску.
- Анимация на слое-маске. Содержимое маскируемого слоя неподвижно. В этом случае внутри перемещающейся маски отображается соответствующий фрагмент содержимого маскируемого слоя.
- Комбинированный эффект. Анимация и на слое-маске и на маскируемом слое. В этом случае внутри перемещающейся маски отображается перемещающееся содержимое маскируемого слоя.

Пример.

Маскирование текстом

На монтажной линейке имеются два слоя. Верхний из них имеет тип **Mask** (Слой-маска), нижний — **Masked** (Маскируемый). В первом ключевом кадре слоя-маски находится текстовый блок и несколько векторных форм.

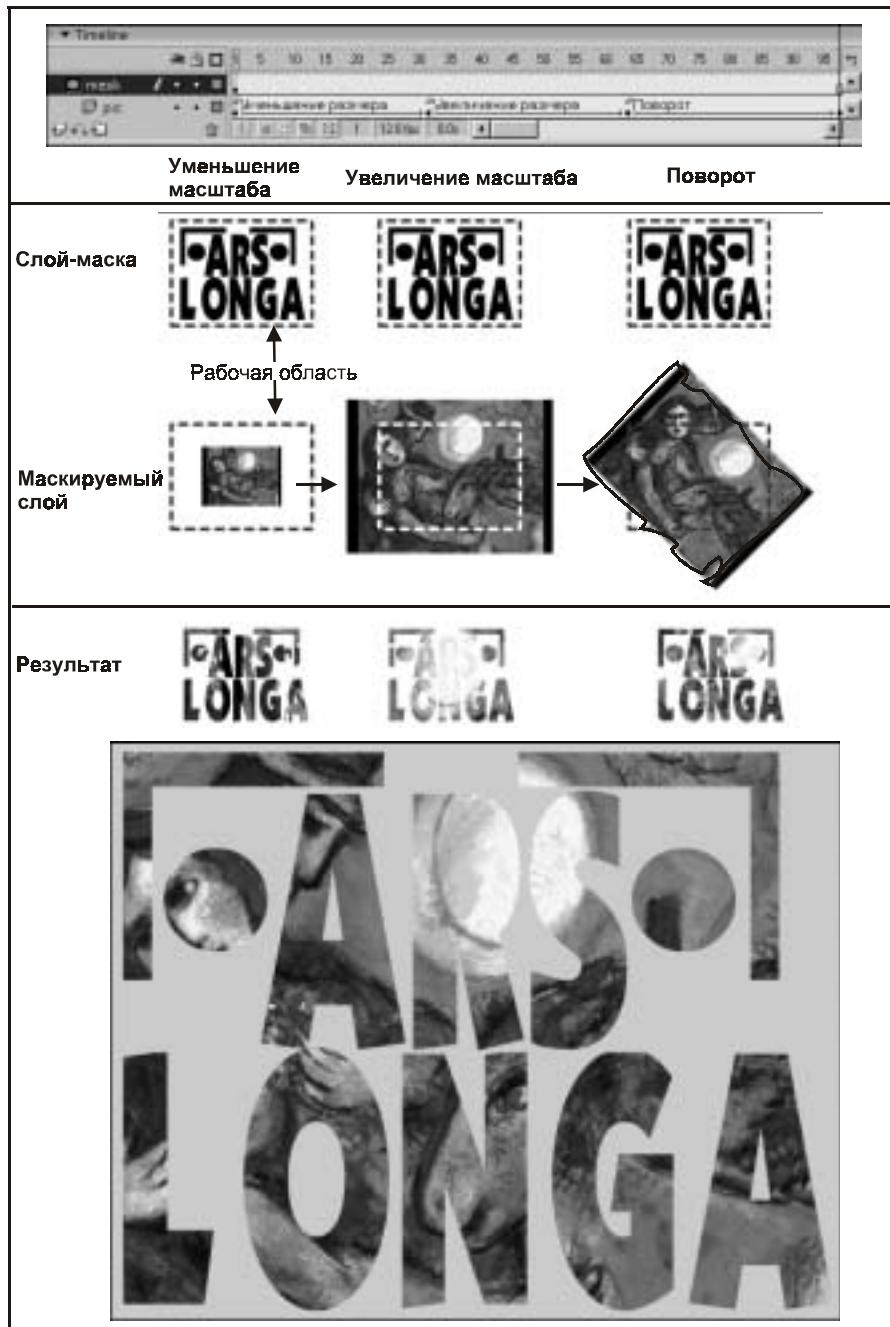


Рис. 13.18. Пример маскирования текстом.
Маска статична, на маскируемом слое — анимация растрового изображения

Поскольку в качестве маски может быть использован либо только один объект наложенного уровня, либо несколько объектов рабочего уровня, текстовый блок должен быть разбит (**Modify>Break Apart**). На маскируемом слое содержится анимация движения растрового изображения. Изображение совершает один полный оборот относительно своего центра, при этом увеличиваясь и затем уменьшаясь в размерах (рис. 13.18). Необходимо убедиться в том, что маска присутствует на сцене на протяжении всей анимации.

В результате внутри форм букв выполняется вращение "дышащего" растрового изображения. Подобный эффект может быть успешно применен при создании анимированной заставки splash-страницы или предзагрузчика (см. гл. 21).

Пример.

Создание слайд-шоу с использованием анимированной маски

Слайд-шоу представляет собой последовательную смену растровых изображений. Каждое новое изображение может открываться определенным способом. Слайд-шоу может быть использовано для демонстрации каталога товаров фирмы-производителя, создания личного портфолио или электронного фотоальбома. Однако следует иметь в виду, что наличие в проекте большого числа растровых изображений может привести к существенному увеличению его объема.

Использование анимации на слое-маске позволяет создать разнообразные способы появления изображений. В простейшем случае для создания слайд-шоу достаточно двух слоев — слоя-маски и маскируемого слоя.

На маскируемом слое последовательно с определенным интервалом размещаются разные изображения (растровые или векторные), включенные в слайд-шоу. Каждое изображение должно быть помещено в отдельный ключевой кадр. Интервал между соседними ключевыми кадрами, определяющий время появления каждого изображения, должен быть заполнен простыми кадрами. Таким образом, на маскируемом слое содержится покадровая анимация, реализующая смену изображений через определенные промежутки времени (как правило, равные).

На слое-маске содержатся последовательные диапазоны автоматической анимации (формы или движения), используемые для плавного открытия (появления) соответствующего изображения маскируемого слоя. Длительность каждого диапазона анимации слоя-маски должна соответствовать длительности присутствия соответствующего изображения на маскируемом слое. Анимация формы может быть использована, если необходимо произвольно трансформировать форму маски; анимация движения используется в

остальных случаях, при этом в качестве маски может применяться экземпляр символа. Существуют различные вариации открытия изображений. В качестве примера можно привести три следующих варианта:

- *Появление изображения слева направо.* В этом случае начальный кадр диапазона анимации слоя-маски содержит экземпляр графического символа с изображением прямоугольника (без контура), высота которого не меньше высоты маскируемого растрового изображения. Мaska помещена слева от изображения и не перекрывает его, т. е. в начальном кадре изображение не видно (рис. 13.19, а). Автоматическая анимация движения позволит сместить (и/или масштабировать) маску, постепенно открывая изображение.
- *Появление изображения из центра по кругу.* Этот эффект напоминает диафрагму фотоаппарата и часто используется в анимационных фильмах для перехода от одного эпизода (сцены) к другому. В этом случае начальный кадр диапазона анимации слоя-маски содержит экземпляр графического символа с изображением круга (без контура), расположенного точно над центром растрового изображения. Размер круга минимален. В конечном кадре круг на слое-маске должен полностью перекрывать растровое изображение, открывая его (рис. 13.19, б). В данном случае также применяется автоматическая анимация движения.
- *Произвольный переход при помощи анимации формы.* В начальный кадр диапазона анимации слоя-маски можно поместить несколько произвольных векторных форм (т. е. объектов рабочего уровня), а в конечный — прямоугольник, перекрывающий изображение целиком. Применение анимации формы позволит реализовать достаточно интересный и в какой-то степени непредсказуемый эффект (рис. 13.19, в).

Используя эти и другие варианты, можно создать необходимое число переходов, однако при просмотре становится заметной резкая смена предыдущего изображения следующим. Для того чтобы смена изображений не происходила скачкообразно, можно сделать так, чтобы каждое следующее изображение появлялось не из пустоты, а открывалось на фоне предыдущего. В этом случае переход будет выглядеть значительно естественнее. Добиться этого можно, создав третий слой, причем нужно назначить ему тип **Normal** и сделать его самым нижним в палитре слоев. Этот слой будет выполнять функцию заднего плана. Далее необходимо на слое заднего плана разместить используемые в слайд-шоу изображения таким образом, чтобы на протяжении всего диапазона открытия нового изображения предыдущее присутствовало бы на сцене (рис. 13.19). В результате каждое следующее изображение будет появляться как бы из предыдущего. При этом важно согласовать размеры всех изображений, приведя их к одинаковым значениям либо еще до импорта в рабочую среду Flash, либо непо-

средственно на сцене, поместив под изображения меньшего размера непрозрачные прямоугольники.

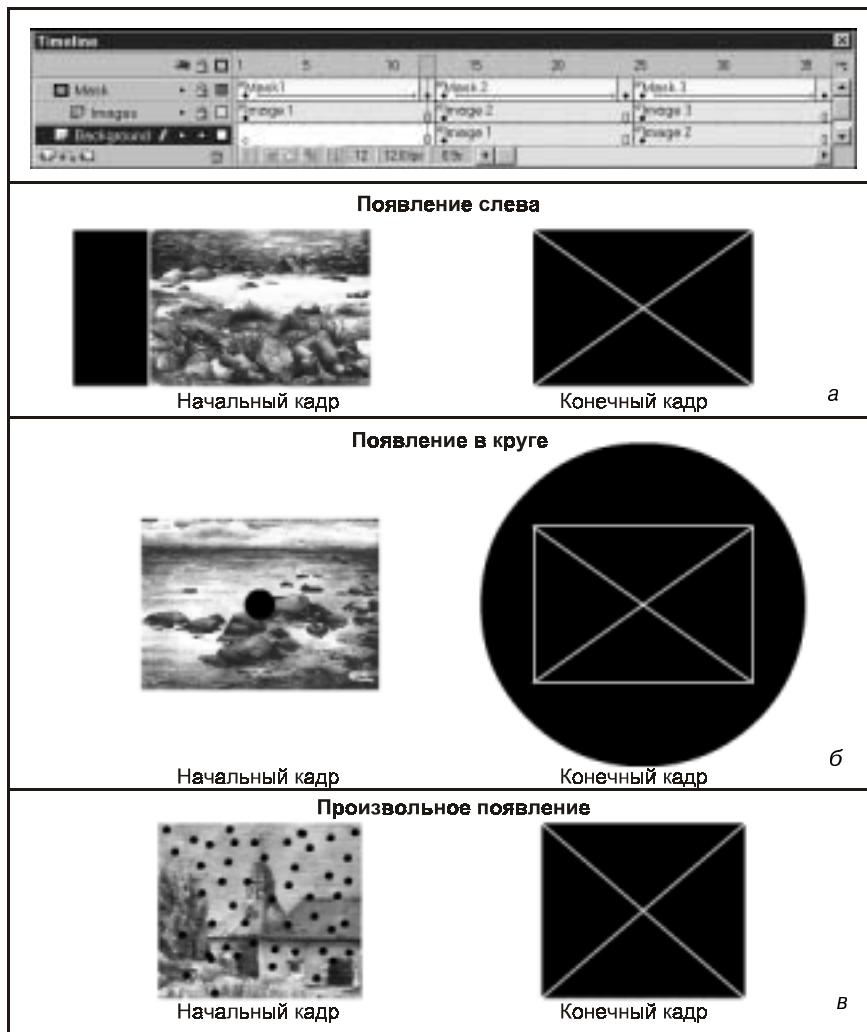


Рис. 13.19. Создание слайд-шоу с использованием маски

Пример. Эффект линзы.

Данный эффект заключается в том, что при перемещении изображения линзы над текстовым блоком его содержимое отображается с увеличением размера (рис. 13.20).

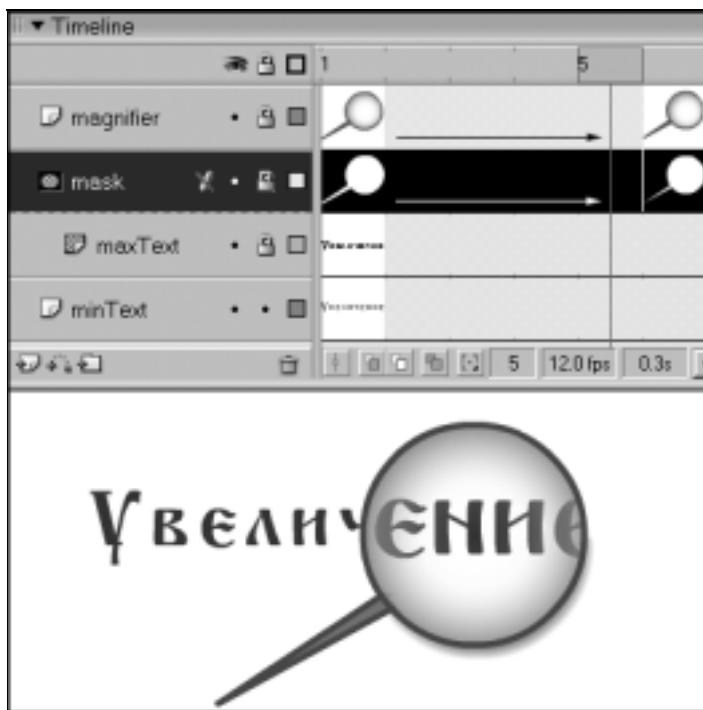


Рис. 13.20. Эффект линзы

Для реализации этого эффекта необходимо использование четырех слоев. Назначение этих слоев описывается далее от верхнего к нижнему.

- Слой **magnifier** содержит анимацию движения изображения линзы слева направо. Слой имеет тип **Normal** (Нормальный). Линза представляет собой экземпляр графического символа. Увеличительное стекло — это круг, залитый радиальным градиентом, прозрачность которого повышается по мере приближения к центру. Центральная область линзы должна быть полностью или частично прозрачной, так чтобы сквозь нее можно было увидеть содержимое нижележащих слоев.
- Слой **mask** содержит анимацию движения круга, по размерам совпадающего с линзой. Слой имеет тип **Mask** (Слой-маска). Анимация маски синхронизирована с анимацией линзы, т. е. эти два объекта всегда находятся точно друг над другом.
- Слой **maxText** содержит изображение увеличенного статичного текстового блока. Слой имеет тип **Masked** (Маскируемый). При воспроизведении монтажной линейки виден только тот фрагмент увеличенного текстового блока, расположенного на маскируемом слое, над которым в данный момент находится маска и, соответственно, синхронизированное с ней изо-

брожение линзы. Для того чтобы сквозь промежутки между буквами текстового блока маскируемого слоя не было видно содержимое нижележащего слоя, под увеличенный текст нужно подложить непрозрачный прямоугольник, цвет которого совпадает с цветом фона. В этом случае, когда маска находится над текстом, объекты нижележащего слоя не видны.

- Слой **minText** содержит изображение уменьшенного статичного текстового блока и играет роль фона. Слой имеет тип **Normal** (Нормальный). В той области, где находится маска и синхронизированное с ней изображение линзы, виден увеличенный текст; в остальной части рабочей области виден фоновый слой, содержащий уменьшенный текст.

Пример.

Эффект осциллографа

Данный эффект имитирует вычерчивание кривой электронным лучом осциллографа. След луча должен постепенно исчезать по мере удаления от него (рис. 13.21). Подобный эффект может быть использован для создания динамического акцента в оформлении элементов интерфейса. Для реализации этой задачи необходимо использовать три слоя.



Рис. 13.21. Эффект осциллографа

Назначение слоев описывается далее от нижнего к верхнему (рис. 13.22).

- Слой **gradient** содержит статичную прямоугольную форму, залитую линейным градиентом. Слой имеет тип **Masked** (Маскируемый). Левый цветовой порог данного градиента полностью прозрачен (**Alpha** = 0%), правый порог полностью непрозрачен (**Alpha** = 1000%). Цвет определяет цвет получившейся в результате кривой. Необходимо убедиться в том, что содержимое маскируемого слоя присутствует на сцене в течение всего диапазона анимации.

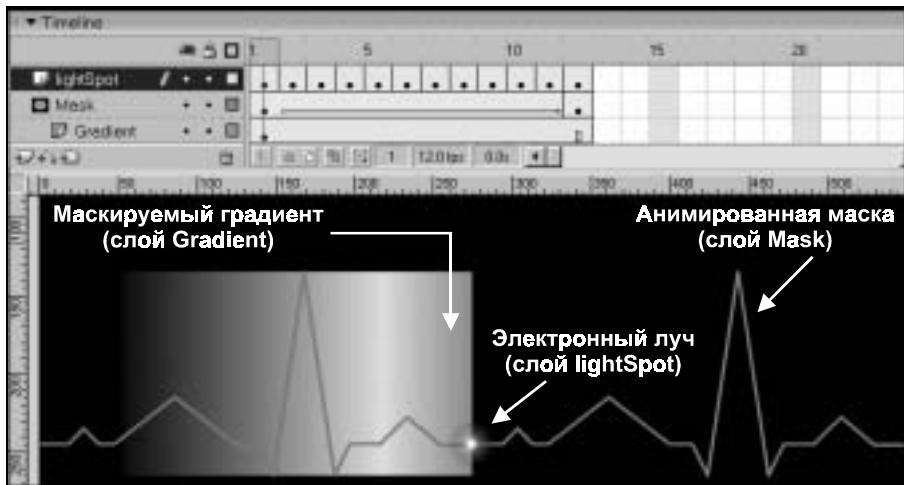


Рис. 13.22. Реализация эффекта осциллографа

- Слой **Mask** содержит анимацию движения графика, реализованную по принципу бесконечной ленты (см. разд. "Пример. Зацикливание фонового изображения" данной главы). Слой имеет тип **Mask** (Слой-маска). График состоит из двух одинаковых фрагментов. Длина одного фрагмента графика должна в точности совпадать с шириной прямоугольника на слое **gradient**. Поскольку в качестве маски могут быть использованы только сплошные области (векторные формы), график, нарисованный как контур, необходимо превратить в форму (**Modify>Shape>Convert Lines to Fills**), после чего его можно конвертировать в графический символ. Затем экземпляр данного символа должен быть помещен в первый ключевой кадр слоя **Mask** так, чтобы его левый край совпадал с левой границей рабочей области. В конечном кадре диапазона анимации необходимо сместить график влево так, чтобы реализовать эффект бесконечного движения.
- Слой **lightSpot** содержит покадровую анимацию, имитирующую движение электронного луча. Слой имеет тип **Normal** (Нормальный). Изображение электронного луча представляет собой экземпляр графического символа, содержащего небольшой по сравнению с размером графика круг, залиятый радиальным градиентом, правый цветовой порог которого полностью прозрачный. В первом ключевом кадре его необходимо позиционировать, поместив над правой (наиболее яркой) границей прямоугольника на слое **gradient** и одновременно совместить с линией графика. Далее в каждом вновь создаваемом ключевом кадре нужно смещать данный экземпляр по вертикали, совмещая его с линией графика, горизонтальная составляющая электронного луча не изменяется.

Встроенные эффекты монтажной линейки

Flash MX 2004 включает встроенные эффекты, позволяющие быстро получить требуемый результат. Эффекты могут быть применены к любым графическим объектам рабочей среды, как рабочего, так и наложенного уровня. В результате применения того или иного эффекта к объекту, он конвертируется в символ типа **Graphic**, который помещается на отдельный слой, получающий название в соответствии с применяемым эффектом. Реализация эффекта осуществляется на монтажной линейке данного символа посредством автоматического создания новых слоев, символов и диапазонов анимации движения, в которых используются их экземпляры. В случае применения эффекта к экземпляру символа типа **Movie Clip**, реализация эффекта осуществляется на его монтажной линейке. Flash предоставляет возможность редактирования назначенных эффектов. При необходимости эффект может быть удален.

Для того чтобы применить встроенный эффект к объекту, необходимо выделить его и выполнить команду главного меню **Insert>Timeline Effects>Категория эффекта>Название эффекта** или соответствующую команду контекстного меню самого объекта. Все эффекты сгруппированы в три категории:

- Assistants** — помощники;
- Effects** — эффекты;
- Transform/Transition** — Трансформация/Переход.

Настройка параметров каждого эффекта осуществляется в отдельном окне, содержащем типовые элементы интерфейса. Для просмотра предварительного результата можно воспользоваться кнопкой **Update Preview** данного окна. Для того чтобы применить эффект, нужно нажать кнопку **OK**. Для отмены применения эффекта нужно нажать кнопку **Cancel**.

Раздел **Assistants**

Данные эффекты позволяют быстро создать эффект мозаики, или перехода, используя любой графический объект.

Copy to Grid (**Копирование в сетку**)

Данный эффект позволяет автоматически скопировать объект и равномерно разместить его копии по строкам и столбцам сетки, создавая заполнение наподобие мозаики.

Эффект содержит следующие параметры настройки (рис. 13.23):

- Grid Size** — параметр, регулирующий число ячеек в сетке по вертикали (**ROWS**) и по горизонтали (**COLUMNS**), т. е. количество копий объекта;
- Grid Spacing** — расстояние между ячейками сетки, т. е. между объектами по вертикали (**ROWS**) и по горизонтали (**COLUMNS**).

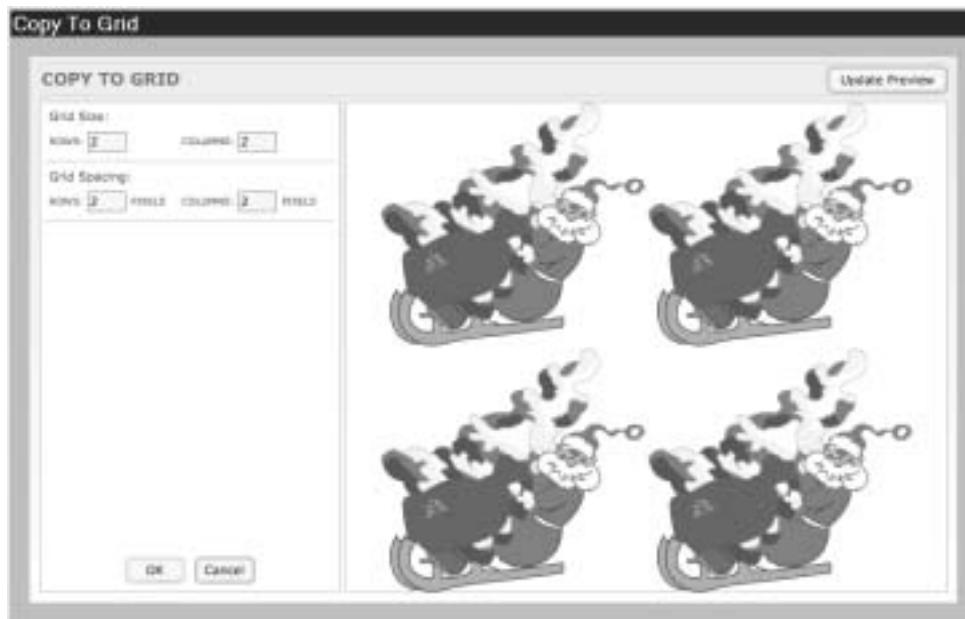


Рис. 13.23. Диалоговое окно **Copy to Grid**

Distributed Duplicate **(Распределение копий)**

Данный эффект позволяет автоматически скопировать объект и разместить его копии, задав шаг смещения и поворота каждой следующей копии по отношению к предыдущей, а также организовать плавный переход цвета и прозрачности. В результате можно получить статичное изображение или анимацию. Эффект содержит следующие параметры настройки (рис. 13.24):

- Number of Copies** — количество копий объекта;
- Offset Distance** — смещение каждой копии по отношению к предыдущей по горизонтали (**X**) и вертикали (**Y**), задаваемое в пикселях;
- Offset Rotation** — угол поворота в градусах каждой следующей копии по отношению к предыдущей;

- Offset Start Frame** — интервал (в кадрах) между появлением копий. Задание значения, отличного от нуля, приводит к созданию анимации;
- параметры масштабирования копий:
 - **Exponential Scaling/Linear Scaling** — экспоненциальное масштабирование/линейное масштабирование;
 - **Lock/Unlock** — пропорциональное/непропорциональное масштабирование;
 - **Scale** — значение масштаба в процентах. При непропорциональном масштабировании можно указать масштаб по горизонтали и вертикали;
- цветовые установки:
 - установка флашка **Change Color** позволяет реализовать плавный переход цвета от начального к конечному. Конечный цвет перехода выбирается из стандартного каталога цветов при помощи цветового образца **Final Color**;
 - **Alpha** — позволяет реализовать плавное изменение степени прозрачности объекта. Конечное значение прозрачности вводится в поле **Final Alpha** или задается при помощи горизонтального слайдера.

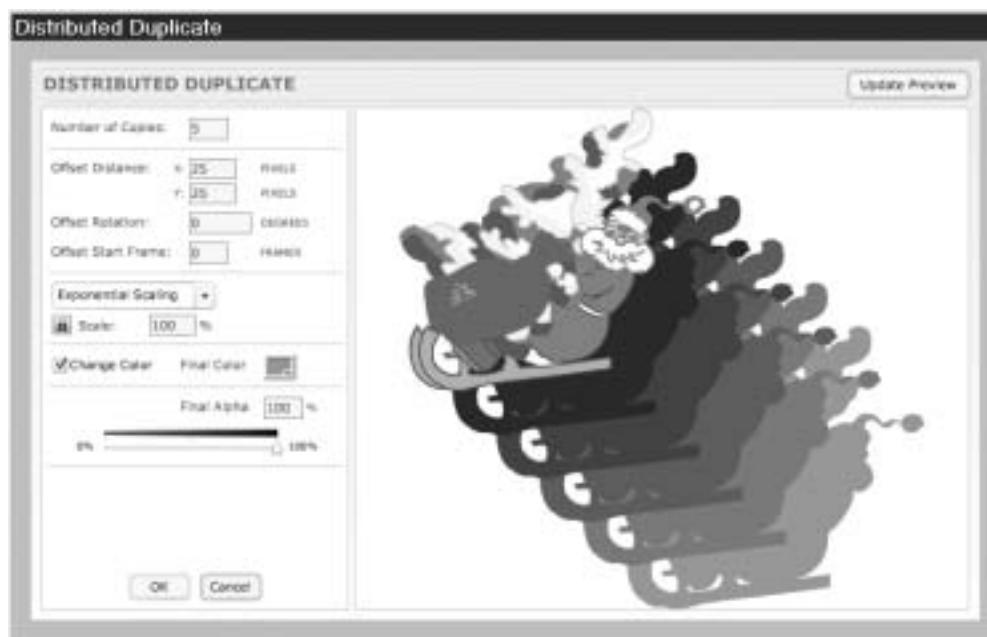


Рис. 13.24. Диалоговое окно **Distributed Duplicate**

Раздел *Effects*

Данная группа операций позволяет быстро создать стандартные статичные и анимированные эффекты размытия, тени, расширения/сжатия, взрыва.

Blur (Размытие)

Данная операция позволяет сымитировать эффект размытия формы путем создания копий начального объекта и изменения их масштаба и степени прозрачности. Эффект содержит следующие параметры настройки (рис. 13.25):

- Effect Duration** — длительность выполнения анимации размытия;
- Resolution** — количество дополнительных форм, используемых для имитации размытия;
- Allow Horizontal Blur** — размытие по горизонтали;
- Allow Vertical Blur** — размытие по вертикали;
- Direction of Movement** — направление размытия.

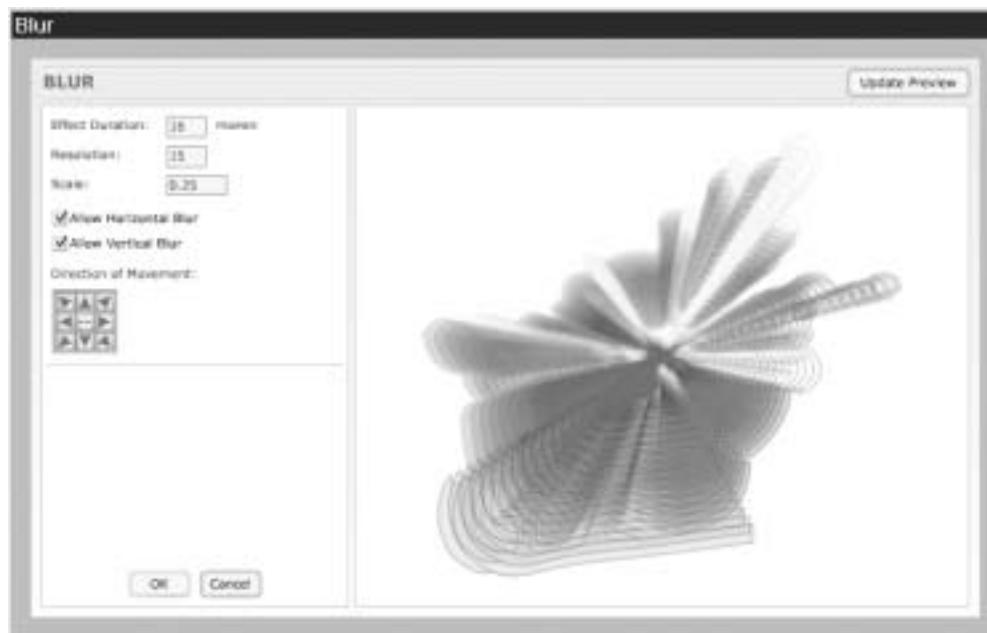


Рис. 13.25. Диалоговое окно **Blur**

Drop Shadow (Тень)

Данная операция позволяет создать тень для любого статичного графического объекта. Тень представляет собой копию объекта, окрашенную заданным цветом и содержащую указанную степень прозрачности, помещенную под оригинал. Эффект содержит следующие параметры настройки (рис. 13.26):

- Color** — цвет тени;
- Shadow Transparency** — степень прозрачности тени;
- Alpha Offset** — смещение (в пикселях) тени по горизонтали и вертикали.



Рис. 13.26. Диалоговое окно **Drop Shadow**

Expand (Расширение)

Данная операция позволяет создать анимационный эффект увеличения/скатия объекта. Эффект не может быть применен к объектам рабочего уровня. Эффект содержит следующие параметры настройки (рис. 13.27):

- Effect Duration** — длительность анимации;
- направление действия: расширение — **Expand**, скатие — **Sneeze**, расширение и затем скатие — **Both**;
- Direction of Movement** — направление движения;

- Shift Group Center by** — смещение центра трансформации;
- Fragment Offset** — смещение объекта при трансформации;
- Change Fragment Size by** — изменение размера по высоте (**Height**) и ширине (**Width**).

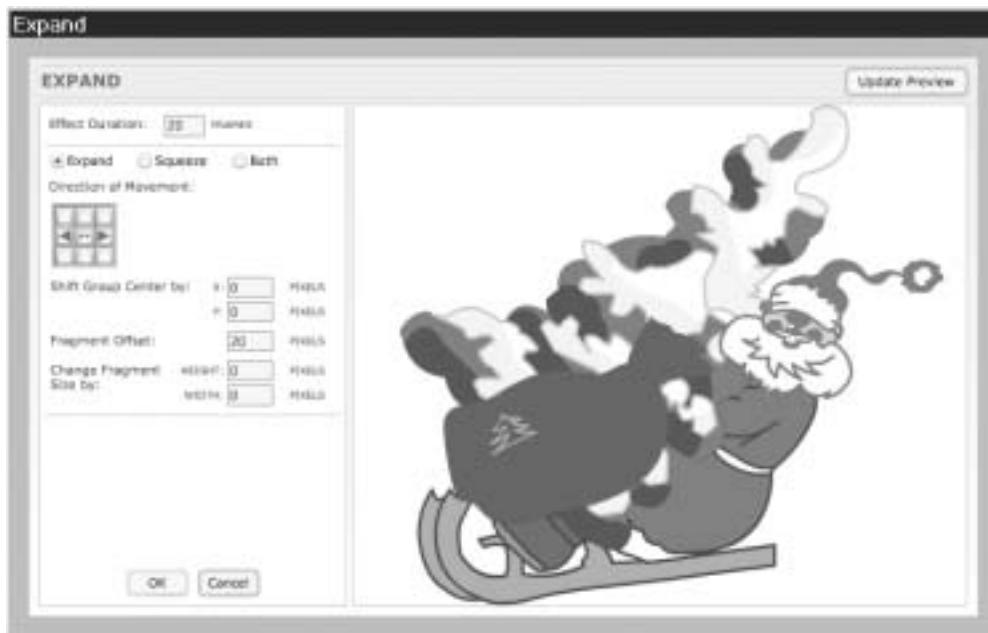


Рис. 13.27. Диалоговое окно **Expand**

Explode (Взрыв)

Данная операция имитирует взрыв и разлетание частей любого графического объекта. Эффект **Explode** содержит следующие параметры настройки (рис. 13.28):

- Effect Duration** — длительность анимации;
- Direction of Explosion** — направление движения частей объекта;
- Arc Size** — величина дуги полета частей по горизонтали и вертикали;
- Rotate Fragments by** — угол поворота частей объекта в процессе анимации;
- Change Fragments Size by** — изменение размеров частей объекта по горизонтали и вертикали;
- Final Alpha** — конечная степень прозрачности объектов.

Рис. 13.28. Диалоговое окно **Explode**

Раздел *Transform/Transition*

Данная группа операций позволяет реализовать эффекты, характерные для автоматической анимации движения.

Transform (Трансформация)

Данная операция позволяет выполнить смещение одновременно с вращением, масштабированием и изменением цвета объекта. Эффект содержит следующие параметры настройки (рис. 13.29):

- **Effect Duration** — длительность анимации;
- параметры движения — раскрывающийся список позволяет либо сместить объект на заданное расстояние (**Change Position by**), либо переместить его в заданные координаты (**Move to Position**);
- **Scale** — масштабирование (пропорциональное или непропорциональное);
- **Rotate/Spin** — вращение на заданный угол (**Rotate**) или поворот в заданном направлении требуемое количество раз (**Spin**);
- **Change Color** — установка данного флагка позволит реализовать изменение цвета объекта. Конечный цвет задается при помощи цветового образца **Final Color**;

- Final Alpha** — задание конечного значения степени прозрачности объекта;
- Motion Ease** — замедление анимации вначале (**SLOW AT START**) или в конце (**SLOW AT END**).

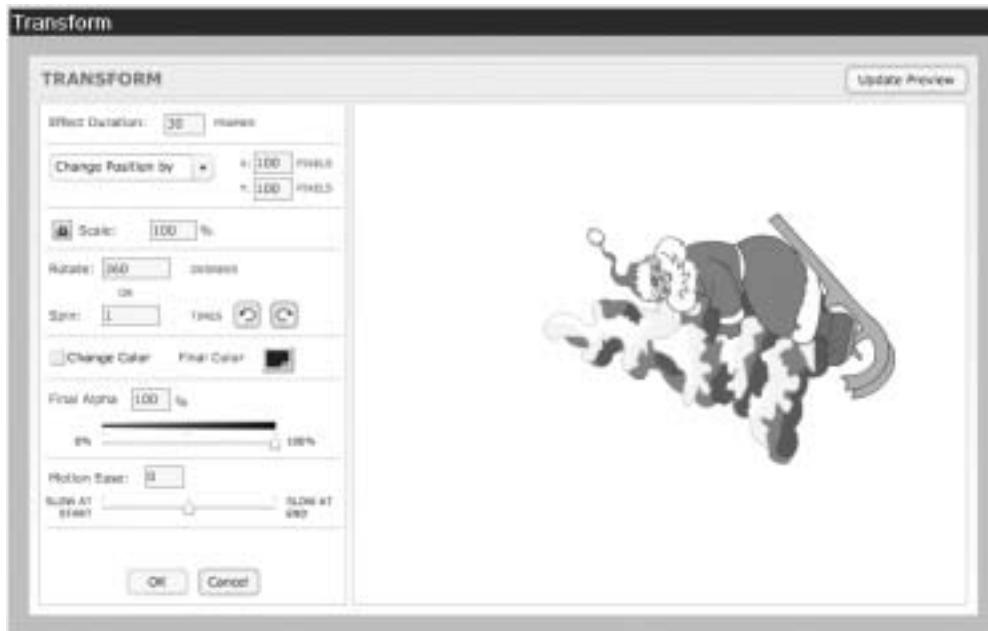


Рис. 13.29. Диалоговое окно **Transform**

Transition (Переход)

Данная операция позволяет реализовать эффект появления или исчезания (стирания) объекта с одновременным усилением тона или переходом в прозрачность. Эффект содержит следующие параметры настройки (рис. 13.30):

- Effect Duration** — длительность анимации;
- Fade/Wipe** — характер выполнения эффекта: **Fade** — переход тона, **Wipe** — появление или исчезание объекта;
- Direction: In** — усиление тона и/или появление объекта, **Out** — ослабление тона (переход в прозрачность) и/или стирание объекта;
- Motion Ease** — замедление анимации вначале (**SLOW AT START**) или в конце (**SLOW AT END**).

Для того чтобы отредактировать созданный эффект, необходимо выделить объект, содержащий эффект, и выполнить одно из следующих действий:

- выполнить команду главного меню **Modify>Timeline Effects>Edit Effect**;
- выполнить команду контекстного меню объекта **Edit Effect**;
- в Инспекторе свойств нажать кнопку **Edit**, расположенную под названием текущего эффекта.

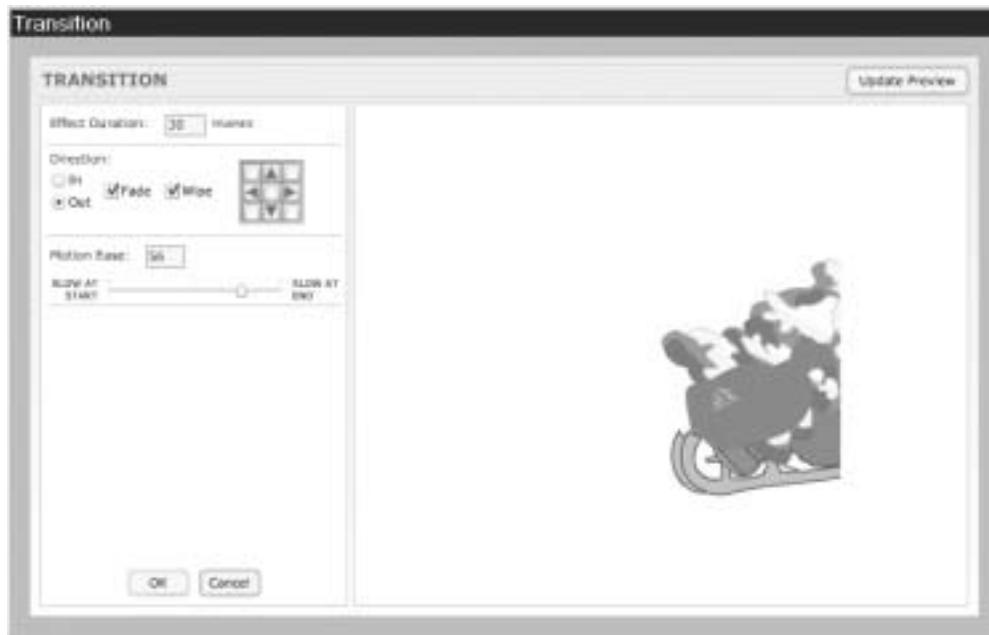
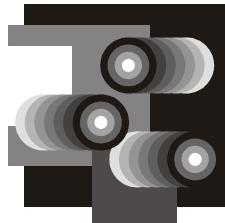


Рис. 13.30. Диалоговое окно **Transition**

Для того чтобы удалить эффект, необходимо выделить соответствующий объект и выполнить команду главного меню **Modify>Timeline Effects>Edit Effect** или аналогичную команду контекстного меню объекта.

Для того чтобы отредактировать эффект непосредственно на монтажной линейке содержащего его графического символа, нужно щелкнуть на экземпляре правой кнопкой мыши и выполнить одну из команд контекстного меню: **Edit**, **Edit in Place** или **Edit in New Window**. Редактирование символа, содержащего эффект, приведет к невозможности дальнейшего редактирования данного эффекта с использованием параметров его настройки.



Глава 14

Работа со звуком

Для каждого сказанного слова есть слушающее ухо.

Арабская пословица

Звук возникает в результате колебаний воздушной среды, возбуждаемых вибрирующим предметом. Волны воздушного давления с определенной частотой и интенсивностью действуют на барабанную перепонку, вызывая слуховые ощущения. Звуковые волны смешиваются, накладываясь друг на друга, и образуют музыку, человеческую речь, всевозможные шумы и т. д.

Звук является важнейшим элементом мультимедийного проекта и может быть использован для озвучивания персонажей, определенных событий или действий, создания фонового сопровождения и всевозможных шумовых эффектов. Зачастую грамотное и удачное применение звукового сопровождения позволяет обеспечить успех всего Flash-фильма.

Основные параметры цифрового звука

Для записи, редактирования и воспроизведения звуковых сигналов в компьютерной среде используется процедура *оцифровки*, или *дискретизации* (Digital sampling). При оцифровке через равные интервалы времени осуществляется ряд замеров звука, результаты которых записываются в цифровой форме. Таким образом, в ходе оцифровки непрерывный звуковой сигнал разбивается на ряд дискретных отсчетов (квантов), называемых *выборками* (samples). Чем выше число замеров звука в единицу времени и чем больше информации используется для описания результатов каждого замера, тем выше качество цифрового звука и, соответственно, объем конечного файла.

Существуют два основных параметра, определяющих качество цифрового звука и влияющих на размер его файла.

- Частота дискретизации (Sampling rate) — характеризует количество отсчетов, производимых в единицу времени при оцифровке, и измеряется в Гц.

- Разрядность (Sample size) или разрешение — характеризует количество информации, используемое для описания каждого отсчета, и измеряется в битах (bit). Так, например, каждый отсчет 8-битного звука может принимать одно из $2^8 = 256$ значений, а каждый отсчет 16-битного звука — одно из $2^{16} = 65\,536$ значений.

На рис. 14.1 представлен график цифрового изображения звука. Каждая вертикальная линия представляет собой один отсчет (квант). Количество таких линий в секунду определяет частоту дискретизации, а диапазон значений высот линий — разрядность.

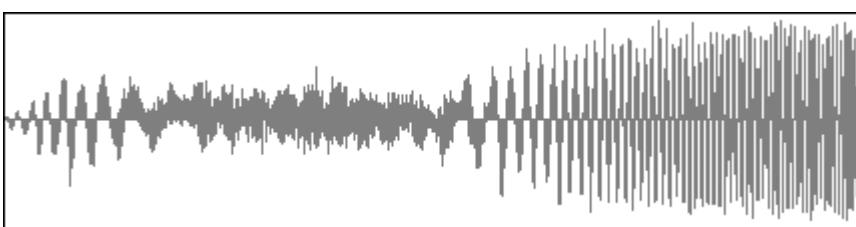


Рис. 14.1. График цифрового изображения звука

Звуковой файл может содержать несколько каналов. Flash поддерживает стерео (2 канала) и моно (1 канал) аудиофайлы. Стереозвук занимает вдвое больший объем, чем аналогичный монозвук. Это связано с тем, что для описания звука, содержащего 2 канала, используется вдвое большее количество информации.

Объем файла цифрового звука в байтах может быть определен по следующей формуле: *Размер файла (bytes) = Частота дискретизации (Гц) × Разрядность (bit)/8 × Продолжительность (с) × Количество каналов.*

Таким образом, чем выше частота дискретизации и разрядность, тем точнее цифровой звук воспроизводит оригинал и тем больше размер файла цифрового звука.

Импорт звука в Flash

Для того чтобы снабдить фильм звуковым сопровождением в рабочей среде, необходимо импортировать соответствующий файл. В рабочую среду Flash можно импортировать звуковые файлы следующих форматов:

- WAV (только для Windows);
- MP3;
- AIFF (только для Macintosh).

При наличии QuickTime 4 или более поздних версий можно также импортировать дополнительные форматы:

- Sound Only QuickTime Movies;
- Sun AU;
- WAV (как для Windows, так и для Macintosh);
- Sound Designer II (только для Macintosh);
- System 7 Sounds (только для Macintosh).

Для того чтобы импортировать звуковой файл, необходимо выполнить команду главного меню **File>Import>Import to Library** или команду **File>Import>Import to Stage**. В результате выполнения любой из данных команд звуковой файл помещается в библиотеку текущего документа. Один и тот же звуковой файл, находящийся в библиотеке, может многократно использоваться в фильме. Причем многократное применение экземпляра одного и того же звука не будет приводить к увеличению объема конечного фильма (при использовании определенных типов синхронизации). Здесь имеет место та же ситуация, что и при работе с растровыми изображениями.

Для того чтобы ввести звук в фильм, необходимо поместить его в ключевой кадр монтажной линейки. Это можно сделать двумя способами.

- Выделить ключевой кадр, в который должен быть помещен звук, и перетащить требуемый звук из библиотеки на сцену, так же как и при создании экземпляра символа.
- Выделить ключевой кадр, в который должен быть помещен звук, и в списке **Sound** Инспектора свойств, содержащем все находящиеся в библиотеке текущего документа звуки, выбрать требуемый (рис. 14.2).

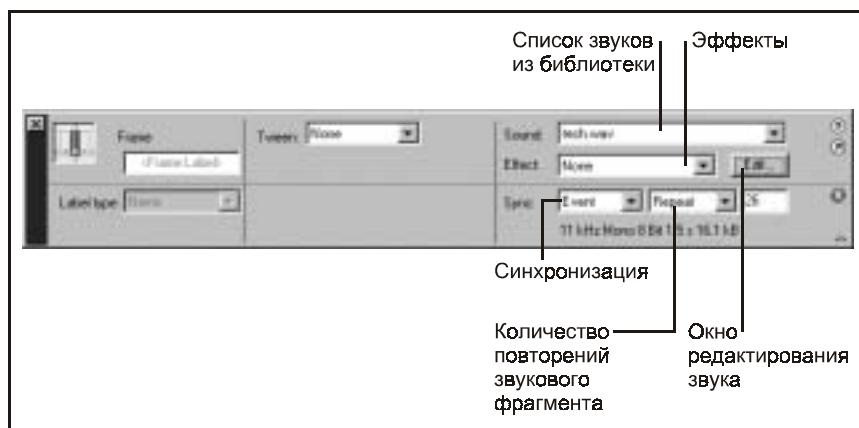


Рис. 14.2. Параметры экземпляра звука. Инспектор свойств

В результате этих действий в соответствующем кадре монтажной линейки появится график цифрового представления звука (рис. 14.3). Если за ключевым кадром, содержащим звук, не следуют простые кадры, то на монтажной линейке будет виден только тот фрагмент графика, который соответствует длительности экспонирования кадра. При выделении кадра, содержащего звук, в нижней части инспектора свойств указываются все параметры данного звукового файла: частота дискретизации, число каналов, разрядность, длительность, занимаемый размер в Кбайт.

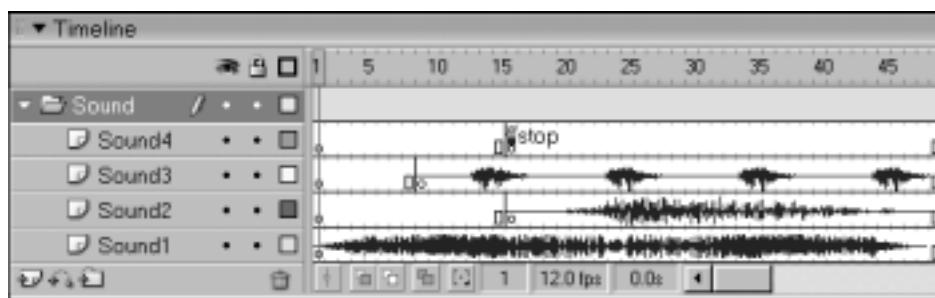


Рис. 14.3. Отображение звука на монтажной линейке

Каждый ключевой кадр монтажной линейки может содержать только один экземпляр звука. Для того чтобы одновременно использовать несколько разных звуков, необходимо каждый из них размещать на отдельном слое.

Типы синхронизации звука в Flash

Звуки, используемые в проекте, могут иметь различное назначение. В соответствии с этим существует возможность задать тип синхронизации, определяющий характерные особенности воспроизведения и обработки звука для каждого применяемого экземпляра. Для того чтобы задать тип синхронизации экземпляра звука, необходимо выделить содержащий его кадр на монтажной линейке и воспользоваться списком **Sync** Инспектора свойств.

Flash позволяет задать следующие типы синхронизации.

Event (Событие) — данный тип синхронизации используется по умолчанию. Экземпляр звука, использующий тип синхронизации **Event**, будем называть *событийным* звуком. Событийные звуки обладают следующими характерными особенностями.

- Воспроизведение событийного звука начинается, когда воспроизводящая головка входит в ключевой кадр, содержащий экземпляр данного звука.

- Событийный звук может быть воспроизведен только после того, как он полностью загрузился. При загрузке фильма по сети с недостаточной скоростью соединения воспроизведение монтажной линейки может быть приостановлено на кадре, содержащем событийный звук, до тех пор, пока данный звук не будет загружен. Этого явления можно избежать при помощи предзагрузчика (см. гл. 21).
 - Событийный звук не зависит от содержащей его монтажной линейки. Если данный звук запущен, то он прозвучит до конца вне зависимости от того, продолжается ли анимация содержащей его монтажной линейки или нет. Так, если экземпляр пятисекундного звука поместить в первый ключевой кадр монтажной линейки, содержащей 12 кадров (при скорости воспроизведения 12 кадров в секунду), то, будучи запущен, данный звук будет воспроизведен целиком, несмотря на то, что анимация монтажной линейки длится только одну секунду.
 - Событийные звуки *микшируются*, т. е. могут накладываться сами на себя. Микширование может иметь место в различных случаях. Например, если событийный звук помещен в первый кадр монтажной линейки, длительность воспроизведения которой меньше продолжительности его звучания (как в предыдущем примере), то при зацикливании монтажной линейки (**Control>Loop Playback**) звук будет накладываться сам на себя. Всякий раз при переходе воспроизводящей головки с последнего кадра в первый звук будет запускаться вновь, при этом накладываясь на ранее запущенный экземпляр.
 - Многократное использование одного и того же событийного звука практически не приводит к существенному увеличению объема конечного файла. На объем конечного фильма влияет только размер звукового файла, находящегося в библиотеке, а не число его экземпляров. В связи с этим, событийные звуки могут быть зациклены, что позволяет использовать их в качестве фонового звукового сопровождения.
- **Start** (Старт) — синхронизация типа **Start** применяется для исключения возможности наложения событийных звуков самих на себя. Экземпляр звука, использующий тип синхронизации **Start**, будем называть звуком типа **Start**. Звук типа **Start** обладает многими свойствами событийного звука, но его характерной особенностью является то, что этот звук будет воспроизводиться только в том случае, если другой экземпляр этого же звука (типа **Event** или **Start**) не воспроизводится в данный момент. Так, если звук типа **Start** помещен в первый кадр монтажной линейки, длительность воспроизведения которой меньше продолжительности его звучания, то при зацикливании монтажной линейки наложения звука не произойдет. При переходе воспроизводящей головки с последнего кадра в первый звук не будет запускаться вновь, если продолжается его воспроизведение, инициированное предыдущим запуском.

- **Stop** (Стоп) — данный тип синхронизации используется для остановки воспроизведения событийного звука. Экземпляр звука, использующий тип синхронизации **Stop**, будем называть звуком типа **Stop**. Как только воспроизводящая головка входит в ключевой кадр монтажной линейки, содержащий звук типа **Stop**, все воспроизводящиеся в этот момент экземпляры того же самого звука будут остановлены (рис. 14.4). Звук типа **Stop** сам не звучит и применяется исключительно для остановки воспроизведения событийных (**Event**) звуков и звуков типа **Start**. Экземпляр звука типа **Stop** может остановить воспроизведение только экземпляров того же самого звука. На монтажной линейке он отображается знаком паузы (синий прямоугольник).

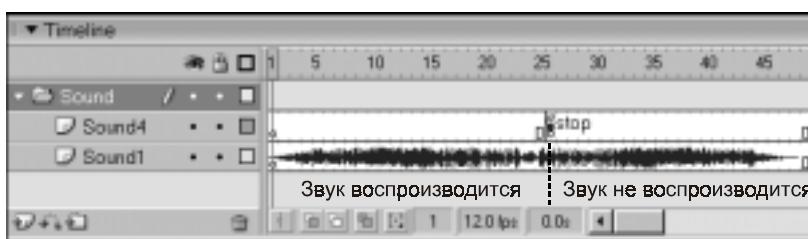


Рис. 14.4. Применение синхронизации звука типа **Stop**

- **Stream** (Поток) — данный тип используется для синхронизации анимации со звуком. Экземпляр звука, использующий тип синхронизации **Stream**, будем называть *потоковым* звуком. Потоковый звук имеет следующие характерные особенности.
- Потоковый звук начинает воспроизводиться, как только загрузится достаточное количество информации для начала звучания.
 - Потоковый звук зависит от содержащей его монтажной линейки, т. е. он воспроизводится только в тех кадрах, в которых он размещен. Для воспроизведения потокового звука в течение определенного временного интервала после ключевого кадра, содержащего данный звук, необходимо поместить на монтажную линейку соответствующее требуемому интервалу число простых кадров (рис. 14.5). В отличие от событийного звука, потоковый звук не будет звучать при остановке воспроизведения содержащей его монтажной линейки. Потоковый звук также не будет звучать дольше, чем воспроизводятся кадры, в которых он размещен. Так, если пятисекундный событийный звук будет помещен на монтажную линейку, содержащую 12 кадров (при скорости воспроизведения 12 кадров в секунду), то при воспроизведении можно будет услышать только первую секунду его звучания.

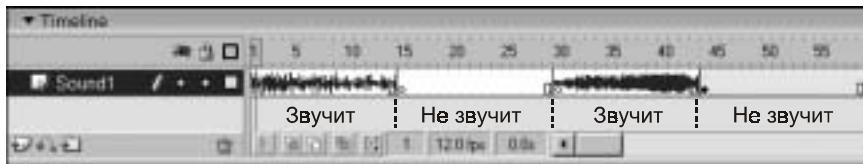


Рис. 14.5. Потоковый звук воспроизводится только в тех кадрах, в которых он размещён

- Потоковый звук имеет приоритет над визуальным содержимым монтажной линейки. При воспроизведении монтажной линейки Flash стремится сохранить целостность звучания потокового звука и пытается обеспечить синхронное воспроизведение звука и анимации. Однако если из-за низкой скорости соединения содержимое некоторых кадров не успевает прорисовываться, они будут пропущены ради нормальной передачи звуковой информации.
- Потоковый звук разбивается на отдельные фрагменты, которые жестко привязываются к содержащим их кадрам монтажной линейки. В связи с этим использование нескольких экземпляров одного и того же потокового звука в разных участках монтажной линейки приведет к пропорциональному увеличению объема конечного файла. Поэтому потоковые звуки не рекомендуется зацикливать.

Редактирование звука

К экземплярам звуков могут быть применены различные звуковые эффекты. Flash содержит набор встроенных эффектов, которые основаны на манипуляциях с уровнем (громкостью) звукового сигнала. Каждому экземпляру одного и того же звукового файла может быть назначен свой звуковой эффект, что позволяет добиться определенного разнообразия, используя ограниченный набор звуков.

Для того чтобы назначить экземпляру звука один из стандартных звуковых эффектов, необходимо на монтажной линейке выделить содержащий данный экземпляр кадр и в списке **Effect** Инспектора свойств выбрать одно из предлагаемых значений:

- None** — отсутствие звукового эффекта;
- Left Channel** — звучит только левый канал;
- Right Channel** — звучит только правый канал;
- Fade Left to Right** — постепенное понижение уровня звука с перетеканием из левого в правый канал;

- Fade Right to Left** — постепенное понижение уровня звука с перетеканием из правого в левый канал;
- Fade In** — нарастание звука, постепенное повышение уровня звука в начале;
- Fade Out** — затухание звука, постепенное понижение уровня звука в конце;
- Custom** — создание пользовательского эффекта (*см. далее*).

Flash позволяет создать пользовательский звуковой эффект, задавая изменение уровня, а также точки начала и конца звучания экземпляра звука. Для создания пользовательского звукового эффекта можно либо выбрать в списке **Effect** значение **Custom**, либо щелкнуть на кнопке **Edit** Инспектора свойств. Любое из этих действий приведет к появлению диалогового окна **Edit Envelope**, представленного на рис. 14.6.

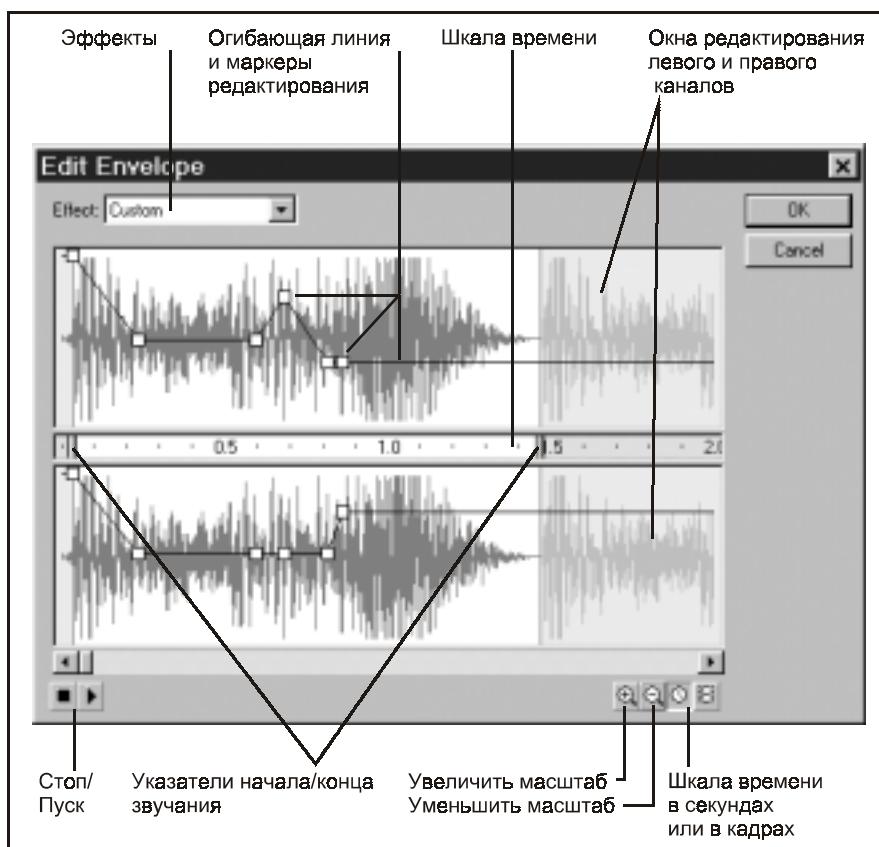


Рис. 14.6. Диалоговое окно **Edit Envelope**

Окно **Edit Envelope** содержит следующие элементы:

- **Effect** — список, идентичный списку инспектора свойств, позволяющий выбрать один из стандартных эффектов, на основе которых можно создать свой собственный;
- огибающая с маркерами редактирования — окно **Edit Envelope** содержит графики цифрового представления звука для левого и правого каналов. Положение огибающей линии определяет уровень звука. Если огибающая находится над графиком — это соответствует максимальному уровню для данного канала; расположение огибающей под наклоном соответствует плавному увеличению или уменьшению уровня звука. Изменение положения огибающей задается при помощи расположенных на ней маркеров редактирования. Для того чтобы добавить новый маркер на огибающую, нужно щелкнуть на ней в соответствующей точке. Максимальное число маркеров редактирования не может превышать восьми. Смещая маркеры по горизонтали и вертикали, можно управлять громкостью звучания данного экземпляра. Для того чтобы удалить маркер редактирования, необходимо вытащить его за пределы области редактирования;
- шкала с маркерами начала и конца звучания — шкала, разделяющая окна левого и правого каналов, содержит маркеры, при помощи которых можно задать точки начала и конца звучания редактируемого экземпляра. Смещая их по горизонтали, можно "отсекать" фрагменты звука в начале и в конце. Такая процедура называется *тримминг* (trimming). С ее помощью можно, например, избавиться от "мертвого воздуха" — тишины в начале и в конце звукового файла. При установке точек начала и конца звучания те фрагменты, которые не будут воспроизведиться, отображаются на сером фоне. В конечный фильм войдет только тот звуковой фрагмент, который находится между маркерами начала и конца звучания, поэтому "отрезая" лишние участки, можно в определенной степени оптимизировать размер конечного файла;
- **Zoom in/Zoom out** — эти кнопки позволяют увеличить или уменьшить масштаб просмотра графика;
- **Seconds/Frames** — при помощи этих кнопок можно в качестве единиц измерения шкалы установить секунды или кадры монтажной линейки;
- **Stop sound/ Play sound** — используя эти кнопки, можно прослушать результат выполненных преобразований.

Оптимизация звука

Применение звука может привести к очень существенному увеличению конечного объема файла. Для минимизации объема конечного фильма необходимо

димо провести оптимизацию звуковых файлов. Оптимизация должна включать в себя два этапа.

- Оптимизация звукового файла во внешнем звуковом редакторе на стадии, предшествующей импорту в Flash. Необходимо удалить из аудиофайла пустые места (тишину), минимизировать длительность его звучания, при необходимости подготовить к зацикливанию. Не стоит импортировать в рабочую среду очень продолжительные звуки.
- Оптимизация средствами Flash, включающая задание точек начала и конца звучания и настройки параметров сжатия (компрессии) звуков, используемых в фильме.

Flash предоставляет возможность задать параметры сжатия для событийных и потоковых звуков, используемых в фильме. Каждому звуку могут быть назначены индивидуальные настройки сжатия, позволяющие минимизировать конечный объем файла. Кроме того, можно задать настройки компрессии для всех событийных и потоковых звуков одновременно непосредственно при публикации (см. гл. 17). Flash позволяет использовать различные алгоритмы сжатия для достижения оптимального соотношения между качеством звука и размером экспортруемого звукового файла.

Для того чтобы задать параметры компрессии звукового файла, необходимо выполнить одно из следующих действий:

- выделить требуемый звуковой файл в библиотеке и выполнить команду его контекстного меню или контекстного меню библиотеки **Properties**;
- дважды щелкнуть на пиктограмме требуемого звука в библиотеке;
- выделить требуемый звуковой файл в библиотеке и щелкнуть на пиктограмме **Properties** в нижней части окна библиотеки.

В результате любого из этих действий появится диалоговое окно **Sound Properties**, представленное на рис. 14.7. В правой части окна содержится ряд кнопок, используемых для выполнения следующих действий:

- **Update** — загрузка новой версии данного звукового файла;
- **Import** — замена данного звукового файла другим;
- **Test** — воспроизведение звукового файла непосредственно в окне **Sound Properties**;
- **Stop** — остановка воспроизведения.

Список **Compression** позволяет выбрать один из алгоритмов сжатия и установить требуемые параметры:

- **Default** — при выборе этого значения звук будет экспортирован в конечный фильм с теми параметрами компрессии, которые задаются непосредственно при публикации в диалоговом окне **Publish Settings** (см. гл. 17).

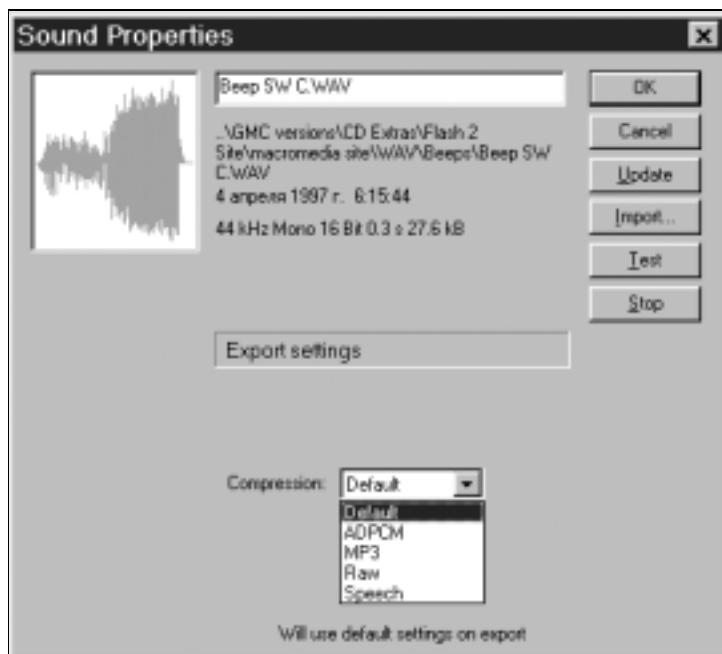


Рис. 14.7. Диалоговое окно **Sound Properties**

□ **ADPCM** (Adaptive-Differential Pulse Code Modulation) — данный алгоритм был разработан для передачи звуковых сигналов по телефонным линиям. Он использует сжатие с фиксированной скоростью (fixed-rate compression). При таком методе компрессии все данные сжимаются на одну и ту же величину, причем последовательность замеров преобразовывается с сохранением лишь разницы между соседними значениями. Оптимизация достигается за счет того, что значение разностей меньше значения самих замеров. Для хранения малых значений разности наряду с большими применяются специальные коды для указания масштаба, который используется последующим набором разностей. Этот коэффициент масштаба в некоторых случаях позволяет представлять большое изменение, используя относительно малые значения разностей. Данный алгоритм содержит следующие параметры настройки (рис. 14.8);

- **Preprocessing** — установка флажка **Convert stereo to mono** позволяет преобразовать стереозвук в моно, автоматически уменьшая объем файла в два раза. Данная установка вызывает эффект только на стереозвуках;
- **Sample rate** — данный параметр влияет на чистоту звука и, соответственно, на конечный объем звукового файла. Чем выше значение частоты дискретизации, тем выше качество звука и занимаемый им объем файла. Установка для данного параметра более высоких значений, чем

у импортированного звука приведет к увеличению объема файла, но на качестве звука не отразится. Ниже представлены предельно допустимые значения частоты дискретизации для различных типов звука:

- ◊ 5 kHz — предельное значение для звуков человеческой речи;
- ◊ 11 kHz — короткие музыкальные фрагменты (четверть качества музыкального компакт-диска);
- ◊ 22 kHz — может быть использовано в проектах, предназначенных для Интернета;
- ◊ 44 kHz — высокое качество, соответствующее качеству компакт-диска.
- **ADPCM bits** — количество бит информации, используемых при кодировании (рис. 14.8). Чем выше данное значение, тем выше качество звука и больше объем занимаемого им файла;

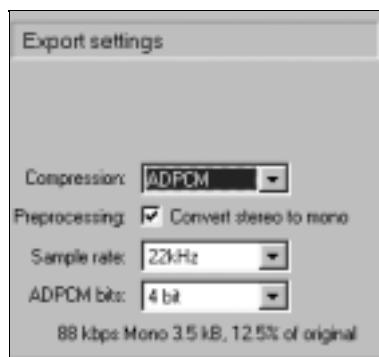


Рис. 14.8. Параметры настройки ADPCM-сжатия

□ **MP3 (MPEG-1 Audio Layer-3)** — наиболее эффективный алгоритм, обеспечивающий высокую степень сжатия и приемлемое качество звука. Алгоритм MP3 уменьшает количество избыточной информации, используемой для описания цифрового звука. При этом учитываются особенности человеческого восприятия — например, неравномерная чувствительность к восприятию звука на разных частотах. Данный алгоритм хорошо подходит для сжатия продолжительных музыкальных фрагментов и содержит следующие параметры настройки (рис. 14.9):

- **Use imported MP3 quality** — этот флажок появляется только при работе со звуками в формате MP3. Его установка позволяет экспортить звук в конечный фильм с теми же параметрами, с которыми он был импортирован в рабочую среду. Для того чтобы задать дополнительные параметры оптимизации, данный флажок должен быть снят;

- **Preprocessing** — установка флажка **Convert stereo to mono** позволяет преобразовать стереозвук в моно, смешивая левый и правый каналы в один;
- **Bit rate** — количество бит в секунду, используемое для кодирования экспортируемого звука. Диапазон используемых значений — от 8 Kbps до 160 Kbps с постоянной скоростью потока данных (CBR — constant bit rate);
- **Quality** — данный список позволяет выбрать соотношение между скоростью сжатия передачи и качеством звука и содержит следующие значения:
 - ◊ **Fast** — наиболее высокая скорость передачи, качество может значительно ухудшиться;
 - ◊ **Medium** — компромисс между скоростью и качеством, наиболее приемлемое решение для интернет-проектов;
 - ◊ **Best** — оптимальное качество, требующее высокой скорости передачи данных.

Выбор того или иного значения параметра **Quality** не отражается на размере конечного файла, а влияет на скорость его передачи и, соответственно, на качество звука;

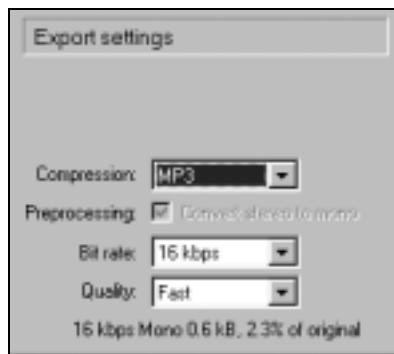


Рис. 14.9. Параметры настройки MP3-сжатия

- **RAW** — данный алгоритм экспортирует звук в конечный фильм, не сжимая его. RAW позволяет задать следующие параметры, аналогичные соответствующим параметрам алгоритма **ADPCM**:

- **Preprocessing** — установка флажка **Convert stereo to mono** позволяет преобразовать стереозвук в моно, смешивая левый и правый каналы в один;
- **Sample Rate** — частота дискретизации;

□ **Speech** — данный алгоритм использует схему сжатия, применяемую для передачи звуков речи. Он позволяет задать значение частоты дискретизации звука, выбрав одно из предлагаемых значений:

- 5 kHz — допустимо для речи;
- 11 kHz — рекомендуется для речи;
- 22 kHz — допустимо для музыкальных фрагментов в Интернете;
- 44 kHz — стандартная частота дискретизации. Используемая на компакт-дисках, однако качество звука ниже, чем качество CD, в связи с тем, что данный алгоритм предполагает сжатие звукового файла.

Параметры оптимизации звукового файла, заданные в библиотеке, будут применены ко всем его экземплярам, используемым в фильме.

В табл. 14.1 обобщены основные возможности работы со звуком.

Таблица 14.1. Операции со звуком

Операция	Способ
Импорт	1. Команда File>Import>Import to Library . 2. Команда File>Import>Open External Library ; <Ctrl>+<Shift>+<O>, перетащить звук в библиотеку своего фильма (импорт звука из внешнего документа Flash)
Экспорт	Команда File> Export >Export Movie... ; <Ctrl>+<Alt>+<Shift>+<S>
Оптимизация (сжатие)	Дважды щелкнуть на пиктограмме звука в библиотеке
Добавление звука в кадр	1. Выделить кадр, перетащить звук из библиотеки на сцену. 2. В Инспекторе свойств выбрать звук из списка
Редактирование длительности	Инспектор свойств — кнопка Edit , в окне редактирования звука можно перемещать маркеры на шкале, устанавливая начало и конец звучания
Выбор типа синхронизации	Инспектор свойств — выбрать из поля Sync тип звука

Event — звук, управляемый событием (мыши или наступлением кадра на монтажной линейке), должен быть загружен полностью, прежде чем он начнет воспроизводиться и будет звучать, пока его не выключат. Назначение — служит для озвучивания кнопок, для зацикленных музыкальных фрагментов; воспроизводится полностью; помещается только в один кадр

Start — экземпляр звука, использующий данный тип синхронизации, будет воспроизведен только, если другой экземпляр этого же звука не воспроизводится в настоящий момент

Таблица 14.1 (окончание)

Операция	Способ
	<p>Stop — прекращает воспроизведение одноименного звука</p> <p>Stream — потоковый звук начинает звучать еще до полной загрузки. Служит для синхронизации с анимацией, а также однократного использования звука в фильме, воспроизводится только в кадрах, куда помещен</p>
Управление громкостью	В списке Effect Инспектора свойств выбрать один из стандартных эффектов звука или создать пользовательский эффект в окне редактирования звука Edit Envelope (кнопка Edit)

Использование звука в фильме

Звук может быть использован в фильме для выполнения различных задач, и исходя из этого, необходимо задать соответствующий тип синхронизации, а также параметры компрессии. Поскольку звуковой файл, помещенный в Flash, может очень существенно увеличить его размер, следует стараться применять не слишком продолжительные звуки. При необходимости введения в фильм фонового звукового сопровождения можно использовать сравнительно короткий событийный звук, зациклив его или задав требуемое число повторов. Для того чтобы это сделать, необходимо выполнить следующие действия:

1. Выделить ключевой кадр, содержащий событийный звук;
2. В списке **Sound Loop**, расположенному в нижней части Инспектора свойств, выбрать один из следующих параметров:
 - **Loop** — зациклить воспроизведение звука. Звук будет циклически повторяться на всем протяжении воспроизведения фильма. Не рекомендуется зацикливать потоковые звуки, поскольку это приведет к увеличению объема конечного фильма, кратному числу повторов;
 - **Repeat** — повторить звук. При выборе этого значения звук будет воспроизведен заданное число раз. Задать количество повторов можно при помощи поля **Times Number of times to loop**.

Резюмируя сказанное, можно привести следующие рекомендации по эффективному применению звука.

- Размещайте экземпляры звуков на отдельных слоях, не объединяя их с графикой. Если в документе одновременно используются несколько звуков, можно поместить все содержащие их слои в общую слой-папку с названием **Sound**. Такой подход позволит лучше организовать структуру документа и обеспечит гибкость и удобство работы с ним.

- Корректируйте точки начала и конца звучания экземпляров звуковых файлов, чтобы не сохранять в конечном фильме тишину, которая тоже занимает объем.
- Модифицируйте один и тот же звук, применяя к его экземплярам различные эффекты на основе управления уровнем сигнала (затухание, нарастание, перетекание из канала в канал и т. д.) Таким образом, можно добиться иллюзии воспроизведения различных звуков, используя только один звуковой файл.
- Зацикливайте короткие событийные звуки для фонового сопровождения.
- Не зацикливайте потоковые звуки, старайтесь избегать их повторов.
- Имейте в виду, что параметры потокового звука, содержащегося во внедренном видеоролике, определяются глобальными звуковыми настройками при публикации в диалоговом окне **Publish Settings**.
- Оптимизируйте отдельные звуковые файлы при помощи библиотеки или все используемые звуки сразу при публикации в диалоговом окне **Publish Settings**.

Пример.

Использование звука в кнопках

Звук может быть помещен в кадры монтажной линейки символа типа **Button**, соответствующие определенным состояниям кнопки. Для того чтобы это сделать, необходимо выполнить следующие действия:

1. Создать новый символ типа **Button** (**Insert>New Symbol**).
2. В среде редактирования этого символа создать два слоя. Один из них можно назвать **graphic**, другой — **sound**.
3. На слое **graphic**, создавая ключевые кадры, нарисовать соответствующие состояния кнопки. Можно оставить все кадры, за исключением кадра **Hit** пустыми. Это приведет к созданию озвученной пустой кнопки (*см. разд. "Символ Button" гл. 10*).
4. На слое **sound** создать пустые ключевые кадры для тех состояний, которые должны быть озвучены, и поместить в них требуемые звуки (рис. 14.10). При этом звуки будут воспроизводиться следующим образом:
 - **Up** — звук, помещенный в этот кадр, будет воспроизведен, как только курсор мыши выйдет за пределы области реагирования кнопки. Таким образом, данный звук будет впервые услышан только после выполнения определенных действий с кнопкой;
 - **Over** — звук, помещенный в этот кадр, будет воспроизведен при попадании курсора мыши в область реагирования кнопки;

- **Down** — звук, помещенный в этот кадр, будет воспроизведен при нажатии левой кнопки мыши, в случае если курсор находится в области реагирования кнопки;
- **Hit** — звук, помещенный в этот кадр, будет воспроизведен при отпускании левой кнопки мыши, в случае если курсор находится в области реагирования кнопки.



Рис. 14.10. Монтажная линейка кнопки, содержащей звук

5. Выйти из среды редактирования кнопки и вытащить из библиотеки на сцену необходимое число ее экземпляров.
6. Протестировать кнопку в рабочей среде (**Control>Enable Simple Buttons**) или в среде тестирования (**Control>Test Movie**).

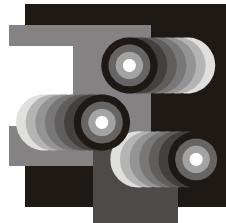
Примечание

В кнопке могут быть использованы только два типа синхронизации: **Event** (Событие) **Start** (Старт). Для того чтобы избежать эффекта микширования, т. е. самоналожения звука при быстром переводе курсора с одного экземпляра озвученной кнопки на другой, нужно применять синхронизацию типа **Start**. В этом случае звук, назначенный определенному состоянию кнопки, будет запущен только в том случае, если его экземпляр не воспроизводится в данный момент.

Экспорт звука

Звук может быть экспортирован из Flash-документа в виде самостоятельного файла в формате WAV (только для Windows). Для того чтобы это сделать, необходимо выполнить команду главного меню **File>Export>Export Movie** и в списке **Save as type** появившегося окна **Export Movie** выбрать тип файла **WAV Audio**. После этого в окне **Export Windows Wav** можно задать параметры экспорта звука.

- Sound format** — данный список позволяет задать значения частоты дискретизации, разрядности звука и число каналов.
- Ignore event sound** — установка данного флагка отключает экспорт событийных звуков.



Глава 15

Работа над фильмом

Техника достигнет такого совершенства,
что человек сможет обходиться без себя.

Станислав Ежи Лец

Работа со сценами

Сцена (Scene) представляет собой элемент структурной организации Flash-фильма. Основная монтажная линейка фильма может быть разбита на несколько фрагментов, называемых сценами (Scenes). Каждая сцена имеет уникальное имя и может содержать неограниченное число слоев и кадров, в пределах которых можно размещать любые объекты рабочей среды и выполнять с ними все доступные операции. В конечном фильме все сцены воспроизводятся последовательно друг за другом в соответствии с порядком, заданным в рабочей среде. Когда воспроизводящая головка достигает последнего кадра монтажной линейки сцены Scene N, она автоматически переходит к первому кадру сцены Scene N + 1. Таким образом, разделение на сцены имеет место только в рабочей среде. В конечном фильме все сцены воспроизводятся как единая монтажная линейка.

Сцена представляет собой изолированную среду, т. е. работа в пределах одной сцены осуществляется совершенно независимо от остальных. Однако все сцены могут непосредственно использовать ресурсы библиотеки (library) документа и обмениваться графическими объектами или целыми кадрами через буфер обмена.

Применение сцен позволяет упорядочить структуру документа и упростить процесс работы с фильмом, содержащим несколько тематических разделов и большое число кадров. Так, например, если проект содержит три раздела: "Introduction", "Main" и "Final", каждый из которых в свою очередь предполагает наличие анимации, интерактивных элементов, звукового сопровождения и т. д., то размещение каждого из разделов в отдельной сцене позволит существенно упростить рабочий процесс. При создании анимационных фильмов в отдельные сцены могут быть помещены различные эпизоды

фильма. Таким образом, применение сцен позволяет легко ориентироваться в рабочей среде и быстро переходить к требуемому эпизоду при необходимости редактирования его содержимого.

При помощи сценариев ActionScript можно легко организовать переход между сценами, остановку воспроизведения монтажной линейки сцены или нелинейное воспроизведение сцен (т. е. переход с одной сцены на другую в произвольном порядке). (*Об этих возможностях будет подробно рассказано в гл. 19.*)

Управление сценами

Для выполнения различных операций, связанных с управлением сценами, служит панель **Scene** (рис. 15.1). Для того чтобы ее открыть, нужно выполнить команду **Window>Design Panels>Scene** или воспользоваться сочетанием клавиш <Shift>+<F2>.

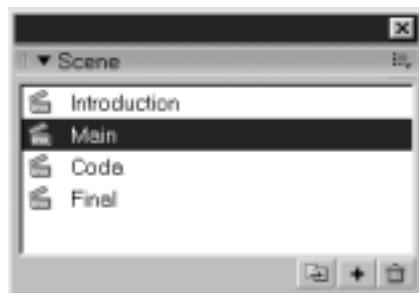


Рис. 15.1. Панель **Scene**

Создание сцен

По умолчанию новый документ Flash содержит только одну сцену с названием **Scene 1**. Для того чтобы создать новую сцену, необходимо в панели **Scene** щелкнуть на кнопке **Add scene** или выполнить команду главного меню **Insert>Scene**. Новая сцена по умолчанию будет названа **Scene N + 1**, где **N** — порядковый номер предыдущей созданной сцены. Для того чтобы переименовать сцену, необходимо дважды щелкнуть на ее названии в панели **Scene** и ввести новое имя. Имена сцен не могут совпадать, т. е. каждая сцена должна иметь свое уникальное название.

Дублирование сцен

Для того чтобы скопировать сцену, необходимо выделить ее в панели **Scene** и щелкнуть на кнопке **Duplicate scene** . В результате будет создана новая сцена с копией содержимого оригинала.

Изменение порядка следования сцен

Для того чтобы изменить последовательность воспроизведения сцен в фильме, нужно изменить порядок их размещения в панели **Scene**, перетащив требуемую сцену в соответствующее положение.

Переход между сценами в рабочей среде

Для того чтобы в рабочей среде перейти на требуемую сцену, можно либо щелкнуть на ее имени в панели **Scene**, либо щелкнуть на кнопке  расположенной в полосе редактирования, и из появившегося перечня всех имеющихся в документе сцен выбрать требуемую.

Перейти с одной сцены на другую в рабочей среде также можно при помощи Проводника фильма **Movie Explorer** (см. далее в этой главе).

Удаление сцен

Для того чтобы удалить сцену, необходимо выделить ее в панели **Scene** и щелкнуть на кнопке **Delete scene** , расположенной в нижней части панели.

Использование макросов

Flash MX 2004 предоставляет достаточно мощные возможности по автоматизации рабочего процесса. При необходимости многократного выполнения последовательности однотипных операций можно записать их в виде макроса и воспроизвести в любом Flash-документе. Появившийся в Flash MX 2004 объектно-ориентированный язык JSFL (JavaScript Flash) позволяет сохранять набор произведенных действий в виде сценария во внешнем файле с расширением jsfl, а также "с нуля" создавать сценарии, используемые в рабочей среде для выполнения различных процедур. Так, например, все встроенные эффекты монтажной линейки (см. гл. 13) созданы при помощи языка JSFL.

Последовательность действий, записанная в виде набора инструкций JSFL, в терминах Flash называется **командой** (Command) или макросом. Создать макрос можно при помощи панели **History** (рис. 15.2), для открытия которой нужно выполнить команду главного меню **Window>Other Panels>History** или использовать сочетание клавиш <Ctrl>+<F10>. Панель **History** отображает всю последовательность выполненных действий, или *шагов* (steps) и может быть использована для отмены предыдущих действий (*об этих возможностях рассказывается в разд. "Палитра History" гл. 3*). Каждое действие имеет свою пиктограмму и текстовый комментарий. Не все действия, выполняемые в рабочей среде, могут быть сохранены в качестве макроса. Пиктограммы таких действий в панели **History** содержат значок красного крестика. К числу таких действий, например, относятся следующие: создание

произвольной векторной формы инструментом **Brush** или контура инструментом **Pen**, изменение размеров рабочей области, заливка градиентом, выделение кадров монтажной линейки и др.

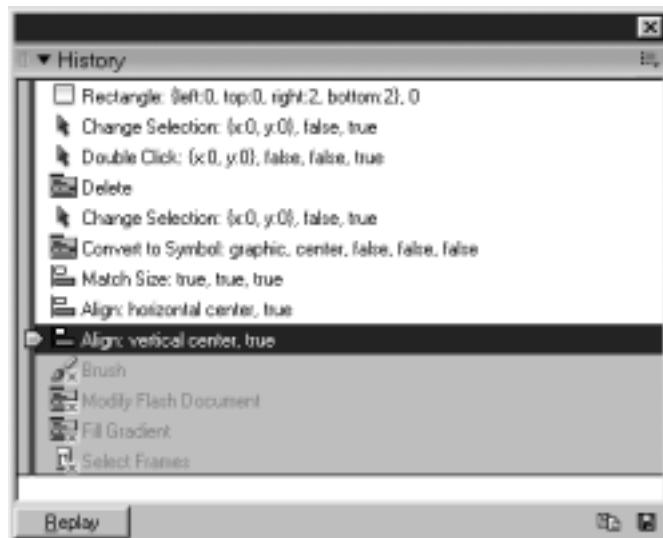


Рис. 15.2. Панель **History**

Контекстное меню панели **History** содержит ряд команд, управляющих последовательностью выполненных действий.

- **Replay Steps** — данная команда позволяет повторно выполнить набор выделенных действий и применить их к любому выделенному объекту сцены. Для того чтобы выделить диапазон последовательных действий в панели **History**, можно либо протащить по ним курсор, либо щелкнуть на первом действии диапазона, а затем, удерживая клавишу <Shift>, — по последнему. Для того чтобы выделить действия вразбивку, необходимо использовать клавишу <Ctrl>. Команда **Replay Steps** повторяет все выделенные действия вне зависимости от положения вертикального слайдера. Выполнить данную команду также можно при помощи клавиши **Replay**, расположенной слева внизу панели. Команда **Replay Steps** неприменима к действиям, которые не могут быть сохранены в качестве макроса.
- **Copy Steps** — данная команда позволяет скопировать в буфер выделенную последовательность действий, для того чтобы впоследствии применить их к объекту другого документа Flash или вставить в текстовый редактор как программный код JSFL. Чтобы применить скопированный набор действий к объекту другого документа, необходимо выделить этот

объект и выполнить команду главного меню **Edit>Paste in Place** или команду **Edit** контекстного меню самого объекта. При вставке скопированных действий в текстовый редактор (например, MS Word) они будут вставлены как набор инструкций JSFL. Данную команду также можно выполнить, щелкнув на пиктограмме **Copy selected steps to the clipboard** . Действия, которые не могут быть сохранены в качестве макроса, скопировать нельзя.

- **Save as Command** — данная команда позволяет создать макрос на основе выделенных действий. В результате ее выполнения появляется одноименное диалоговое окно, предлагающее задать имя макроса. Макрос сохраняется в виде внешнего файла JSFL, имя которого совпадает с именем макроса. Данный файл по умолчанию помещается в папку конфигурации, адрес которой может выглядеть следующим образом: C:/WINDOWS/Application Data/Macromedia/Flash MX 2004/en/Configuration/Commands. Доступ ко всем макросам, помещенным в эту папку, может быть получен из любого документа Flash при помощи команды главного меню **Commands>Run Command**. Данную команду также можно выполнить, щелкнув на пиктограмме **Save selected steps as a Command** .
- **View** — данная команда позволяет выбрать один из режимов отображения комментариев, расположенных рядом с пиктограммой, соответствующей тому или иному действию в панели **History**.
 - **Default** — значение по умолчанию. Рядом с пиктограммой указывается содержание выполненного действия. При подведении курсора к названию данной операции появляется всплывающая подсказка (tooltip), содержащая название операции и аргументы соответствующих функций JSFL, используемых для ее реализации. Если действие не может быть описано посредством языка JSFL, аргументы отсутствуют.
 - **Arguments in Panel** — рядом с пиктограммой указывается содержание выполненного действия и аргументы соответствующих функций JSFL.
 - **JavaScript in Panel** — рядом с пиктограммой выводится набор инструкций JSFL, используемых для реализации данного действия.
 - **Argument in Tooltip** — при подведении курсора к названию операции всплывающая подсказка содержит название операции и аргументы соответствующих функций JSFL.
 - **JavaScript in Tooltip** — при подведении курсора к названию операции всплывающая подсказка содержит набор инструкций JSFL, используемых для реализации данного действия.
- **Clear History** — данная команда очищает панель истории, удаляя всю информацию о предыдущих действиях.

Для того чтобы применить сохраненный макрос в любом документе, включая текущий, необходимо выполнить команду **Commands** и в нижней части списка выбрать любой из сохраненных макросов. Макрос может быть создан на основе выполненных действий при помощи панели **History** или создан вручную в любом текстовом редакторе и сохранен с расширением jsfl. Листинг 15.1 содержит пример сценария, позволяющего создать графический символ, размеры которого будут соответствовать размерам рабочей области текущего документа, и, поместив его экземпляр на сцену, отцентрировать его относительно рабочей области. Две косые черты "//" используются для создания комментариев, т. е. произвольного текста, который игнорируется интерпретатором. Такой макрос позволит быстро создать пользовательский фон (изменять его цветовые характеристики можно при помощи цветовых эффектов экземпляра) или маску, используемую для блокировки отображения содержимого вспомогательной области. Приведенный в листинге 15.1 код был сгенерирован автоматически, а затем отредактирован в текстовом редакторе. Внешний вид панели истории при создании данного макроса представлен на рис 15.2.

Листинг 15.1. Программный код JSFL для создания пользовательского фона

```
// setBg – Создание пользовательского фона
//Макрос, сгенерирован Macromedia Flash MX 2004
//Создать квадрат размером 2x2 пикселя с обводкой (1, solid)
fl.getDocumentDOM().addNewRectangle({left:0, top:0, right:2, bottom:2},
    0);
//Выделить обводку
fl.getDocumentDOM().mouseDblClk({x:1, y:1}, false, false, true);
//Удалить обводку
fl.getDocumentDOM().deleteSelection();
//Выделить заливку
fl.getDocumentDOM().mouseClick({x:0, y:0}, false, true);
//Конвертировать в графический символ с точкой регистрации в центре
fl.getDocumentDOM().convertToSymbol('graphic', '', 'center');
var lib = fl.getDocumentDOM().library;
if (lib.getItemProperty('linkageImportForRS') == true) {
    lib.setItemProperty('linkageImportForRS', false);
}
else {
    lib.setItemProperty('linkageExportForAS', false);
    lib.setItemProperty('linkageExportForRS', false);
}
```

```
//Привести размер экземпляра к размерам рабочей области  
fl.getDocumentDOM().match(true, true, true);  
//Поместить в центр рабочей области  
fl.getDocumentDOM().align('horizontal center', true);  
fl.getDocumentDOM().align('vertical center', true);
```

Для того чтобы выполнить любой внешний сценарий JSFL, находящийся в другой директории, нужно выполнить команду главного меню **Commands>Run Command**. После этого необходимо указать путь к требуемому файлу.

Для организации сохраненных в стандартном каталоге Commands макросов применяется команда **Commands>Manage Saved Commands**. Ее выполнение приводит к появлению одноименного диалогового окна, позволяющего переименовать (**Rename**) требуемый макрос или удалить (**Delete**) его. Удаление макроса в данном диалоговом окне влечет за собой удаление соответствующего JSFL-файла из каталога Commands.

Команда главного меню **Commands>Get More Commands** позволяет связаться с разделом сайта компании Macromedia www.macromedia.com/go/flash_exchange и загрузить макросы, созданные другими Flash-разработчиками.

Команды *Find/Replace*

В Flash MX 2004 имеются новые возможности поиска и замены элементов рабочей среды, облегчающие процесс редактирования документа. При помощи функций **Find** и **Replace** можно быстро найти и заменить текстовую строку, шрифт, цвет, экземпляр символа, звук, видео или растровое изображение, используемые в документе.

Поиск и замена объектов могут осуществляться как в пределах одной сцены, так и во всем документе. Кроме того, функция поиска позволяет отредактировать найденный объект, организуя быстрый доступ к нему.

Поиск и замена требуемых элементов выполняются при помощи панели **Find and Replace**, которую можно открыть командой главного меню **Edit>Find and Replace** (рис. 15.3).

Панель **Find and Replace** состоит из следующих элементов:

- Search in** — данный список служит для указания области, в которой осуществляется поиск: **Current Document** (Весь документ) или **Current Scene** (Текущая сцена);
- For** — данный список служит для выбора объекта или параметра поиска и замены;

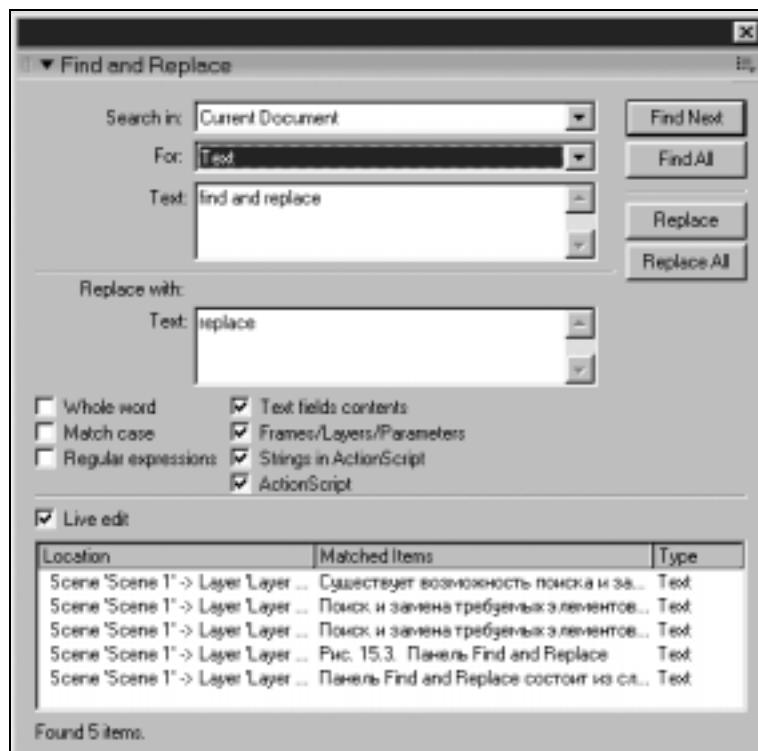


Рис. 15.3. Панель **Find and Replace**.
Поиск текста

- **Replace with** — данный список служит для выбора объекта или параметра, который должен заменить найденный объект или параметр;
- **Find Next** — переход к следующему объекту или параметру, удовлетворяющему условиям поиска;
- **Find All** — найти сразу все объекты или параметры, удовлетворяющие условиям поиска;
- **Replace** — заменить объект или параметр, удовлетворяющий условиям поиска, объектом или параметром, указанным в списке **Replace with**;
- **Replace All** — заменить все объекты или параметры, удовлетворяющие условиям поиска, объектом или параметром, указанным в списке **Replace with**;
- **Live edit** — данная опция позволяет получить доступ к редактированию найденного объекта или параметра. Она работает только при использовании команды **Find Next**;

- **Find and Replace Log** — поле в нижней части панели служит для вывода результатов поиска. В столбце **Location** указывается местоположение найденного объекта, в столбце **Matched Items** — имя найденного объекта или значение параметра, в столбце **Type** приводится тип найденного элемента.

Поиск и замена текстовой строки (*Text*)

Данную функцию можно использовать для поиска текстового фрагмента в различных элементах документа и замены его на другой текст. Для задания условий поиска текста используются следующие параметры (см. рис. 15.3):

- **Text** — данное поле служит для введения образца текста, который требуется найти;
- **Whole word** — поиск целого слова. При установке этого флажка будет производиться поиск фрагмента текста, ограниченного пробелами или знаками пунктуации, в точности совпадающего с текстом, введенным в поле **Text**. Так, например, при поиске слова *Rock* будут найдены слова *(Rock)* и *"Rock"*, но не *Rock 'n' Roll*. При снятии данного флажка могут быть найдены все случаи вхождения данного слова (или фрагмента), в том числе и как части другого слова. Так, например, при поиске слова *Rock* будут найдены слова *ArtRock* и *Rock 'n' Roll*;
- **Match case** — установка данного флажка приведет к тому, что при поиске будет учитываться регистр (прописные или строчные буквы). Так, например, при поиске слова *ROCK*, слова *rock* или *Rock* найдены не будут;
- **Regular expressions** — установка данного флажка позволяет осуществлять поиск текста в регулярных выражениях ActionScript;
- **Text field contents** — при установке данного флажка будет осуществляться поиск текста в текстовых блоках;
- **Frames/Layers/Parameters** — при установке данного флажка поиск текста будет осуществляться в метках кадров, именах слоев, параметрах компонентов;
- **Strings in ActionScript** — поиск текста осуществляется в строках сценариев ActionScript. Под строкой (*String*) понимается строковый литерал, т. е. произвольная последовательность символов, заключенная в кавычки;
- **ActionScript** — поиск в сценариях ActionScript.

Установка флажка **Live edit** позволяет сразу войти в режим редактирования найденного текста. Для выполнения этой функции при поиске или замене нужно использовать кнопки **Find Next** и **Replace** соответственно.

Кроме того, войти в режим редактирования найденного текста, находящегося в текстовом блоке, можно, дважды щелкнув на содержащей его строке в поле **Find and Replace Log** в нижней части панели.

Поиск и замена шрифта (*Font*)

Данную функцию можно использовать для поиска шрифта требуемого размера и начертания и замены его на другой шрифт. Для задания условий поиска шрифта используются следующие параметры (рис. 15.4):

- Font name** — данный список позволяет выбрать шрифт, который требуется найти (раздел **Search in**), и шрифт, который заменит найденный (раздел **Replace with**);
- Font style** — данный список позволяет выбрать тип начертания шрифта, который требуется найти (раздел **Search in**), и шрифта, который заменит найденный (раздел **Replace with**);
- Size** — данный список позволяет указать диапазон размеров шрифта, используемый при поиске (**Min** — наименьший, **Max** — наибольший), в разделе **Search in** и задать размер шрифта в разделе **Replace with**, которым будет заменен найденный шрифт.

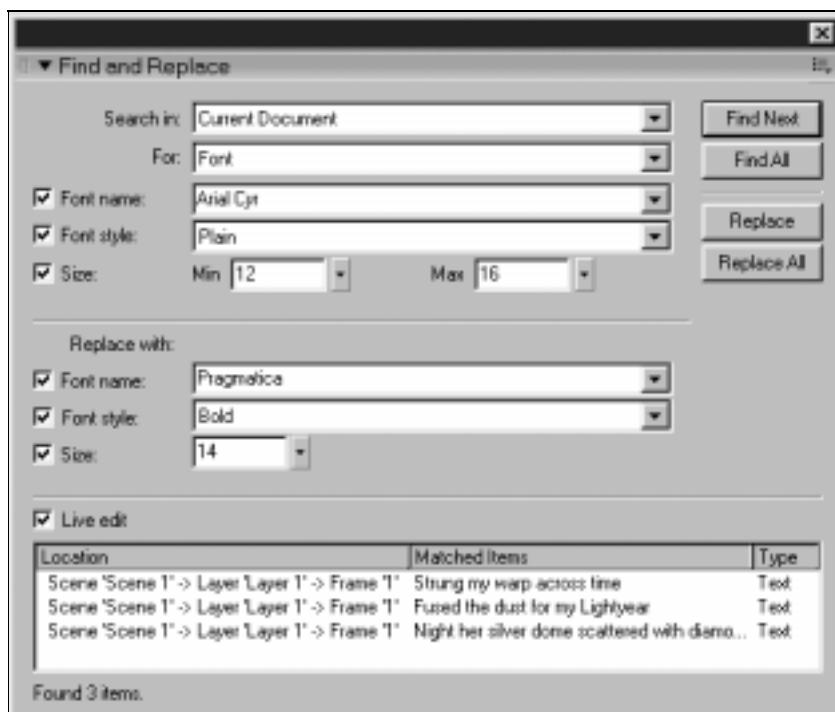


Рис. 15.4. Панель **Find and Replace**.
Поиск шрифта

Поиск и замена цвета (Color)

Данную функцию можно использовать для поиска и замены сплошного цвета. Поиск цвета не может быть применен к сгруппированным векторным объектам, однако он выполняется для экземпляров символов. Для задания условий поиска цвета используются следующие параметры (рис. 15.5):

- Color** — щелчок на цветовом образцу в разделе (**Search in**) позволяет выбрать цвет, который требуется найти или заменить, из текущего каталога цветов; задать его шестнадцатеричное значение либо синтезировать при помощи стандартного окна синтеза цвета **Color**. Кроме того, можно в качестве образца "подобрать" цвет с любой точки экрана, если, не отпуская кнопки мыши, переместить курсор, принявший вид пипетки, в соответствующую точку. Цветовой образец в разделе **Replace with** позволяет аналогичным образом задать цвет, который должен заменить найденный;
- Fills** — поиск цвета заливок (векторных форм);
- Strokes** — поиск цвета обводок (контуров);
- Text** — поиск цвета текста.

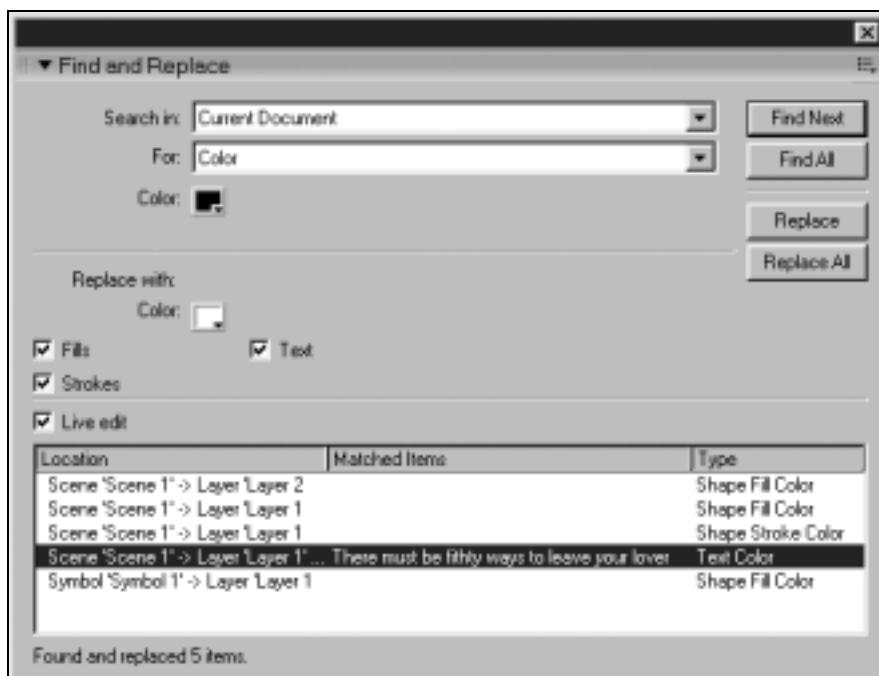


Рис. 15.5. Панель **Find and Replace**.
Поиск цвета

Поиск и замена экземпляра символа (*Symbol*), звука (*Sound*), видео (*Video*) или растрового изображения (*Bitmap*)

Данную функцию можно использовать для поиска и замены экземпляров символов и импортированных объектов. В списке **For** в качестве объекта поиска нужно выбрать требуемый элемент: **Symbol** (экземпляр символа), **Sound** (экземпляр звука), **Video** (экземпляр видеоролика) или **Bitmap** (экземпляр растрового изображения) (рис. 15.6). Для поиска экземпляра необходимо выбрать его имя в списке **Name** раздела **Search in**. Для указания экземпляра, который заместит найденные экземпляры, необходимо выбрать его имя в списке **Name** раздела **Replace with**.

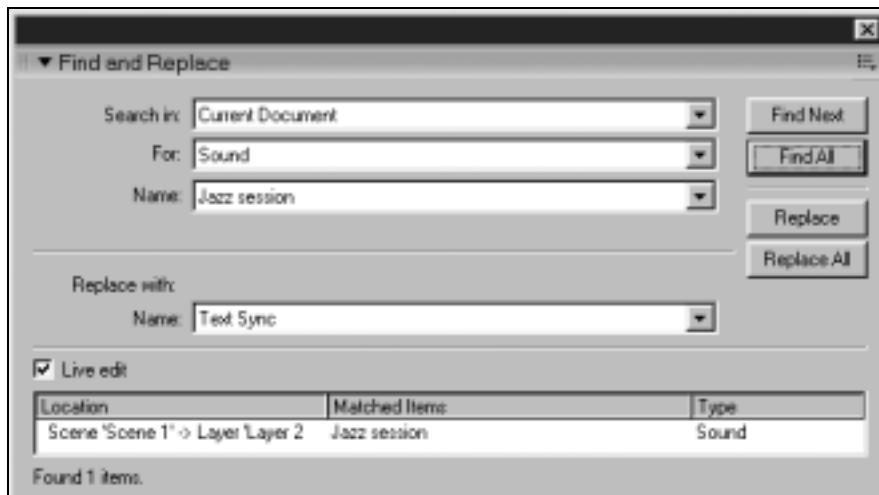


Рис. 15.6. Панель **Find and Replace**.
Поиск звука

Использование Проводника *Movie Explorer*

Проводник фильма **Movie Explorer** является мощным и удобным инструментом при работе с большими проектами, обладающими сложной и разветвленной структурой. С его помощью можно быстро выполнять целый ряд различных операций. Проводник **Movie Explorer** отображает структуру документа в виде графической схемы, позволяя получить наглядное представление о его организации, и обеспечивает быстрый доступ к требуемому эле-

менту фильма. Элементы, используемые в фильме, представлены в окне Проводника в виде иерархического дерева. Проводник позволяет отображать графические элементы, звуки и сценарии ActionScript, а также структуру монтажной линейки основного фильма и монтажных линеек используемых в нем символов. Существует возможность отображать только требуемые элементы, входящие в состав сцен или символов. При помощи Проводника можно быстро отредактировать требуемый элемент вне зависимости от уровня его вложенности.

Для того чтобы открыть панель Проводника фильма, необходимо выполнить команду главного меню **Window>Other Panels>Movie Explorer** или воспользоваться сочетанием клавиш **<Alt>+<F3>**. Панель **Movie Explorer** представлена на рис. 15.7.



Рис. 15.7. Панель **Movie Explorer**

Панель **Movie Explorer** содержит контекстное меню, включающее различные команды для управления элементами, отображаемыми в окне данной панели. Кроме того, щелчок правой кнопкой мыши на соответствующем элементе дерева приводит к появлению контекстного меню, аналогичного контекстному меню панели.

Настройка отображения структуры документа

В окне Проводника могут одновременно отображаться различные элементы документа. Каждый элемент имеет свою характерную пиктограмму, по которой его можно быстро идентифицировать. Рядом с пиктограммой элемента указывается его имя и дополнительные сведения (например, имя экземпляра символа или метка кадра). В структурном дереве Проводника фильма могут быть отображены следующие элементы:

- сцены 
- слои 
- кадры 
- текст 
- экземпляры символов типа **Graphic** , **Movie Clip** , **Button** 
- экземпляры растровых изображений 
- экземпляры внедренных и связанных видеофайлов и видеороликов , 
- экземпляры звуков 
- сценарии ActionScript 

Примечание

Объекты рабочего уровня Проводником не отображаются.

Проводник позволяет отображать две различные категории организации элементов: **Movie elements** (Содержимое сцен) и **Symbol definition** (Содержимое символов). Содержимое сцен отображается в виде дерева, включающего в себя элементы, непосредственно размещенные в кадрах монтажной линейки основного фильма. При этом содержимое монтажных линеек символов не отображается. Содержимое символов отображается в виде дерева, включающего в себя все элементы, находящиеся на монтажных линейках символов, экземпляры которых используются в фильме, включая вложенные символы. Обе категории могут использоваться одновременно, отображая

содержимое в различных областях Проводника. В этом случае вверху находится дерево (или несколько деревьев), отображающее содержимое сцен (оно помечено пиктограммой), под ним находится дерево, отображающее содержимое символов (оно помечено пиктограммой).

Отображение элементов структуры

В верхней части панели **Movie Explorer** находятся клавиши, позволяющие управлять отображением соответствующих объектов в окне Проводника. Для отображения требуемых элементов одновременно можно использовать любые сочетания этих клавиш. Щелчок на крайней справа пиктограмме приводит к появлению диалогового окна **Movie Explorer Settings** (рис. 15.8), где в разделе **Show** можно указать элементы, отображаемые в окне Проводника фильма. Раздел **Context** позволяет выбрать категорию отображения содержимого сцен и/или символов. Данный режим отображения также может быть установлен при помощи команд **Show Movie elements** и **Show Symbol definition** контекстного меню панели **Movie Explorer** или соответствующих команд контекстного меню элемента структуры, которое появляется при щелчке на нем правой кнопкой мыши.



Рис. 15.8. Диалогового окна **Movie Explorer Settings**

Команда **Show All Scenes** контекстного меню Проводника или элемента структуры позволяет отобразить в окне содержимое всех сцен фильма.

Строка **Find**, расположенная в верхней части окна Проводника, позволяет осуществлять поиск элемента фильма по названию. Для того чтобы найти требуемый элемент, необходимо ввести в данную строку его название (название сцены, слоя, имя символа или его экземпляра, номер или метку кадра, название шрифта, название импортированного объекта, фрагмент

сценария); при этом нужно убедиться, что режим отображения искомого элемента включен. Данная функция действует непосредственно при вводе значения в строку **Find**. Для того чтобы отключить поиск элементов, нужно удалить из этой строки все содержимое. В противном случае Проводник будет отображать только объекты, удовлетворяющие условиям поиска.

Пиктограмма "+" рядом с элементом указывает на то, что он содержит вложенные элементы. Для того чтобы увидеть вложенные элементы, необходимо развернуть соответствующую ветвь. Развернуть ветвь можно, щелкнув на пиктограмме с плюсом или выполнив команду контекстного меню (панели или объекта) **Expand Branch**. Соответственно, свернуть ветвь можно, щелкнув на пиктограмме "-" или выполнив команду контекстного меню **Collapse Branch**. Команда контекстного меню панели или элемента **Collapse Others** позволяет свернуть все ветви, кроме текущей.

Редактирование элементов документа

Контекстное меню панели **Movie Explorer**, помимо рассмотренных выше команд, содержит ряд команд, позволяющих выполнять различные операции с элементами фильма, отображаемыми в окне Проводника. Эти команды также можно выполнить при помощи контекстного меню самого элемента, щелкнув на нем правой кнопкой мыши.

Для выделения элемента, отображаемого в Проводнике, можно щелкнуть на нем, а для выделения нескольких элементов вразброс нужно воспользоваться клавишей <Ctrl>. Выделить диапазон элементов можно, протянув через него курсор или при помощи клавиши <Shift>.

Примечание

Проводник **Movie Explorer** не может выполнять никакие операции с графическими элементами фильма, являющимися членами вложенных групп (nested groups).

Рассмотрим эти команды:

- Go to Location** — позволяет перейти к сцене, слою и кадру, в которых содержится выделенный элемент;
- Go to Symbol Definition** — позволяет перейти в раздел **Symbol Definition** к ветви, отображающей содержимое выделенного символа (режим **Show Symbol Definition** должен быть включен);
- Select Symbol Instances** — может быть применена к символу, находящемуся в разделе **Symbol Definition**, экземпляры которого расположены непосредственно на основной монтажной линейке. В результате выполнения команды все экземпляры данного символа, находящиеся на текущей сцене,

- будут выделены в разделе Проводника **Movie elements** (режим **Show Movie elements** должен быть включен) и, соответственно, на сцене;
- **Find in Library** — выделяет текущий элемент в библиотеке фильма (Library). Если панели Library нет на экране, она будет открыта автоматически;
 - **Rename** — позволяет переименовать элемент. При изменении имени символа или импортированного объекта соответствующие изменения будут внесены в библиотеку. Применение данной команды к текстовому блоку позволит отредактировать его содержимое. Новый текст может быть введен непосредственно в соответствующую строку прямо в Проводнике;
 - **Edit in Place/Edit in new Window** — команды позволяют отредактировать выделенный в Проводнике символ в контексте сцены или в новом окне;
 - **Copy All Text to Clipboard** — копирует иерархический список всех отображаемых в Проводнике элементов структуры в буфер. Скопированные данные можно затем вставить во внешний текстовый редактор для обработки (например, для проверки правописания при использовании русского языка);
 - **Cut, Copy, Paste, Clear** — данные команды используются для вырезания, копирования, вставки и удаления элементов из окна Проводника и из их местоположения в фильме. Действие команд распространяется только на экземпляры, а не на родительские объекты (master objects), находящиеся в библиотеке;
 - **Print** — выводит на печать содержимое Проводника.

При выделении элементов, находящихся в одном и том же кадре, в области элементов фильма (**Movie elements**) щелчками мыши, они выделяются на сцене при условии, что они не сгруппированы.

Для того чтобы попасть в среду редактирования символа, находящегося на любом уровне вложенности, достаточно дважды щелкнуть на названии любого экземпляра этого символа в панели **Movie Explorer**. Эта возможность не может быть реализована в случае, если данный экземпляр является членом вложенной группы.

Для того чтобы отредактировать содержимое текстового блока и задать параметры его форматирования, можно дважды щелкнуть на соответствующем элементе в панели **Movie Explorer**. Новый текст может быть введен непосредственно в Проводнике, а все параметры форматирования заданы при помощи инспектора свойств. Если текстовый блок сгруппирован, то он не может быть форматирован указанным способом.

В окне Проводника фильма можно непосредственно просмотреть используемые в фильме сценарии ActionScript. Для того чтобы отредактировать

сценарий, необходимо дважды щелкнуть на соответствующей строке в панели **Movie Explorer**. В результате данный сценарий будет выведен в автоматически открытой панели **Actions**.

Оптимизация фильма

При разработке проекта, предназначенного для размещения в Интернете, размер конечного файла является критическим параметром. Для уменьшения размера фильма и минимизации времени его загрузки следует пользоваться следующими рекомендациями.

- Ограничивайте использование сложных стилей контуров (пунктир, прерывистый и т. д.).
- Применяйте **Modify>Shape>Optimize** или команды **Modify>Shape>Smooth**, **Modify>Shape>Straighten**, чтобы минимизировать число опорных точек, которые используются для описания векторной формы. Избегайте использования большого числа сложных векторных форм (например, полученных в результате применения команд **Modify>Shape>Softens Fill Edges** или **Modify>Bitmap>Trace Bitmap**).
- Ограничьте использование градиентных и растровых заливок. Заполнение области градиентной заливкой требует памяти приблизительно на 50 байт больше, чем заполнение сплошным цветом.
- Импортируйте растровые изображения в рабочую среду Flash, предварительно приведя их к минимально допустимым размеру и разрешению.
- Оптимизируйте растровую графику в среде библиотеки.
- По возможности избегайте анимации растровых элементов, используйте растровые изображения как фоновые или статичные элементы.
- Применяйте символы для каждого элемента, который встречается в фильме более одного раза. Аналогичным образом следует поступать и с многократно используемыми фрагментами анимации. Для видоизменения экземпляров можно использовать операции трансформации и цветовые эффекты.
- Ограничивайте число шрифтов, используемых в фильме.
- По возможности используйте машинонезависимые шрифты (device fonts).
- Избегайте анимации больших фрагментов текста.
- Для динамического (Dynamic) и пользовательского (Input) текста встраивайте очертания только тех символов шрифта, которые будут использованы в фильме.

- Отделяйте статику от динамики при помощи слоев.
- Используйте вложенные символы для повторяющихся элементов движения при разработке сложных анимационных процессов.
- Страйтесь избегать одновременного выполнения множества сложных анимационных действий при помощи автоматической анимации. Это может привести к снижению скорости воспроизведения на менее мощных машинах.
- Не используйте продолжительные звуковые файлы и видеоролики.
- Оптимизируйте звуки в среде библиотеки фильма.
- Разбивайте проект на несколько тематических разделов, сохраняя каждый раздел в отдельном документе Flash.
- Сжимайте (Compress) файл при публикации.

В табл. 15.1 содержится краткое описание типовых средств дизайна, применяемых при разработке интернет-проектов.

Таблица 15.1. Средства дизайна

Средства дизайна	Возможности использования
Цветовая схема	Цветовая схема должна повторяться во всех разделах (страницах) проекта, это создаст у посетителя ощущение связности. Необходимо тщательно проработать цветовое решение, придать цветам самостоятельное звучание и вместе с тем организовать цветовую поддержку остальных элементов композиции
Шрифт	Гарнитура (семейство шрифтов), кегль (высота), цвет. Принцип контраста: сочетание различных начертаний (жирное и курсивное), различных типов шрифтов (рубленых – sans serif и с зачеками – serif). Анимационные эффекты
Размещение текста на странице	Верстка текста в несколько колонок, цветные подложки под текст – он выделяется на общем фоне страницы, акцентируя на себе внимание, рамка вокруг фрагмента текста, применение фоновых рисунков
Шапка страницы	Следование определенному стилю в заголовках, вставка логотипа, основного меню, динамических элементов, использование разделителей и линеек
Конец страницы	Ссылки на авторские права, логотип разработчика, почтовые адреса
Списки, таблицы, меню, ссылки	Стилизация, подбор типовых элементов дизайна (иконки, элементы оформления текста – маркеры (bullets), разделители параграфов или страниц, рамки), цветовое оформление, динамические элементы дизайна

Таблица 15.1 (окончание)

Средства дизайна	Возможности использования
Растровые изображения в качестве иллюстраций	Перед использованием следует обработать их — сделать в случае необходимости тоновую и цветовую коррекцию, кадрирование, привести к требуемому размеру, обработать края фотографии (рамки, форма), данное оформление используется на протяжении всего проекта. Можно применить операцию трансформации, добиваясь стилизованного декоративного эффекта. Применение эффектов анимации, использование масок, изменение прозрачности
Панель навигации	Для удобной навигации по проекту разрабатывается панель навигации — набор кнопок с пиктограммами и подписями, а также специальный раздел (страница) — карта сайта. Дублирование навигационной системы на всех разделах, возможность перехода на главную страницу из любого раздела
Анимационные ролики	Создают акцент, могут быть использованы в качестве начальной страницы (splash) или в каждом тематическом разделе проекта. Должны отражать общую идею и стиль

В табл. 15.2 содержится перечень основных средств разработки Flash, используемых при создании интернет-проектов.

Таблица 15.2. Обзор основных средств разработки Flash

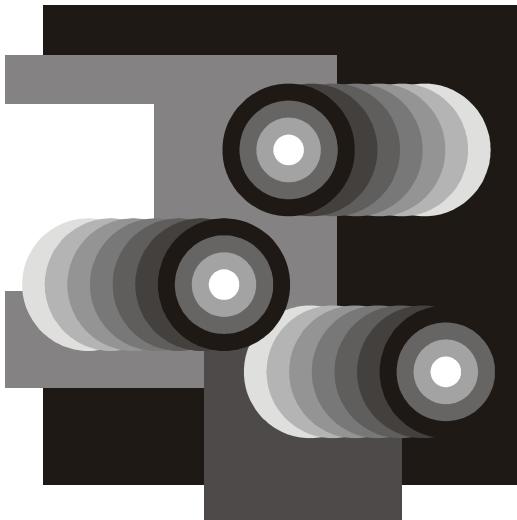
Средство	Применение	Преимущества
Библиотека	Средство создания, хранения и доступа к символам и импортированным файлам, реализующее возможность их многократного использования в проекте в качестве экземпляров. Содержит все импортированные в документ объекты (растровую графику, звук и видео)	
а) графический символ	Создание типовых элементов дизайна (рамок, пиктограмм, логотипов и других статических элементов), используемых в фильме более одного раза 	Применение символов позволяет существенно оптимизировать размер конечного файла, поскольку многократное включение их экземпляров в очень несущественной степени влияет на конечный объем фильма.
б) кнопка	Интерактивный элемент, способный реагировать на действия пользователя, применяемый для реализации взаимодействия и для инициирования определенных действий. Используется для создания навигации по проекту, управления анимацией, манипуляций с объектами и т. д. Обрабатывается программно 	Обмен символами между документами позволяет использовать ранее созданную графику, анимацию и интерактивность в новых проектах

Таблица 15.2 (продолжение)

Средство	Применение	Преимущества
в) муви-клип	<p></p> <p>Создание повторяющихся элементов анимации, которые воспроизводятся независимо от основной монтажной линейки фильма (мини-фильм). Разработка программной анимации, обеспечение интерактивности. Содержит звук, программный код, другие муви-клипы, кнопки, графику. Обрабатывается программно</p>	
Экземпляры:		
а) символов любых типов из библиотеки	<p>При редактировании экземпляров применяются цветовые эффекты, операции трансформации: масштабирование, поворот, скос и зеркальное отражение. Это дает возможность значительно разнообразить внешний вид экземпляров</p>	<p>Используя только один символ и видоизменяя его экземпляры на сцене, можно добиться значительного разнообразия типовых элементов дизайна на странице.</p> <p>Осуществление принципа: <i>"один объект — много применений"</i></p>
б) растровой графики и текста		
в) звука	<p></p> <p>Озвучивание событий и кнопок. Создание фонового звукового сопровождения. Синхронизация звука с анимацией, озвучивание персонажей</p>	
Ключевой кадр	<p>Ключевой кадр может содержать элементы дизайна целой web-страницы (включая панель навигации, типовые элементы дизайна, текст, растровую графику, анимированные элементы). Навигация (переход между разделами) осуществляется при помощи кнопок, инициирующих переходы между соответствующими кадрами</p>	<p>В одном документе Flash теоретически может содержаться целый динамический Web-проект. При этом можно не прибегать к использованию других программных средств, таких как редакторы HTML и языки программирования для создания динамических Web-страниц (JavaScript, VBScript)</p>
Слой		
а) как средство организации структуры и разделения статических и динамических элементов	<p>Упорядочивание элементов для более рационального управления содержимым документа. Различные элементы размещаются по слоям в соответствии со своими функциями. Разделение одновременно выполняемых анимационных действий</p>	<p>Рациональная организация документа, дающая гибкость при работе с элементами фильма. Возможность реализации различных эффектов (маска, движение по траектории и т. д.). Создание сложных анимационных процессов</p>

Таблица 15.2 (окончание)

Средство	Применение	Преимущества
б) направляю- щий слой	Служит для создания макетов, шаблонов, используемых для позиционирования объектов на других слоях	Содержимое данного слоя не экспортится в конечный фильм, поэтому не требует удаления перед каждой публикацией
Динамический и пользо- вательский текст	Используются для организации взаимодействия с пользователем, обмена информацией, для вывода информации как реакции на выполненные действия или протекающие в ходе воспроизведения фильма процессы	Обеспечение высокой степени интерактивности. Разработка разнообразных динамических эффектов
Язык сценари- ев ActionScript	Используется для организации навигации, загрузки внешних ресурсов, создания программной анимации, при создании сложных динамических эффектов, при разработке комплексных схем интерактивного взаимодействия, для обеспечения взаимодействия с клиентским и серверным сце-нариями	Обладает широкими возможностями, позволяющими снабдить проект высокой степенью инте-рактивности

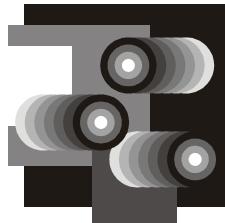


ЧАСТЬ III

ТЕСТИРОВАНИЕ И ПУБЛИКАЦИЯ ПРОЕКТА

Разработка проекта и работа с документом осуществляются в рабочей среде. На этой стадии бывает трудно оценить, насколько функционирование конечного проекта будет соответствовать ожидаемым результатам. Тестирование позволяет оценить функциональность конечного фильма и его отдельных элементов.

Публикация представляет собой процедуру, в результате которой исходный документ транслируется в один или сразу в несколько результирующих форматов, предназначенных для распространения в Интернете или иных средах. Таким образом, тестирование и публикация — это два взаимосвязанных процесса, позволяющих оценить работоспособность проекта и преобразовать его в формат, предназначенный для целевой аудитории.



Глава 16

Тестирование фильма

Чтобы понять что-нибудь, нужно это сделать.

Софокл

Тестирование позволяет оценить функционирование конечного фильма и всех его элементов, а также получить представление о том, как будет выглядеть процесс загрузки данного фильма по сети. Тестирование — это чрезвычайно важный этап разработки Flash-проекта, поскольку оно дает представление о том, каким образом данный проект будет выглядеть для конечного пользователя. Тестирование позволяет проанализировать, насколько эффективно организована структура проекта, и выявить проблемные места. В процессе разработки часто возникает необходимость тестировать отдельные элементы фильма на предмет соответствия поставленным задачам результатов их функционирования. Перед публикацией проекта необходимо тщательно протестировать его и убедиться в его полной работоспособности и отсутствии ошибок и неполадок.

Тестирование в рабочей среде

Рабочая среда Flash позволяет увидеть лишь малую часть функциональности создаваемого проекта. Непосредственно в рабочей среде можно протестировать следующие элементы фильма:

- анимацию основной монтажной линейки;
- анимацию графических (Graphic) символов;
- кнопки, расположенные непосредственно на основной монтажной линейке;
- звуки, помещенные в кадры основной монтажной линейки или кнопки;
- простые сценарии кадров основной монтажной линейки или кнопок, связанные с управлением воспроизведением монтажной линейки фильма.

В рабочей среде с основной монтажной линейки нельзя воспроизвести анимацию и звуки, помещенные на монтажные линейки муви-клипов, а также

увидеть функционирование вложенных кнопок. В рабочей среде анимацию муви-клипа можно просмотреть, только находясь в среде его редактирования, однако при этом анимация вложенных клипов и кнопок будет не видна. Сценарии ActionScript не выполняются в рабочей среде (за исключением простейших функций управления воспроизведением). Таким образом, можно сказать, что в рабочей среде возможности тестирования в основном ограничиваются просмотром содержимого текущей монтажной линейки.

Тестирование документа в рабочей среде осуществляется при помощи команд главного меню **Control** и/или контроллера (**Controller**). Меню **Control** содержит следующие команды, позволяющие управлять отображением и функционированием элементов фильма в рабочей среде:

- **Play/Stop** — включение/остановка воспроизведения текущей монтажной линейки, начиная с текущего кадра;
- **Rewind** — переход к первому кадру текущей монтажной линейки. Если режим **Play All Scenes** активен, то выполнение данной команды на основной монтажной линейке приведет к переходу на первый кадр первой сцены документа;
- **Go to End** — переход к последнему кадру текущей монтажной линейки. Если режим **Play All Scenes** активен, то выполнение данной команды на основной монтажной линейке приведет к переходу на последний кадр последней сцены документа;
- **Step Forward One Frame** — переход на следующий кадр и остановка воспроизведения;
- **Step Backward One Frame** — переход на предыдущий кадр и остановка воспроизведения;
- **Loop Playback** — зациклить воспроизведение. Выполнение этой команды приводит к тому, что воспроизводящая головка, достигнув последнего кадра, автоматически переходит к первому кадру, и воспроизведение возобновляется;
- **Play All Scenes** — данный режим позволяет автоматически переходить с предыдущей сцены на последующую при воспроизведении основной монтажной линейки документа;
- **Enable Simple Frame Actions** — данный режим позволяет реализовать выполнение простых сценариев кадров и кнопок (при включенном режиме **Enable Simple Buttons**), управляющих воспроизведением монтажной линейки и переходом между кадрами. К числу действующих в рабочей среде функций относятся `stop()`, `play()`, `gotoAndStop()`, `gotoAndPlay()`, `prevFrame()`, `nextFrame()`, `prevScene()`, `nextScene()`;
- **Enable Simple Buttons** — режим включения кнопок. При активизации данного режима можно просмотреть функционирование кнопки непо-

средственно в рабочей среде. Однако если кнопка содержит вложенные муви-клипы, их анимация не будет воспроизводиться в рабочей среде;

- Enable Live Preview** — режим, позволяющий отображать компоненты в рабочей среде такими же, как и в конечном фильме;
- Mute Sounds** — запретить воспроизведение звуков в рабочей среде.

Контроллер позволяет управлять воспроизведением текущей монтажной линейки. Для того чтобы открыть контроллер, необходимо выполнить команду главного меню **Window>Toolbars>Controller**.

Работа в среде тестирования

Среда тестирования позволяет просмотреть конечный фильм, включая все его элементы и полную функциональность. В среде тестирования можно оценить, как будет выглядеть процесс загрузки данного фильма по сети при различных скоростях соединения, и получить наглядное представление о распределении данных по кадрам. Среда тестирования также предоставляет ряд возможностей для отладки сценариев ActionScript, используемых в фильме.

При переходе в среду тестирования Flash автоматически генерирует файл SWF, используя текущие настройки публикации, заданные в окне **Publish Settings** (см. гл. 17). Этот файл дописывается в тот же каталог, где был сохранен исходный документ. Для перехода в среду тестирования можно выполнить одну из двух команд главного меню **Control**:

- Test Movie** ($<\text{Ctrl}>+<\text{Enter}>$) — тестирование фильма. В результате выполнения данной команды весь фильм экспортируется в файл SWF, который открывается в среде тестирования;
- Debug Movie** — данная команда используется для отладки сценариев. Ее выполнение приводит к переходу в среду тестирования и запуску отладчика (**Debugger**);
- Test Scene** ($<\text{Ctrl}>+<\text{Alt}>+<\text{Enter}>$) — тестирование сцены. В результате выполнения данной команды текущая сцена экспортируется в файл SWF, который открывается в среде тестирования. Выполнение команды **Test Scene** в среде редактирования символа позволяет осуществить тестирование данного символа. При этом его монтажная линейка экспортируется в самостоятельный файл SWF.

Среда тестирования открывается в отдельном окне, и таким образом тестирование проекта выполняется независимо от среды его разработки. Окно среды тестирования в развернутом виде всегда несколько больше, чем рабочая область, размеры которой задаются в рабочей среде Flash. В связи с этим в среде тестирования может быть видна часть вспомогательной области с расположенными на ней объектами. Для того чтобы привести окно среды

тестирования к размерам рабочей области, можно щелкнуть на стандартной кнопке **Restore**  в правом верхнем углу окна. В среде тестирования имеется собственное главное меню, содержащее набор команд, используемых в процессе тестирования и отладки.

Управление отображением и воспроизведением фильма

Для изменения масштаба просмотра фильма в среде тестирования используются команды главного меню **View**:

- Zoom In** — увеличение масштаба просмотра;
- Zoom Out** — уменьшение масштаба просмотра;
- Magnification** — выбор одного из стандартных значений масштаба.

Установить требуемый масштаб просмотра можно также при помощи контекстного меню, для вызова которого необходимо щелкнуть на окне фильма правой кнопкой мыши.

Поскольку Flash использует технологию сглаживания (antialiasing) по умолчанию, это может вызвать снижение скорости воспроизведения, если ресурсов процессора недостаточно для прорисовки кадров с требуемой скоростью. Такая ситуация может иметь место при одновременном выполнении нескольких сложных анимационных процессов с применением автоматической анимации. Чтобы задать соотношение между качеством прорисовки графики и скоростью воспроизведения, можно воспользоваться командой главного меню **View>Quality** или соответствующей командой контекстного меню среды тестирования. Данная команда содержит три значения, определяющие приоритет качества или быстродействия: **Low** — отключение сглаживания, максимальная скорость; **Medium** — компромисс между качеством и скоростью, **High** — сглаживание включено, скорость прорисовки может быть снижена.

Управление воспроизведением фильма в среде тестирования осуществляется при помощи команд меню **Control**, аналогичных соответствующим командам рабочей среды фильма:

- Play/Stop** — включение/остановка воспроизведения, начиная с текущего кадра;
- Rewind** — переход к первому кадру монтажной линейки;
- Loop** — зациклить воспроизведение;
- Step Forward One Frame** — переход на следующий кадр и остановка воспроизведения;
- Step Backward One Frame** — переход на предыдущий кадр и остановка воспроизведения.

Данные команды также могут быть выполнены при помощи контекстного меню среды тестирования.

Как и в рабочей среде, в среде тестирования имеются свои горячие клавиши (клавиатурные эквиваленты команд). В некоторых случаях при необходимости программного контроля клавиатуры использование горячих клавиш может приводить к нежелательным результатам, поскольку они обладают приоритетом над сценарием. Отключить горячие клавиши среды тестирования можно при помощи команды **Control>Disable Keyboard Shortcuts**.

Команда главного меню **Debug** позволяет вывести сведения обо всех используемых в фильме переменных или объектах. Эти данные выводятся в открывшемся окне **Output** и бывают весьма полезны при отладке сценариев. Для того чтобы вывести сведения о переменных или объектах фильма, необходимо выполнить одну из двух команд соответственно:

- List Variables** — выводит список переменных;
- List Objects** — выводит список объектов.

Эмуляция загрузки

При загрузке по сети Flash-фильм представляет собой поток данных (data streaming), организованный таким образом, что первыми загружаются те данные, которые должны отображаться в первую очередь. Вне зависимости от скорости соединения проигрыватель Flash Player пытается поддерживать заданную в рабочей среде скорость воспроизведения. Если скорость передачи данных недостаточно высока, то для того, чтобы обеспечить своевременную загрузку содержимого кадров, на соответствующем участке воспроизведение будет приостановлено до тех пор, пока не загрузится информация, необходимая для отображения следующего кадра. Это приводит к нарушению ритма фильма и асинхронизации звукового сопровождения, разрушая впечатление целостности. Если размер кадра меньше, чем объем информации, которая может быть загружена при заданной скорости за время экспонирования кадра, то в процессе его воспроизведения будет происходить загрузка следующих кадров. Элементы, которые многократно используются в фильме, загружаются только в кадре их первого вхождения в фильм. В дальнейшем для их повторного отображения загрузка не должна выполняться повторно. Таким образом, реальная картина загрузки проекта по сети будет существенно отличаться от запуска Flash-фильма, размещенного локально на компьютере пользователя.

Среда тестирования предоставляет возможность имитации процесса загрузки по сети с заданной скоростью. Используя эту возможность, можно смоделировать процесс загрузки с использованием различных модемов, по цифровым линиям или по локальной сети. Для того чтобы включить режим эмуляции загрузки, необходимо выполнить команду главного меню **View>Simulate Download** или воспользоваться ее клавиатурным эквивалентом **<Ctrl>+<Enter>**.

Скорость загрузки зависит от скорости соединения. Команда **View>Download Settings** позволяет задать одно из предустановленных значений скорости передачи данных, на основании которого будет реализована имитация загрузки фильма. При моделировании процесса загрузки учитывается не номинальная скорость модема, а более реальная величина, имеющая место на практике. Так, например, если скорость модема составляет 56 Кбит/с, то теоретически он может передавать информацию со скоростью около 7 Кбайт/с, однако на практике скорость будет иметь величину порядка 4,7 Кбайт/с или 4800 байт/с. При эмуляции загрузки также учитывается компрессия SWF-файла (если данная опция была задана при публикации), которая приводит к уменьшению размера конечного файла и, соответственно, к увеличению скорости его передачи. Выбор команды **Customize** приводит к появлению диалогового окна настройки скорости передачи данных **Custom Download Settings**, представленного на рис. 16.1.

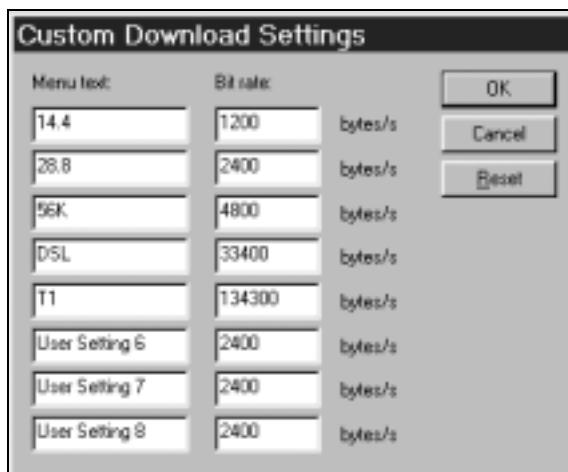


Рис. 16.1. Диалоговое окно **Custom Download Settings**

Здесь в колонке **Menu text** можно указать обозначение требуемой скорости, причем эти данные не влияют на результат, а используются для вывода в меню **View>Download Settings**. Колонка **Bit rate** позволяет задать величину скорости соединения, на основании которой Flash будет эмулировать процесс загрузки.

Окно **Bandwidth Profiler**

Важнейшим инструментом среди тестирования является окно **Bandwidth Profiler**, позволяющее произвести анализ процесса загрузки фильма. Для того чтобы открыть данное окно, необходимо выполнить команду главного

меню **View>Bandwidth Profiler** или воспользоваться ее клавиатурным эквивалентом <Ctrl>+. Окно **Bandwidth Profiler** состоит из двух разделов: слева приводятся параметры загрузки фильма, справа находится один из двух видов графиков, иллюстрирующий процесс загрузки или размер кадров фильма (рис. 16.2).

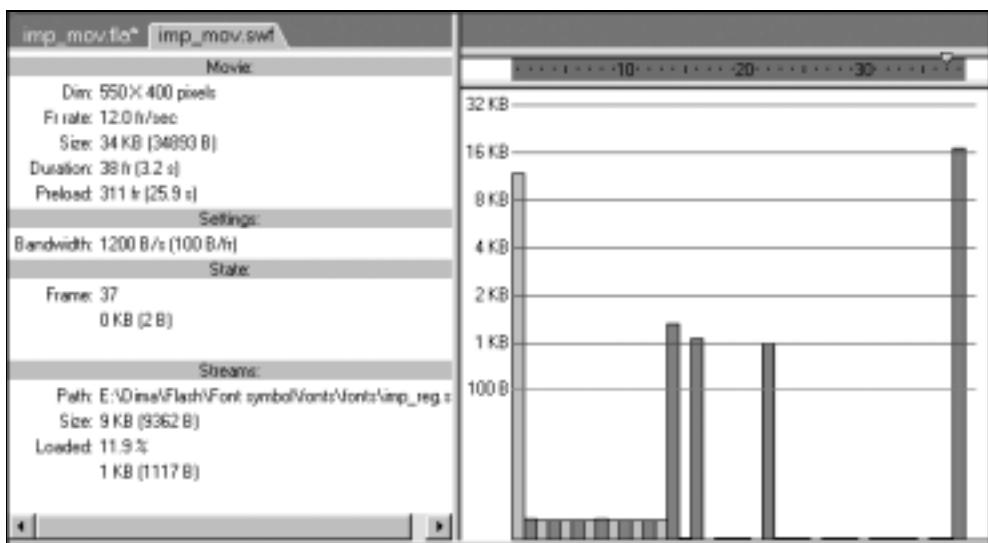


Рис. 16.2. Окно **Bandwidth Profiler**

Информационная панель

Информационная панель в левой части окна **Bandwidth Profiler** состоит из нескольких разделов и содержит следующие сведения.

Movie — параметры фильма:

- **Dim** — размер фильма, определяемый размерами рабочей области;
- **Fr rate** — скорость воспроизведения монтажной линейки фильма;
- **Size** — размер конечного файла SWF в килобайтах и байтах;
- **Duration** — длительность фильма в кадрах и секундах;
- **Preload** — суммарное время ожидания загрузки данных, на протяжении всего фильма. Если скорость передачи данных при заданной скорости соединения ниже, чем требуется для поддержания требуемой скорости воспроизведения, то на соответствующих кадрах воспроизведение приостанавливается, ожидая загрузки. Данный параметр указывает суммарную длительность всех таких задержек и измеряется в секундах и в кад-

рах (исходя из используемого значения скорости воспроизведения монтажной линейки). Если скорость передачи данных достаточна для обеспечения требуемой скорости воспроизведения на протяжении всего фильма, то значение данного параметра равно нулю.

- **Settings** — заданные настройки скорости соединения, для которой выполняется тестирование. Первая цифра указывает скорость в байтах в секунду, вторая (в скобках) — максимальный размер каждого кадра фильма, при котором будет обеспечена требуемая скорость воспроизведения и не будут возникать остановки при загрузке с заданной скоростью.
- **State** — информация о размере кадров и состоянии загрузки.
 - **Frame** — номер кадра и его размер.
 - **Loaded** — эти данные отображаются только в режиме эмуляции загрузки **Simulate Download** на протяжении загрузки фильма и указывают, какой объем фильма (в процентах и килобайтах) загрузился на текущий момент. После того как фильм загружен, эта информация не видна.
- **Streams** — этот раздел появляется только в режиме эмуляции загрузки **Simulate Download** в том случае, если в процессе воспроизведения в текущий фильм подгружаются внешние данные (например, растровое изображение, звуковой файл, внешний ролик SWF, элементы общей библиотеки и т. д.).
 - **Path** — путь к загружаемому файлу.
 - **Size** — размер загружаемого файла.
 - **Loaded** — состояние загрузки внешнего файла (в процентах и килобайтах).

Работа с графиками окна **Bandwidth Profiler**

В правой части окна **Bandwidth Profiler** расположены шкала с нумерацией кадров и один из графиков: **Streaming Graph** (потоковый график) или **Frame by Frame Graph** (покадровый график). В верхней части окна находится шкала с нумерацией кадров. Треугольный бегунок, находящийся на этой шкале, указывает на текущее положение воспроизводящей головки. Его можно перемещать, так же как и указатель текущего кадра в рабочей среде фильма, таким образом переходя из одного кадра в другой. По вертикальной оси (расположенной справа) откладывается размер кадров. Красная горизонтальная линия показывает максимальный размер кадра, обеспечивающий нормальную передачу потока данных без прерываний воспроизведения. Это значение рассчитывается как отношение скорости передачи данных к скорости воспроизведения монтажной линейки. Так, если заданная в настрой-

ках **Custom Download Settings** скорость соединения составляет 1200 байт в секунду, то при скорости воспроизведения монтажной линейки 12 кадров в секунду объем каждого кадра не должен превышать 100 байт. Здесь, однако, существует один нюанс, который необходимо иметь в виду. Дело в том, что если объем предыдущих кадров меньше критического объема и число таких кадров достаточно велико, то в процессе их воспроизведения следующий за ними кадр, превышающий критический размер, может успеть загрузиться и не вызовет паузы.

При активизации режима эмуляции загрузки **Simulate Download** поток данных графически отображается в виде растущей зеленой полосы загрузки на шкале в верхней части окна **Bandwidth Profiler**. Если полоса загрузки опережает указатель текущего кадра, это означает, что скорость передачи информации достаточна для обеспечения заданной скорости воспроизведения; в противном случае воспроизведение может быть приостановлено.

Потоковый график (**Streaming Graph**)

Потоковый график является графическим представлением процесса загрузки кадров фильма и позволяет определить те участки фильма, которые могут вызвать задержку при загрузке. Для того чтобы отобразить этот график в правой части окна **Bandwidth Profiler**, необходимо выполнить команду главного меню **View>Streaming Graph** (<Ctrl>+<G>). Потоковый график представлен на рис. 16.3. Вертикальные светлые и темные серые полосы (или блоки) представляют собой кадры фильма. Каждый из них может быть выделен по отдельности; при этом в разделе **State** информационной панели будет выведен номер данного кадра и его размер. Теоретически загрузка будет выполняться без остановок и прерываний воспроизведения в том случае, если время загрузки содержимого кадра равно (или меньше) времени его экспонирования, которая определяется исходя из заданной скорости воспроизведения (frame rate). Так, если скорость воспроизведения составляет 12 кадров в секунду, то на загрузку каждого кадра отводится 1/12 секунды. Максимальный размер кадра, который может быть загружен за время экспонирования при заданной скорости передачи данных, будем называть в дальнейшем *критическим размером*. Критический размер кадра на графике указывается красной горизонтальной полосой.

Если размер кадра меньше критического размера, отмеченного красной полосой, то это означает, что данный кадр загружается за более короткий интервал времени, чем составляет длительность экспонирования (задаваемая скоростью воспроизведения); таким образом, он не только не вызывает паузы при загрузке, но и предоставляет дополнительное время для загрузки следующих после него кадров. На графике блоки, соответствующие таким кадрам, располагаются друг над другом в одном или нескольких расположенных рядом столбцах (рис. 16.3, а).

Если размер кадра в точности равен критическому размеру, то такой кадр будет загружаться в течение всего отведенного ему времени, не вызывая паузы при загрузке, но и не предоставляя дополнительное время для загрузки других кадров. Такой кадр отображается столбцом, высота которого равна критическому размеру кадра (рис. 16.3, б).

Если размер кадра превышает критический размер, то для его загрузки требуется больше времени, чем составляет длительность экспонирования кадра. Соответственно, такой кадр либо вызовет приостановку воспроизведения, либо использует дополнительное время, "сэкономленное" предшествующими кадрами небольшого размера (а если накопленный запас времени не сможет компенсировать размер кадра, то произойдет и одно, и другое одновременно). Такой кадр отображается в виде широкого блока. Если часть этого блока выходит за пределы красной линии, это означает, что в данном месте воспроизведение будет прерываться, ожидая загрузки данных (рис. 16.3, в). Если весь блок укладывается в допустимый диапазон, не пересекая красную полосу, то он не будет вызывать паузы при воспроизведении (рис. 16.3, г).

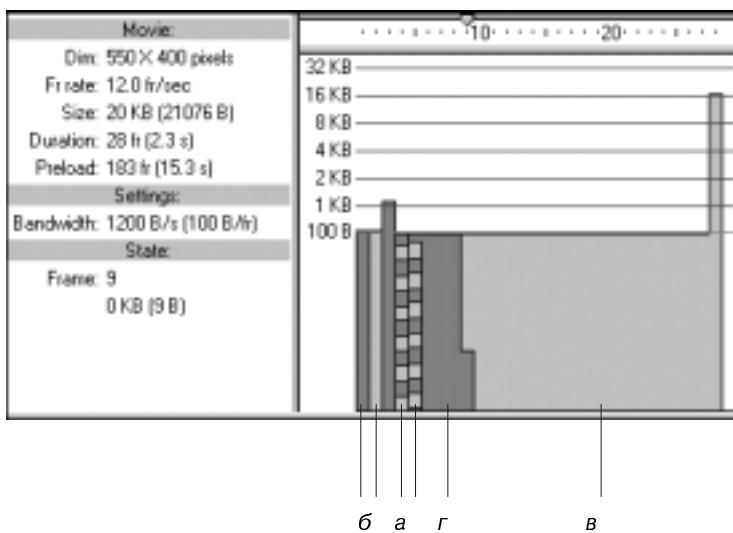


Рис. 16.3. Потоковый график

На практике, как правило, не удается добиться плавной загрузки фильма без приостановок воспроизведения. Для того чтобы избежать связанных с этим негативных последствий, создается сценарий, который называется "предзагрузчик", или preloader. Предзагрузчик предотвращает воспроизведение фильма до тех пор, пока он не будет полностью загружен (или пока не будет

загружена определенная его часть). *Подробная информация о создании предзагрузчика содержится в гл. 21.*

Покадровый график (Frame by Frame Graph)

Для того чтобы отобразить этот график в правой части окна **Bandwidth Profiler**, необходимо выполнить команду главного меню **View>Frame by Frame Graph** (<Ctrl>+<F>). Покадровый график представляет собой столбчатую диаграмму, высота каждого столбца которой соответствует размеру кадра, находящегося в текущей временной позиции (см. рис. 16.2). Участки покадрового графика с отсутствующими столбцами соответствуют кадрам, в которых не происходит никаких изменений. Для получения информации о размере каждого кадра можно щелкнуть на соответствующем столбце, при этом в разделе **State** информационной панели будет выведен номер данного кадра и его размер.

Отчет о размерах элементов фильма

Еще одна возможность, позволяющая проанализировать эффективность организации фильма, заключается в создании отчета о размерах всех используемых в фильме элементов. Данный отчет представляет собой текстовый документ (файл с расширением txt), который генерируется автоматически и дописывается в каталог рядом с исходным документом. Так, если документ был сохранен с именем myMovie.fla, то файл отчета будет назван myMovie Report.txt. Отчет содержит информацию о размерах фильма по кадрам и по сценам, а также о размерах используемых символов, шрифтов, сценариев ActionScript и импортированных объектов. Анализ отчета позволяет увидеть, какой вклад в размер каждой сцены и символа вносят векторные объекты, текст и сценарии. Кроме того, здесь приводится информация обо всех используемых в фильме шрифтах и указываются внедряемые символы. Отчет также отображает информацию о местоположении всех сценариев фильма и о параметрах компрессии импортированных объектов.

Для того чтобы создать отчет, необходимо в диалоговом окне настройки параметров публикации **File>Publish Settings** на вкладке **Flash** установить флаjkок **Generate Size Report**. В результате всякий раз при публикации или тестировании проекта отчет будет создаваться и автоматически записываться на диске. Кроме того, при тестировании проекта в среде тестирования отчет будет выводиться в диалоговом окне **Output**.

Пример отчета о размере фильма представлен в листинге 16.1.

Листинг 16.1. Пример отчета о размере фильма

heartbeat.swf Movie Report

Frame #	Frame Bytes	Total Bytes	Scene
1	16576	16576	Scene 1 (AS 2.0 Classes Export Frame)
2	19	16595	
3	1664	18259	
4	18	18277	
5	18	18295	
6	22	18317	
7	2995	21312	Scene 2

Scene	Shape Bytes	Text Bytes	ActionScript Bytes
Scene 1	605	138	897
Scene 2	37	0	885

Symbol	Shape Bytes	Text Bytes	ActionScript Bytes
buttonNav	84	0	0
logo	115	73	104
lightSpot	77	0	0
graph	0	0	0
car	1364	0	0
Flash	0	0	0
headlights	573	0	0

Tweened Shapes: 192 bytes

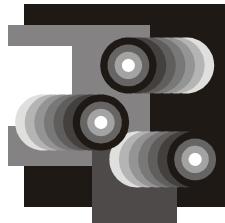
Font Name	Bytes	Characters
Courier New	1888	SWacefghiklmmostuwyz
_sans	18	

ActionScript Bytes	Location
885	Scene 1:actions:1
12	Scene 1:buttons:1>No instance name assigned(buttonNav)
104	logo:morphing:1
885	Scene 2:Actions:1

Bitmap	Compressed	Compression
--------	------------	-------------

backImg.jpg	12019	312512	JPEG Quality=70
Event Sounds: 22KHz Mono 16 kbps MP3			
Sound name	Bytes	Format	
<hr/>			
soundSync.WAV	1835	22KHz Mono 64 kbps MP3	

Анализ отчета зачастую позволяет выявить проблемные места, которые могут вызвать задержку загрузки фильма. Используя информацию, предоставленную отчетом, можно попытаться оптимизировать фильм за счет более рациональной организации его структуры (например, за счет использования символов, компрессии импортированных объектов и т. д.).



Глава 17

Публикация и экспорт документа

Наибольшие трудности встречаются там, где мы их не ожидаем.

Гете

Настройка параметров публикации

Настройка параметров публикации документа осуществляется при помощи диалогового окна **Publish Settings**. Для того чтобы открыть это окно, необходимо выполнить команду главного меню **File>Publish Settings** или воспользоваться сочетанием клавиш <Ctrl>+<Shift>+<F12>. Данное окно представлено на рис. 17.1.

Столбец **Type** содержит перечень всех форматов файлов, которые могут быть получены в результате публикации. Установка соответствующего флажка приводит к тому, что при выполнении команды **Publish** исходный документ будет опубликован в указанном формате. При установке нескольких флажков исходный документ будет транслирован во все выбранные форматы, в результате чего Flash автоматически генерирует требуемые файлы.

По умолчанию файлы, генерированные при публикации, получают имя исходного документа и сохраняются в том же каталоге, где был сохранен Flash-документ (с расширением fla). Так, например, если исходный документ был сохранен в каталоге myProject с именем myMovie.fla, то в результате публикации в каталог myProject будут помещены файлы myMovie.swf, myMovie.html, myMovie.gif и т. д.

При необходимости можно сохранить результирующий файл под другим именем, отличным от имени исходного документа. Для этого необходимо воспользоваться полем **Filename**, соответствующим требуемому формату, расположенному в столбце **File**. Для того чтобы сохранить результирующий файл (или несколько файлов) в произвольном каталоге, необходимо задать путь к этому каталогу, воспользовавшись кнопкой **Select Publish Destina-**

tion. Кнопка **Use Default Names** позволяет вновь применить названия по умолчанию.



Рис. 17.1. Диалоговое окно **Publish Settings**

По мере выбора требуемых форматов публикации в верхней части окна **Publish Settings** появляются ярлыки, позволяющие переместиться на вкладку, содержащую соответствующие параметры настройки. Совокупность настроек, задаваемых в окне **Publish Settings**, называется *профилем публикации* (Publish profile). При сохранении документа профиль публикации сохраняется вместе с ним. Один документ может содержать несколько различных профилей. Flash MX 2004 позволяет сохранять профиль во внешнем файле и импортировать в другие документы. Это дает возможность создания набора стандартных настроек, которые могут быть многократно использованы при работе над различными документами.

Название текущего профиля публикации указывается в списке **Current profile**, расположеннном в верхней части окна **Publish Settings**. По умолчанию новый документ использует профиль **Default**, содержащий настройки публикации в форматы Flash (SWF) и HTML. Кнопки, расположенные справа от

списка **Current profile**, позволяют выполнить ряд операций с профилями публикации.

- Для того чтобы создать новый профиль, нужно щелкнуть на кнопке **Create New Profile**  и в появившемся диалоговом окне **Create New Profile** указать имя нового профиля. Созданный профиль будет добавлен в список **Current profile** и автоматически сделан текущим.
- Для того чтобы создать копию существующего профиля, нужно воспользоваться кнопкой  **Duplicate Profile**.
- Для того чтобы изменить профиль, нужно выбрать его в списке **Current profile** и произвести все необходимые изменения. Новые установки будут автоматически сохранены в данном профиле.
- Кнопка  **Profile Properties** позволяет переименовать существующий профиль.
- Для того чтобы получить возможность использования профиля публикации текущего документа в других документах Flash, данный профиль необходимо экспорттировать и сохранить во внешнем файле XML. Это можно сделать при помощи кнопки **Import/Export Profile** . В выпадающем меню нужно выбрать команду **Export**, а появившемся вслед за этим окне **Export Profile** указать имя сохраняемого профиля. По умолчанию все экспортруемые профили сохраняются в каталоге **Publish Profiles**. При работе с операционной системой Windows 2000 или Windows XP данный каталог расположен по адресу: <drive>/Documents and Settings/<username>/Local Settings/Application Data/Macromedia/Flash MX 2004/en/Configuration/Publish Profiles. При работе с операционной системой Windows 98 он расположен по адресу: <drive>/Windows/Application Data\Macromedia/Flash MX 2004/en/ Configuration/Publish Profiles.
- Для того чтобы импортировать профиль, нужно щелкнуть на кнопке **Import/Export Profile** и в выпадающем меню выбрать команду **Import**. Далее в диалоговом окне **Import Profile** нужно найти требуемый XML-файл. При импорте профиля Flash автоматически открывает каталог Publish Profiles.
- Удалить профиль из документа можно при помощи кнопки  **Delete Profile**, при этом соответствующий XML-файл остается в каталоге Publish Profiles.

После выполнения всех настроек документ можно опубликовать. Для этого необходимо выполнить одно из следующих действий:

- щелкнуть на клавише **Publish** в диалоговом окне **Publish Settings**;
- выполнить команду главного меню **File>Publish** или воспользоваться ее клавиатурным эквивалентом <Shift>+<F12>;

- для того чтобы просмотреть результат публикации в один из выбранных форматов, нужно выполнить команду главного меню **File>Publish Preview**. В раскрывающемся списке можно выбрать требуемый формат. Выполнение данной команды приводит к публикации документа только в одном выбранном формате;

При тестировании проекта (**Control>Test Movie**, **Control>Test Scene**) происходит публикация документа в формат SWF.

Формат Flash (swf)

Файл SWF включает в себя всю функциональность, которой исходный документ был снабжен на стадии разработки. В результате публикации происходят реорганизация и оптимизация данных, в результате чего SWF-файл занимает значительно меньший объем, чем исходный авторский документ. Формат SWF используется для размещения Flash-фильма в сети Интернет. Для воспроизведения фильма в формате SWF применяется проигрыватель Flash Player. Для просмотра Flash-содержимого при помощи web-браузера в зависимости от его типа необходимо наличие соответствующей версии элемента управления Flash Player ActiveX (для браузера Internet Explorer) или встраиваемого модуля расширения Flash Player (для браузера Netscape Navigator). Локально размещенный Flash-фильм может быть воспроизведен при помощи самостоятельного проигрывателя Flash (Stand Alone Player), который называется проектором (projector) и поставляется вместе с Flash MX 2004. Его можно найти в каталоге Players, расположенному в директории, куда были установлены файлы Flash приложения. Файл проектора называется SAFlashPlayer.exe. Файл SWF также может быть воспроизведен в приложении Macromedia Director при наличии соответствующего расширения (Flash Asset Xtra).

Вкладка **Flash** окна **Publish Settings** позволяет установить следующие параметры публикации (рис. 17.2).

- **Version** — данный список позволяет указать версию проигрывателя Flash Player, для которой будет выполнена публикация. Следует иметь в виду, что фильм, опубликованный для более поздней версии проигрывателя, не может быть воспроизведен более ранней версией. При публикации для более ранней версии (ниже Flash Player 7) часть функциональности проекта, связанная с новыми возможностями Flash, может быть утрачена.
- **Load order** — данный список позволяет указать порядок загрузки элементов первого кадра фильма по слоям: **Bottom up** — снизу вверх или **Top down** — сверху вниз. При медленном соединении бывает важно, чтобы содержимое слоев загружалось в определенном порядке.

- **ActionScript version** — данный список позволяет указать версию языка сценариев ActionScript, которая использовалась при написании сценариев в документе. В связи с тем, что во второй версии ActionScript появилось несколько существенных отличий (строгая типизация данных, объявление классов во внешних файлах и др.), сценарии в духе ActionScript 1.0 могут не работать или выполняться некорректно. Для того чтобы реализовать обратную совместимость, нужно выбрать первую версию ActionScript 1.0. Если в документе используется ActionScript 2.0, становится доступной кнопка **Settings**, приводящая к появлению диалогового окна **ActionScript Settings**. Здесь можно задать кадр, в котором будут экспортированы классы, используемые сценарием фильма, а также путь (Classpath) к каталогу, содержащему файлы AS с определением классов.

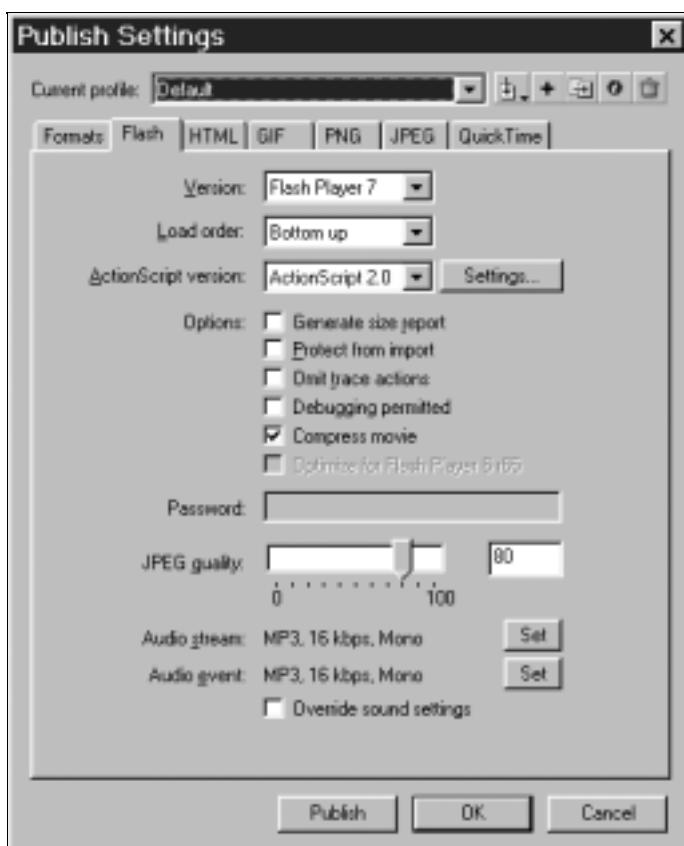


Рис. 17.2. Настройка параметров публикации в формат SWF (Flash)

□ **Options** — данный раздел содержит ряд опций, используемых при публикации.

- **Generate size report** — создать отчет о размере всех элементов фильма. Отчет будет сохранен в формате TXT в том же каталоге, где находится исходный документ (см. гл. 16).
- **Protect from import** — установка данного флашка приведет к тому, что сгенерированный при публикации SWF-файл не сможет быть импортирован обратно в рабочую среду. При попытке импортировать или даже открыть защищенный файл в рабочей среде будет выведено сообщение о том, что данная операция не может быть выполнена. При установке данного флашка становится доступным поле **Password**, в которое можно ввести пароль, позволяющий осуществить импорт защищенного файла. Следует иметь в виду, что данная защита предусматривает использование легальных средств и может быть достаточно легко разблокирована.
- **Omit trace actions** — установка данного флашка приведет к тому, что при тестировании фильма все вызовы функции `trace()` будут проигнорированы. Функция `trace()` используется при отладке сценария ActionScript и позволяет в среде тестирования выводить в окно **Output** различную информацию, помогающую проанализировать ход выполнения программы (сообщения о выполнении определенных условий, результаты вычисления выражений, принимаемые переменными значения и т. д.).
- **Debugging permitted** — установка данного флашка позволяет осуществлять отладку сценария Flash-фильма, размещенного на сервере. Для того чтобы воспользоваться отладчиком сценариев (Debugger), необходимо использовать проигрыватель Flash Debug Player, который устанавливается автоматически при установке Flash MX 2004. При установке данного флашка становится доступным поле **Password**, позволяющее задать пароль для удаленной отладки.
- **Compress movie** — установка данного флашка позволяет применить к SWF-файлу компрессию, и таким образом уменьшить его размер. Эта возможность может быть использована только при работе с проигрывателем Flash Player 6 или выше. Наиболее успешно сжатие может быть применено к фильму, содержащему большие объемы текста и/или сценариев ActionScript.
- **Optimize for Flash Player 6 r65** — эта опция становится доступной только при публикации документа для шестой версии Flash Player. Установка данного флашка позволяет оптимизировать фильм для промежуточной версии проигрывателя.

- **JPEG quality** — этот параметр позволяет задать степень сжатия всех используемых в фильме растровых изображений, которые не были оптимизированы при помощи библиотеки, т. е. для которых в окне **Bitmap Properties** был установлен флажок **Use document default quality**. Требуемое значение может быть установлено при помощи слайдера или введено вручную в поле, расположенное справа от него. Значение параметра **quality** задается в диапазоне от 0 до 100 и характеризует собой соотношение качества изображения и размера занимаемого им файла. Чем выше значение качества, тем больший объем будет занимать файл данного изображения.
- **Audio Stream/Audio Event** — кнопки **Set** позволяют задать тип и параметры компрессии для используемых в фильме событийных и потоковых звуков в отдельности. Данные настройки компрессии применяются только к тем звуковым файлам, которые не были индивидуально оптимизированы при помощи библиотеки, т. е. для которых в списке **Compression** окна **Sound Properties** было установлено значение **Default**. Нажатие кнопки **Set** приводит к появлению окна **Sound Settings**, где в списке **Compression** можно выбрать требуемую схему сжатия. Типы компрессии и параметры настройки аналогичны соответствующим параметрам окна **Sound Properties**. При выборе значения **Disable** звук не экспортируется в конечный фильм.
- **Override sound settings** — установка этого флажка позволяет переопределить параметры оптимизации звуков, заданные в библиотеке, текущими настройками окна **Publish Settings**.

Формат HTML (html/htm)

При размещении Flash-фильма в Интернете на сервер необходимо отправить сам SWF-файл и документ HTML, содержащий набор тегов, необходимых для описания параметров Flash-фильма, связанного с данным документом. Flash позволяет автоматически генерировать HTML-страницу на основе одного из стандартных шаблонов, поставляемых вместе с Flash MX 2004.

При публикации в формате HTML автоматически выделяется формат Flash. Взаимное расположение HTML-страницы и Flash-фильма в результате публикации определяется значениями, заданными в полях **Filename** вкладки **Formats**. На основе этого значения в соответствующих тегах HTML-страницы указывается путь к отображаемому на ней файлу SWF. Если после публикации взаимное расположение данных файлов будет изменено, HTML-код необходимо отредактировать вручную, в противном случае Flash-фильм не сможет быть отображен на странице.

Вкладка окна **Publish Settings** позволяет выполнить ряд настроек, на основе которых будет сгенерирована HTML-страница, содержащая соответствующий код для описания параметров размещения и воспроизведения Flash-фильма (рис. 17.3).

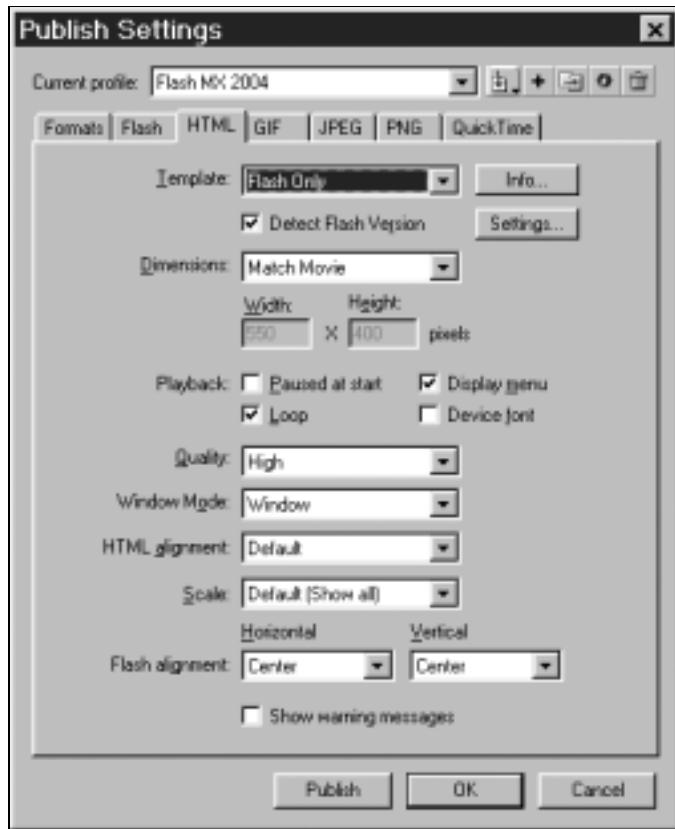


Рис. 17.3. Настройка параметров публикации в формат HTML

Вкладка **HTML** позволяет задать следующие параметры, на основе которых будет сгенерирован документ **HTML**.

- Раскрывающийся список **Template** позволяет выбрать один из предлагаемых шаблонов, поставляемых вместе с Flash MX 2004, на основе которого будет сгенерирована HTML-страница. Код, создаваемый в результате публикации на основе поставляемых шаблонов, соответствует стандарту XHTML. Список содержит следующий набор шаблонов.
 - **Flash For Pocket PC 2003** — данный шаблон используется для размещения Flash-содержимого на HTML-странице, предназначенный для

просмотра в браузере Pocket Internet Explorer на платформе Pocket PC. Страница, сгенерированная на основе данного шаблона, может быть также просмотрена в обычных браузерах IE и Netscape Navigator.

- **Flash HTTPS** — данный шаблон позволяет генерировать HTML-страницу и разместить на ней Flash-фильм. Если при загрузке данной страницы на компьютере пользователя будет отсутствовать проигрыватель Flash Player, он будет автоматически загружен с сайта компании Macromedia по защищенному каналу с использованием протокола HTTPS.
- **Flash Only** — данный шаблон используется по умолчанию и генерирует все необходимые теги для размещения Flash-фильма на странице. Шаблон Flash Only можно использовать, если предполагается создание web-страницы исключительно средствами Flash. В случае если на странице, кроме Flash-содержимого, должны находиться другие элементы HTML, данный шаблон удобно использовать для получения исходного кода, описывающего параметры размещения файла SWF, который в дальнейшем может быть отредактирован вручную и включен в соответствующий фрагмент кода конечной страницы.
- **Flash With AICC Tracking** — данный шаблон используется при разработке интерактивных обучающих программ (e-learning) на основе поставляемых вместе с Flash MX 2004 шаблонов (Quiz) и элементов общих библиотек (Learning Interactions). Данные элементы отслеживают выполняемые пользователем действия и отправляют соответствующую информацию управляющей системе (LMS — learning management system), расположенной на сервере и поддерживающей протокол AICC (Aviation Industry CBT Committee protocol). Информация об использовании поставляемых вместе с Flash интерактивных обучающих элементов содержится в документации к Flash MX 2004.
- **Flash With FSCommand** — данный шаблон помещает на страницу сценарий, позволяющий организовать взаимодействие Flash-фильма со сценарием Java Script, расположенным на данной странице при помощи функции `fscommand();`
- **Flash With Named Anchors** — данный шаблон необходимо использовать для того, чтобы получить возможность перемещаться между кадрами фильма, содержащими указатели или якоря (Named Anchors) при помощи кнопок браузера Forward (Вперед) и Back (Назад). Эта возможность позволяет пользователю переходить из одного раздела Flash-проекта в другой так же, как если бы это были просмотренные web-страницы. При этом, правда, все разделы должны находиться в пределах одного фильма. При публикации Flash автоматически генерирует HTML-страницу, содержащую сценарий JavaScript (для браузера Netscape Navigator), и якоря HTML, соответствующие указателям Flash. Ис-

пользование якорей поддерживается проигрывателем Flash только начиная с версии Flash Player 6.

- **Flash With SCORM Tracking** — данный шаблон, так же как и Flash With AICC Tracking, используется при разработке интерактивных обучающих программ. Применение этого шаблона предполагает использование стандарта SCORM (Shareable Content Object Reference Model).
 - **Image Map** — данный шаблон используется для создания карты изображения. Карта изображения представляет собой растровое изображение, размещаемое на HTML-странице, содержащее набор активных областей произвольной формы, выполняющих функции гиперссылок. Каждой из таких областей сопоставляется адрес интернет-ресурса; при попадании курсора в активную область он принимает вид руки; нажатие левой кнопки мыши позволяет перейти к соответствующему ресурсу. Карта изображения генерируется на основе содержимого кадра, помеченного меткой **#Static**. В данном кадре необходимо разместить кнопки и назначить им сценарии с использованием функции загрузки сетевого ресурса `getURL()` (см. гл. 19). Следует принять во внимание, что созданные активные области будут иметь прямоугольную форму, а не форму области реагирования кнопки Flash. При публикации, кроме форматов Flash и HTML, необходимо выбрать один из предлагаемых графических форматов GIF, JPEG или PNG — тот, который будет использован для создания изображения. При использовании изображения в формате JPEG воспроизводящая головка в момент публикации должна находиться в кадре, играющем роль изображения. В результате публикации будет создана HTML-страница, содержащая не Flash-фильм, а карту изображения.
 - **Quick Time** — данный шаблон позволяет разместить на HTML-странице фильм в формате Quick Time.
- Detect Flash Version** — установка данного флагка позволит реализовать механизм автоматической проверки наличия у пользователя необходимой для просмотра публикуемого SWF-файла версии проигрывателя Flash Player. Использование этой опции приведет к тому, что Flash автоматически сгенерирует и поместит на HTML-страницу (Detection File) SWF-файл (`Flash_detection.swf`), содержащий сценарий ActionScript, который проверяет номер версии проигрывателя Flash Player, установленного в системе у пользователя. Сценарий использует синтаксис ActionScript версии Flash 4, благодаря чему данный файл может быть воспроизведен у большинства пользователей Интернета. Страница, содержащая файл-определитель, должна быть загружена в первую очередь. В этом случае, если проверка покажет, что у пользователя установлена более ранняя версия Flash Player, то произойдет переход на альтернативную страницу (Alternate File), содержащую, например, HTML-версию сайта. Если про-

верка покажет, что номер версии Flash Player пользователя равен или выше версии, для которой выполнялась публикация, то будет автоматически выполнен переход к основной странице (Content File). Для настройки параметров определения версии необходимо воспользоваться кнопкой **Settings**, нажатие на которую приведет к появлению диалогового окна **Version Detection Settings** (рис. 17.4). Данное окно включает следующие настройки.

- ◊ В разделе **Requirements** указываются основной (**Major Revision**) и дополнительный (**Minor Revision**) номера версии Flash Player наличие которых будет проверяться.
- ◊ В поле **Detection File (first target file)** указывается местоположение страницы HTML, на которой будет размещен файл SWF, содержащий сценарий, проверяющий номер версии.

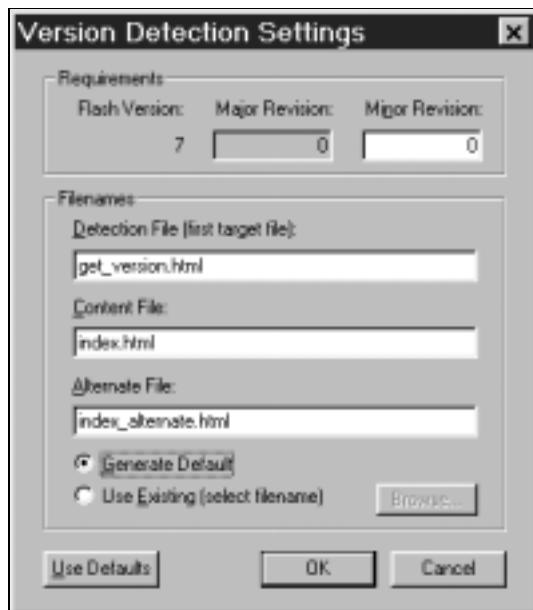


Рис. 17.4. Диалоговое окно **Version Detection Settings**

- ◊ В поле **Content File** указывается местоположение страницы HTML, содержащей сам Flash-фильм.
- ◊ В поле **Alternate File** указывается местоположение альтернативной страницы HTML, которая будет загружена в случае отсутствия у пользователя требуемой версии проигрывателя Flash Player. Альтернативная страница может быть создана вручную при помощи HTML-

редактора; в этом случае нужно установить переключатель в положение **Use Existing (select filename)** и указать ее местоположение, щелкнув на кнопке **Browse**. Альтернативная страница также может быть сгенерирована автоматически при публикации. Для этого нужно установить переключатель в положение **Generate Default**. Созданная таким образом страница будет содержать текст, сообщающий об отсутствии требуемой версии проигрывателя Flash Player, и кнопку, нажатие на которую позволит перейти к разделу сайта компании Macromedia, предлагающему загрузить последнюю версию Flash Player.

□ **Dimensions** — данный список позволяет задать размер окна Flash Player, в котором будет воспроизводиться фильм. Окно Flash Player представляет собой прямоугольную область без границ — ее можно условно сравнить с киноэкраном. Фильм, который воспроизводится в данном окне, можно сравнить с пятном света, проецируемым на экран. Список **Dimensions** содержит три варианта задания размеров окна:

- **Match Movie** — размер окна Flash Player задается равным размеру рабочей области документа. В этом случае фильм будет занимать все окно целиком;
- **Pixels** — произвольный размер окна Flash Player в пикселях. В этом случае размер окна и размер фильма могут отличаться;
- **Percent** — произвольный размер окна Flash Player в процентах от размера окна браузера. В этом случае размер окна и размер фильма могут отличаться. При изменении размера окна браузера будет происходить соответствующее масштабирование окна Flash Player.

□ **Playback** — параметры воспроизведения включают следующие установки.

- **Paused at start** — при установке этого флажка воспроизведение фильма автоматически не начинается. Запуск монтажной линейки основного фильма происходит в результате выполнения определенного действия (запуска фильма пользователем при помощи команды **Play** контекстного меню проигрывателя Flash Player, нажатия на спрограммированную кнопку, расположенную в фильме, выполнения сценария ActionScript и т. д.). Анимация монтажных линеек муви-клипов, расположенных в первом кадре основной монтажной линейки, будет выполняться вне зависимости от того, установлен данный флажок или нет.
- **Loop** — при установке этого флажка воспроизведение основной монтажной линейки фильма будет выполняться циклически.
- **Display menu** — в результате установки этого флажка, при щелчке правой кнопкой мыши внутри окна Flash Player будет выведено контекстное меню, содержащее ряд команд, при помощи которых можно

управлять воспроизведением основной монтажной линейки фильма и изменять масштаб просмотра содержимого данного окна. Если использование этих возможностей является нежелательным, флажок необходимо снять. Правда, при этом контекстное меню все равно будет появляться, но в нем будет присутствовать только один пункт — *About Macromedia Flash Player 7*.

- **Device font** — установка данного флажка приводит к замене отсутствующих в системе пользователя шрифтов сглаженными системными шрифтами. Данный параметр воздействует только на статический горизонтальный текст, для которого в рабочей среде был установлен режим **Use device fonts**.
- **Quality** — данный параметр позволяет задать соотношение между быстродействием и качеством прорисовки кадров. При включенном сглаживании одновременное выполнение нескольких сложных анимационных заданий на недостаточно мощном компьютере может привести к снижению скорости воспроизведения фильма. Данный параметр может принимать одно из следующих значений.
 - **Low** — приоритет полностью отдается быстродействию. В результате достигается максимальная скорость воспроизведения, но границы форм выглядят ступенчатыми и неровными вследствие полного отключения сглаживания.
 - **Auto Low** — приоритет на стороне быстродействия, но, по возможности, качество прорисовки поддерживается на приемлемом уровне. В начале воспроизведения сглаживание отключено, однако, если ресурсов процессора оказывается достаточно для более качественной прорисовки, сглаживание может быть включено.
 - **Auto High** — в начале воспроизведения устанавливается равное соотношение между скоростью воспроизведения и качеством прорисовки кадров; сглаживание включено. Однако если ресурсов процессора недостаточно для поддержания требуемой скорости воспроизведения, сглаживание отключается.
 - **Medium** — частичное сглаживание графики. Растворные изображения не сглаживаются вообще.
 - **High** — приоритет отдается качеству прорисовки, сглаживание включено всегда. Если фильм не содержит анимации, растворные изображения сглаживаются; при наличии анимации сглаживание растворных изображений отключено.
 - **Best** — приоритет полностью отдается качеству прорисовки. Полное сглаживание включено всегда вне зависимости от того, обеспечивается ли требуемая скорость воспроизведения или нет.

□ **Window Mode** — данный список позволяет задать параметры размещения Flash-фильма на странице HTML относительно ее содержимого.

- **Window** — Flash-фильм воспроизводится в своем собственном прямоугольном окне.
- **Opaque Windowless** — окно Flash-фильма имеет непрозрачный фон и перекрывает другие элементы HTML-страницы.
- **Transparent Windowless** — окно Flash-фильма имеет прозрачный фон, сквозь который просвечивает содержимое нижележащих элементов HTML. Использование данного параметра в сочетании с DHTML слоями позволяет реализовать достаточно интересные анимационные и интерактивные эффекты, связанные с тем, что Flash-содержимое "плавает" над элементами HTML.

Безоконный режим (Windowless) поддерживается только следующими версиями браузеров: Internet Explorer 5.0 и выше, Netscape Navigator 7.0 и выше, Опера 6 и выше, Mozilla 1.0 и выше, AOL/Compuserve.

□ **HTML alignment** — данный список предназначен для задания параметров выравнивания окна Flash Player относительно границ окна браузера. Позиционирование Flash-фильма на странице имеет смысл осуществлять в соответствующем HTML-редакторе (например, Dreamweaver) или вручную, поскольку данный параметр не позволяет добиться желаемых результатов.

□ **Scale** — данный список позволяет указать параметры масштабирования фильма внутри окна Flash Player. Различные варианты масштабирования имеют смысл в ситуации, когда размер окна и размер фильма не совпадают. Предлагаются следующие варианты масштабирования (рис. 17.5).

- **Default (Show all)** — этот параметр используется по умолчанию. Фильм масштабируется пропорционально таким образом, чтобы все содержимое рабочей области отображалось в окне. При этом возможна ситуация, когда будет видна часть вспомогательной области.
- **No Border** — фильм масштабируется пропорционально таким образом, чтобы полностью заполнить все окно Flash Player. При этом возможна ситуация, когда часть фильма будет срезана границами окна Flash Player.
- **Exact Fit** — фильм масштабируется таким образом, чтобы полностью заполнить все окно Flash Player. Размер фильма подгоняется к размеру окна, при этом, если рабочая область фильма и окно имеют разные пропорции, фильма будет сжат или растянут.
- **No Scale** — масштабирования фильма не происходит; он отображается в окне Flash Player с теми же размерами, которые были заданы в рабочей среде.

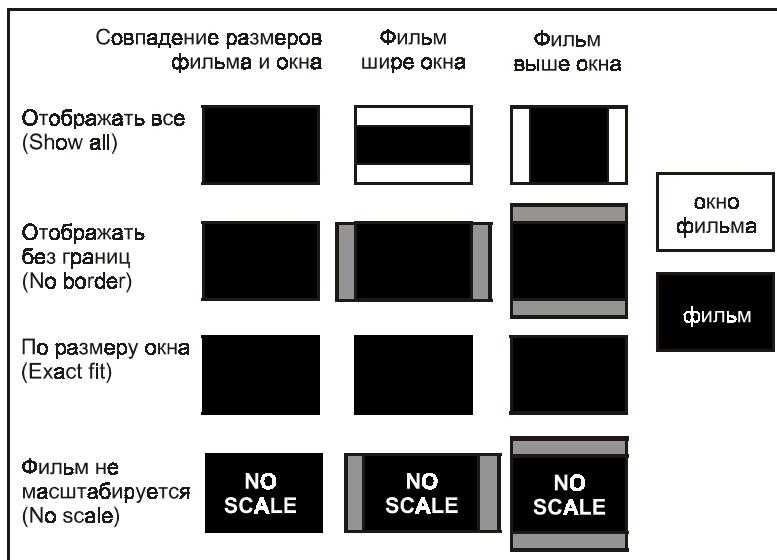


Рис. 17.5. Параметры масштабирования фильма в окне проигрывателя

- Flash Alignment — два данных списка позволяют задать параметры выравнивания фильма внутри окна Flash Player по вертикали и по горизонтали соответственно (рис. 17.6).

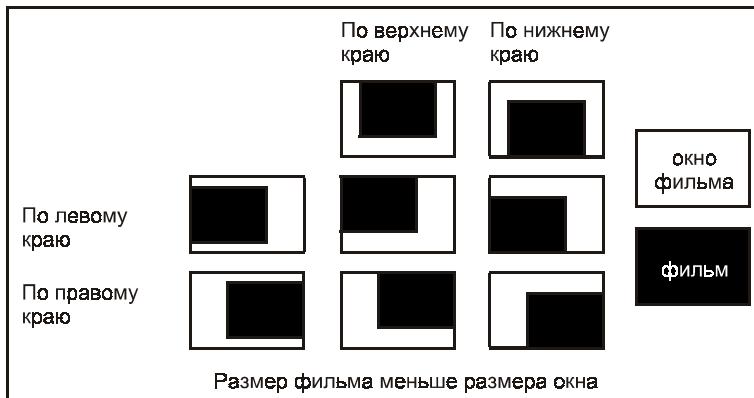


Рис. 17.6. Выравнивание фильма в окне проигрывателя

- Show Warning Messages — вывод сообщений об ошибках, возникающих вследствие конфликта заданных параметров публикации.

Формат GIF Image (gif)

Документ Flash может быть опубликован в графическом формате GIF (Graphic Interchange Format). Данный формат предназначен для хранения изображений, использующих индексированную цветовую палитру. В связи с тем, что данный формат не поддерживает глубину цвета более 8 бит на пиксель, максимальное число цветов, входящих в палитру, не может превышать 256, при этом допускается наличие прозрачных областей. Файл в формате GIF может содержать анимацию, представляющую собой набор сменяющих друг друга растровых изображений. Flash позволяет опубликовать исходный документ в формате GIF89a в виде статичного изображения или анимированной последовательности. Формат GIF использует мощный алгоритм сжатия LZW (Lemple-Zif-Welch), позволяющий уменьшить размер графического файла и, соответственно, время его загрузки по сети. Формат GIF хорошо подходит для изображений, содержащих сплошные цвета и четкие формы (например, для логотипов). При публикации в данный формат происходит потеря части цветовой информации, звука и интерактивности.

Вкладка **GIF** окна **Publish Settings** позволяет задать следующие параметры публикации (рис. 17.7).

- **Dimensions (Width и Height)** — данные поля содержат размеры результирующего изображения, заданные в пикселях. Если флажок **Match movie** установлен, размеры автоматически задаются равными размерам рабочей области. Снятие флажка **Match movie** позволяет задать произвольный размер результирующего изображения.
- **Playback** — данный раздел позволяет задать параметры воспроизведения результирующего GIF-файла.
 - **Static** — данное значение приводит к созданию статичного изображения, полученного на основе кадра основной монтажной линейки Flash-документа, содержащего метку **#Static**. Если данная метка отсутствует, изображение будет сгенерировано на основе первого кадра фильма.
 - **Animated** — данное значение позволяет сгенерировать анимированный GIF-файл на основе диапазона кадров основной монтажной линейки, первый кадр которого содержит метку **#First**, последний — метку **#Last**. Если данные метки отсутствуют, в результирующий GIF-файл войдут все кадры основной монтажной линейки исходного фильма. Следует иметь в виду, что программная анимация (созданная при помощи ActionScript) выполниться не будет.
 - **Loop continuously** — данное значение зацикливает воспроизведение GIF-файла.
 - **Repeat** — данное значение позволяет указать число повторов воспроизведения, которое задается в поле **times**.

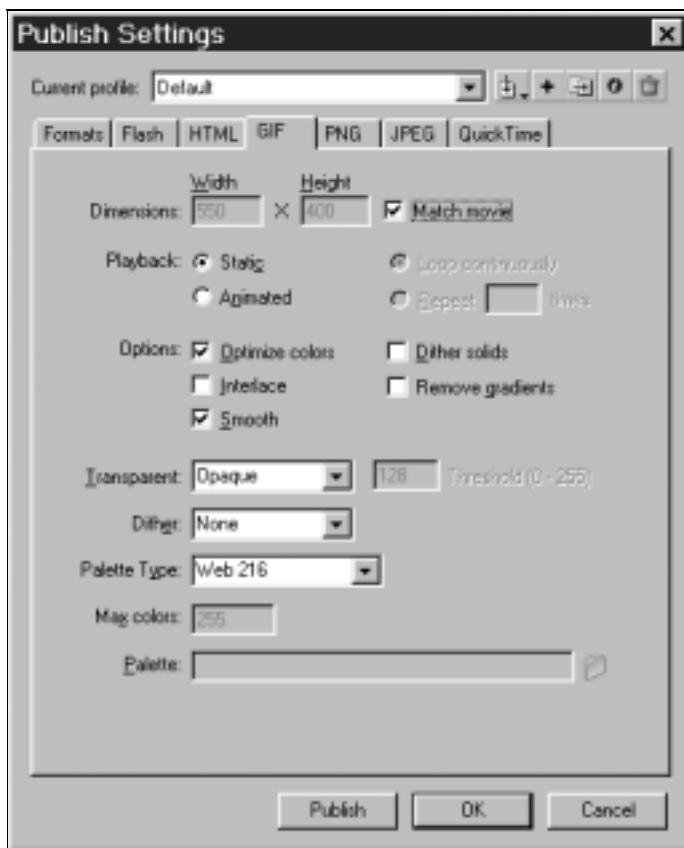


Рис. 17.7. Настройка параметров публикации в формат GIF

□ **Options** — данный раздел позволяет задать параметры, влияющие на передачу цветовой информации исходного документа в результирующий GIF-файл.

- **Optimize colors** — установка данного флагшка приводит к удалению из палитры GIF-файла всех неиспользуемых цветов, что позволяет уменьшить его размер. Применение этой возможности может уменьшить размер файла на 1–1,5 Кбайт без ухудшения качества изображения. Данная опция не оказывает влияния при использовании адаптированной (Adaptive) палитры.
- **Smooth** — установка данного флагшка позволяет добиться более высокого качества передачи изображения (снизить ступенчатость краев форм) и приводит к незначительному увеличению размера файла. В результате сглаживание вокруг границ формы, помещенной на

цветной фон, может возникнуть ореол (halo) из серых точек. В этом случае нужно либо отключить сглаживание, либо использовать прозрачный фон в экспортируемом файле GIF.

- **Interlace** — установка данного флашка приводит к применению чересстрочного чередования (Interlace), в результате чего изображение будет постепенно формироваться при загрузке по сети. Сначала происходит загрузка каждой восьмой строки изображения, затем каждой четвертой и т. д. Таким образом, при медленном соединении зритель может сначала оценить изображение, построенное по одной восьмой данных. Данный параметр не рекомендуется применять для анимированного GIF-файла.
 - **Dither solids** — обеспечивает использование смешения (dithering) для передачи сплошных цветов, отсутствующих в цветовой палитре файла. Поскольку GIF использует индексированную цветовую палитру, содержащую ограниченное число цветов, смешение позволяет имитировать отсутствующий цвет путем размещения в соответствующей области изображения пикселов, окрашенных цветами,ключенными в палитру. В результате оптического смешения человеческий глаз усредняет цветовую информацию и на определенном расстоянии воспринимает такую область как сплошной цвет.
 - **Remove gradients** — данная опция удаляет все используемые в документе градиенты, заменяя их сплошными (solid) цветами, соответствующими крайним слева цветовым порогам градиента.
- **Transparent** — данный список позволяет установить режим прозрачности результирующего GIF-изображения.
- **Opaque** — изображение не будет содержать прозрачных областей.
 - **Transparent** — области фона Flash-фильма в результирующем GIF-изображении будут сделаны прозрачными. При размещении такого изображения на HTML-странице сквозь области его фона будет виден фон страницы.
 - **Alpha** — данный режим позволяет задать пороговое значение прозрачности **Threshold**, на основании которого цвета, используемые в исходном фильме, будут сделаны либо полностью прозрачными, либо полностью непрозрачными. Порог прозрачности может быть задан в диапазоне от 0 до 255, что соответствует диапазону значений прозрачности от 0 до 100%, задаваемых в рабочей среде. Если значение прозрачности цвета в исходном фильме меньше указанного порогового значения, в результирующем GIF-изображении данный цвет будет сделан полностью прозрачным (т. е. его не будет видно). Если значение прозрачности цвета в исходном фильме больше указанного порогового значения, в результирующем GIF-изображении данный цвет

будет сделан полностью непрозрачным. Следует иметь в виду, что при перекрывании нескольких объектов, содержащих прозрачность, имеет место кумулятивный эффект, т. е. прозрачность данной области оценивается как сумма прозрачностей данных объектов.

- **Dither** — данный список позволяет задать метод смешения для имитации цветов, отсутствующих в палитре GIF. Применение смешения позволит добиться удовлетворительной передачи градиентов и фотографических изображений, используемых в исходном документе, однако при этом имеет место увеличение размера файла.
 - **None** — смешение не используется. Цвет, отсутствующий в палитре, будет заменен ближайшим к нему сплошным цветом.
 - **Ordered** — данный метод использует регулярную схему размещения точек, имитирующих отсутствующий цвет, и позволяет получить удовлетворительное качество передачи полутонаов при минимальном увеличении конечного объема файла.
 - **Diffusion** — данный метод применяет случайное размещение точек, имитирующих отсутствующий цвет, и обеспечивает наилучшую передачу полутонаов при значительном увеличении объема файла. Данный метод действует только при работе с безопасной Web-палитрой, состоящей из 216 цветов.
- **Palette Type** — данный список позволяет указать цветовую палитру, используемую результирующим GIF-файлом.
 - **Web 216** — использование безопасной Web-палитры, включающей 216 цветов, одинаково отображающихся на различных платформах и в различных браузерах.
 - **Adaptive** — применение адаптированной палитры, в которую войдут цвета, используемые в исходном фильме. Данная палитра обеспечивает точную передачу исходных цветов, но приводит к увеличению объема файла. Поле **Max colors** позволяет указать максимальное число используемых в палитре цветов, для того чтобы уменьшить объем конечного файла.
 - **Web Snap Adaptive** — данная палитра создается на основе используемых в исходном фильме цветов, но по возможности предпочтение отдается цветам из безопасной Web-палитры. Поле **Max colors** позволяет указать максимальное число используемых в палитре цветов.
 - **Custom** — это значение позволяет в качестве используемой палитры задать палитру, сохраненную во внешнем файле с расширением act (Adobe Color Table). Для этого в списке **Palette** необходимо указать путь к требуемому файлу, содержащему палитру.

Формат JPEG Image (jpeg)

Формат JPEG (Joint Photographic Experts Group) предназначен для хранения фотографических растровых изображений, обеспечивая при этом высокую степень сжатия. Формат JPEG позволяет сохранять изображения с глубиной цвета 24 бита. JPEG использует подход сжатия с потерями (lossy), в результате которого часть информации, в определенной степени несущественная для восприятия, идентифицируется и отбрасывается. В результате может быть достигнут высокий уровень сжатия без значительных потерь качества. В отличие от GIF формат JPEG не допускает возможность хранения анимированных изображений. При публикации Flash генерирует изображение на основе текущего кадра фильма (т. е. кадра, в котором во время публикации расположена воспроизводящая головка) и сохранит его во внешнем файле с расширением jpg.

Вкладка JPEG окна **Publish Settings** позволяет задать следующие параметры публикации (рис. 17.8).

- **Dimensions (Width и Height), Match movie** — данные поля и флажок позволяют задать размеры результирующего изображения и аналогичны соответствующим настройкам формата GIF.



Рис. 17.8. Настройка параметров публикации в формат JPEG

- **Quality** — этот параметр позволяет задать соотношение между степенью сжатия и качеством результирующего изображения в виде числа от 0 до 100. Малые значения параметра приводят к тому, что при сжатии отбрасывается большое количество информации, что позволяет получить файл меньшего размера и, соответственно, изображение худшего качества. Вы-

сокие значения ограничивают количество той информации, которой можно пренебречь при сжатии, что обеспечивает лучшее качество результирующего изображения и приводит к получению файла большего размера.

- **Progressive** — этот параметр аналогичен параметру **Interlace** для файла GIF. Использование данного режима приводит к тому, что при загрузке по сети изображение формируется постепенно по строкам. Изображения, использующие данную опцию, требуют больше ресурсов оперативной памяти компьютера для их отображения.

Формат PNG Image (png)

Формат PNG (Portable Network Graphics) предназначен для хранения растровых изображений. Данный формат поддерживает глубину цвета 24 бит на пикセル и позволяет хранить изображения, содержащие различные уровни прозрачности. Формат PNG использует метод сжатия без потерь (lossless), не приводящий к визуально заметному ухудшению качества.

Вкладка **PNG** окна **Publish Settings** позволяет задать следующие параметры публикации (рис. 17.9).

- **Dimensions (Width и Height)** — данные поля позволяют задать размеры результирующего изображения и аналогичны соответствующим настройкам форматов GIF и JPEG.
- **Bit depth (8-bit, 24-bit, 24-bit with Alpha)** — данный список позволяет задать глубину цвета результирующего изображения. Предлагаются несколько значений: **8-bit** — максимальное количество цветов не превышает 256, **24-bit** — применяется для хранения полноцветных фотографических изображений, **24-bit with Alpha** — добавляет к 24-битовому изображению 8-битный канал прозрачности (alpha channel), позволяющий реализовать 256 уровней прозрачности.
- **Options (Optimize colors, Smooth, Interlace, Dither solids, Remove gradients)** — данные параметры отвечают за передачу цветовой информации исходного документа в результирующий PNG-файл и аналогичны соответствующим параметрам формата GIF (*см. выше*).
- **Dither, Palette Type, Max colors, Palette** — данные параметры связаны с настройкой индексированной палитры и могут быть применены только для 8-битных изображений. Эти параметры аналогичны соответствующим параметрам формата GIF (*см. выше*).
- **Filter options** — список позволяет выбрать фильтрующую функцию, обрабатывающую данные изображения. Формат PNG использует алгоритм сжатия без потерь Deflation (Deflation Compression Algorithm). Для улучшения сжатия формат PNG использует специальные фильтры (filters), совершающие определенные преобразования данных, описывающих изо-

бражение. Программа кодирования указывает для каждой строки развертки, какая фильтрующая функция была использована. В результате после распаковки данных программа декодирования с помощью процесса, обратного фильтрации, восстанавливает исходное изображение.

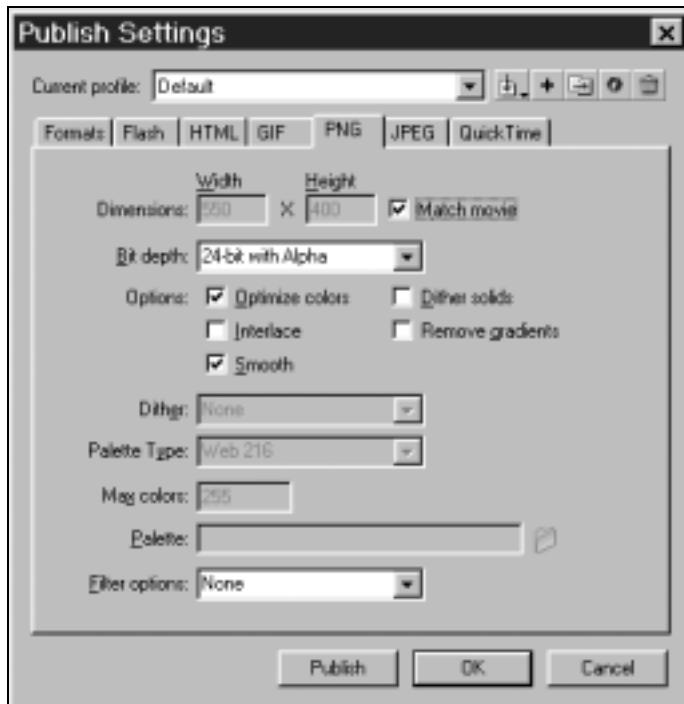


Рис. 17.9. Настройка параметров публикации в формат PNG

- **None** — фильтрация не применяется.
- **Sub** — передается значение разности между каждым байтом и значением соответствующего байта предыдущей точки раstra данной строки для изображений, у которых вдоль горизонтальной оси содержится однородная цветовая информация.
- **Up** — передается значение разности между каждым байтом и значением соответствующего байта вышележащей точки раstra. Данный фильтр может быть успешно применен к изображениям, цветовая информация в которых мало изменяется по вертикали.
- **Average** — значение текущей точки раstra вычисляется как среднее значение из значений соседних точек, расположенных слева и сверху.

- **Paeth** — используется линейная функция, которая строится по значениям трех соседних точек (левой, верхней и левой верхней), после чего в качестве значения текущей точки используется наиболее близкое к вычисленному (данный метод был предложен Аланом Паэтом (Alan W. Paeth)).
- **Adaptive** — данный фильтр создает цветовую палитру для результирующего файла, основываясь на цветах, используемых в исходном документе. При этом исходные цвета передаются более точно, однако применение данного фильтра приводит к увеличению объема файла.

Формат Windows Projector (exe)

Projector представляет собой исполняемый файл (с расширением exe), который включает в себя Flash-фильм, содержащий полную функциональность и автономный проигрыватель Flash Player, позволяющий воспроизвести данный фильм, даже если у пользователя не установлено программное обеспечение для просмотра Flash-содержимого. Размер проектора всегда больше размера соответствующего SWF-файла на тот объем, который занимает автономный проигрыватель. Для седьмой версии он составляет 964 Кбайт. Применение проектора чрезвычайно удобно для распространения Flash-проектов, предназначенных для локального просмотра, на электронных носителях или через сеть Инtranет (инtranet). При работе с проектором существует возможность контролировать параметры воспроизведения фильма (качество, режим экрана и т. д.) и выполнять различные действия (закрывать проектор, запускать другой исполняемый файл, находящийся в системе, и пр.) при помощи сценария ActionScript.

Публикация в данный формат не требует задания дополнительных параметров, необходимо только установить соответствующий флажок на вкладке **Formats**. Для создания проектора, предназначенного для просмотра на платформе Macintosh, необходимо установить флажок **Macintosh Projector**.

Примечание

Проектор также может быть создан непосредственно при воспроизведении SWF-файла из проигрывателя Flash Player при помощи команды **File>Create Projector**.

Формат Quick Time (mov)

Программный пакет Flash позволяет осуществлять публикацию исходного документа в формате Quick Time. При этом будет использован формат той версии Quick Time, которая установлена в системе. Quick Time поддерживает воспроизведение SWF-файлов, включая базовую интерактивность. Для

того чтобы Flash-содержимое корректно воспроизводилось в проигрывателе Quick Time, необходимо осуществлять публикацию для четвертой версии Flash Player. Если Flash-фильм содержит импортированный файл Quick Time, то он будет помещен на отдельную дорожку.

Вкладка **Quick Time** окна **Publish Settings** позволяет задать следующие параметры публикации (рис. 17.10).

- **Dimensions (Width и Height)** — данные поля используются для задания размеров фильма Quick Time

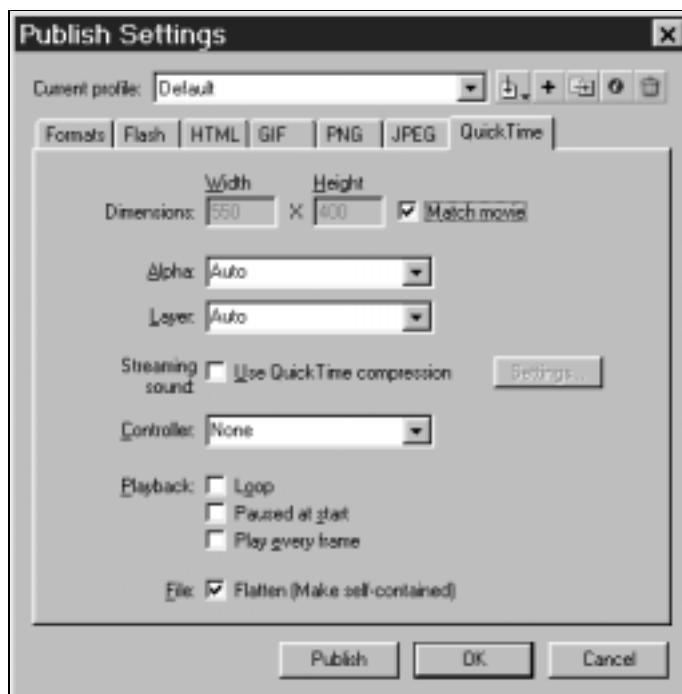


Рис. 17.10. Настройка параметров публикации в формат Quick Time

- **Alpha** — данный параметр позволяет настроить прозрачность дорожки Quick Time-фильма, на которую будет помещен Flash-фильм.
 - **Auto** — если Flash-фильм будет помещен поверх других дорожек Quick Time-фильма, то его фон делается прозрачным, если он находится на самой нижней дорожке или если кроме него других дорожек нет, то фон Flash-фильма делается непрозрачным.
 - **Alpha-transparent** — делает фон Flash-фильма полностью прозрачным.
 - **Copy** — делает фон Flash-фильма полностью непрозрачным.

- **Layer** — данный параметр управляет размещением дорожки Flash-фильма относительно других дорожек Quick Time.
 - **Auto** — дорожка Flash-фильма размещается перед всеми остальными, если фильм содержит импортированные в рабочую среду видеоролики Quick Time, расположенные на переднем плане по отношению к остальным объектам. В противном случае Flash-фильм будет помещен на самую нижнюю дорожку.
 - **Top** — дорожка Flash-фильма размещается над всеми остальными дорожками Quick Time-фильма.
 - **Bottom** — дорожка Flash-фильма размещается позади остальных дорожек Quick Time-фильма.
- **Streaming sound** — установка флагка **Use Quick Time compression** приводит к тому, что все потоковые звуки (к которым был применен тип синхронизации **Stream**) будут помещены на отдельную звуковую дорожку Quick Time-фильма и к ним будут применены стандартные параметры сжатия Quick Time. Для настройки этих параметров можно воспользоваться кнопкой **Settings** (информация о стандартных настройках компрессии звука Quick Time содержится в документации к данному приложению).
- **Controller** — данный параметр позволяет указать, какой тип контроллера будет использован для управления воспроизведением фильма: **None** (отсутствует), **Standart** (стандартные элементы управления), **Quick Time VR** (панель управления для панорамных фильмов Quick Time).
- **Playback** — параметры воспроизведения фильма.
 - **Loop** — при установке флагка воспроизведение фильма Quick Time будет зациклено.
 - **Paused at start** — воспроизведение не начнется до того момента, пока не будет выполнено действие, инициирующее начало воспроизведения (использование элемента управления проигрывателя Quick Time или нажатие кнопки Flash-фильма, содержащей соответствующий сценарий).
 - **Play every frame** — все кадры фильма будут воспроизведены, даже если при этом не обеспечивается требуемая скорость воспроизведения. В этом режиме звуки воспроизводиться не будут.
- **File Flatten (Make self-contained)** — установка этого флагка приведет к тому, что все содержимое исходного документа будет помещено в один фильм в формате Quick Time. Если данный флагок снят, все импортированные в исходный документ видеофайлы не будут встроены в фильм Quick Time, а будут связаны с ним внешним образом. В этом случае для корректного воспроизведения необходимо наличие всех связанных файлов.

Экспорт документа

В Flash MX 2004 имеется возможность экспорта содержимого Flash-документа в различные форматы, которые могут быть использованы в других приложениях. Исходный документ может быть экспортирован в виде статичного изображения, сформированного на основе текущего кадра, либо в виде анимационной последовательности, сгенерированной на основе содержимого кадров основной монтажной линейки. Кроме того, существует возможность экспорта звука, используемого в документе в формат WAV. Форматы файлов, которые поддерживаются при экспорте исходного документа, представлены в табл. 17.1.

Таблица 17.1. Форматы файлов, поддерживаемые при экспорте Flash-документа

Формат файла	Возможность экспорта последовательности файлов	Расширение	Windows	Macintosh
Векторные форматы				
Документ Flash (SWF)	+	swf	+	+
Adobe Illustrator	+	ai	+	+
AutoCAD DXF	+	dxf	+	+
Растровые форматы				
Анимированный GIF-файл и статичное GIF-изображение	+	gif	+	+
JPEG	+	jpg	+	+
PNG	+	png	+	+
Bitmap (BMP)	+	bmp	+	
EPS 3.0 с возможностью Preview	+	eps	+	+
Метафайлы (содержат информацию о векторном и растровом представлении изображения)				
Windows Metafile	+	wmf	+	
Enhanced Metafile (Windows)	+	emf	+	

Таблица 17.1 (окончание)

Формат файла	Возможность экспорта последовательности файлов	Расширение	Windows	Macintosh
Видеоформаты				
Macromedia Flash Video (FLV)		flv	+	+
Windows AVI		avi	+	
QuickTime		mov	+	+
Звуковые форматы				
WAV audio		wav	+	

Экспорт изображения

Для того чтобы экспортировать содержимое текущего кадра (т. е. кадра, в котором в момент экспортации находится воспроизводящая головка), необходимо выполнить команду главного меню **File>Export>Export Image**. Следует иметь в виду, что при выполнении данной команды будет экспортироваться текущий кадр текущей монтажной линейки (фильма или символа).

Документ Flash (SWF)

Параметры настройки экспорта в данный формат аналогичны соответствующим параметрам публикации. В результате экспортации будет получен SWF-файл, содержащий текущий кадр основной монтажной линейки или монтажной линейки символа. При этом будет выполняться только та часть функциональности фильма, которая содержится в текущем кадре и не зависит от воспроизведения основной монтажной линейки (например, анимация экземпляров анимированных символов типа муви-клип, содержащихся в текущем кадре или сценарии объектов текущего кадра).

Adobe Illustrator

В Flash можно осуществлять экспорт изображений в формат Adobe Illustrator (ai) версии 6.0 и ранее. Следует иметь в виду, что поддержка градиентов осуществляется, только начиная с пятой версии Adobe Illustrator, а поддержка растровой графики с версии Adobe Illustrator 6. При экспортации в данный формат появляется диалоговое окно **Export Adobe Illustrator**, предлагающее выбрать версию Adobe Illustrator 88, 3.0, 5.0 или 6.0.

GIF

При экспорте в данный формат в диалоговом окне **Export GIF** можно задать следующие параметры.

- Dimensions** — размеры изображения.
- Resolution** — разрешение изображения (этот параметр влияет на размеры изображения, полученного в результате растирования). Кнопка **Match Screen** задает размер изображения, соответствующий его отображению на экране монитора при данном экранном разрешении.
- Include** — данный параметр определяет, какая часть содержимого кадра войдет в результирующее изображение.
 - **Minimum Image Area** — в изображение будут включены все находящиеся на сцене объекты, причем его размер определяется размером прямоугольной области, которую занимают графические объекты сцены (расположенные как в рабочей, так и во вспомогательной областях).
 - **Full Document Size** — в изображение будет включена вся рабочая область с расположенными в ней объектами и фоном; объекты, размещенные во вспомогательной области, не войдут в результирующее изображение.
- Colors** — данный список позволяет задать число используемых в цветовой палитре цветов.
- Interlace, Smooth, Transparent, Dither Solid colors** — чересстрочное чередование, сглаживание, прозрачность фона, смешение для сплошных цветов.
(Эти параметры были рассмотрены в разд. "Настройка параметров публикации" данной главы.)

JPEG, PNG, Bitmap

Параметры экспорта в данные форматы аналогичны рассмотренным в разд. "Настройка параметров публикации" данной главы.

Windows Metafile, Enhanced Metafile, EPS 3.0, AutoCAD DXF

Экспорт в данный формат не требует настройки параметров.

Экспорт фильма

Экспорт фильма осуществляется при помощи команды главного меню **File>Export>Export Movie**.

Документ Flash (SWF)

Результат экспорта аналогичен публикации проекта в формате Flash. Параметры экспорта аналогичны параметрам публикации в данный формат (см. разд. "Настройка параметров публикации" данной главы).

Последовательности изображений Adobe Illustrator, AutoCAD DXF, GIF, JPEG, PNG, Bitmap, EPS 3.0, Windows Metafile, Enhanced Metafile

В результате экспорта в любой из данных форматов будет сгенерирована последовательность файлов, пронумерованных четырехзначным числом. Каждый кадр основной монтажной линейки фильма будет помещен в отдельный графический файл. Следует иметь в виду, что в последовательности будет отражена только анимация основной монтажной линейки, анимация вложенных муви-клипов не будет видна.

Параметры, задаваемые при экспорте, аналогичны параметрам экспорта изображения в соответствующий формат и будут применены к каждому члену результирующей последовательности файлов.

Animated GIF

Экспорт в данный формат приводит к созданию анимированного GIF-файла, включающего последовательность изображений, сгенерированных на основе кадров основной монтажной линейки. При этом в результирующем файле будет видна только анимация основной монтажной линейки, анимация муви-клипов отрабатываться не будет. Параметры экспорта анимированного GIF-файла аналогичны соответствующим параметрам, задаваемым для экспорта статичного GIF-изображения. Параметр **Animation Repetition** позволяет задать количество повторов при воспроизведении конечного файла. Значение 0 приводит к зацикливанию анимации.

Macromedia Flash Video (FLV)

В Flash имеется возможность экспортировать видеофайл из исходного документа в формате FLV (Macromedia Flash Video). Это действие выполняется не при помощи команды **Export Movie**, а посредством контекстного меню библиотеки документа, содержащего требуемый видеоролик. Для выполнения экспорта необходимо выделить внедренный видеофайл в окне библиотеки и, выполнив команду **Properties** контекстного меню библиотеки, в появившемся диалоговом окне **Embedded Video Properties** нажать кнопку **Export**.

Windows AVI

При экспорте в данный видеоформат в конечном файле будет отражена только анимация основной монтажной линейки. Анимация муви-клипов

работать не будет. Следует иметь в виду, что при экспорте документа, содержащего большое количество кадров, размер результирующего файла может достигать достаточно большого объема. При экспорте могут быть заданы следующие параметры.

- Dimensions** — размеры кадров AVI-фильма. Флажок **Maintain Aspect Ratio** позволяет сохранить исходные пропорции.
- Video Format** — глубина цвета.
- Compress Video** — при установке данного флажка результирующий видеофайл будет сжат с использованием одного из предлагаемых алгоритмов, который можно выбрать в окне **Video Compression**, появляющемся при экспорте.
- Smooth** — сглаживание позволяет получить изображение лучшего качества, однако может привести к появлению ореола вокруг границ форм на цветном фоне.
- Sound Format** — данный список позволяет задать частоту дискретизации, разрядность и количество каналов (стерео или моно) для используемого в фильме звука. Значение **Disable** отключает экспорт звука.

QuickTime

Параметры экспорта в данные форматы аналогичны рассмотренным в *разд. "Настройка параметров публикации"* данной главы.

WAV audio

При экспорте звука, используемого в фильме, в диалоговом окне **Export Windows WAV** можно задать следующие параметры:

- Sound format** — данный список позволяет задать частоту дискретизации, разрядность и количество каналов (стерео или моно) для используемого в фильме звука. Значение **Disable** отключает экспорт звука;
- Ignore event sound** — событийные звуки не экспортируются.

Печать документа

Для настройки параметров печати необходимо выполнить команду главного меню **File>Page Setup**.

В диалоговом окне **Page Setup** доступны следующие параметры (рис. 17.11).

- Margins (inches)** — поля от границы изображения до края печатного листа (**Left** — слева, **Right** — справа, **Top** — сверху, **Bottom** — снизу). Единицы измерения полей даны в дюймах.

- Center (Horizontal/Vertical)** — отцентрировать по горизонтали/вертикали относительно печатного листа.

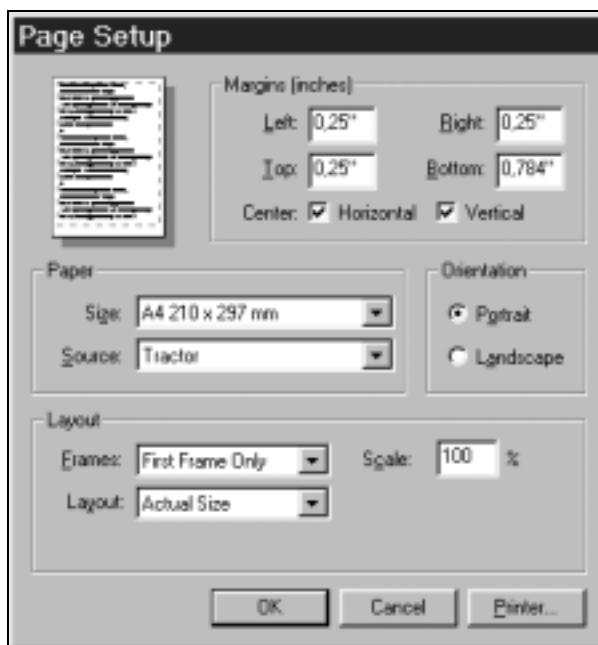


Рис. 17.11. Диалоговое окно настройки параметров печати **Page Setup**

Paper — печатный лист.

- **Size** — данный список позволяет задать стандартные размеры печатного листа.
- **Source** — вид подачи бумаги на принтер.

Orientation — ориентация изображения на странице (**Portrait/Landscape** — книжная/альбомная).

Layout — порядок размещения кадров фильма.

- **Frames** — кадры (**First Frames Only** — печатать только первый кадр, **All frames** — печатать все кадры).
- **Actual Size** — печать изображения с размерами, заданными в фильме, **Scale** — масштаб изображения.
- **Fit On One Page** — уместить все на одной странице.

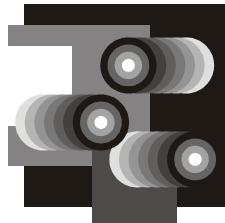
- **Storyboard-Boxes** — печать раскадровки. Каждый кадр помещается в прямоугольную рамку.
 - ◊ **Frames across** — количество кадров в строке.
 - ◊ **Frame margin** — размер поля между кадрами в текущих единицах, используемых в рабочей среде.
 - ◊ **Label frames** — указать имя сцены и номер кадра при печати.
- **Storyboard-Grid** — печать раскадровки. Каждый кадр помещается в ячейку сетки.
- **Storyboard-Blank** — печать раскадровки. Кадры не разделяются ограничителями.

Для печати кадров фильма необходимо выполнить команду главного меню **File>Print**

Кроме печати документа непосредственно из рабочей среды Flash, можно выполнять печать из проигрывателя Flash Player. Для этого используется команда меню проигрывателя **File>Print** либо создается сценарий ActionScript, позволяющий выполнять печать требуемых кадров.

Для того чтобы при печати из Flash Player обеспечить возможность распечатки отдельных кадров, необходимо в рабочей среде снабдить их метками **#p**. Это позволит пользователю при выполнении команды **File>Print** главного или контекстного меню проигрывателя указать диапазон кадров, которые будут выведены на печать, в диалоговом окне **Print**. Следует иметь в виду, что нумерация кадров диапазона печати указывается в соответствии с расстановкой меток **#p**, а не с их порядковым номером на монтажной линейке. При тестировании документа наличие одинаковых меток приведет к выводу предупреждающего сообщения в окне **Output**, однако когда речь идет о метках **#p**, их многократное применение является вполне корректным.

Вывод на печать из проигрывателя Flash Player можно также организовать программным образом при помощи функций `print()` и `printAsBitmap()`, а также методов нового класса ActionScript `PrintJob`, поддерживаемого только проигрывателем Flash Player 7 (и выше). Информацию о данных возможностях можно найти в справочной системе ActionScript Dictionary, поставляемой вместе с Flash MX 2004.



Глава 18

Flash и HTML

Нет правил без исключений, но исключения не нарушают правила.

Сенека Луций Анней Младший

Теги HTML, используемые для размещения Flash-ролика

Для того чтобы поместить объект Flash на HTML-страницу, необходимо использовать набор определенных тегов, указывающих браузеру, какую программу необходимо применить для его обработки и какие параметры отображения и воспроизведения должны быть при этом установлены. Браузеры Microsoft Internet Explorer и Netscape Navigator используют различные подходы к обработке Flash-содержимого, и в связи с этим для отображения Flash-фильма в этих браузерах используются различные теги.

Браузер Internet Explorer использует для обработки объекта Flash элемент управления ActiveX Flash Player. Язык ActiveX представляет собой набор технологий, разработанных компанией Microsoft, которые позволяют реализовать широкие возможности интерактивности в сети Интернет. Элементы управления ActiveX — это программные компоненты, создаваемые различными фирмами-разработчиками, которые можно вставить в Web-страницу для использования функциональных возможностей, реализованных в различных программных пакетах. Для размещения элемента ActiveX на HTML-странице используется парный тег `<object>`. Для просмотра Flash-фильма в браузере Internet Explorer необходимо наличие элемента ActiveX Flash Player, версия которого соответствует версии, указанной при публикации исходного документа.

Браузер Netscape Navigator использует для обработки Flash-содержимого подключаемый модуль (plug-in), который загружается самим браузером и размещается на HTML-странице при помощи парного тега `<embed>`. Для просмотра Flash-фильма в браузере Netscape Navigator необходимо наличие соответствующей версии подключаемого модуля.

Для того чтобы обеспечить возможность корректного отображения Flash-фильма как в одном, так и в другом браузере, необходимо разместить на странице оба набора тегов с соответствующими атрибутами, причем тег `<embed>` должен быть вложенным по отношению к тегу `<object>`. Браузер будет обрабатывать только тот набор тегов, который предназначен для него, игнорируя остальные. Шаблоны публикации, поставляемые вместе с Flash MX 2004, позволяют автоматически генерировать все необходимые теги для обоих браузеров (листинг 18.1).

Следует отметить, что при использовании HTML-шаблонов, поставляемых вместе с Flash MX 2004, генерированный код разметки соответствует стандарту XHTML. XHTML — это обычный HTML, который записывается в соответствии с синтаксическими правилами оформления XML-документов, делая web-страницу XML-совместимой. Для соответствия стандарту XHTML в HTML должны выполняться следующие правила:

- все теги должны быть записаны строчными буквами;
- все теги должны быть закрыты. Если элемент не имеет закрывающего тега, то после его названия необходимо использовать наклонную черту, например ``;
- вложенность тегов должна быть корректной;
- все атрибуты тегов должны быть заключены в кавычки.

Листинг 18.1. Образец кода HTML, генерированного на основе шаблона Flash Only

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=koi8-r" />
<title>walkingMan</title>
</head>
<body bgcolor="#999999">
<!--url's used in the movie-->
<a href="http:www.wherever.com"></a>
<a href="http:www.anywhere.com"></a>
<!--text used in the movie-->
<!--
<p align="right"><font face="Crystal" size="38" color="#000000">Text
 content</font></p>
-->
```

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=7,0,0,0" width="550" height="400" id="walkingMan"
  align="left">

<param name="allowScriptAccess" value="sameDomain" />
<param name="movie" value="walkingMan.swf" />
<param name="play" value="false" />
<param name="loop" value="false" />
<param name="menu" value="false" />
<param name="quality" value="medium" />
<param name="scale" value="noscale" />
<param name="salign" value="lt" />
<param name="wmode" value="opaque" />
<param name="devicefont" value="true" />
<param name="bgcolor" value="#999999" />

<embed src="walkingMan.swf" play="false" loop="false" menu="false"
  quality="medium" scale="noscale" salign="lt" wmode="opaque"
  devicefont="true" bgcolor="#999999" width="550" height="400"
  name="walkingMan" align="left" allowScriptAccess="sameDomain"
  type="application/x-shockwave-flash"
  pluginspage="http://www.macromedia.com/go/getflashplayer" />

</object>
</body>
</html>
```

В результате публикации в формате HTML автоматически сгенерированный код, кроме тегов `<object>` и `<embed>`, описывающих собственно Flash-объект, может содержать еще ряд тегов, созданных на основе используемого шаблона.

Тег `<!DOCTYPE>` содержит указание на DTD (Document Type Definition, определение типа документа), содержащее формальное описание языка разметки, используемого данным документом. Тег `<meta>` включает информацию о содержимом документа и может быть использован поисковыми системами для вывода результатов поиска. Используемые в фильме ссылки на внешние сетевые ресурсы (реализуемые, например, с помощью функции `getURL()`) оформляются в виде комментариев. Это позволяет осуществлять проверку правильности ссылок в конечном проекте, но приводит к увеличению объема файла. Кроме того, Flash помещает в код страницы комментарии, содержащие перечень используемого в фильме текста с указанием параметров его форматирования. Эта возможность позволяет поисковым системам обрабатывать данную информацию, но также приводит к увеличению размера файла.

При использовании тега `<object>` параметры, управляющие отображением и воспроизведением Flash-фильма, и принимаемые ими значения, указываются при помощи атрибутов `name` и `value` вложенных в него тегов `<param>`. Таким образом, значения параметров задаются в виде:

```
name = "параметр" value = "значение"
```

При использовании тега `<embed>` параметры указываются непосредственно как пары `параметр = "значение"` при помощи соответствующих атрибутов.

Атрибуты и параметры тегов `<object>` и `<embed>`

Атрибут `classid`

Данный атрибут является *обязательным*.

Атрибут `classid` содержит уникальный алфавитно-цифровой идентификатор (GUID — Global Unique IDentifier), связывающий элемент управления ActiveX Flash Player с реестром Windows. Эта информация необходима для того, чтобы указать браузеру, какой именно компонент требуется для обработки Flash-содержимого. Обнаружив тег `<object>`, браузер просматривает содержимое реестра на клиентском компьютере в поисках данного идентификатора и при его наличии загружает соответствующий компонент ActiveX. Если идентификатор не найден в реестре, то соответствующий компонент может быть загружен с web-сервера компании Macromedia по адресу, заданному в атрибуте `codebase`. Данный атрибут может быть использован только для тега `<object>`.

Допустимые значения атрибута `classid` таковы:

```
clsid:d27cdb6e-ae6d-11cf-96b8-444553540000
```

Атрибут `type`

Данный атрибут является *обязательным*.

Атрибут `type` указывает браузеру Netscape Navigator, каким типом MIME обладает загружаемый на страницу элемент. Каждый тип файла содержит уникальный заголовок, называемый типом MIME (Multipurpose Interchange Mail Extensions), указывающий на встраиваемый модуль (plug-in), обрабатывающий данный файл. Реестр Windows содержит перечень всех типов MIME и соответствующих им модулей расширения. Обнаружив тег `<embed>`, браузер обрабатывает тип MIME, заданный атрибутом `type`, в соответствии с настройками реестра, вызывая требуемый модуль расширения. Данный атрибут может быть использован только для тега `<embed>`.

Допустимые значения атрибута `type` таковы:

```
application/x-shockwave-flash
```

Атрибут **codebase**

Данный атрибут является *обязательным*.

Атрибут `codebase` содержит URL, по которому находится элемент управления ActiveX Flash Player. Если данный элемент управления еще не установлен у пользователя, он будет автоматически загружен с указанного адреса. Данный атрибут может быть использован только для тега `<object>`.

Допустимые значения атрибута `codebase` таковы:

`http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,0,0`

Атрибут **pluginspage**

Данный атрибут является *обязательным*.

Атрибут `pluginspage` содержит URL, где находится встраиваемый модуль, который может быть автоматически загружен, если он не установлен у пользователя.

Допустимые значения атрибута `pluginspage` таковы:

`http://www.macromedia.com/go/getflashplayer`

Атрибут **width**

Данный атрибут является *обязательным*.

Атрибут `width` задает ширину окна Flash Player при отображении на HTML-странице. Значение может быть указано в пикселях или в процентах относительно ширины окна браузера. Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе значения, заданного в поле **Width** окна настройки параметров публикации **Publish Settings>HTML**.

Допустимые значения атрибута `width`:

Положительные целые значения в пикселях или процентах.

Атрибут **height**

Данный атрибут является *обязательным*.

Атрибут `height` задает высоту окна Flash Player при отображении на HTML-странице. Значение может быть указано в пикселях или в процентах относительно ширины окна браузера. Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе значения, заданного в поле **Height** окна настройки параметров публикации **Publish Settings>HTML**.

Допустимые значения атрибута `height`:

Положительные целые значения в пикселях или процентах

Атрибут *id*

Атрибут *id* является *необязательным*.

Данный атрибут предназначен для идентификации элемента ActiveX Flash Player в сценариях или VBS. Значение атрибута *id* представляет собой имя, идентифицирующее Flash-фильм, при помощи которого к нему можно обращаться из сценариев JavaScript и VBS. По умолчанию идентификатор задается как строка, соответствующая имени SWF-файла.

Атрибут *align*

Атрибут *align* является *необязательным*.

Данный атрибут задает параметры выравнивания окна фильма относительно других элементов HTML-страницы. Результат действия этого параметра бывает трудно предсказать в ситуации, когда на странице имеется достаточно сложное содержание (помимо Flash-фильма). Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе значения, заданного в поле **HTML_alignment** окна настройки параметров публикации **Publish Settings>HTML**.

Допустимые значения атрибута *align*:

baseline, top, middle, bottom, texttop, absmiddle, absbottom, left, right

Параметр *movie*

Атрибут *movie* является *обязательным*.

Этот параметр определяет путь к файлу SWF, который должен быть отображен на странице. При использовании относительной адресации ссылка разрешается относительно местоположения HTML-страницы, содержащей Flash-фильм. Если путь к файлу SWF указан неверно, при загрузке в строке состояния браузера будет выводиться информация, свидетельствующая о выполнении загрузки, однако Flash-фильм так и не сможет быть загружен. Параметр *movie* можно использовать только для тега `<object>`. Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе взаимного расположения файлов SWF и HTML, определяемого значениями, заданными в соответствующих полях **Filename** окна настройки параметров публикации **Publish Settings>Formats**.

Допустимые значения атрибута *movie*:

Путь к файлу SWF, например, `swf/intro/logo.swf`

Атрибут *src*

Атрибут *src* является *обязательным*.

Данный атрибут определяет путь к файлу SWF, который должен быть отображен на странице. Атрибут *src* является аналогом параметра *movie*, но

может быть использован только для тега <embed>. Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе взаимного расположения файлов SWF и HTML, определяемого значениями, заданными в соответствующих полях **Filename** окна настройки параметров публикации **Publish Settings>Formats**.

Допустимые значения атрибута src:

Путь к файлу SWF, например, swf/intro/logo.swf

Параметр/атрибут play

Параметр **play** является *необязательным*.

Этот параметр определяет, будет ли воспроизведение Flash-фильма начато автоматически или нет. Если данный параметр отсутствует, по умолчанию используется значение `true`. Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе флажка **Paused at start** окна настройки параметров публикации **Publish Settings>HTML**.

Допустимые значения атрибута play:

`true, false`

Параметр/атрибут loop

Параметр **loop** является *необязательным*.

Этот параметр определяет, будет ли воспроизведение монтажной линейки Flash-фильма зациклено или нет. Если данный параметр отсутствует, по умолчанию используется значение `true`. Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе флажка **Loop** окна настройки параметров публикации **Publish Settings>HTML**.

Допустимые значения параметра loop:

`true, false`

Параметр/атрибут menu

Параметр **menu** является *необязательным*.

Этот параметр определяет, будут ли доступны команды изменения масштаба просмотра и управления воспроизведением фильма, содержащиеся в контекстном меню проигрывателя Flash Player, которое выводится при щелчке по окну проигрывателя правой кнопкой мыши. Если данный параметр отсутствует, по умолчанию используется значение `true`. Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе флажка **Display menu** окна настройки параметров публикации **Publish Settings>HTML**.

Допустимые значения параметра menu:

`true, false`

Параметр/атрибут *quality*

Параметр *quality* является *необязательным*.

Этот параметр позволяет задать соотношение между качеством прорисовки кадров фильма и скоростью воспроизведения. Если данный параметр отсутствует, по умолчанию используется значение *.autohigh*. Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе значения, указанного в поле **Quality** окна настройки параметров публикации **Publish Settings>HTML**.

Допустимые значения параметра *quality*:

low, high, autolow, autohigh, best

Параметр/атрибут *scale*

Параметр *scale* является *необязательным*.

Этот параметр указывает, каким образом фильм будет отмасштабирован в пределах окна Flash Player. Если данный параметр отсутствует, по умолчанию используется значение *showall*. Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе значения, указанного в поле **Scale** окна настройки параметров публикации **Publish Settings>HTML**.

Допустимые значения параметра *scale*:

showall, noborder, exactfit, noscale

Параметр/атрибут *salign*

Параметр *salign* является *необязательным*.

Этот параметр определяет режим выравнивания фильма в пределах окна Flash Player. Применение данного параметра имеет смысл в том случае, когда размер фильма отличается от размера окна. Если параметр *salign* опущен, фильм размещается в центре окна. Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе значений, указанных в полях **Flash alignment** окна настройки параметров публикации **Publish Settings>HTML**.

Допустимые значения параметра *salign* приводятся в табл. 18.1.

Таблица 18.1. Значения параметра/атрибута *salign*

Значения	Выравнивание по вертикали	Выравнивание по горизонтали
"T"	Верх	Центр
"B"	Низ	Центр
"L"	Центр	По левому краю
"R"	Центр	По правому краю

Таблица 18.1 (окончание)

Значения	Выравнивание по вертикали	Выравнивание по горизонтали
"TL"	Верх	По левому краю
"TR"	Верх	По правому краю
"BL"	Низ	По левому краю
"BR"	Низ	По правому краю

Параметр/атрибут *wmode*

Параметр *wmode* является *необязательным*.

Данный параметр определяет, будет ли фильм воспроизводиться в своем собственном прямоугольном окне на странице или будет помещен над остальными ее элементами; при этом фон фильма может быть непрозрачным или прозрачным. Если параметр *wmode* опущен, то будет использоваться оконный режим *window*. Данный фрагмент кода может быть сгенерирован автоматически при публикации на основе значения, указанного в поле **Window Mode** окна настройки параметров публикации **Publish Settings>HTML**.

Допустимые значения параметра *wmode*:

window, opaque, transparent

Параметр/атрибут *bgcolor*

Параметр *bgcolor* является *необязательным*.

Данный параметр позволяет задать цвет фона фильма и таким образом переопределить настройки, заданные в окне **Document Properties** рабочей среды. Если данный параметр опущен, в качестве цвета фона будет использован цвет, заданный в рабочей среде. При публикации данному параметру по умолчанию присваивается значение, соответствующее цвету фона фильма.

Допустимые значения параметра *bgcolor*:

Любое шестизначное шестнадцатеричное число, перед которым стоит префикс *#*, например, *#FF0000*

Параметр/атрибут *devicefont*

Параметр *devicefont* является *необязательным*.

Данный параметр указывает, будет ли статический текст, для которого в рабочей среде не был установлен флажок **Use Device Fonts**, отображаться с использованием системного шрифта при наличии соответствующего шрифта у пользователя. Если данный параметр опущен, по умолчанию используется значение *false*.

Допустимые значения параметра `devicefont`:

true или false

Параметр/атрибут *base*

Параметр `base` является *необязательным*.

Данный параметр используется при наличии в фильме относительных ссылок на внешний сетевой ресурс (например HTML-страницу). При использовании относительной адресации по умолчанию все ссылки разрешаются относительно местоположения HTML-страницы, содержащей Flash-фильм. Применение параметра `base` позволяет задать базовую директорию или произвольный URL, используемый в качестве базового, для разрешения относительных ссылок. Эта возможность бывает достаточно полезна в ситуациях, когда различные файлы проекта находятся в разных директориях.

Допустимые значения параметра `base`:

Относительный путь к директории или абсолютный URL, относительно которых будут разрешаться все используемые в фильме относительные ссылки, например, `../html_gallery/images`.

Параметр/атрибут *allowScriptAccess*

Параметр `allowScriptAccess` является *необязательным*.

Этот параметр связан с новой междоменной политикой безопасности, введенной с выходом седьмой версии Flash Player. Он указывает, в каких случаях разрешается осуществлять программное взаимодействие Flash-фильма со сценарием JavaScript посредством функций `getURL()` и `fscommand()`. По умолчанию данный параметр принимает значение `samedomain`.

Допустимые значения параметра `allowScriptAccess`:

- `always` — программное взаимодействие осуществляется без ограничений;
- `never` — программное взаимодействие не осуществляется;
- `samedomain` — программное взаимодействие осуществляется только в случае, когда фильм Flash расположен в том же домене, что и HTML-страница.

Работа с шаблонами HTML

При необходимости создания множества однотипных HTML-страниц, содержащих фильмы Flash, бывает удобно создать свой собственный шаблон, на основе которого могут быть генерированы требуемые HTML-документы. Шаблон публикации представляет собой обычный документ XHTML (или HTML), который наряду с фиксированными тегами и их атрибутами содержит

жит специально определенные переменные (*variables*). Каждая переменная состоит из трех знаков: символа доллара (\$) и двухбуквенного алфавитного идентификатора. При необходимости применения в коде шаблона символа \$ нужно использовать escape-последовательность, т. е. перед данным символом необходимо поместить обратную косую черту \\$. При публикации вместо переменных подставляются соответствующие им значения параметров, задаваемые в окне настройки параметров публикации **Publish Settings**. Поскольку все содержимое шаблона, за исключением переменных, остается неизменным, это означает, что шаблон может содержать любые элементы разметки HTML, а также сценарии JavaScript и VBS. Для создания шаблона можно воспользоваться любым текстовым или HTML-редактором. В качестве основы может быть использован один из стандартных шаблонов, поставляемых вместе с Flash MX 2004.

При работе с операционной системой Windows 2000 или XP файлы шаблонов находятся по адресу: <drive>/Documents and Settings/<user>/Local Settings/Application Data/Macromedia/Flash MX 2004/en/Configuration/HTML. При работе с Windows 98 файлы шаблонов находятся по адресу: <drive>/Program Files/Macromedia/Flash MX 2004/en/First Run/HTML. Для того чтобы получить доступ к пользовательскому шаблону из окна **Publish Settings**, необходимо сохранить файл шаблона в каталоге HTML, расположенным по указанному адресу. Перечень переменных шаблонов и их назначение представлены в табл. 18.2.

Таблица 18.2. Переменные шаблонов и их назначение

Тег/атрибут/ параметр	Переменная шаблона	Назначение
Название шаблона	\$TT	Данное название будет выведено в списке Template вкладки HTML диалогового окна Publish Settings
Начало описания шаблона	\$DS	Информация, помещенная между данными переменными, будет отображаться в окне HTML Template Info , выводимом при нажатии кнопки Info , расположенной во вкладке HTML диалогового окна Publish Settings
Окончание описания шаблона	\$DF	
Заглавие страницы <title> </title>	\$T1	Заглавие страницы выводится в заголовке окна браузера и размечается при помощи парного тега. Вместо данной переменной в коде страницы будет выведено название файла SWF, заданное при публикации в поле SWF Filename вкладки Formats диалогового окна Publish Settings

Таблица 18.2 (продолжение)

Тег/атрибут/ параметр	Переменная шаблона	Назначение
width	\$WI	Ширина и высота окна Flash Player, задаются в соответствии с установками окна Publish Settings
height	\$HE	
movie, src	\$MO	Путь к фильму, задаваемый параметром <i>movie</i> для тега <code><object></code> и атрибутом <i>src</i> для тега <code><embed></code>
align	\$HA	Выравнивание в окне браузера относительно других элементов страницы, задается атрибутом <i>align</i> для тегов <code><object></code> и <code><embed></code>
loop	\$LO	Режим зацикливания воспроизведения фильма
Параметры для тега <code><object></code>	\$PO	В коде HTML данная переменная будет заменена набором тегов <code><param></code> , содержащих все параметры тега <code><object></code> , заданные во вкладке HTML окна Publish Settings , значения которых отличны от используемых по умолчанию
Параметры для тега <code><embed></code>	\$PE	В коде HTML данная переменная будет заменена набором атрибутов тега <code><embed></code> , заданных во вкладке HTML окна Publish Settings , значения которых отличны от значений, используемых по умолчанию
play	\$PL	Указывает, будет ли воспроизведение фильма начато автоматически
quality	\$QU	Соотношение качества прорисовки и скорости воспроизведения
scale	\$SC	Масштабирование фильма в пределах окна Flash Player
salign	\$SA	Выравнивание фильма в пределах окна Flash Player
wmode	\$WM	Оконный режим (<code>window/windowless</code>)
devicefont	\$DE	Применение системных шрифтов
bgcolor	\$BG	Цвет фона фильма
Текст фильма	\$MT	Вывод используемого в фильме текста в виде комментариев
Ссылки фильма на URL	\$MU	Вывод используемых в фильме ссылок на внешние сетевые ресурсы в виде комментариев
id	\$TI	Идентификатор, позволяющий обращаться к фильму Flash из сценариев JavaScript и VBS, совпадает с именем файла SWF

Таблица 18.2 (окончание)

Тег/Атрибут/ Параметр	Переменная шаблона	Назначение
<meta>	\$CS	Данная переменная будет заменена тегом <meta>, содержащим атрибуты http-equiv и content
cab#version	\$FV, \$JR, \$NR	Данные переменные будут заменены номером версии Flash Player, указываемым атрибутом codebase тега <object>
width	\$IW	Ширина растрового изображения
height	\$IH	Высота растрового изображения
src	\$IS	Имя файла растрового изображения
usemap	\$IU	Имя карты изображения
<map></map>, <area>	\$IM	Данная переменная будет заменена тегами карты изображения <map> со всеми необходимыми атрибутами, значения которых задаются на основе содержимого кадра, используемого в качестве карты
width	\$QW	Ширина фильма QuickTime
height	\$QH	Высота фильма QuickTime
src	\$QN	Путь к файлу QuickTime
width	\$GW	Ширина изображения GIF
height	\$GH	Высота изображения GIF
src	\$GN	Путь к файлу GIF
width	\$JW	Ширина изображения JPEG
height	\$JH	Высота изображения JPEG
src	\$JN	Путь к файлу JPEG
width	\$PW	Ширина изображения PNG
height	\$PH	Высота изображения PNG
src	\$PN	Путь к файлу PNG

Листинг 18.2 содержит код шаблона, позволяющего разместить Flash-фильм на HTML-странице и отцентрировать его в окне браузера по горизонтали.

Листинг 18.2. Шаблон HTML для размещения и горизонтального центрирования Flash-фильма

```
$TTFflash Centered
$DS
Display Flash Movie in HTML centered horizontally
$DF
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>$TI</title></head>
<body bgcolor="$BG">
<div align="center">
  <object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/
  swflash.cab#version=$FV,$JR,$NR,0" width="$WI" height="$HE">
    $PO
    <embed $PEwidth="$WI" height="$HE" type="application/x-shockwave-flash"
    pluginspage="http://www.macromedia.com/go/getflashplayer" />
  </object>
</div>
</body>
</html>
```

Разрешение вопросов политики безопасности

С выходом новой версии Flash Player 7 изменились правила политики безопасности. В частности, обмен данными между фильмами Flash, загрузка ресурсов общих библиотек и внешних SWF-файлов могут быть непосредственно осуществлены только при условии, что фильмы расположены в одном и том же домене. Таким образом, фильм, расположенный по адресу www.wherever.com, не сможет загрузить ресурсы общей библиотеки с сайта www.anywhere.com.

Для того чтобы получить возможность обмена данными между файлами, находящимися в различных доменах, а также осуществлять загрузку ресурсов общей библиотеки, расположенной в другом домене, необходимо использовать файл с описанием междоменной политики безопасности (cross-

domain security). Этот файл представляет собой документ XML, содержащий информацию, указывающую серверу, что находящиеся на нем данные и документы доступны SWF-файлам с определенных доменов (или со всех доменов). Таким образом, SWF-файл с домена, указанного в файле с *политиками*, действующими для данного сервера, сможет получить доступ к соответствующим данным с этого сервера.

Когда Flash-фильм пытается осуществить загрузку данных с другого домена, Flash Player автоматически загружает файл с политиками, расположенный в этом домене. Если домен, с которого осуществляется попытка получить доступ к данным, включен в файл с политиками, данные автоматически будут доступны.

Файл с политиками должен быть назван crossdomain.xml и помещен в корневую директорию сервера, предоставляющего данные. Файлы с политиками действуют только на серверах, использующих протоколы HTTP, HTTPS или FTP и только в пределах конкретного порта и протокола.

Листинг 18.3 содержит пример файла с описанием политики, позволяющей осуществить доступ к данным **www.whenever.com** SWF-файлам, расположенным в любом поддомене домена anywhere.com.

Листинг 18.3. Файл политики, предоставляющий доступ SWF-файлам с определенного домена

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy
SYSTEM http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd>
<!-- http://www.whenever.com/crossdomain.xml -->
<cross-domain-policy>
<allow-access-from domain="*.anywhere.com"/>
</cross-domain-policy>
```

Для того чтобы организовать доступ к данным **www.whenever.com** с любого домена, можно воспользоваться кодом, представленным в листинге 18.4.

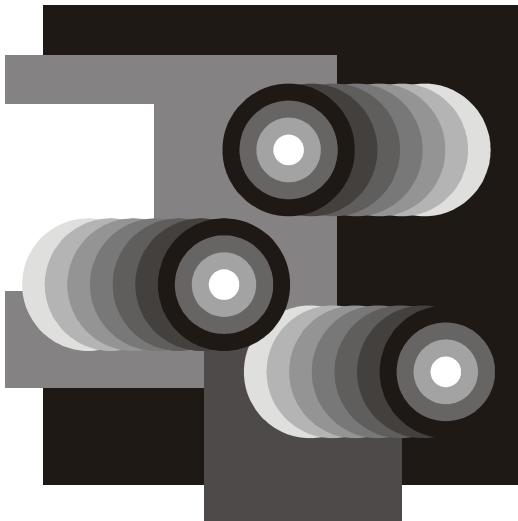
Листинг 18.4. Файл политики, предоставляющий доступ SWF-файлам с любого домена

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy
SYSTEM http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd>
<!-- http://www.whenever.com/crossdomain.xml -->
```

```
<cross-domain-policy>
<allow-access-from domain="*"/>
</cross-domain-policy>
```

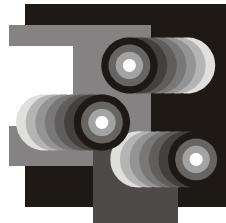
Тег `<cross-domain-policy>` содержит необходимое количество вложенных тегов `<allow-access-from>`, каждый из которых содержит атрибут `domain`, указывающий точный IP-адрес, точный домен или задает домен по шаблону (с использованием знака `"*"`). При использовании шаблона "звездочка" соответствует всем доменам. Если после звездочки следует суффикс, отделенный точкой, это соответствует всем доменам, оканчивающимся данным суффиксом.

Если файл с политиками не содержит тега `<allow-access-from>`, это равносильно отсутствию данного файла.



ЧАСТЬ IV

ИНТЕРАКТИВНОСТЬ



Глава 19

Введение в ActionScript

Узкий специалист узнает все больше о все меньшем и так до тех пор, пока не будет знать все ни о чем и ничего обо всем.

Бернард Шоу

Представление о сценариях ActionScript

Сценарий представляет собой набор инструкций (actions), которые осуществляют программное управление элементами фильма. Сценарии, используемые в фильме, обрабатываются интерпретатором ActionScript, который проверяет корректность кода, обрабатывает программные события, выполняет соответствующие инструкции, вычисляет требуемые значения и возвращает результат вычислений. Сценарии ActionScript могут быть непосредственно встроены в фильм или размещаться во внешнем текстовом файле, имеющем расширение as. В последнем случае фильм должен содержать указание на внешний сценарий, который будет загружен в него непосредственно в процесс воспроизведения.

Характерной особенностью сценариев ActionScript является их рассредоточенность по различным элементам структуры фильма. Достаточно часто Flash-фильм содержит несколько фрагментов программного кода, расположенных в различных частях фильма, каждый из которых выполняет специфическую для его местоположения задачу. В этом случае программный код бывает сильно децентрализован, и попытка представить его в виде единого листинга (текста сценария) может быть сопряжена со значительными трудностями (в такой ситуации каждый фрагмент кода должен быть снабжен комментариями, указывающими его местоположение в структуре фильма). Настоящая версия ActionScript располагает возможностями, позволяющими обеспечить высокую степень централизации кода, однако при решении

конкретных задач на практике часто приходится прибегать к рассредоточению сценариев.

В рабочей среде документа Flash существуют *только* три объекта, которые могут содержать сценарий ActionScript. К их числу относятся:

- ключевой кадр основной монтажной линейки или монтажной линейки символа типа **Movie Clip**. В этом случае сценарий называется сценарием кадра — Frame Action. Кадр монтажной линейки, содержащий сценарий, имеет метку "a" 
- экземпляр символа типа **Button**. В этом случае сценарий называется сценарием кнопки — Button Action;
- экземпляр символа типа **Movie Clip**. В этом случае сценарий называется сценарием клипа — Movie Clip Action.

Любой документ может содержать неограниченное число сценариев. При создании сценария очень важно удостовериться в том, что он размещен именно в том объекте, для которого предназначен. Поскольку сценарии разных объектов оформляются по-разному, размещение, например, сценария кнопки в кадре вызовет ошибку при компиляции. Такой сценарий не будет выполняться.

При воспроизведении фильма сценарии ActionScript выполняются только в результате наступления определенного события или одного из нескольких событий. Все события могут быть разделены на две категории.

- Системные события (System events) — это события, которые генерируются автоматически в процессе воспроизведения фильма. Примером такого события является переход воспроизводящей головки на новый кадр фильма.
- События, инициированные пользователем (User events), — это события, которые генерируются в результате действий, выполняемых пользователем. Примерами таких событий являются, например, нажатие кнопки или клавиши клавиатуры, перемещение курсора мыши, перемещение фокуса ввода или изменение содержимого пользовательского текстового блока.

Сценарий кадра выполняется автоматически, как только воспроизводящая головка входит в кадр, содержащий этот сценарий. Данное событие не указывается явно в тексте сценария, поскольку размещение сценария в соответствующем кадре подразумевает его наступление.

Сценарий кнопки или клипа должен содержать явное указание на событие, наступление которого повлечет за собой выполнение данного сценария. Для этого необходимо использовать конструкцию, которая называется *обработчиком событий* (Event handler). Обработчик событий содержит наименование

события (или нескольких) и набор инструкций, которые будут выполнены интерпретатором при наступлении любого из указанных событий. В процессе воспроизведения Flash-фильма может генерироваться (происходить) множество событий, как системных, так и пользовательских, однако выполнение программного кода кнопки или клипа осуществляется только при наличии обработчика событий. Так, например, нажатие кнопки всегда приводит к генерации события `press`, однако для того чтобы нажатие кнопки привело к выполнению определенных действий, необходимо создать соответствующий сценарий с использованием обработчика событий, контролирующего событие `press`.

Так же как и человеческая речь, сценарий ActionScript состоит из предложений. В конце каждого предложения сценария содержится знак точки с запятой ";", указывающий интерпретатору, что предложение закончено. Каждое предложение принято начинать с новой строки. Несмотря на то, что перевод строки автоматически указывает интерпретатору на конец предложения, рекомендуется, тем не менее, использовать точку с запятой, поскольку это позволит избежать возможных ошибок.

При создании сценариев фрагменты программного кода могут быть сопровождены комментариями, поясняющими их назначение. Комментарии представляют собой различные пометки и замечания, помещенные непосредственно в программный код, которые игнорируются интерпретатором. Комментарии бывают весьма полезны при работе в команде, поскольку они дают возможность другому разработчику быстро разобраться в логике реализации сценария. Кроме того, комментарии позволяют легче ориентироваться в собственном коде, особенно при необходимости его редактирования спустя некоторое время после создания. В ActionScript существуют два способа обозначения комментариев:

- для создания одностroочного комментария используются две наклонные черты "///". Интерпретатор игнорирует все символы, расположенные между символом "///" и окончанием строки;
- для создания многострочного комментария используются обозначения "/*" и "*/". Все символы, расположенные между данными обозначениями, игнорируются интерпретатором.

Следующий код иллюстрирует различные варианты использования комментариев.

```
//Это однострочный комментарий  
/*Это еще один однострочный комментарий */  
/*Это пример  
комментария, расположенного  
на нескольких строках */
```

Использование панели *Actions* для создания сценариев

Создание сценариев в рабочей среде Flash выполняется при помощи редактора ActionScript. Редактор также позволяет быстро перемещаться между сценариями фильма. Для того чтобы открыть редактор ActionScript, необходимо выполнить команду главного меню **Window>Development Panels>Actions** или воспользоваться ее клавиатурным эквивалентом <F9>.

Панель **Actions** состоит из трех частей (рис. 19.1).

- **Script pane** — окно, расположенное в правой части панели **Actions**, предназначеннное для ввода и отображения сценария. Работа с программным кодом в этом окне осуществляется так же, как и с обычным текстом в текстовом редакторе. Фрагменты сценария могут быть выделены, скопированы, вставлены из буфера или удалены. Эти действия выполняются при помощи команд контекстного меню, которое появляется при щелчке правой кнопкой мыши внутри раздела **Script pane** или при помощи стандартных клавиатурных сочетаний (<Ctrl>+<A>, <Ctrl>+<C>, <Ctrl>+<V>, <Ctrl>+<X>, <Delete>). Для того чтобы выделить целую строку сценария, можно щелкнуть слева от требуемой строки, удерживая при этом клавишу <Ctrl>. Для того чтобы выделить сразу несколько строк или многострочный фрагмент кода, можно воспользоваться клавишей <Shift>.
- **Actions toolbox** — данный раздел расположен слева в верхней части панели **Actions** и содержит перечень всех встроенных классов, задокументированных методов, свойств, функций, предложений, операторов и т. д., т. е. элементов языка ActionScript. Все элементы разбиты на категории и иерархически упорядочены при помощи папок. Щелчок на папке раскрывает ее, предоставляя доступ к вложенным папкам или соответствующим элементам. Любой элемент может быть помещен из раздела **Actions toolbox** непосредственно в сценарий. Для этого в окне **Script pane** нужно поместить курсор в то место, куда должен быть вставлен требуемый элемент и либо дважды щелкнуть на нем в разделе **Actions toolbox**, либо перенести в окно сценария, либо, щелкнув правой кнопкой мыши, выполнить команду контекстного меню **Add to script**. Поместить в программный код инструкцию, не набирая ее текст вручную, также можно при помощи кнопки , нажатие которой приводит к появлению меню, содержащего различные категории действий.
- **Script Navigator** — данный раздел расположен слева в нижней части панели **Actions** и содержит схематичное отображение элементов структуры документа, содержащих сценарии. **Script Navigator** напоминает Проводник фильма **Movie Explorer**. Раздел **Current Selection** отображает текущий выделенный элемент, раздел **Scene** содержит перечень всех объектов сцены, содержащих сценарии, а раздел **Symbol Definition(s)** содержит пере-

чень всех объектов, содержащих сценарии и входящих в состав символов типа муви-клип. Используя эту информацию, можно быстро сориентироваться в структуре сценариев фильма и перейти к сценарию другого элемента сцены.

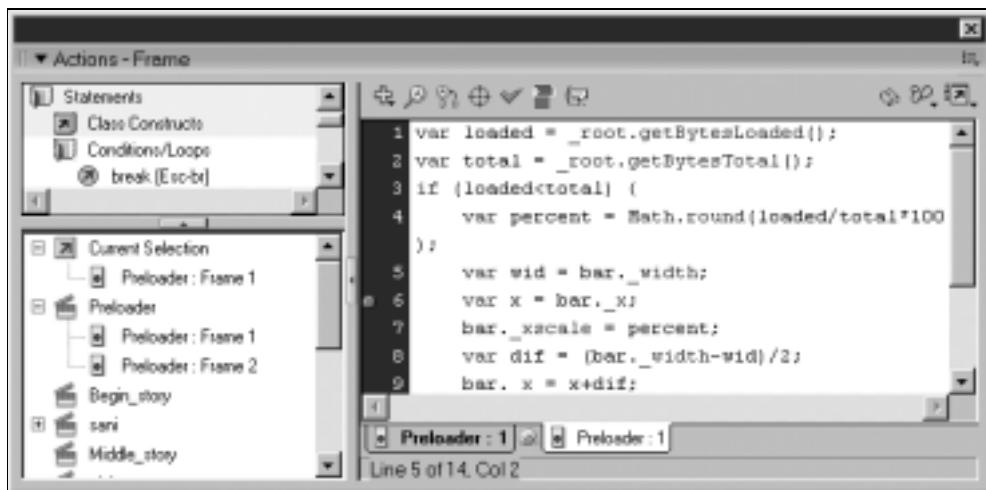


Рис. 19.1. Панель редактора ActionScript **Actions**

Для того чтобы назначить сценарий кадру, экземпляру кнопки или муви-клипу, или отобразить имеющийся сценарий, необходимо явным образом выделить требуемый объект, щелкнув на нем. После этого в заголовке панели **Actions** появится указание на тип выделенного в данный момент объекта (**Actions-Frame**, **Actions-Button** или **Actions-Movie Clip**). Указание на выделенный объект будет также содержаться в категории **Current Selection** раздела **Script Navigator** и на закладке, расположенной слева под окном сценария. Если выделенный объект не относится к объектам, которым можно назначить сценарий, его тип не будет указан в заголовке панели **Actions**, а в окне **Script pane** и в категории **Current Selection** раздела **Script Navigator** будет выведено сообщение "Current selection cannot have actions applied to it". Каждому экземпляру кнопки или муви-клипа может быть назначен свой сценарий.

Сценарий кадра рекомендуется помещать в пустой ключевой кадр на отдельный слой. Обычно такой слой называют **actions**.

Примечание

При создании сценария следует тщательно следить за тем, чтобы он назначался именно тому элементу, для которого предназначен.

Одна из удобных возможностей, которыми располагает редактор ActionScript, заключается в использовании подсказок по коду (code hints). Подсказки автоматически появляются в процессе ввода текста сценария (если данная опция не отключена). Редактор ActionScript выводит два типа подсказок: всплывающая подсказка по синтаксису используемой конструкции (tooltip style) и подсказка в виде меню (menu style). Подсказка по синтаксису появляется при вводе открывающей круглой скобки, размещенной после элемента сценария, требующего наличия круглых скобок (функция, условное предложение, предложение цикла и т. д.). Всплывающая подсказка указывает возможные варианты формата данного элемента. При наличии нескольких вариантов подсказка содержит стрелки, позволяющие просмотреть различные схемы его использования (рис. 19.2).

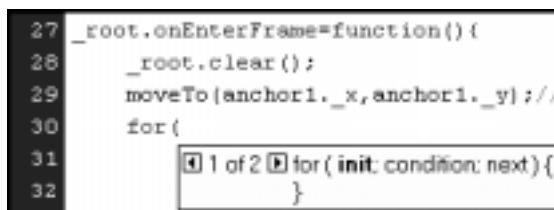


Рис. 19.2. Всплывающая подсказка по коду

Подсказка в виде меню появляется при вводе точки после имени переменной или объекта, после ввода открывающей круглой скобки обработчика событий, после двоеточия в объявлении переменной или функции и позволяет выбрать из списка один из требуемых в данном контексте элементов языка (рис. 19.3). Подсказка будет выведена только в том случае, если используемый синтаксис соответствует задокументированному (в частности, подсказки чувствительны к регистру). Настройка параметров отображения подсказок осуществляется во вкладке **ActionScript** окна **Preferences** (см. разд. "Настройки программы" гл. 3).

Другая полезная функция редактора — это подсветка элементов сценария (syntax coloring). Функции, ключевые слова, строки и т. п. при вводе окрашиваются определенным цветом. Это позволяет легче ориентироваться в программном коде и контролировать правильность написания команд непосредственно при вводе. Подсветка выполняется только в том случае, если используемый синтаксис соответствует задокументированному (в частности, подсказки чувствительны к регистру). Настройка используемой цветовой схемы подсветки осуществляется во вкладке **ActionScript** окна **Preferences** (см. разд. "Настройки программы" гл. 3).

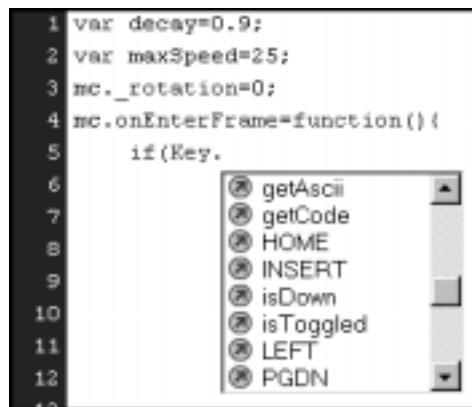


Рис. 19.3. Подсказка по коду в виде меню

Контекстное меню палитры **Actions** содержит ряд команд, позволяющих настроить редактор сценариев:

- **Pin Script** (Закрепить сценарий) — данная команда закрепляет текущий сценарий, что позволяет быстро перейти к нему в любой момент. В результате этого действия в нижней части окна **Script pane** справа появится закладка с указанием объекта, содержащего сценарий. Закрепленные сценарии отображаются при помощи пиктограммы с изображением воткнутой кнопки . К закрепленному сценарию можно обратиться вне зависимости от того, какой объект выделен в данный момент. Для того чтобы отобразить данный сценарий, достаточно щелкнуть на соответствующей закладке. Одновременно можно закрепить любое количество сценариев, при этом каждый из них будет отражен при помощи соответствующей закладки. Закрепить текущий сценарий можно также при помощи пиктограммы с изображением кнопки . При нажатии на нее кнопка "воткнется" и сценарий будет закреплен. Для того чтобы перейти от закрепленного сценария к сценарию текущего объекта, нужно щелкнуть на его закладке, расположенной слева;
 - **Close Script** (Закрыть сценарий) — данная команда открепляет текущий закрепленный сценарий. Открепить текущий закрепленный сценарий также можно при помощи пиктограммы . После открепления закладка сценария в нижней части окна будет удалена;
 - **Close All Scripts** (Закрыть все сценарии) — данная команда открепляет все закрепленные сценарии;
 - **Go to Line** (Перейти к строке) — данная команда позволяет осуществить переход к заданной строке сценария;

- **Find** (Найти) — данная команда позволяет найти заданное сочетание символов в сценарии. Флажок **Match case** позволяет осуществлять поиск с учетом регистра. Для выполнения данной команды можно воспользоваться кнопкой , расположенной над окном **Script pane**;
- **Find Again** (Найти снова) — данная команда позволяет найти сочетание символов, заданное при предыдущем поиске;
- **Replace** (Заменить) — данная команда позволяет найти заданное сочетание символов в сценарии и заменить его на другое. Для выполнения данной команды можно воспользоваться кнопкой , расположенной над окном **Script pane**;
- **Auto Format** (Автоматическое форматирование) — выполнение данной команды позволяет единообразно отформатировать программный код и сделать его более читаемым. Для выполнения данной команды можно воспользоваться кнопкой , расположенной над окном **Script pane**;
- **Check Syntax** (Подсветка) — данная команда позволяет выполнить автоматическую проверку синтаксиса. При обнаружении ошибки будет выведено соответствующее сообщение, а в автоматически открывающейся панели **Output** будет указан адрес и содержание ошибки. Для выполнения данной команды можно воспользоваться кнопкой , расположенной над окном **Script pane**;
- **Show Code Hint** (Вспывающая подсказка) — данная команда позволяет вывести подсказку по элементу кода, на котором в данный момент находится курсор. Для отображения подсказки курсор должен быть помещен после открывающей круглой скобки либо после точки или двоеточия. Для выполнения данной команды можно воспользоваться кнопкой , расположенной над окном **Script pane**;
- **Import Script** (Импортировать сценарий) — данная команда позволяет импортировать сценарий из внешнего файла с расширением as. Текст импортированного сценария будет помещен в текущий сценарий, начиная с местоположения курсора. Сценарий импортируется с использованием кодировки, указанной во вкладке **ActionScript** окна **Preferences** (см. разд. "Настройки программы" гл. 3).
- **Export Script** (Экспортировать сценарий) — данная команда позволяет сохранить текущий сценарий во внешнем файле с расширением as. Сценарий экспортируется с использованием кодировки, указанной во вкладке **ActionScript** окна **Preferences** (см. разд. "Настройки программы" гл. 3).
- **Print** (Печать) — данная команда позволяет вывести сценарий на печать;
- **View Esc Shortcut Keys** (Показать горячие клавиши действий) — активизация данного режима включает отображение клавиатурных эквивалентов элементов языка в разделе **Actions toolbox**. Для быстрого ввода некоторых

инструкций можно воспользоваться соответствующим им сочетанием клавиш с использованием клавиши <Escape>. Для выполнения данной команды можно воспользоваться кнопкой  , расположенной над окном **Script pane**;

- **View Line Numbers** (Показать номера строк) — активизация данного режима позволяет отобразить нумерацию строк сценария. Эта возможность бывает полезна, например, при поиске ошибки, адрес которой указывается в окне **Output** и включает номер строки. Для выполнения данной команды можно воспользоваться кнопкой  , расположенной над окном **Script pane**;
- **Word Wrap** (Перенос слов) — данный режим позволяет автоматически переносить текст сценария на новую строку, если он не умещается в окно по ширине. Перевод строки в результате применения данного режима не является указанием на окончание предложения. Для выполнения данной команды можно воспользоваться кнопкой  , расположенной над окном **Script pane**;
- **Auto Format Options** (Параметры автоматического форматирования) — выполнение данной команды приводит к появлению одноименного диалогового окна, позволяющего задать некоторые параметры автоматического форматирования текста сценария.
- **Preferences** (Настройки) — данная команда открывает вкладку **ActionScript** окна **Preferences**, содержащую настройки редактора сценариев.

Flash MX 2004 содержит обширную справочную информацию по применению **ActionScript**. Для вызова справки можно воспользоваться командой главного меню **Help>ActionScript Dictionary**. Для того чтобы вывести справку по элементу программного кода, нужно поместить на него курсор и щелкнуть на кнопке  , расположенной над окном **Script pane**, либо воспользоваться командой **View Help** контекстного меню окна **Script pane**.

Встроенные функции управления воспроизведением

При отсутствии сценариев кадры монтажной линейки воспроизводятся последовательно друг за другом. Также последовательно выполняется переход с одной сцены на другую, согласно порядку следования сцен, заданному в панели **Scene**. Монтажные линейки анимированных муви-клипов всегда воспроизводятся циклически. Редактор ActionScript содержит несколько встроенных функций, позволяющих останавливать и запускать анимацию монтажных линеек, осуществлять управление порядком воспроизведения кадров и выполнять переходы между сценами.

Для вызова функции необходимо использовать следующий формат записи:

```
functionName(arg1, arg2, arg3,...);
```

Здесь:

- functionName — имя вызываемой функции,
- arg1, arg2, arg3, — значения аргументов, которые принимает функция.

Некоторые функции не содержат аргументов, но несмотря на это, при вызове таких функций также необходимо использовать открывающую и закрывающую круглые скобки, в противном случае интерпретатор не идентифицирует соответствующую инструкцию как функцию. Точка с запятой указывает на окончание предложения.

Функции воспроизведения и остановки

Данные функции позволяют остановить или запустить воспроизведение монтажной линейки. При использовании функций воспроизведения и остановки в сценариях кадра их действие распространяется на монтажную линейку, содержащую сценарий с вызовом функции.

Функция `stop()` останавливает воспроизведение монтажной линейки, содержащей данную функцию. Эта функция не принимает аргументов. Для того чтобы остановить воспроизводящую головку в требуемом кадре, нужно поместить в него следующий сценарий:

```
stop();
```

Такой сценарий может быть использован, например, в одной из следующих ситуаций:

- для того чтобы предотвратить автоматический запуск воспроизведения фильма, данный сценарий можно поместить в первый кадр его монтажной линейки;
- при необходимости предотвратить циклическое воспроизведение монтажной линейки муви-клипа данный сценарий можно поместить в его последний кадр;
- для того чтобы предотвратить самопроизвольный переход с одной сцены на другую, можно поместить данный сценарий в последний кадр монтажной линейки соответствующей сцены.

Функция `play()` запускает воспроизведение монтажной линейки, с которой осуществлен ее вызов. Данная функция также не принимает аргументов.

Функции выбора кадра

Функции выбора кадра позволяют осуществить переход к кадру, указанному в качестве аргумента. Существуют две функции выбора кадра, выполнение которых приводит к различным результатам.

Функция `gotoAndStop()` переводит воспроизводящую головку в заданный кадр и останавливает воспроизведение текущей монтажной линейки.

Функция `gotoAndPlay()` переводит воспроизводящую головку в заданный кадр, и воспроизведение монтажной линейки возобновляется (или продолжается).

Обе функции имеют схожий формат:

```
gotoAndStop(Scene, frame)  
gotoAndPlay(Scene, frame)
```

Здесь:

- `Scene` — название сцены, к кадру которой осуществляется переход. Название сцены должно быть заключено в кавычки. Этот параметр является необязательным. Если название сцены опущено, происходит переход к кадру текущей сцены;
- `frame` — номер или метка кадра (label), к которому осуществляется переход. При использовании метки для указания целевого кадра, она заключается в кавычки (*информация о создании меток и указателей содержится в гл. 9*). Принципиально на кадр можно ссылаться как по его номеру, так и по метке или указателю (anchor), однако использование меток в некоторых случаях является более предпочтительным, поскольку не требует корректировки сценариев при изменении порядкового номера кадра.

Ниже представлены примеры употребления функций выбора кадра.

```
gotoAndStop("Main_Scene", 1); /*переход к первому кадру сцены Main_Scene с последующей остановкой */  
gotoAndStop("start"); /*переход к кадру с меткой start текущей сцены с последующей остановкой */  
gotoAndPlay("Introduction", "start"); /*переход к кадру с меткой start сцены Introduction и продолжение воспроизведения */  
gotoAndPlay(24); /*переход к 24-му кадру текущей сцены и продолжение воспроизведения */
```

Функция `gotoAndPlay()` может быть использована для того, чтобы зациклить воспроизведение монтажной линейки в определенном диапазоне кадров. Для этого в последний кадр данного диапазона нужно поместить следующий сценарий:

```
gotoAndPlay(beginningFrame);
```

Здесь `beginningFrame` — номер или метка начального кадра требуемого диапазона.

Функции безусловного перехода

Функции безусловного перехода перемещают воспроизводящую головку на соседние по отношению к кадру или сцене, в которых осуществлен вызов функции, кадр или сцену с последующей остановкой.

Функция `nextFrame()` — перемещает воспроизводящую головку в следующий кадр по отношению к кадру, в котором производится вызов данной функции. После перехода воспроизведение остановлено.

Функция `prevFrame()` — перемещает воспроизводящую головку в предыдущий кадр по отношению к кадру, в котором производится вызов данной функции. После перехода воспроизведение остановлено.

Функция `prevFrame()` — перемещает воспроизводящую головку в первый кадр сцены, следующей за сценой, в которой осуществляется вызов данной функции. После перехода воспроизведение монтажной линейки остановлено. Если текущая сцена является последней, в результате выполнения данной функции будет осуществлен переход на ее последний кадр. Эта функция может быть вызвана только с основной монтажной линейки.

Функция `prevScene()` — перемещает воспроизводящую головку в первый кадр сцены, предшествующей сцене, в которой осуществляется вызов данной функции. После перехода воспроизведение монтажной линейки остановлено. Если текущая сцена является первой, в результате выполнения данной функции будет осуществлен переход на ее первый кадр. Эта функция может быть вызвана только с основной монтажной линейки.

Функция `trace()`

Данная функция носит служебный характер, действует только в среде тестирования и используется при отладке сценариев. Функция `trace()` выводит в панель **Output** результат вычисления выражения, заключенного в круглые скобки в качестве ее аргумента, преобразованного в строку, т. е. в последовательность чисел и/или букв. Функция `trace()` является мощным инструментом отладки, используемым на стадии разработки сценария. С ее помощью можно проверить, какое значение принимает та или иная переменная в определенный момент, выполняется ли заданное условие и т. д. При необходимости можно отключить все вызовы данной функции, установив флагок **Omit trace actions** вкладки **Flash** диалогового окна **Publish Settings**. Следующий сценарий, будучи помещен в первый кадр монтажной линейки фильма, в среде тестирования выведет в панель **Output** сообщение о начале воспроизведения:

```
trace("Воспроизведение фильма началось");
```

Программирование кнопок

Для того чтобы снабдить кнопку возможностью выполнять определенные действия в ответ на воздействие пользователя, ее необходимо снабдить сценарием. Сценарий кнопки *всегда* размещается внутри обработчика событий (event handler), который инициирует выполнение данного сценария только при наступлении определенного события. Для того чтобы назначить сценарий экземпляру кнопки, необходимо выделить его на сцене и, убедившись в том, что в заголовке панели **Actions**, в категории **Current Selection** раздела **Script Navigator** и на закладке, расположенной слева под окном **Script Pane**, появилось указание на данный объект, ввести текст сценария в окно **Script Pane**.

Обработчики событий кнопок

Обработчик событий кнопки называется `on()` и имеет следующий вид:

```
on(event) {  
    текст сценария;  
}
```

Здесь `event` — это название события, наступление которого повлечет за собой выполнение сценария, помещенного внутрь блока обработчика событий, ограниченного фигурными скобками.

Существует ряд событий, которые могут быть обработаны или "перехвачены" обработчиком событий `on()`. Такие события называются *событиями кнопки*. События связаны с различными вариантами воздействия на область реагирования кнопки курсором и использования *левой* кнопки мыши. Нажатие других кнопок мыши не обрабатывается. Табл. 19.1 содержит перечень стандартных событий кнопки.

Таблица 19.1. События кнопки

Событие	Описание*	
	Режим Track as Button	Режим Track as Menu Item
press	Кнопка мыши нажата при нахождении курсора в области реагирования кнопки	
release	Кнопка мыши нажата и затем отпущена при нахождении курсора в области реагирования кнопки	Кнопка мыши отпущена в области реагирования кнопки вне зависимости от местонахождения курсора в момент ее нажатия
rollOver	Курсор входит в область реагирования кнопки и, если клавиша мыши нажата, она отпускается	Курсор входит в область реагирования кнопки при ненажатой кнопке мыши

Таблица 19.1 (окончание)

Событие	Описание*	
	Режим Track as Button	Режим Track as Menu Item
rollOut	Курсор выходит за пределы области реагирования кнопки при ненажатой кнопке мыши	
dragOut	Курсор выходит за пределы области реагирования кнопки при нажатой кнопке мыши	
releaseOutside	Курсор выходит за пределы области реагирования кнопки при нажатой кнопке мыши, после чего кнопка мыши отпускается вне области реагирования	Событие не генерируется
dragOver	Курсор выходит за пределы области реагирования кнопки при нажатой кнопке мыши, после чего вновь возвращается в область реагирования, кнопка мыши остается нажатой	Курсор входит в область реагирования кнопки при нажатой кнопке мыши
keyPress	Нажатие указанной клавиши клавиатуры	

*О режимах (свойствах) экземпляров кнопки говорится в разд. "Свойства экземпляра символа типа *Button*" гл. 10.

Событие keyPress, формально являясь событием кнопки, тем не менее, с самой кнопкой не связано. Оно генерируется при нажатии указанной клавиши клавиатуры, но инициирует выполнение сценария кнопки, обрабатывающей данное событие. Существует возможность контролировать нажатие любой буквенной или цифровой клавиши, а также специальных клавиш. В случае буквенной или цифровой клавиши требуемый символ должен быть указан в кавычках, для специальных клавиш используются клавиатурные константы. Следующий пример демонстрирует варианты использования события keyPress.

```
on(keyPress "s") { /*нажатие клавиши <s> останавливает воспроизведение текущей монтажной линейки */
stop();
}
on(keyPress "<Enter>") { /* нажатие клавиши <Enter> возобновляет воспроизведение */
play();
}
```

Следует иметь в виду, что событие keyPress чувствительно к регистру, т. е. нажатия клавиш <S> и <S> — это разные события. Список клавиатурных констант выводится во всплывающей подсказке по коду. Выбрать требуемую константу (как и любое другое событие кнопки) можно, дважды щелкнув на соответствующей строке всплывающего меню подсказки.

Обработчик `on()` может одновременно перехватывать несколько различных событий. Различные события указываются через запятую. В этом случае сценарий кнопки выполняется при наступлении любого из них. Исключение составляет событие keyPress, каждый обработчик может контролировать нажатие только одной клавиатурной клавиши.

```
on(press, release, keyPress "<Space>") {  
    gotoAndPlay(1); /* переход на первый кадр текущей линейки будет выполнен в  
    результате любого из используемых событий */  
}
```

Примечание

Следует иметь в виду, что в среде тестирования действуют горячие клавиши, которые имеют приоритет над сценариями ActionScript. При использовании программного контроля клавиатуры имеет смысл отключить горячие клавиши среды тестирования при помощи команды ее главного меню **Control>Disable Keyboard Shortcuts**.

Сценарий кнопки может содержать несколько обработчиков событий. Таким образом, одна и та же кнопка может выполнять различные действия в зависимости от производимых с ней манипуляций. Следующий сценарий кнопки (листинг 19.1) представляет собой тренажер, позволяющий закрепить информацию о стандартных событиях кнопки. Здесь используется функция трассировки `trace()`, выводящая в панель **Output** информацию о генерации события.

Листинг 19.1. Контроль событий кнопки (сценарий экземпляра символа типа **Button**)

```
on (press) {  
    trace("Событие press");  
}  
on (release) {  
    trace("Событие release");  
}  
on (rollOver) {  
    trace("Событие rollOver");  
}
```

```
on (rollOut) {
    trace("Событие rollOut");
}

on (dragOut) {
    trace("Событие dragOut");
}

on (releaseOutside) {
    trace("Событие releaseOutside");
}

on (dragOver) {
    trace("Событие dragOver");
}

on (keyPress "<Space>") {
    trace("Событие keyPress \"<Space>\\""); /*для использования кавычек
в строке применяется символ обратной наклонной черты*/
}
```

Следует иметь в виду, что областью видимости (scope) обработчика событий кнопок `on()` является монтажная линейка, содержащая экземпляр кнопки, которому назначен сценарий. Это означает, что все команды, используемые внутри обработчика, распространяются только на монтажную линейку, содержащую кнопку. Для того чтобы из сценария кнопки обратиться к другой монтажной линейке, необходимо использовать адресацию. Так, для того чтобы при помощи кнопки остановить воспроизведение монтажной линейки экземпляра клипа, необходимо явно сослаться на него по имени. Об адресации клипов и кнопок будет сказано далее в этой главе.

Использование муви-клипов в качестве кнопок

Начиная с шестой версии Flash появилась возможность использования экземпляров клипов в качестве кнопок. Экземпляру символа может быть назначен сценарий с использованием обработчика событий `on()`. В результате этого действия экземпляр клипа будет функционировать как кнопка, реагируя на воздействия курсора и левой кнопки мыши. Такой клип называется *кнопка-клип*. Область реагирования кнопки-клипа совпадает с формой самого экземпляра. Если муви-клип содержит анимацию формы, то область реагирования такого клипа будет изменяться в соответствии с трансформацией входящих в его состав форм.

Для того чтобы внешний вид клипа-кнопки изменялся при воздействии курсора так же, как это происходит с обычной кнопкой, на его монтажной линейке должны находиться три кадра, описывающие его внешний вид при отсутствии взаимодействия, при наведении и при нажатии. Эти кадры должны содержать метки `_up`, `_over` и `_down` соответственно. Роль области реагирования в этом случае будет играть содержимое кадра с меткой `_up`. Для того чтобы воспроизведение монтажной линейки клипа-кнопки не начиналось самопроизвольно, в ее первый кадр нужно поместить функцию `stop()`.

Примечание

В качестве области реагирования клипа-кнопки может быть использован экземпляр другого клипа, при этом взаимное расположение клипов на сцене может быть совершенно произвольным. Для этого обоим экземплярам нужно присвоить уникальные идентификаторы **Instance Name**, задав их при помощи Инспектора свойств, и воспользоваться свойством `hitArea`. Пусть экземпляру клипа-кнопки будет присвоено имя `mc_but`, а экземпляру муви-клипа, используемому в качестве области реагирования, будет присвоено имя `mc_hitArea`. Если оба экземпляра находятся на одной и той же монтажной линейке, в кадр, содержащий эти экземпляры, нужно поместить следующий сценарий:

```
mc_but.hitArea=mc_hitArea;
```

В результате в качестве области реагирования клипа `mc_but` будет использован клип `mc_hitArea`.

Следует иметь в виду, что при использовании обработчика событий `on()` в клипах областью его видимости является не монтажная линейка, содержащая экземпляр клипа (как это происходит с кнопками), а монтажная линейка самого клипа. Это означает, что все команды, используемые внутри обработчика, в этом случае распространяются только на его монтажную линейку. Для того чтобы из сценария клипа обратиться к другой монтажной линейке, необходимо использовать адресацию. Ниже приводятся два примера, иллюстрирующие различия в областях видимости обработчика `on()`, используемого в кнопках и клипах.

```
//Сценарий кнопки
on(press, release, keyPress "<Enter>" ){
  stop(); /*остановка монтажной линейки, содержащей данную кнопку*/
}

//Сценарий клипа
on(press, release, keyPress "<Space>" ){
  stop(); /*остановка монтажной линейки данного клипа*/
}
```

Навигация внутри фильма при помощи монтажной линейки

Навигацией называется перемещение между различными частями проекта, осуществляемое пользователем. Любой проект, как правило, состоит из нескольких тематических разделов, освещдающих различные его стороны. Так, например, электронная презентация портфолио может включать следующие разделы: "Об авторе", "Живопись", "Графика", "Дизайн" и т. д. Flash предоставляет различные варианты организации структуры проекта и реализации навигации по его разделам. Один из таких вариантов заключается в размещении содержимого разделов в соответствующих кадрах монтажной линейки или сценах документа. Функции управления воспроизведением позволяют реализовать переход между разделами, перемещая воспроизводящую головку к указанному кадру. Применение данного подхода позволяет поместить весь проект целиком в один конечный файл SWF.

Управление воспроизведением монтажной линейки

Для того чтобы дать пользователю возможность управлять воспроизведением монтажной линейки фильма, можно поместить в него кнопки, снабдив их соответствующими сценариями. Следует иметь в виду, что панель навигации с кнопками, управляющими воспроизведением, должна присутствовать на сцене в течение всего времени воспроизведения фильма. В качестве примера можно рассмотреть организацию управления воспроизведением основной монтажной линейки, содержащей анимацию. Для этого предлагаются снабдить фильм следующими элементами управления:

- кнопка запуска анимации ();
- кнопка паузы ();
- кнопка перехода к началу фильма с последующей остановкой ();
- кнопки покадровой перемотки вперед () и назад () .

Для краткости предположим, что фильм состоит только из одной сцены. Чтобы реализовать данный пример, необходимо создать два слоя, которые можно назвать **movie** и **controls**. Слой **movie** будет содержать анимацию, а на слое **controls** нужно разместить кнопки для управления фильмом. В качестве кнопок управления можно использовать кнопки стандартной библиотеки **Window>Other Panels>Common Libraries>Buttons**, расположенные в папке Playback. Ниже приводятся сценарии используемых кнопок. Требуемое дейст-

вие происходит при нажатии соответствующей кнопки или при использовании клавиатурной клавиши, указанной событием keyPress.

```
// Сценарий кнопки запуска анимации
on(press, keyPress "<Enter>") {
    play();
}

// сценарий кнопки паузы
on(press, keyPress "<Space>") {
    stop();
}

// сценарий кнопки покадровой перемотки вперед
on(press, keyPress "<Right>") {
    nextFrame();
}

// сценарий кнопки покадровой перемотки назад
on(press, keyPress "<Left>") {
    prevFrame();
}

// сценарий кнопки перехода к первому кадру с последующей остановкой
on(press, keyPress "<Backspace>") {
    gotoAndStop(1);
}
```

Организация структуры проекта

Один из подходов к организации структуры проекта заключается в размещении его частей в различных кадрах или сценах одного и того же документа. Каждая часть, являясь самостоятельным тематическим разделом, должна быть снабжена панелью навигации, позволяющей переместиться к другой части проекта, размещенной в указанном кадре заданной сцены. В общем случае каждый раздел проекта может занимать на монтажной линейке один кадр или целый диапазон кадров, что зависит от его внутренней организации.

Каждый тематический раздел проекта может быть оформлен как муви-клип, содержащий на своей монтажной линейке все составные элементы фильма в виде вложенных символов, других графических объектов, звука и сценариев. В этом случае экземпляру данного символа достаточно отвести один кадр основной монтажной линейки документа. При таком подходе кадры, содержащие различные разделы, могут находиться в пределах одной сцены или каждый из них может быть помещен на свою отдельную сцену. В первом случае кадры имеют смысл снабдить метками, указывающими на содержание

кадра и позволяющими обратиться к данному кадру вне зависимости от его порядкового номера.

Отдельные разделы проекта могут занимать целый диапазон кадров основной монтажной линейки. В этом случае целесообразно размещать такие разделы на разных сценах. Это позволит лучше организовать структуру фильма и легче ориентироваться в документе на стадии его разработки.

Кроме того, возможен вариант, представляющий собой комбинацию описанных случаев. Некоторые разделы проекта могут состоять из одного единственного кадра (например, раздел "Об авторе"), другие могут включать множество кадров основной монтажной линейки (например, раздел "Живопись"). В этой ситуации также имеет смысл прибегнуть к организации структуры документа при помощи сцен.

Размещение различных частей проекта в одном документе, как правило, приводит к получению конечного файла достаточно большого размера и, соответственно, к увеличению времени его передачи по сети. В связи с этим, данный подход может быть реализован при разработке мультимедийной презентации, предназначенной для распространения на компакт-дисках и иных электронных носителях, либо при создании web-узла в расчете на аудиторию, обладающую каналом с высокой пропускной способностью. Этот подход, однако, обладает некоторыми преимуществами. Во-первых, он прост в реализации и, во-вторых, после того как файл проекта загружен по сети, переходы между разделами осуществляются мгновенно, и зрителю не приходится дожидаться загрузки каждой части в отдельности.

Создание сценариев

Размещая несколько тематических разделов проекта в одном Flash-документе, следует иметь в виду следующие правила.

- Каждый раздел должен быть снабжен панелью навигации, позволяющей пользователю переместиться к другой части проекта. Технически данный переход реализуется при помощи кнопок, содержащих сценарии с использованием функций выбора кадра `gotoAndStop()` или `gotoAndPlay()`.
- Необходимо предотвратить самопроизвольный переход между кадрами фильма и/или сценами документа, содержащими разные тематические разделы проекта. Для этого в последний кадр каждого раздела необходимо поместить функцию `stop()`.
- Если один из разделов содержит анимацию, расположенную на основной монтажной линейке фильма, то для того, чтобы зациклить ее воспроизведение, необходимо в последний кадр этой анимации поместить функцию `gotoAndPlay()`, отправляющую воспроизводящую головку в начальный кадр диапазона анимации.

- При переходе к кадру, соответствующему началу требуемого раздела, нужно использовать функцию `gotoAndPlay()` для того, чтобы после перехода начать воспроизведение, или функцию `gotoAndStop()` для того, чтобы после перехода воспроизведение было остановлено.

Рис. 19.4 иллюстрирует организацию структуры проекта, содержащего три части, каждая из которых помещена на отдельную сцену.

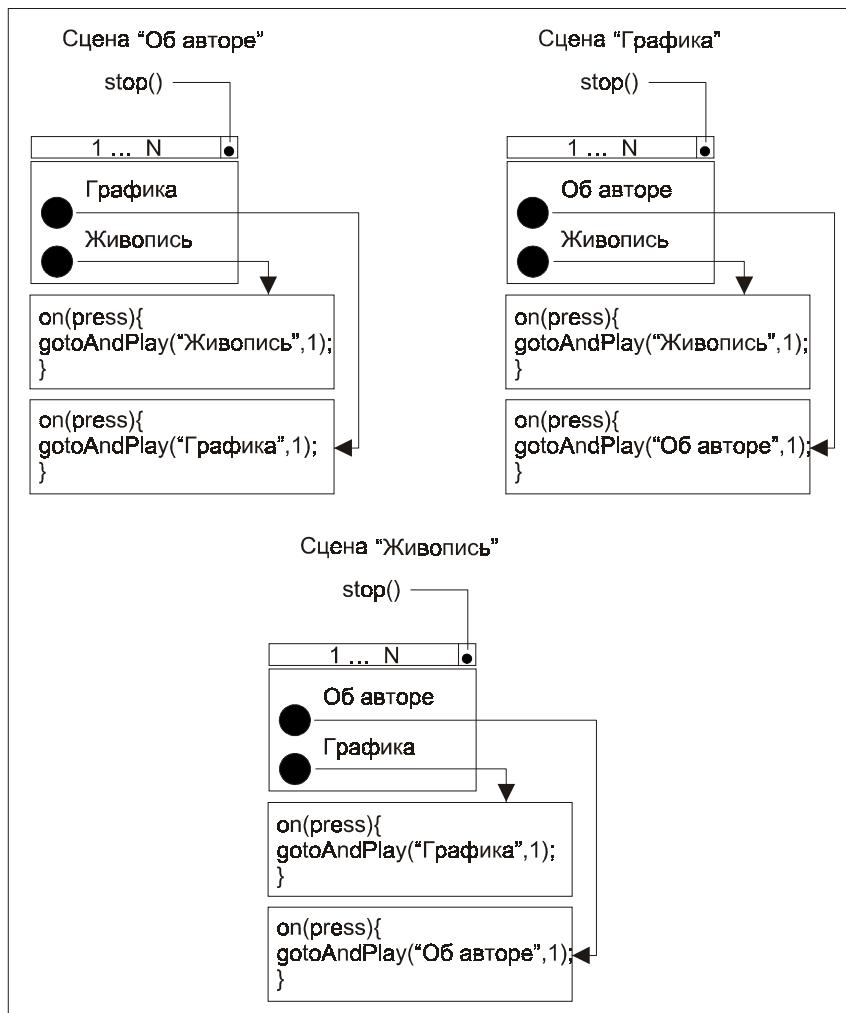


Рис. 19.4. Навигация посредством перехода между сценами

Навигация между фильмами при помощи гиперссылок

Если Flash-проект, предназначенный для размещения в сети Интернет, содержит несколько тематических разделов, каждый из них может быть помещен в отдельный файл SWF, который затем, в свою очередь, помещается на HTML-страницу. Таким образом, проект разбивается на ряд отдельных файлов, размещаемых на разных HTML-страницах. Навигация между частями проекта осуществляется посредством перехода с одной HTML-страницы на другую.

Для реализации данного подхода используется встроенная функция `getURL()`. Данная функция позволяет загрузить ресурс, указанный в качестве ее аргумента, в окно браузера или фрейм. Функция `getURL()` имеет следующий формат:

```
getURL(URL, window, method)
```

Здесь:

- *URL* — строка, содержащая абсолютный или относительный путь к документу, который требуется загрузить для отображения в окне браузера. Абсолютный путь начинается с указания протокола или логического диска (для локально расположенного файла), например, `http://www.avalon.ru` или `E:/Dima/Flash/Works/ArtStudio.swf`. При использовании относительных ссылок они разрешаются относительно местоположения HTML-страницы, содержащей файл SWF, в котором осуществляется вызов функции `getURL()`. При помощи параметра/атрибута `base` тегов `<object>` и `<embed>`, соответственно, можно задать базовую директорию, относительно которой будут разрешаться все относительные ссылки, используемые в SWF-файле, размещенном на данной HTML-странице (см. гл. 18). Поскольку данный аргумент является строкой, он должен быть заключен в кавычки;
- *window* — строка, указывающая окно браузера или фрейм, в который будет загружен документ. В качестве данного аргумента можно использовать одно из зарезервированных значений:

- `_blank` — загрузка в новое окно браузера;
- `_self` — загрузка в текущее окно браузера или фрейм;
- `_parent` — загрузка в родительский фрейм по отношению к текущему фрейму;
- `_top` — загрузка в самый верхний в иерархии фрейм.

Поскольку данный аргумент является строкой, он должен быть заключен в кавычки. Этот параметр является необязательным. Если его не указы-

вать, документ будет загружен в текущее окно браузера или фрейм (при фреймовой структуре);

- *method* — строка, задающая метод передачи переменных текущей монтажной линейки (т. е. линейки, откуда вызывается данная функция) во внешний удаленный сценарий, указанный в качестве URL. В качестве данного аргумента можно указать один из следующих методов:
 - GET — переменные отправляются в виде строки запроса, которая присоединяется к URL;
 - POST — переменные отправляются в виде отдельного блока данных после заголовка запроса HTTP POST.

Данный параметр является необязательным.

Следующий пример содержит сценарий кнопки, нажатие которой приведет к открытию нового окна браузера и загрузке в него страницы, расположенной по адресу **www.avalon.ru**.

```
on(press) {  
    getURL("http://www.avalon.ru","_blank");  
}
```

Функцию `getURL()` можно использовать для вызова почтовой программы и автоматической передачи ей требуемых данных для отправки почтового сообщения. Для этого в аргументе `URL` необходимо использовать ключевое слово `mailto:`. Следующий сценарий кнопки при ее нажатии откроет используемую в системе пользователя почтовую программу и отправит ей адрес, по которому можно будет послать письмо (для этого необходимо выполнить соответствующую команду почтового приложения).

```
on(press) {  
getURL("mailto:book_flash@mail.ru");  
}
```

Несколько усложнив данную конструкцию, можно переслать почтовому приложению имя адресата так, что оно будет выведено в строке адреса:

```
on(press) {  
getURL("mailto:Dmitri.I.Albert<book_flash@mail.ru>");  
}
```

Для того чтобы указать тему сообщения и переслать его текст, можно воспользоваться следующей записью:

```
on(press) {  
getURL("mailto:Dmitri.I.Albert<book_flash@mail.ru>?subject=Тема  
сообщения&body=Текст сообщения");  
}
```

При помощи функции `getURL()` также можно вызывать функции JavaScript или VBScript, используя следующий формат:

```
getURL("javascript:functionName") /* вызов функции JavaScript */  
getURL("vbscript:functionName") /* вызов функции VBScript */
```

Здесь `functionName` — вызываемая функция. Следующий пример содержит сценарий кнопки, нажатие которой приведет к появлению системного окна с сообщением:

```
on(press) {  
    getURL("javascript:alert(\"Для просмотра содержимого данного  
сайта необходимо наличие элемента управления Flash Player 7\");"  
}
```

Переход между HTML-страницами, содержащими различные части проекта, может быть также осуществлен при помощи текстовых гиперссылок (см. разд. "Статический текст" гл. 8).

Организация структуры проекта

В качестве альтернативы размещению всех частей проекта в одном файле SWF может быть применен подход, в соответствии с которым каждый раздел проекта помещается в отдельный SWF-файл, размещаемый на собственной HTML-странице. Каждый фильм, являющийся частью проекта, должен быть снабжен панелью навигации, позволяющей осуществить переход к другому разделу посредством загрузки HTML-страницы, содержащей соответствующий SWF-файл. В результате на сервере необходимо разместить файлы SWF и содержащие их HTML-страницы. Данный подход позволяет обеспечить достаточно быструю загрузку файлов проекта и может быть использован при разработке web-узла.

Создание сценариев

При размещении частей проекта на нескольких HTML-страницах следует иметь в виду следующие особенности.

- При создании сценариев, обеспечивающих возможность перехода с одной HTML-страницы на другую, целесообразно использовать относительную адресацию (относительный путь может выглядеть, например, следующим образом: `../mainHTML/logo.html`). Относительные ссылки задаются в качестве аргумента функции `getURL()` в рабочей среде Flash, но разрешаются относительно фактического местоположения HTML-страницы, содержащей SWF-файл, в котором производится вызов данной функции. В связи с этим, на стадии разработки необходимо четко представлять структуру взаимного расположения HTML-страниц, на которых

будут размещены различные части конечного проекта. С использованием параметра/атрибута `base` тегов `<object>` и `<embed>`, соответственно, можно задать базовую директорию, относительно которой будут разрешаться все относительные ссылки, используемые в SWF-файле, размещенном на данной HTML-странице (см. гл. 18). Этой возможностью можно воспользоваться, если HTML-страницы, содержащие различные части проекта, находятся в разных каталогах. Если относительный путь в аргументе функции `getURL()` указан неверно, требуемая страница загружена не будет.

- Для того чтобы SWF-файл мог быть загружен на HTML-страницу, необходимо верно указать путь к этому файлу при помощи параметров `movie` и `src` тегов `<object>` и `<embed>` соответственно (см. гл. 18). Если путь будет указан неверно, файл не сможет быть загружен.

Представление об объектно-ориентированном программировании

Ключевым понятием объектно-ориентированного программирования (ООП) является понятие *объекта* (*object*). Формально объект является структурой данных, объединяющей набор специфических для данного типа объектов свойств и функций. Объект может иметь визуальное выражение или являться абстракцией, позволяющей осуществлять управление и обработку данных с помощью своих *свойств* (*properties*) и функций, называемых *методами* (*methods*).

Объект создается на основе *класса* (*class*), описывающего характерные особенности целой категории объектов. Класс представляет собой шаблон, в котором определяются все свойства и методы категории объектов. *Свойства* объектов представляют именованные контейнеры данных, описывающие характеристики объекта или его состояние. *Методы* представляют собой функции, присущие данному классу объектов, позволяющие реализовывать его функциональность.

Объект, создаваемый на базе класса, называется *экземпляром* (*instance*) класса. Экземпляр класса получает доступ к свойствам и методам, определенным в данном классе. Каждый объект обладает именем и может иметь различные значения свойств и вызывать методы, определенные в его классе. Вся функциональность свойств и методов объекта реализуется внутри определения его класса, поэтому внутренняя механика выполнения объектом своих функций может быть не видна снаружи. Таким образом, при разработке сценариев можно просто обращаться к свойствам или вызывать определенные методы объекта, совершенно не заботясь о подробностях их внутренней реализации.

Язык ActionScript содержит набор встроенных классов, определяющих ряд свойств и методов, которые могут быть использованы в сценариях. Все встроенные классы (их свойства и методы) перечислены в папке **Built-in Classes** раздела **Actions toolbox** панели **Actions**. Наряду с прочими здесь содержаться классы MovieClip, Button, TextField и т. д. Эти классы обладают характерной особенностью, которая заключается в том, что экземпляр любого из них автоматически создается при введении в фильм экземпляра символа типа **MovieClip**, **Button** или при создании динамического или пользовательского текста соответственно. Для того чтобы присвоить имя объекту, являющемуся экземпляром одного из указанных классов, необходимо выделить муми-клип, кнопку или динамический/пользовательский текстовый блок и ввести имя в поле **Instance Name**, расположенное в левом верхнем углу Инспектора свойств. Так, в результате размещения на сцене экземпляра символа типа **Movie Clip** с именем mc создается объект mc, являющийся экземпляром класса MovieClip, получающий доступ к встроенным свойствам и методам, определенным в данном классе.

Примечание

Имя муми-клипа, кнопки или текстового блока (динамического или пользовательского) задается в поле **Instance Name** Инспектора свойств. Именно это имя используется для программного управления и при адресации объекта. Название символа как элемента библиотеки интерпретатором не обрабатывается. Зачастую причиной неполадок при выполнении сценария является отсутствие имени экземпляра (**Instance Name**).

Для обращения к свойству объекта можно использовать оператор точки или квадратные скобки (в этом случае в качестве имени свойства может быть использовано любое выражение, значением которого является строка). Следующий пример иллюстрирует различные форматы обращения к свойству объекта.

```
objectName.propertyName  
objectName["propertyName"]
```

Здесь:

- objectName — имя объекта, обращающегося к свойству;
- propertyName — название свойства.

Для вызова метода объекта применяется оператор точки. При этом используется следующая запись:

```
objectName.methodName
```

Здесь:

- objectName — имя объекта, для которого вызывается метод;
- methodName — название вызываемого метода.

Так, например, встроенный класс MovieClip содержит определение ряда методов, к числу которых относятся методы play(), stop(), gotoAndPlay(), gotoAndStop() и т. д. Название данных методов совпадает с названиями глобальных функций, рассмотренных выше в этой главе. Тем не менее, вызов функции и метода, имеющих одинаковые названия, приводит к различным результатам. Действие функции распространяется на монтажную линейку, содержащую данную функцию, а метод объекта, являющегося экземпляром символа типа муви-клип, воздействует на монтажную линейку данного экземпляра. Следующий пример содержит сценарий кадра основной монтажной линейки и иллюстрирует различие в вызове глобальной функции и метода объекта:

```
gotoAndPlay("start"); /*переход к кадру с меткой "start", расположенному на основной монтажной линейке*/  
mc.gotoAndPlay("start"); /*переход к кадру с меткой "start", расположенному на монтажной линейке клипа mc*/
```

Встроенные свойства клипов и кнопок

Обращение к свойству осуществляется из объекта, и действие этого свойства распространяется на вызвавший его объект. Следующие примеры иллюстрируют результаты обращения к свойствам различных объектов:

```
//сценарий кадра, содержащего клип mc  
mc.property = value; /*свойству property клипа mc присвоено значение value*/  
  
//сценарий кадра  
property = value; /* свойству property объекта, на монтажной линейке которого находится данный сценарий, присвоено значение value*/  
  
//сценарий кнопки  
on(press) {  
  
    property = value; /* свойству property объекта, на монтажной линейке которого находится данная кнопка, присвоено значение value*/  
}  
  
//сценарий клипа mc  
on(press) {  
    property = value; /* свойству property клипа mc присвоено значение value*/  
}
```

Два последних сценария формально абсолютно одинаковы, однако приводят к разным результатам. Это связано с тем, что обработчик событий on() име-

ет различные области видимости в зависимости от того, используется ли он в сценарии кнопки или клипа.

Использование встроенных свойств объектов позволяет реализовать множество возможностей программного управления объектами. В табл. 19.2 содержится перечень основных встроенных свойств клипов и кнопок (многие из данных свойств также могут быть применены к экземплярам динамических и пользовательских текстовых блоков). Перечень всех встроенных свойств содержится в разделе **Actions toolbox** панели **Actions**, для получения информации об их назначении можно обратиться к справочной системе Flash.

Таблица 19.2. Основные встроенные свойства клипов и кнопок

Название свойства	Описание свойства
_x	Задают или возвращают координаты объекта по горизонтали и по вертикали относительно текущего начала координат в пикселях. Если объект находится на основной монтажной линейке, его координаты отсчитываются относительно левого верхнего угла рабочей области. Если объект находится на монтажной линейке клипа, его координаты отсчитываются относительно точки регистрации родительского клипа. В качестве координат объекта всегда используются координаты его точки регистрации (перекрестья)
_y	
_width	Задают или возвращают значение ширины и высоты объекта в пикселях
_height	
_xscale	Задают или возвращают значение ширины и высоты объекта в процентах от исходного размера. Изменение масштаба экземпляра в рабочей среде документа влияет на значение данных свойств
_yscale	
_rotation	Задает или возвращает значение угла поворота объекта в градусах. Положительные значения соответствуют повороту по часовой стрелке, отрицательные — против часовой стрелки.
_alpha	Задает степень прозрачности объекта. Допустимые значения лежат в диапазоне от 0 до 100
_visible	Управляет отображением объекта. Принимает два значения: <code>false</code> или <code>true</code> . Значение <code>false</code> отключает отображение объекта (делает его невидимым). Значение <code>true</code> включает отображение объекта
enabled	Включает или отключает функционирование кнопки или кнопки-клипа (т. е. клипа, использующего обработчик <code>on()</code>). Принимает два значения: <code>false</code> или <code>true</code> . Значение <code>false</code> отключает функционирование кнопки, в результате этого кнопка перестает реагировать на воздействия мыши, однако событие <code>keyPress</code> , связанное с клавиатурой по-прежнему обрабатывается. Значение <code>true</code> активизирует кнопку и устанавливается по умолчанию
_xmouse	Возвращают текущие координаты курсора мыши относительно начала координат объекта, вызывающего данные свойства. Доступны только для чтения
_ymouse	

Таблица 19.2 (продолжение)

Название свойства	Описание свойства
<code>_totalframes</code>	Возвращает количество кадров монтажной линейки основного фильма или клипа, вызвавшего данное свойство. Доступно только для чтения. Принимает целочисленные значения
<code>_currentframe</code>	Возвращает номер текущего кадра монтажной линейки основного фильма или клипа, вызвавшего данное свойство, т. е. кадра, в котором находится воспроизводящая головка. Принимает целочисленные значения от 1 до <code>_totalframes</code> . Доступно только для чтения
<code>_framesloaded</code>	Возвращает количество загруженных кадров монтажной линейки основного фильма или клипа, вызвавшего данное свойство. Используется для проверки состояния загрузки основного фильма или внешних SWF-файлов, загружаемых в проигрыватель в процессе воспроизведения. Доступно только для чтения. Принимает целочисленные значения от 0 до <code>_totalframes</code>
<code>_parent</code>	Содержит ссылку на родительский объект. Родительским является объект, на монтажной линейке которого содержится объект, вызвавший данное свойство
<code>_name</code>	Возвращает идентификатор объекта, т. е. имя экземпляра (Instance Name) в виде строки
<code>_url</code>	Возвращает сетевой или локальный адрес текущего SWF-файла, откуда он был загружен в виде строки. Если данное свойство вызывается клипом, содержащим внешний загруженный файл SWF, оно возвращает адрес внешнего файла
<code>_droptarget</code>	Возвращает путь к клипу, над которым находится перетаскиваемый клип в виде строки. Путь указывается в нотации, использующей наклонную черту, и всегда начинается с символа "/". Для преобразования в точечный синтаксис можно использовать функцию <code>eval()</code> . Считается, что клип находится над другим клипом только, если его точка регистрации находится внутри области, занимаемой целевым клипом. Данное свойство работает только для клипов
<code>_target</code>	Возвращает путь к объекту в нотации, использующей наклонную черту, в виде строки
<code>useHandCursor</code>	Определяет внешний вид курсора при наведении на кнопку или кнопку-клип. Принимает два значения <code>false</code> или <code>true</code> . Значение <code>true</code> устанавливается по умолчанию и приводит к тому, что при наведении на кнопку курсор принимает вид указателя (руки). Значение <code>false</code> приводит к тому, что при наведении на кнопку внешний вид курсора не изменяется
<code>trackAsMenu</code>	Устанавливает для кнопки или клипа-кнопки режим Track as Menu Item (см. гл. 10). Принимает два значения: <code>false</code> или <code>true</code> . По умолчанию устанавливается значение <code>false</code>

Таблица 19.2 (окончание)

Название свойства	Описание свойства
hitArea	Указывает клип, используемый в качестве области реагирования кнопки-клипа, вызвавшего данное свойство. В качестве значения принимает ссылку на клип. Если данное свойство отсутствует или его значение не определено, в качестве области реагирования используется сам клип

При помощи встроенных свойств можно программным образом перемещать объекты, изменять их размеры, степень прозрачности и т. д. Так, пусть на сцене имеется экземпляр клипа (или кнопки) с именем `mc`. Для того чтобы наглядно продемонстрировать работу свойств, можно поместить `mc` за пределы рабочей области. Таким образом, без использования ActionScript экземпляр в конечном фильме виден не будет. Листинг 19.2 содержит сценарий, который, будучи помещен в кадр монтажной линейки, отцентрирует `mc` относительно рабочей области (при условии, что она имеет размер, заданный по умолчанию как 550×400 пикселов, и что точка регистрации `mc` совпадает с его геометрическим центром), пропорционально увеличит его масштаб в полтора раза, повернет на 30 градусов и установит пятидесятипроцентную степень прозрачности.

Листинг 19.2. Использование встроенных свойств объекта

```
//Сценарий кадра фильма, содержащего mc
//Позиционирование
mc._x=275;
mc._y=200;
//Масштабирование
mc._xscale=150;
mc._yscale=150;
//Задание угла поворота
mc._rotation=30;
//Задание степени прозрачности
mc._alpha=50;
```

Программирование клипов

Экземпляр муви-клипа, как и кнопка, может содержать сценарий. Сценарий клипа *всегда* размещается внутри обработчика событий (event handler), который инициирует выполнение данного сценария только при наступле-

нии определенного события. Для того чтобы назначить сценарий экземпляру символа типа муви-клип, необходимо выделить его на сцене и, убедившись в том, что в заголовке панели **Actions**, в категории **Current Selection** раздела **Script Navigator** и на закладке, расположенной слева под окном **Script Pane**, появилось указание на данный объект, ввести текст сценария в окно **Script Pane**.

Обработчики событий клипов

При назначении сценария экземпляру клипа можно использовать либо обработчик событий кнопок `on()` (см. выше), либо специальный обработчик событий клипов `onClipEvent()`. Формат использования обработчика `onClipEvent()` аналогичен формату применения обработчика `on()`:

```
onClipEvent(event) {  
    текст сценария;  
}
```

В отличие от обработчика `on()`, обработчик событий `onClipEvent()` не может обрабатывать несколько событий, однако сценарий клипа может содержать несколько обработчиков.

Существует целый ряд событий, которые могут быть перехвачены обработчиком `onClipEvent()` и повлечь выполнение соответствующего сценария клипа. Такие события называются *событиями клипа*. События клипов могут быть инициированы либо пользователем, либо процессом воспроизведения фильма, а также загрузкой и выгрузкой данных. В табл. 19.3 содержится перечень стандартных событий клипов.

Таблица 19.3. События клипов

Событие	Описание
<code>enterFrame</code>	Данное событие генерируется постоянно с частотой, равной скорости воспроизведения монтажной линейки. Сценарий клипа, использующего данный обработчик, будет многократно выполняться до тех пор, пока клип присутствует в фильме вне зависимости от того, воспроизводится его монтажная линейка или монтажная линейка основного фильма или нет. Это событие может быть успешно использовано для выполнения циклических процедур, т. е. многократно исполняемого фрагмента кода. Сценарий, содержащийся в обработчике события <code>enterFrame</code> , выполняется перед любым сценарием, расположенным на монтажной линейке клипа, содержащего данный обработчик
<code>mouseMove</code>	Данное событие генерируется при перемещении курсора мыши в пределах окна Flash Player с частотой, которую может обеспечить процессор

Таблица 19.3 (окончание)

Событие	Описание
mouseDown	Нажатие левой кнопки мыши при нахождении курсора в любой точке в пределах окна Flash Player
mouseUp	Отпускание левой кнопки мыши при нахождении курсора в любой точке в пределах окна Flash Player
keyDown	Нажатие любой клавиши клавиатуры
keyUp	Отпускание нажатой клавиши клавиатуры
load	Данное событие генерируется при появлении клипа в фильме, например, в результате вхождения воспроизведяющей головки в кадр, в котором содержится данный клип
unload	Данное событие генерируется при удалении клипа из фильма, т. е. при переходе воспроизведяющей головки из кадра, содержащего данный клип, в кадр, где он отсутствует
data	Данное событие генерируется при загрузке в фильм внешних данных, например, при помощи методов MovieClipLoader.loadClip(), loadMovie() или loadMovieNum() (см. гл. 21)

При помощи обработчика события `enterFrame` можно реализовать программную анимацию объекта, поскольку данное событие генерируется постоянно вне зависимости от воспроизведения монтажной линейки. Это дает возможность многократного выполнения сценария клипа, находящегося в единственном кадре монтажной линейки фильма. Листинг 19.3 содержит сценарий, перемещающий клип `mc` по горизонтали, с одновременным вращением его по часовой стрелке. Это достигается за счет периодического изменения значений его свойств `_x` и `_rotation` с частотой, равной скорости воспроизведения фильма. Приведенный сценарий является не совсем корректным, поскольку в результате его выполнения в определенный момент клип выйдет за пределы видимости, хотя изменение значений его свойств будет продолжаться. О возможностях ActionScript, позволяющих устранить этот недостаток, будет сказано в следующей главе.

Листинг 19.3. Линейное перемещение клипа

```
//Сценарий клипа
onClipEvent(enterFrame) {
    _x = _x + 5; /*увеличение текущего значения свойства _x на 5 перемещает
    клип на 5 пикселов вправо*/
    _rotation = _rotation - 15; /*уменьшение текущего значения свойства
    _rotation изменяет угол поворота на 15 градусов против часовой стрелки*/
}
```

Адресация клипов и кнопок

При обращении к свойствам и вызове методов объектов в сценариях необходимо использовать ссылки на соответствующие экземпляры. Если экземпляр объекта располагается на монтажной линейке, содержащей сценарий, ссылка на него осуществляется непосредственно по имени. Например, если в кадре основной монтажной линейки находится экземпляр клипа с именем mc, из сценария данного кадра к нему можно обратиться следующим образом:

```
mc._alpha=50; //Установка 50-процентной прозрачности mc  
mc.play(); //Запуск монтажной линейки mc
```

Для обращения к свойствам и методам объекта из сценария, назначенного этому объекту, указывать его имя не требуется. Сценарий кадра или обработчика событий может непосредственно ссылаться на содержащую его монтажную линейку. При этом важно иметь в виду нюансы, связанные с областью видимости обработчиков событий.

```
//Сценарий кадра основной монтажной линейки  
_alpha=50; /*Установка 50-процентной прозрачности всего содержимого  
фильма*/  
play(); /* Запуск основной монтажной линейки */  
//Сценарий кнопки  
on(press){  
    _alpha=50; /* Установка 50-процентной прозрачности объекта, на монтажной  
линейке которого содержится данная кнопка */  
    play(); /*Запуск монтажной линейки, на которой содержится данная кнопка */  
}  
//Сценарий клипа  
on(press){  
    _alpha=50; /* Установка 50-процентной прозрачности данного клипа */  
    play(); /*Запуск монтажной линейки данного клипа */  
}  
//Сценарий клипа  
onClipEvent(mouseDown){  
    _alpha=50; /* Установка 50-процентной прозрачности данного клипа */  
    play(); /*Запуск монтажной линейки данного клипа */  
}
```

Flash-фильм может содержать вложенные объекты. Так, клип, расположенный на основной монтажной линейке, может содержать на своей линейке другой клип, который, в свою очередь, содержит кнопку. Зачастую возникает необходимость использования ссылок на объекты, расположенные на других монтажных линейках. В этом случае приходится использовать адре-

сацию объектов, т. е. указание пути к требуемому объекту. Клип или кнопка, вложенные в другой клип, интерпретируются как свойства содержащего их клипа, называемого *родительским*. В связи с этим доступ к вложенным клипам или кнопкам может быть осуществлен при помощи оператора точки или при помощи квадратных скобок (оператора, обеспечивающего доступ к элементам массива, []). В последнем случае имена адресуемых объектов должны быть заключены в кавычки.

Абсолютная адресация

Адресация объектов может быть *абсолютной* или *относительной*. Абсолютная адресация предполагает указание пути, начиная с самого верхнего уровня, т. е. с основной монтажной линейки фильма. Абсолютный путь не зависит от местоположения объекта, содержащего сценарий, в иерархии объектов. Абсолютные ссылки всегда разрешаются относительно основной монтажной линейки. Применение абсолютных ссылок аналогично указанию пути к файлу, расположенному в файловой системе, начиная с указания диска. Для ссылки на основную монтажную линейку фильма используется глобальное свойство `_root`. Абсолютный путь всегда предполагает опускание вниз по иерархии вложенных объектов, начиная с основной монтажной линейки. Рис. 19.5 иллюстрирует иерархию, состоящую из экземпляра клипа, расположенного на основной монтажной линейке, и двух последовательно вложенных экземпляров клипов.

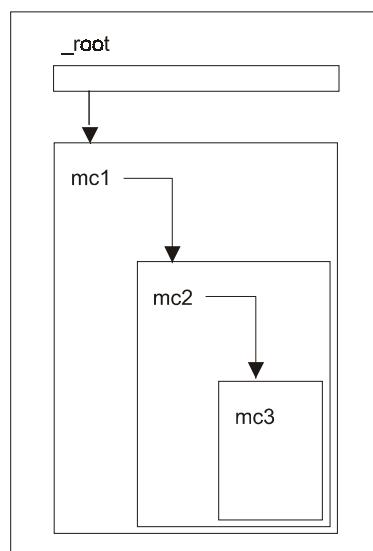


Рис. 19.5. Иерархия вложенных клипов

Для того чтобы обратиться к экземпляру mc3, используя абсолютную адресацию, можно применить один из следующих вариантов записи:

```
//Сценарий кадра  
_root.mc1.mc2.mc3._rotation=30;  
_root["mc1"] ["mc2"] ["mc3"].gotoAndPlay("start");
```

Поскольку в данном случае используется относительная адресация, ссылка будет корректно разрешаться вне зависимости от ее местоположения в иерархии объектов. То есть такая запись может быть использована для обращения к mc3 из сценария, расположенного на любом уровне иерархии.

Для того чтобы, находясь на любом уровне иерархии, обратиться к основной монтажной линейке, нужно просто вызвать глобальное свойство `_root`. Следующий сценарий может быть помещен в кадр монтажной линейки любого из клипов mc1, mc2 или mc3. Вне зависимости от местоположения он остановит воспроизведение основной монтажной линейки:

```
_root.stop();
```

Таким образом, при использовании абсолютной адресации имеет значение только местоположение целевого объекта относительно основной монтажной линейки (`_root`), а не местоположение сценария, обращающегося к данному объекту.

Относительная адресация

Относительная адресация предполагает указание пути относительно местоположения сценария, содержащего обращение к целевому объекту. При использовании относительной адресации имеет значение местоположение сценария, обращающегося к объекту, относительно местоположения этого объекта. При этом никакой привязки к `_root` в этом случае не существует. В отличие от абсолютной адресации, предлагающей только спуск вниз по иерархии объектов, относительная адресация позволяет перемещаться и вверх, и вниз по иерархическому дереву. При этом для перемещения на нижележащий уровень используется оператор точки (или `[]`), а для перемещения на вышележащий уровень используется свойство `_parent`, возвращающее ссылку на родительский объект. Следующие примеры иллюстрируют использование относительной адресации (см. рис. 19.5):

```
//Сценарий кадра основной монтажной линейки  
mc1.mc2.mc3._visible=false;//включение отображения mc3  
// сценарий кадра монтажной линейки mc1  
mc2.mc3._alpha=50;//установка 50-процентной прозрачности mc3  
// сценарий кадра монтажной линейки mc3  
_parent._parent._rotation=45;//поворот mc1 на 45 градусов
```

```
// сценарий кадра монтажной линейки mc1
_parent.stop(); // остановка воспроизведения основной монтажной линейки
```

При применении относительной адресации для ссылки на текущий объект можно использовать ключевое слово `this`. Эта возможность является альтернативой использованию неявных ссылок в обработчиках событий кнопок и клипов. При использовании в обработчике событий `on()`, прикрепленному к кнопке, `this` является ссылкой на монтажную линейку, содержащую данную кнопку. При использовании в обработчике событий `on()` или `onClipEvent()`, прикрепленному к клипу, `this` является ссылкой на монтажную линейку этого клипа. Следующие предложения, помещенные в кадр монтажной линейки, приводят к одинаковому результату.

```
stop(); // неявная ссылка
this.stop(); // явная ссылка
```

Редактор ActionScript позволяет автоматически вставлять в сценарий путь к требуемому объекту. Для этого необходимо воспользоваться кнопкой **Insert a target path** , расположенной над окном **Script pane**. В результате нажатия появляется диалоговое окно **Insert Target Path**, содержащее графическое представление структуры фильма. При выделении пиктограммы требуемого объекта в строке автоматически появляется путь к этому объекту. Для отображения абсолютного или относительного пути нужно установить переключатель в положение **Absolute** или **Relative** соответственно. Нажатие кнопки **OK** вставляет путь в сценарий.

Листинг 19.4 содержит сценарий клипа, заставляющий его следовать за курсором мыши. При перемещении мыши свойствам `_x` и `_y` клипа присваиваются координаты курсора мыши относительно монтажной линейки, содержащей данный клип.

Листинг 19.4. Следование объекта за курсором

```
// Сценарий клипа, следующего за курсором
onClipEvent(mouseMove) {
    this._x=_parent._xmouse;
    this._y=_parent._ymouse;
    updateAfterEvent();
}
```

Регистрация координат мыши относительно начала координат родительской монтажной линейки (`_parent`) позволяет поместить клип на любой уровень вложенности, при этом сценарий будет выполняться корректно. Используемая в сценарии функция `updateAfterEvent()` позволяет обновлять (перерисовывать) экран с частотой, равной частоте генерации события `mouseMove`, делая перемещение клипа более плавным.

Предложение *with*

Предложение *with* позволяет осуществлять обращение к свойствам и методам объекта без многократного указания его имени. Это позволяет обеспечить большую компактность записи. Предложение *with* имеет следующий синтаксис:

```
with (object) {  
    statements;  
}
```

Здесь *object* является ссылкой на объект, в контексте которого будут выполняться все предложения *statements*, заключенные в фигурные скобки. Если у объекта, указанного в круглых скобках предложения *with*, отсутствует свойство или метод, обращение к которым выполняется из блока, ограниченного фигурными скобками, то интерпретатор будет искать данное свойство или метод на монтажной линейке, содержащей предложение *with*.

Следующие два сценария являются эквивалентными и демонстрируют преимущество использования предложения *with*.

```
//Сценарий 1. Многократная ссылка на объект  
myClip._x=275;  
myClip._y=200;  
myClip._rotation=30;  
myClip._alpha=50;*/  
//Сценарий 2. Использование предложения with  
with(myClip){  
    _x=275;  
    _y=200;  
    _rotation=30;  
    _alpha=50;  
}
```

Управление автономным проигрывателем Projector

При работе с автономным проигрывателем, называемым *проектором* (Projector), можно непосредственно из Flash-фильма управлять режимом просмотра, параметрами масштабирования, а также выполнять другие операции (табл. 19.4). Для управления проектором используется глобальная функция *fscommand()*.

Данная функция может принимать от одного до двух строковых аргументов и имеет следующий формат:

```
fscommand("command", "parameters")
```

Здесь:

- command* — строка, указывающая название выполняемой команды;
- parameters* — строка, передаваемая проигрывателю Flash Player или содержащему его приложению (web-браузер, Macromedia Director и др.) в качестве параметра.

Таблица 19.4. Команды и параметры функции *fscommand()*

Команда	Параметр	Назначение
fullscreen	true или false	Значение <i>true</i> устанавливает для Flash Player полноэкранный режим, значение <i>false</i> возвращает обычный режим
allowscale	true или false	Значение <i>false</i> исключает возможность масштабирования фильма при изменении размеров окна проектора. В этом случае его размеры строго соответствуют размерам рабочей области, заданным в рабочей среде. Значение <i>true</i> приводит к тому, что фильм масштабируется (пропорционально) в окне таким образом, чтобы занимать максимальную область
showmenu	true или false	Включает или отключает отображение главного меню автономного проигрывателя. Значение <i>true</i> также приводит к тому, что контекстное меню, которое появляется при щелчке правой кнопкой мыши в окне Flash Player, будет содержать полный набор команд (управление воспроизведением, масштабирование, качество и т. д.). При установке значения <i>false</i> в контекстном меню будет доступна только команда About Macromedia Flash Player 7
trapallkeys	true или false	Значение <i>true</i> приводит к тому, что нажатие любых клавиш (в том числе и их сочетаний) передается непосредственно фильму. Это исключает возможность использования горячих клавиш автономного проигрывателя для выполнения различных действий (например, выход из проектора <i><Ctrl>+<Q></i> и т. д.)
quit	Нет	Закрывает автономный проектор
exec	Путь к приложению в файловой системе	Позволяет запустить внешнее приложение (исполнимый файл с расширением <i>exe</i> , <i>bat</i> и т. д.). Вызываемое приложение должно находиться в поддиректории с названием <i>fscommand</i> , размещенной в одной директории с файлом проектора

Листинг 19.5 демонстрирует пример использования функции `fscommand()`.

Листинг 19.5. Применение функции `fscommand()`

```
fscommand("fullscreen","false"); //Полноэкранный режим  
fscommand("allowscale","false"); //Запрет на изменение размера фильма  
fscommand("showmenu","false"); //Отключение команд контекстного меню  
fscommand("trapallkeys","true"); //Перехват клавиш включен  
launchBut.onPress=function() {  
    fscommand("exec", "coda.exe"); //Запуск внешнего приложения по нажатию  
    // кнопки launchBut  
};  
quitBut.onPress=function() {  
    fscommand("quit"); //Закрытие проектора по нажатию кнопки quitBut  
};
```

Поведения

Поведения (Behaviors) представляют собой готовые фрагменты программного кода, которые можно поместить в фильм для обеспечения выполнения стандартных процедур, таких как переход к другому кадру, загрузка внешнего файла, загрузка сетевого ресурса и т. д. Для работы с поведениями используется панель **Behaviors**, которую можно открыть при помощи команды главного меню **Window>Development Panels>Behaviors** или сочетания клавиш **<Shift>+<F3>**. Внешний вид панели представлен на рис. 19.6.

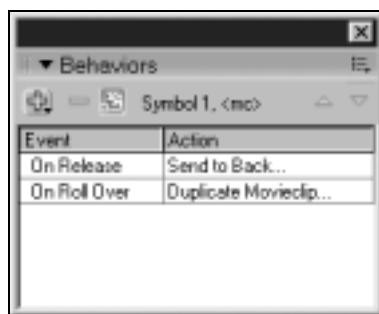


Рис. 19.6. Панель **Behaviors**

Для того чтобы назначить сценарий, необходимо выделить целевой объект на сцене (кадр монтажной линейки, экземпляр клипа, кнопки или другой графический объект), при этом его название будет указано в верхней части

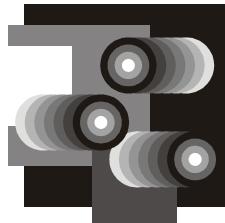
панели **Behaviors**. Если поведение назначается графическому объекту (экземпляру символа типа **Graphic**, растровому изображению, векторной форме), то в результате назначения он будет автоматически конвертирован в муви-клип. Далее нужно щелкнуть курсором на кнопке **Add Behavior**  и выбрать из появившегося меню требуемое действие (табл. 19.5). Некоторые действия требуют указания дополнительных параметров, например, ссылки на кадр, в который должен быть выполнен переход или адрес загружаемого ресурса. В этом случае будет выведено диалоговое окно, предлагающее задать требуемые параметры. В результате выбора действия и задания параметров в столбце **Action** появится его название. После этого можно щелкнуть на той же строке в столбце **Event**, для того чтобы выбрать событие, наступление которого повлечет за собой выполнение выбранного сценария. Сценарий, помещенный при помощи панели **Behavior**, как и сценарий, введенный вручную, может быть просмотрен в панели **Actions**. Для того чтобы удалить поведение, можно воспользоваться кнопкой **Delete Behavior**  или просто удалить сценарий в панели **Actions**. В табл. 19.5 представлен перечень поведений экземпляров клипов.

Таблица 19.5. Поведения экземпляров

Поведение	Назначение	Параметры
Load Graphic	Загружает внешний файл JPEG	Путь к загружаемому файлу JPEG, а также ссылка на целевой объект (клип или <code>_root</code>), в который осуществляется загрузка
Load External Movie Clip	Загружает внешний файл SWF	Путь к загружаемому файлу SWF, а также ссылка на целевой объект (клип или <code>_root</code>), в который осуществляется загрузка
Unload Flash Movie	Удаляет динамически загруженный файл SWF	Имя экземпляра выгружаемого клипа
Duplicate Movieclip	Создает дубликат клипа	Имя экземпляра клипа, который должен быть дублирован, а также смещение дубликата относительно оригинала по X и по Y
GotoAndPlay at frame or label	Переход на указанный кадр и продолжение воспроизведения	Имя объекта, к кадру монтажной линейки которого осуществляется переход, а также номер или метка требуемого кадра
GotoAndStop at frame or label	Переход на указанный кадр и остановка воспроизведения	Имя объекта, к кадру монтажной линейки которого осуществляется переход, а также номер или метка требуемого кадра
Bring to Front	Перемещает клип на передний план по отношению к остальным клипам, находящимся в стеке	Имя экземпляра клипа

Таблица 19.5 (окончание)

Поведение	Назначение	Параметры
Bring Forward	Перемещает клип на один уровень вверх в стеке	Имя экземпляра клипа
Send to Back	Перемещает клип на задний план по отношению к остальным клипам, находящимся в стеке	Имя экземпляра клипа
Send Backward	Перемещает клип на один уровень вниз в стеке	Имя экземпляра клипа
Start Dragging movieclip	Включает режим перетаскивания клипа	Имя экземпляра клипа



Глава 20

Разработка сценариев

Ошибаться человеку свойственно, но окончательно
все запутать может только компьютер.

Мерфи, Пятый закон ненадежности

Переменные. Типы данных. Операторы

Создание переменных

Переменные являются одним из ключевых элементов языка программирования. Переменная (Variable) представляет собой именованный контейнер, содержащий определенный фрагмент данных (datum). Переменная состоит из двух элементов: *имени* (name) и *значения* (value). Имя переменной позволяет осуществлять доступ к хранимому в ней значению и используется для извлечения этого значения или его модификации (изменения).

В качестве значения переменной может быть использован *литерал* (символьное выражение), т. е. простые данные, не требующие вычисления и используемые в неизменном виде, или выражение, т. е. фрагмент программного кода, требующий вычисления, в результате которого выдается одиночное значение. Ниже представлены различные варианты оформления литералов, принятые в ActionScript, и примеры выражений:

3.1415926 // Числовой литерал

"Альберт Д.И." // Строковый литерал

function (args) { statements } // Литерал функции

[e11,e12,...] // Литерал массива

2+3 // Выражение, дает значение 5

x+y // Дает сумму значений переменных x и y

"Дмитрий"+"_Альберт" // Дает значение Дмитрий_Альберт

Процесс создания переменной называется *объявлением* (declaration) переменной. Для объявления переменной используется ключевое слово `var`, за которым следует имя создаваемой переменной. Переменная содержит данные определенного типа (число, строка, логический (булев), функция и т. д.). Тип данных, помещаемых в переменную, указывается при ее объявлении после двоеточия. Эта особенность связана со *строгим контролем типов данных* (Strict Data Typing), введенным в последней версии ActionScript 2.0. Объявление переменной имеет следующий формат:

```
var nVar:Number; //Объявление переменной числового типа с именем nVar
```

Допускается объявление сразу нескольких переменных, причем их типы могут различаться, например:

```
var nVar1:Number, nVar2:String, nVar3:Boolean; /*Объявление переменных  
nVar1, nVar2, nVar3 числового, строкового и логического типов  
соответственно*/
```

При задании имен переменных следует придерживаться следующих правил:

- имя переменной должно состоять *только* из букв (латинского алфавита), цифр, а также символа нижнего подчеркивания и знака \$;
- имя переменной может начинаться *только* с буквы, символа нижнего подчеркивания или знака \$;
- имена переменных являются *регистrozависимыми* (case sensitive), т. е. `nVar` и `nvar` — разные имена. Эта особенность появилась в ActionScript с выходом Flash Player 7.0 и имеет место вне зависимости от того, используется ли ActionScript 1.0 или ActionScript 2.0. В предыдущих версиях имена переменных не зависели от регистра;
- в качестве имен переменных нельзя использовать специальные и зарезервированные (ключевые) слова, такие как `true`, `false`, `null` и т. д. При использовании режима подсветки **Syntax coloring** ключевые слова окрашиваются в соответствии с цветом, заданным для элементов **Keywords**.

Имя переменной должно быть "говорящим" и отражать ее назначение. При именовании переменных, состоящих из нескольких слов, удобно использовать заглавные буквы для выделения каждого следующего слова, кроме первого (этот принцип применяется для большинства встроенных элементов ActionScript), например — `maxSpeedUp`. Кроме того, для большей наглядности кода удобно использовать префиксы, отражающие тип данных, хранимых в переменной (например, `nSpeed` — числовая (`Number`) переменная, `sName` — строковая (`String`) переменная).

Строгая типизация данных позволяет избежать возможных ошибок, связанных с попыткой присвоения переменной данных, тип которых отличен от типа переменной, указанного при ее объявлении. В процессе компиляции (публикации) происходит автоматическая проверка соответствия типов дан-

ных. При обнаружении несоответствия типов в панель **Output** в среде тестирования выводится сообщение об ошибке с указанием места ее возникновения. Кроме того, применение строгого контроля типов позволяет отображать подсказки по коду (*code hints*), появляющиеся при вводе точки после имени объекта, при создании которого был указан его тип. Всплывающая подсказка будет содержать перечень всех доступных для данного объекта свойств и методов. Применение строгого контроля типов данных не является обязательным, однако рекомендуется для обеспечения лучшего стиля программирования и приведения сценариев к большему соответствуанию стандарту ECMA 262.

Использование строгой типизации данных поддерживается только версией ActionScript 2.0. В связи с этим при публикации исходного документа необходимо убедиться в том, что в поле **ActionScript version** вкладки **Flash** окна **Publish Settings** установлено значение ActionScript 2.0.

При создании переменной ей автоматически присваивается особое значение `undefined`, указывающее на то, что переменная не содержит данных. Для того чтобы присвоить переменной определенное значение (поместить в нее данные), необходимо выполнить операцию *присваивания* (*assignment*). Присваивание осуществляется при помощи оператора присваивания `=`, причем слева от него указывается имя переменной (`variableName`), а справа — присваиваемое ей значение (`value`):

```
variableName = value;
```

Следующий код объявляет переменную числового типа с именем `nVar` и присваивает ей значение 5:

```
var nVar:Number;  
nVar=5;
```

Объявление и присваивание можно выполнить в одном предложении. Следующая запись эквивалентна предыдущей:

```
var nVar:Number=5;
```

При объявлении нескольких переменных в одном предложении им также сразу могут быть присвоены соответствующие значения:

```
var nVar1:Number = 5, sVar2:String = "Flash", bVar3:Boolean=true;
```

Если в сценарии осуществляется присваивание значения переменной, которая не была предварительно объявлена, то интерпретатор автоматически создаст переменную, тип которой будет определен типом присваиваемого ей значения, оформленного в соответствии с правилами оформления литералов. Однако при использовании переменной, которая не была объявлена, в выражениях она принимает значение `undefined` или `NaN` (*Not-a-Number*, не число), что может привести к нежелательному результату.

Например:

```
var nSquare:Number=x*x;  
trace(nSquare); /*Выводит NaN, т.к. переменная x не объявлена и не  
содержит значения*/
```

Если объявление переменной и присвоение ей значения осуществляется до использования переменной в выражении, результат будет корректным:

```
var x:Number=5;  
var nSquare:Number=x*x;  
trace(nSquare); //Выводит 25
```

При помощи команды главного меню среди тестирования **Debug>List Variables** можно вывести в панели **Output** список всех объявленных в фильме переменных и принимаемых ими значений. Кроме переменных, используемых в фильме, данный список всегда содержит автоматически созданную переменную \$version, указывающую номер версии Flash Player, в котором воспроизводится ролик. Следует иметь в виду, что информация о переменных, выводимая в панели **Output**, не обновляется динамически. Перечень переменных и их значений указывается на момент выполнения команды **List Variables**. Для получения информации об изменении значений переменных нужно использовать функцию `trace()`.

Область видимости и срок жизни переменных

Переменная может быть создана в сценарии кадра основной монтажной линейки или клипа, либо в сценарии клипа или кнопки. После создания переменная является непосредственно доступной всем сценариям монтажной линейки, на которой она была определена (или, как иногда говорят, к которой она прикреплена). Так, если переменная определена в кадре монтажной линейки, к ней можно непосредственно обратиться из сценария кадра данной монтажной линейки или из сценария кнопки, расположенной в пределах данной монтажной линейки. Однако данная переменная будет не видна сценарию клипа и сценарию кадра монтажной линейки данного клипа, даже если клип находится на монтажной линейке, содержащей объявление этой переменной.

Допустим, что в первом кадре основной монтажной линейки определена переменная `nVar`.

```
var nVar:Number=13;
```

Далее во второй кадр основной монтажной линейки поместим следующий сценарий:

```
trace(nVar);
```

При тестировании фильма в панели **Output** будет выведено значение переменной `nVar`.

Если назначить данный сценарий кнопке, расположенной на основной монтажной линейке, результат будет точно таким же, поскольку областью видимости сценария кнопки является содержащая ее монтажная линейка:

```
on(press) {  
    trace(nVar); //Выводит 13 в панели Output  
}
```

Если же предложение `trace(nVar)` будет помещено в блок обработчика событий клипа (`on()` или `onClipEvent()`), расположенного на основной монтажной линейке, либо в кадр его монтажной линейки, то результат будет иным:

```
onClipEvent(keyDown) {  
    trace(nVar); /*Выводит undefined, поскольку в области видимости сценария переменной nVar не существует*/  
}
```

Примечание

Доступ к переменным основной монтажной линейки может быть непосредственно осуществлен из всех сценариев основной монтажной линейки вне зависимости от их принадлежности к той или иной сцене (`scene`).

Для того чтобы получить возможность обращаться к переменной, определенной на одной монтажной линейке (основной или клипа), из сценариев других монтажных линеек необходимо использовать адресацию с применением оператора точки, точно так же, как и при обращении к клипам и кнопкам (см. разд. "Адресация клипов и кнопок" гл. 19).

Пусть в кадре основной монтажной линейки имеется три последовательно вложенных клипа с именами `mc1`, `mc2`, `mc3` (см. рис. 19.5). Следующий сценарий, помещенный в кадр основной монтажной линейки, приводит к автоматическому созданию переменной `nVar` на монтажной линейке клипа `mc3` и присваивает ей значение 10:

```
//Сценарий кадра основной монтажной линейки  
mc1.mc2.mc3.nVar=10;  
  
//Сценарий кадра монтажной линейки клипа mc3  
trace(nVar); //Выводит 10
```

Если переменная определена на основной монтажной линейке, то обратиться к ней из вложенного клипа можно при помощи свойства `_root`.

```
//Сценарий кадра основной монтажной линейки  
var sVar:String; //Создана переменная sVar
```

```
sVar="Flash"; //Переменной sVar присвоено значение "Flash"  
//Сценарий кадра монтажной линейки клипа mc2  
root.sVar="Flash MX 2004"; /*Переменной sVar присвоено значение "Flash  
MX 2004", которое переопределю предыдущее значение*/  
trace (sVar); //Выводит Flash MX 2004
```

В качестве альтернативы использованию адресации существует возможность создания глобальных переменных, доступ к которым может быть осуществлен непосредственно с любой монтажной линейки без указания целевого пути. Для создания глобальной переменной используется идентификатор `_global`, являющийся ссылкой на глобальный объект, который содержит основные классы ActionScript. Для создания глобальной переменной используется следующий формат:

```
_global.variableName = value;
```

При создании глобальной переменной ключевое слово `var` не используется. Доступ к значению глобальной переменной может осуществляться непосредственно по имени без указания идентификатора `_global`, однако в этом случае глобальная переменная может быть перекрыта переменной с таким же именем, определенной на монтажной линейке. Для того чтобы присвоить значение глобальной переменной, нужно использовать идентификатор `_global`. Следующий пример иллюстрирует принцип использования глобальных переменных (см. рис. 19.5).

```
//Сценарий кадра основной монтажной линейки  
_global.glVar = "Flash"; //Создание глобальной переменной  
//Сценарий клипа mc3  
trace (glVar); //Выводит Flash  
_global.glVar = "Flash MX 2004"; /*Переопределение значения глобальной  
переменной*/  
trace (glVar); //Выводит Flash MX 2004  
glVar = "Macromedia" //Создание переменной монтажной линейки клипа mc3  
trace (glVar); /*Выводит Macromedia. Глобальная переменная перекрыта  
внутренней переменной*/  
trace (_global.glVar); //Выводит Flash MX 2004, т. е. значение глобальной  
переменной
```

Следует иметь в виду, что при воспроизведении фильма сценарии кадров монтажных линеек выполняются в исходящем порядке. Сначала выполняется сценарий кадра основной монтажной линейки, затем сценарий кадра монтажной линейки клипа, расположенного на основной монтажной линейке, затем сценарий кадра вложенного в него клипа и т. д. В некоторых случаях это может привести к нежелательному результату. Так, например, пусть в кадре монтажной линейки клипа `mc3` содержится следующий сценарий:

```
_root.sVar = "Flash" //Создание переменной основной монтажной линейки
```

Пускай монтажная линейка основного фильма состоит только из одного кадра. В этот кадр поместим следующее предложение:

```
trace (sVar); //Выводит undefined
```

В панель **Output** выводится значение `undefined`, поскольку на момент выполнения сценария кадра основной монтажной линейки переменная еще не определена; если на основную монтажную линейку добавить еще один кадр, то при повторном воспроизведении в панель **Output** будет выведено значение `Flash`, поскольку при первом воспроизведении переменная `sVar` была создана и следующий вызов функции `trace()` выведет ее значение.

Переменная, определенная в кадре монтажной линейки, становится доступной только после того, как воспроизводящая головка впервые войдет в этот кадр.

Переменные основной монтажной линейки существуют на протяжении всего того времени, пока содержащий их документ присутствует в проигрывателе Flash Player. При выгрузке документа из проигрывателя (осуществляемой с помощью метода `MovieClipLoader.unloadClip()`, `removeMovieClip()` или `unloadMovie()`) все его переменные теряются.

Переменные клипов существуют на протяжении всего того времени, пока клип присутствует в фильме. Как только клип удаляется со сцены (или заменяется другим клипом), все его переменные теряются.

Понятие о типах данных и операциях с ними

При обработке информации в компьютере все данные разбиваются на категории. Категория, к которой относятся те или иные данные, определяется заложенным в них контекстом информации. К основным типам данных ActionScript относятся следующие:

- числовой (Number) — используется для представления чисел и выполнения различных математических операций;
- строковый (String) — используется для представления произвольных последовательностей символов, состоящих из букв, чисел или знаков пунктуации;
- логический или булевский (Boolean) — используются для проверки выполнения условий, вывода результата сравнения. Данные этого типа могут принимать только два значения: `true` (истина) или `false` (ложь);
- `null`, `undefined` — особый тип, используемый для указания на отсутствие данных;
- мани-клип (MovieClip) — применяется для обработки экземпляров клипов;
- объект (Object) — используется для хранения и обработки сложных структур данных.

Типы данных подразделяются на *примитивные* (primitive) и *ссыльочные* (reference). Примитивные типы содержат неизменное фактическое значение представляемого ими элемента. Ссыльочные данные позволяют обрабатывать несколько взаимосвязанных фрагментов данных и содержат ссылку на значение, хранимое в представляемом ими элементе. К примитивным типам относятся типы Number, String, Boolean, null и undefined. Типы MovieClip и Object являются ссыльочными типами данных.

Данные могут быть описаны непосредственно при помощи литерала, представляющего собой последовательность букв, цифр и символов пунктуации и являющегося буквальным представлением данных в программном коде. Данные также могут быть созданы при помощи сложных выражений, которые в отличие от литералов должны быть вычислены в ходе выполнения программы. В результате вычисления выражения получается единичный элемент данных.

Для каждого типа данных существуют определенные правила создания литералов. Данные различного типа допускают различные возможности обращения с представляемыми ими значениями.

Тип литерала или выражения может быть определен при помощи оператора `typeof()`:

```
var nVar:Number=5;
var sVar:String="Flash";
var bVar:Boolean=true;
trace(typeof(nVar)); //Выводит number
trace(typeof(sVar)); //Выводит string
trace(typeof(bVar)); //Выводит boolean
trace(typeof(gotoAndPlay)); //Выводит function
```

Числовой тип данных

Числовой литерал включает числовые символы, знак десятичной точки и порядок (десятичный логарифм):

```
//Числовые литералы
12//Целое число
3.1415926//Число с плавающей точкой
.5//Вариант записи дробного числа меньше единицы (0,5)
2e3//2000 = 103.2
2e-3//0,002 = 1/103.2
```

Для записи шестнадцатеричного числа нужно использовать префикс 0x:

```
0xFF//255
```

Числовой тип данных содержит несколько специальных значений. К их числу относятся: `NaN` (указывает на то, что данные относятся к числовому типу, но не являются числом), `Infinity` и `-Infinity` (значения, превышающие максимально и минимально допустимые значения соответственно) и др.

Для того чтобы преобразовать нечисловой тип данных в число, можно воспользоваться функцией `Number()`. Если преобразование в числовой тип невозможно, данная функция возвращает значение `NaN`:

```
trace(Number("3.1415")); //Выводит 3.1415
var sVar:String="24";
trace(Number(sVar)); //Выводит 24
trace(Number("Can't do it")); //Выводит NaN
```

Арифметические операции

Для выполнения арифметических операций используются операторы `+` (сложение), `-` (вычитание), `*` (умножение), `/` (деление), а также `%` (деление по модулю). Последняя операция возвращает остаток от деления одного операнда на другой. Операции могут выполняться непосредственно с числами или с переменными, содержащими числовые данные. При использовании такой переменной в выражении будет подставлено ее значение:

```
var s:Number=5*6;
trace(s) //Выводит 30
var x:Number=2;
var y:Number=3;
var z:Number=x+y;
trace(z); //Выводит 5
```

Модификация значения переменной

Для изменения значения переменной используется запись следующего вида:

```
nVar=nVar+10; //Приращение значения nVar на 10
nVar=nVar*5; //Увеличение значения nVar в 5 раз
```

При обработке такого предложения интерпретатор выполняет две операции: в первой строке выполняется сложение, а затем присваивание нового значения, а во второй — умножение и присваивание. Существует вариант более компактной записи подобных конструкций с использованием так называемых *составных операций* (*compound assignment*), которые объединяют обе операции в одну. Для этого в сочетании с оператором присваивания используются соответствующие арифметические операторы. Следующие предложения эквивалентны приведенным выше:

```
nVar+=10; //Приращение значения nVar на 10. Сложение с присваиванием
nVar*=5; //Увеличение значения nVar в 5 раз. Умножение с присваиванием
```

Совершенно аналогично можно использовать операторы `-=` (вычитание с присваиванием), `/=` (деление с присваиванием) и `%=` (деление по модулю с присваиванием).

Еще одна возможность модификации значения переменной заключается в использовании **унарных** (т. е. принимающих только один операнд) операторов **инкрементирования** `++` и **декрементирования** `--`. Инкрементирование увеличивает значение своего операнда на единицу, а декрементирование, соответственно, уменьшает значение своего операнда на единицу. Следующие предложения попарно эквивалентны:

```
nVar+=1;           //Текущее значение nVar увеличивается на единицу
nVar++;            //Текущее значение nVar увеличивается на единицу
nVar-=1;            //Текущее значение nVar уменьшается на единицу
nVar--;            //Текущее значение nVar уменьшается на единицу
```

Инкрементирование и декрементирование может быть *префиксным* или *постфиксным*:

```
++ nVar;          // префиксное инкрементирование
nVar++;            // постфиксное инкрементирование
```

Различия между типами инкрементирования/декрементирования имеют место только при использовании данных операторов в составе выражения. Следующий пример иллюстрирует это различие.

```
x=5;
y=x--; /* у принимает значение переменной x, равное 5, после чего x
уменьшается на единицу и становится равным 4*/
y=-x; /* x уменьшается на единицу и становится равным 3, после чего у
принимает значение x, равное 3 */
```

Сравнение значений переменных

Для сравнения значений переменных используются операторы сравнения `>` (больше), `<` (меньше), `>=` (больше или равно), `<=` (меньше или равно); оператор равенства `==` (равно), а также оператор неравенства `!=` (не равно). Результатом сравнения является логическое значение `true` (истина) или `false` (ложь), указывающее на то, верно ли проверяемое отношение. Следует обратить внимание, что при проверке равенства двух значений необходимо использовать оператор равенства `==`, а не оператор присваивания `=`. В противном случае вместо сравнения произойдет присваивание значения операнда, расположенного справа от оператора присваивания, операнду, расположенному слева от него. Следующие примеры иллюстрируют применение данных операторов:

```
5<4 //false
var x=2, y=3
```

```
x>0; //true
x*y==6; //true
x!=y; //true
```

Строковый тип данных

Строковый тип данных используется для хранения произвольных последовательностей символов, состоящих из цифр, букв и знаков пунктуации. Строковый литерал представляет собой набор символов, заключенный в кавычки (двойные или одинарные), например:

```
"Flash MX 2004"
"Дмитрий & Елена Альберт"
'dmitri@avalon.ru' /*Допускается использование одинарных кавычек
(апострофов) */
"3.1415926" //Это тоже строка
```

Если в строке должны быть использованы двойные кавычки, ее можно заключить в одинарные, и наоборот:

```
'To replay the movie press the "S" key'//Допустимые строки
"Rock'n'Roll"
```

При необходимости использования обоих типов кавычек внутри строки нужно использовать escape-последовательность, представляющую значение строки с помощью обратной наклонной черты, за которой следует сам символ или его код, например:

```
"To replay the movie press the \"S\"key"//Использование кавычек в строке
```

Кроме того, существует возможность использование escape-последовательностей Unicode, позволяющих поместить в строку символы Unicode, отсутствующие на клавиатуре. Для этого после символа обратной наклонной черты необходимо указать префикс `u` и четырехзначное шестнадцатеричное число, соответствующее кодовой позиции символа:

```
trace("\u00A9"); //Символ Copyright ©
trace("\u00AE"); //Символ Registered Trademark ®
```

Для того чтобы преобразовать в строку другой тип данных, можно воспользоваться функцией `String()`:

```
String(3.1415); //Преобразует в строку "3.1415"
var nVar:Number=24;
String(nVar); //Преобразует в строку "24"
```

Конкатенация и сравнение строк

Конкатенация представляет собой процедуру формирования строки из нескольких строк путем их соединения ("склеивания"). Конкатенация выполняется при помощи оператора +. При сложении строк они объединяются в новую строку. Для того чтобы использовать пробелы между словами в результирующей строке, можно включить их в состав строк, которые должны быть соединены, либо сложить эти строки со строками, состоящими только из пробелов:

```
trace("Hello"+ "Flash"); //Выводит HelloFlash  
var name:String="Dmitri";  
var lastName:String="Albert";  
var fullName:String=name+" "+lastName;  
trace(fullName); //Выводит Dmitri Albert
```

Следует иметь в виду, что при сложении строкового и числового значения в результате автоматического преобразования типов данных число будет преобразовано в строку, после чего будет выполнена конкатенация. При этом тип переменной не изменится, преобразование выполняется только с копией данных переменной:

```
var sVar:String = "PI=";  
var nVar:Number = 3.14;  
var mes = sVar+nVar; //Выводит PI=3.14  
trace(typeof(mes)); //Выводит string  
trace(typeof(nVar)); //Выводит number, переменная не претерпела изменений
```

Для сравнения строк используются те же операторы, что и для сравнения чисел. При сравнении строк последовательно сравниваются значения кодовых позиций входящих в них символов. Строки считаются равными, только если они состоят из одинакового набора символов (с учетом регистра), расположенных в одинаковом порядке. Следует иметь в виду, что в кодировке Unicode цифры предшествуют заглавным буквам, за которыми, в свою очередь, следуют строчные:

```
trace("0"<"9");//true  
trace("9"<"A");//true  
trace("A"<"Z");//true  
trace("Z"<"a");//true  
trace("a"<"z");//true  
trace("Flash MX">"Flash mX");//false, т.к. M < m  
trace("Flash"=="Flash");//true
```

Некоторые встроенные функции для работы со строками

Свойство `length` позволяет узнать количество символов, содержащихся в строке, включая пробелы. Доступ к данному свойству осуществляется при помощи оператора точки точно так же, как и к свойствам клипов и кнопок:

```
trace("Macromedia".length); //Выводит 10  
var sPhrase:String="Vita Brevis Ars Longa";  
trace(sPhrase.length); //Выводит 21
```

Каждый символ в строке имеет определенный индекс, который указывает его местоположение относительно начала строки. Символы в строке нумеруются, начиная с нуля. Таким образом, первый символ в строке всегда имеет индекс 0, второй — индекс 1 и т. д. Индекс последнего символа в строке равен значению свойства `length - 1`.

Функция `charAt()`

Функция `charAt()` возвращает символ, находящийся в строке по заданному индексу. Данная функция имеет следующий формат:

```
string.charAt(index)
```

Здесь:

- `string` — это строковый литерал или идентификатор;
- `index` — целое число или выражение, задающее индекс требуемого символа в строке.

```
trace("Flash".charAt(0)); //Выводит F — первый символ  
var sVar:String="Versatility";  
trace(sVar.charAt(0)); //Выводит V — первый символ  
trace(sVar.charAt(sVar.length-1)); //Выводит у — последний символ
```

Функция `indexOf()`

Функция `indexOf()` позволяет определить, входит ли в строку заданное сочетание символов, и возвращает индекс его первого вхождения. Если искомая подстрока не найдена, функция возвращает значение `-1`. Данная функция имеет следующий формат:

```
string.indexOf(substirng, startIndex)
```

Здесь:

- `string` — это строковый литерал или идентификатор, содержащий строку;
- `substirng` — это строковый литерал или идентификатор, содержащий подстроку, вхождение которой проверяется;
- `startIndex` — необязательный параметр, задающий индекс символа, с которого будет начат поиск.

При отсутствии *startIndex* поиск выполняется с начала строки.

```
var email:String="dmitri@avalon.ru";
trace(email.indexOf("@")); //Выводит 6, индекс 7-го по счету символа
```

Функция *lastIndexOf()*

Функция *lastIndexOf()* позволяет определить, входит ли в строку заданное сочетание символов, но в отличие от функции *indexOf()* возвращает индекс его последнего вхождения. Формат данной функции точно такой же, как и у функции *indexOf()*, однако различие заключается в том, что поиск выполняется справа налево. Таким образом, *startIndex* — это самый правый символ в строке, с которого будет осуществлен поиск.

```
var searchFor:String="tet";
trace("tet-a-tet".lastIndexOf(searchFor, 5)); /*Выводит 0, поскольку поиск начинается с символа — с индексом 5*/
```

Функция *substring()*

Функция *substring()* позволяет извлечь из строки последовательность символов, основываясь на начальном и конечном индексах. Данная функция имеет следующий формат:

```
string.substring(startIndex, endIndex)
```

Здесь:

- *string* — это строковый литерал или идентификатор, содержащий строку;
- *startIndex* — индекс первого символа извлекаемой подстроки;
- *endIndex* — необязательный параметр, задающий индекс символа, следующего за последним символом подстроки, т. е. индекс первого символа, расположенного после подстроки. Индексы могут быть заданы в виде целочисленных литералов или идентификаторов, либо как выражения. Если параметр *endIndex* опущен, то его значение автоматически устанавливается равным *length* – 1.

```
trace("Дмитрий Альберт".substring(8)); //Выводит Альберт
```

Следующий пример позволяет установить номер основной версии проигрывателя Flash Player, в котором осуществляется воспроизведение. Используемая здесь встроенная функция *getVersion()* возвращает информацию о платформе и версии проигрывателя, которая может иметь следующий вид: "WIN 7,0,14,0". При необходимости установить номер версии проигрывателя, имеющегося у пользователя, информация о платформе и о дополнительном номере, а также номере сборки проигрывателя может быть излишней, поэтому в данной ситуации удобно извлечь только номер основной версии:

```
trace(getVersion().substring(4,5)); //Возвращает 7
```

Метод *fromCharCode()*

Метод класса `String fromCharCode()` позволяет создать символ на основе значения его ASCII-кода. В отличие от всех предыдущих функций, данный метод является статическим, а это означает, что он вызывается непосредственно из объекта. Вызов данного метода имеет следующий формат:

```
String fromCharCode(code1, code2, code3,...)
```

Здесь `code1, code2, ..., codeN` — кодовые значения символов.

```
trace ("Freeway"+String.fromCharCode(153)); //Выводит Freeway™  
trace(String.fromCharCode(68, 105,109, 97)); //Выводит Dima
```

Функции *toUpperCase()* и *toLowerCase()*

Функции `toUpperCase()` и `toLowerCase()` позволяют перевести символы строки в верхний или нижний регистр соответственно. Если для символа не существует верхнего регистра, он остается без изменений. В качестве строки можно использовать литерал или идентификатор, содержащий строку:

```
var sMixed:String="ReNaIsSanCe";  
trace(sMixed.toUpperCase()); //Выводит RENAISSANCE  
trace(sMixed.toLowerCase()); //Выводит renaissance
```

Функция *eval()*

Функция `eval()` принимает в качестве аргумента строку и преобразует ее в идентификатор — например, имя переменной, экземпляра клипа, свойство и т. д. Если элемента с таким идентификатором не существует, функция возвращает значение `undefined`. Данная функция имеет достаточно мощные возможности, поскольку позволяет динамически генерировать идентификаторы. Следующий пример иллюстрирует применение функции `eval()` для получения значения свойства объекта. При этом имя объекта и название свойства передаются как строки:

```
var clip:String="mc";  
var prop:String="_x";  
var val:Number=eval(clip+"."+prop); //Эквивалентно var val=mc._x;  
trace(val); //Выводит текущее значение свойства _x объекта mc
```

Булев тип данных

Логические, или булевые данные используются для управления логикой выполнения сценария. Логический тип данных содержит только два допустимых значения: `true` и `false`. Эти значения являются специально зарезервированными словами. Их нельзя использовать в качестве имен идентификаторов.

Логические данные используются в условных предложениях для проверки выполнения определенных условий. Результат сравнения значений также

представляется в виде логических данных. Эти данные могут быть использованы для контроля за состоянием объекта.

В результате преобразования любого числового значения, отличного от нуля, в булев тип получается значение `true`. Ноль преобразуется в булево значение `false`.

При проверке выполнения нескольких условий используются логические операторы. Существуют три логических оператора: `||` (логическое ИЛИ), `&&` (логическое И) и `!` (логическое НЕ).

Оператор ИЛИ

Оператор ИЛИ (`||`) принимает два операнда. Если оба операнда являются булевыми выражениями (т. е. выдают логическое значение), то логическое ИЛИ возвращает значение `true`, если хотя бы один из его operandов имеет значение `true`. Логическое ИЛИ возвращает значение `false`, только если оба его operandы имеют значение `false`.

```
trace(false||true); //true
trace(2<3||5==6); //true
var x:Number=10;
trace(x>=11||x<=50); //true
trace(false||false); //false
```

Оператор И

Оператор И (`&&`) также принимает два операнда. Если оба операнда являются булевыми выражениями, то логическое И возвращает значение `true`, только если оба его operandы имеют значение `true`. В противном случае логическое И возвращает значение `false`.

```
trace(false&&true); //false
trace(2<3&&5==6); //false
var x:Number=10;
trace(x>=10||x<=50); //true
trace(true||true); //true
```

Оператор НЕ

Оператор НЕ (`!`) принимает только один operand и возвращает булево значение, противоположное значению этого operandса.

```
trace(!true); // false
trace(!false); // true
var i:Number=0;
trace(!i); // true, поскольку ноль преобразуется в false
```

Условные предложения

Обычно предложения в сценарии выполняются поочередно, сначала до конца, т. е. *линейно*. Условные предложения позволяют обеспечить нелинейное выполнение фрагментов сценария, когда определенный набор инструкций может быть выполнен только при выполнении заданного условия. Условное предложение начинается со слова *if* и имеет следующий формат:

```
if(condition) {
    statements;
}
```

Внутри фигурных скобок может быть заключен блок кода, состоящий из неограниченного числа предложений *statements*. Эти предложения выполняются только в случае, если результатом проверки условия *condition* является булевское значение *true*. В качестве условия обычно используется выражение, включающее операторы сравнения и возвращающее булевское значение. При необходимости проверки нескольких согласованных условий нужно использовать логические операторы И или ИЛИ.

Следующий сценарий кнопки при нажатии переведет воспроизведяющую головку к кадру с меткой "start" только при условии, что значение строковой переменной *pass* равно строке "myPassword":

```
on(press) {
    if(pass=="myPassword") {
        gotoAndPlay("start");
    }
}
```

Если в блоке условного предложения, ограниченном фигурными скобками, содержится только одно предложение, то фигурные скобки можно опустить. В этом случае данное предложение должно размещаться в одной строке с ключевым словом *if*. Предыдущий сценарий в более компактном виде можно записать так:

```
on(press) {
    if(pass=="myPassword") gotoAndPlay("start");
}
```

Приведенный ниже сценарий клипа, расположенного на основной монтажной линейке, превращает его в движущуюся мишень, заставляя перемещаться слева направо по прямой, причем когда клип выходит за пределы видимости, он автоматически начинает движение из левого верхнего угла рабочей области. При нажатии левой кнопки мыши в момент, когда курсор находится на клипе, он увеличивается в размерах и затем снова уменьшается. Для контроля условия выхода клипа за пределы сцены используются статические свойства *width* и *height* встроенного класса *Stage*, вызываемые непосред-

ственno из объектa, которые возвращают ширину и высоту сцены (с учeтом масштaбирования окна Flash Player). Предполагается, что точка регистрации клипа находится в его геометрическом центре. Переменные, используемые в сценарии клипа, определяются в сценарии кадра основной монтажной линейки:

```
//Сценарий кадра основной монтажной линейки
var xSpeed:Number=5; //Смещение по X
var ySpeed:Number=8; //Смещение по Y
var hitScale:Number=150; /*Увеличение размера объекта при попадании в
процентах*/
//Сценарий клипа-мишени
onClipEvent (enterFrame) {
    _x += _root.xSpeed; //Перемещение по X
    _y += _root.ySpeed; //Перемещение по Y
    if (_x-.5*_width>=Stage.width || _y-.5*_height>=Stage.height) {
        _x =-.5*_width;
        _y =-.5*_height;
    }
}
on(press){//Увеличение размера при попадании
    _xscale=_root.hitScale;
    _yscale=_root.hitScale;
    trace("Попадание");
}
on(release, rollOut, dragOut){//Возврат прежних размеров
    _xscale=100;
    _yscale=100;
}
```

Для того чтобы движение начиналось с начала только после выхода всего клипа за пределы видимости (а не только его точки регистрации), в условном выражении используются не координаты клипа, возвращаемые его свойствами `_x` и `_y`, а координаты его крайней точки слева (`_x-.5*_width`) и снизу (`_y-.5*_height`) соответственно.

Допускается использование нескольких последовательных условных предложений. При этом сценарии, находящиеся в блоках данных предложений, выполняются последовательно в случае, если проверка соответствующих условий возвращает логическое значение `true`. Если какое-то из условий возвращает `false`, сценарий данного условного предложения игнорируется.

Зачастую вместе с условным предложением используется предложение `else`, позволяющее выполнить альтернативный набор инструкций в случае невыполнения условия предложения `if`.

При этом используется следующая конструкция:

```
if(condition) {  
    statements;  
}  
else {  
    alternative statements;  
}
```

Интерпретатор обрабатывает такую конструкцию следующим образом. Если условие *condition* возвращает *true*, то выполняются инструкции условного предложения *statements*, а набор предложений *alternative statements* игнорируется. Если же условие *condition* возвращает *false*, то инструкции условного предложения *statements* игнорируются и будет выполнен набор инструкций *alternative statements*.

В качестве примера использования предложения *else* можно привести сценарий кнопки-флага. Кнопка-флаг позволяет контролировать несколько состояний объекта и выполняет различные манипуляции с объектом в зависимости от его текущего состояния. Листинг 20.1 содержит сценарий кнопки, останавливающей воспроизведение клипа *mc*, расположенного на основной монтажной линейке. Предполагается, что при воспроизведении фильма клип запускается автоматически, поэтому первое нажатие останавливает его воспроизведение, следующее запускает, и т. д. Для того чтобы знать, запущен клип или остановлен в момент нажатия кнопки, вводится логическая переменная *i*, которая принимает значение *true*, если анимация клипа остановлена или значение *false*, если клип воспроизводится. Такая переменная называется *флагом*, отсюда и название кнопки. Таким образом, при нажатии кнопки проверяется значение переменной *i* и в зависимости от принимаемого ею значения осуществляется либо запуск, либо остановка монтажной линейки клипа, после чего переменной присваивается соответствующее значение. При первом нажатии будет выполнен блок условного предложения *if*, останавливающий воспроизведение клипа, поскольку переменная *i* не была заранее объявлена и в результате автоматического создания принимает значение *undefined*, которое разрешается в булево значение *false*. Таким образом, выражение *!i* при первом нажатии разрешается в *true*.

Листинг 20.1. Кнопка-флаг

```
on(press) {  
if(!i){ /*При первом нажатии результат проверки true,  
_root.mc.stop();  
i=true;  
trace("Выполняется блок предложения if, клип остановлен, i='"+i);  
}  
}
```

```
else{
    _root.mc.play();
    i=false;
    trace("Выполняется блок предложения else, клип запущен, i='"+i+"');
}
}
```

Конструкция `if...else...` может быть еще более усложнена при помощи предложения `else if`, позволяющего выполнять один из нескольких блоков предложений, причем количество таких блоков практически не ограничено. Данное предложение имеет следующий формат:

```
if(condition 1){
    statements 1;
}
else if (condition 2){
    statements 2;
}
...
else {
    alternative statements;
}
```

Все условия проверяются последовательно сверху вниз. Если хотя бы одно из условий `condition n` выполняется, то будет выполнен блок соответствующего условного предложения `statements n`, при этом все остальные блоки предложений `statements` будут проигнорированы. Если не одно из условий `condition` не выполняется, тогда будет выполнен блок `alternative statements` предложения `else`. Если предложение `else` отсутствует, то при невыполнении всех условий ни один блок предложений не будет выполнен.

Предложения цикла

Циклы позволяют многократно выполнять определенный блок кода до тех пор, пока выполняется заданное условие. Язык ActionScript содержит несколько предложений, которые могут применяться для циклического выполнения определенного набора инструкций. Рассмотрим два из них: предложение `while` и предложение `for`.

Предложение `while`

Предложение `while` позволяет многократно выполнять набор подпредложений, до тех пор пока выполняется определенное условие, заданное в заголовке цикла.

Предложение имеет следующий формат:

```
while (condition) {  
    statements;  
}
```

Первая строка, содержащая ключевое слово `while`, называется *заголовком цикла*. *Condition* — это проверочное выражение, возвращающее логическое значение. Предложения *statements* будут многократно выполняться до тех пор, пока результатом проверки *condition* остается значение `true`. Блок предложений *statements*, заключенных в фигурные скобки, называется *телом цикла*. Следующий сценарий позволяет вывести значения десятичного кода для первых 255 символов кодировки Unicode (поскольку часть из них являются управляющими символами, при выводе результата они не отображаются):

```
var i=0;  
while(i<=255) {  
    trace("Символ "+String.fromCharCode(i)+" имеет код "+i);  
    i++;  
}
```

Здесь переменная *i* используется при формировании проверочного выражения, в качестве *счетчика цикла*, а также непосредственно в предложениях для формирования результата, выводимого в панель **Output**. При каждом выполнении тела цикла или *итерации* значение переменной *i* увеличивается на единицу. После того как ее значение становится равным 256, выполнение тела цикла прекращается. Если проверочное выражение цикла всегда возвращает значение `true`, то такой цикл называется *бесконечным* и выводит сообщение об ошибке. Так, если в приведенном примере опустить предложение `i++`, то будет получен бесконечный цикл, поскольку значение переменной *i* будет всегда оставаться меньше 255.

Предложение `for`

Предложение `for` выполняет ту же функцию, что и предложение `while`, однако использует отличный от него синтаксис и имеет следующий формат:

```
for(initialization; condition; next) {  
    statements;  
}
```

В заголовке цикла `for` указываются основные элементы цикла, разделенные точкой с запятой:

- *initialization* — выражение, задающее начальное значение счетчика цикла. Это выражение вычисляется перед выполнением тела цикла и

обычно содержит присваивание. При инициализации можно использовать предложение `var`:

- `condition` — проверочное выражение или условие, выполняемое перед каждой следующей итерацией цикла, возвращает логическое значение. Предложения, расположенные в теле цикла, выполняются до тех пор, пока проверочное выражение возвращает значение `true`;
- `next` — выражение, модифицирующее значение счетчика после каждой итерации цикла;
- `statements` — набор предложений, называемых телом цикла, которые циклически выполняются до тех пор, пока не перестанет выполняться условие `condition`.

Предположим, что на сцене имеется 10 произвольно размещенных экземпляров клипа одинакового размера, названных `mc1`, `mc2`, ..., `mc10`. Следующий сценарий кадра разместит эти экземпляры в одну линию по горизонтали, установив между ними равный промежуток.

```
var xSpace:Number=2;//Расстояние между клипами по X
for(i=1;i<=10;i++){
    this["mc"+i]._x = i*(mc1._width + xSpace);
}
```

В данном примере осуществляется динамическая генерация имен клипов, которым задаются значения координат по горизонтали. Как было сказано ранее, клип является свойством содержащей его монтажной линейки, поэтому обратиться к клипу можно при помощи оператора `[]`, используя строковое значение. В результате конкатенации требуемые имена "склеиваются" из постоянной части `"mc"` и номера клипа, для указания которого удобно использовать переменную счетчика цикла. Поскольку вначале было оговорено, что все клипы имеют равные размеры (по крайней мере, по горизонтали), то в качестве значения ширины каждого из клипов можно использовать значение, возвращаемое свойством `_width` любого из них, например `mc1`.

Применение вложенных циклов

Цикл `for` может содержать в своем теле другой цикл. Такой цикл называется **вложенным** (*nested*). При использовании вложенных циклов сначала осуществляется инициализация счетчика внешнего цикла, а затем вложенного. Модификация (изменение) счетчика внешнего цикла выполняется только после выполнения всех итераций вложенного цикла. Таким образом, для каждой итерации внешнего цикла выполняются все проходы или итерации внутреннего цикла. Например, если внешний цикл выполняет 10 итераций, а вложенный — 20, то блок предложений вложенного цикла будет выполнен 200 раз — по 20 итераций на каждое значение счетчика внешнего цикла.

Используя вложенный цикл, можно сформировать двумерную матрицу элементов, распределив их по строкам и по столбцам. Листинг 20.2 содержит сценарий кадра, использующий вложенный цикл, который создает дубликаты клипа mc, расположенного в этом же кадре, и равномерно размещает их в сетку по столбцам и по строкам.

Листинг 20.2. Размещение элементов в сетке

```
//Инициализация переменных
var nCol:Number = 5;//Количество столбцов
var nRow:Number = 4;//Количество рядов
var xSpace:Number = 2;//Расстояние по X
var ySpace:Number = 10;//Расстояние по Y
var initX:Number = 0;//Координата левого верхнего клипа по X
var initY:Number = 0;//Координата левого верхнего клипа по Y
//Формирование матрицы
for (i=1; i<=nRow; i++) {//Формирование строк
    for (j=1; j<=nCol; j++) {//Формирование столбцов
        mc.duplicateMovieClip("mc"+i+"_"+j,i*nCol+j);
        this["mc"+i+"_"+j]._x = (j - 1) * (this["mc"+i+"_"+j]._width + xSpace) +
        0.5 * mc._width + initX;
        this["mc"+i+"_"+j]._y = (i - 1) * (this["mc"+i+"_"+j]._height + ySpace) +
        0.5 * mc._height + initY;
    }
}
mc._visible = false;//Отключение видимости оригинала mc
```

Внешний цикл размещает элементы по строкам, а вложенный — по столбцам. Таким образом, сначала происходит полное заполнение строки, и только после этого — переход на новую строку и ее заполнение. Переменные, создаваемые в начале кода, задают параметры размещения клипов. В качестве родительского клипа может быть использован абсолютно любой клип, содержащий на своей монтажной линейке любые элементы. При помощи данного сценария можно создавать сетки (если клип имеет вид крестика) или регулярные узорные заполнения (рис. 20.1).

Для дублирования клипа используется встроенный метод `duplicateMovieClip()`, позволяющий динамически дублировать клипы, присваивая им новые имена, и размещать копии на заданных уровнях стека программно генерируемых клипов (*подробнее об этом методе говорится в гл. 21*). Первый параметр метода `duplicateMovieClip()` задает имя копии клипа, второй указывает уровень в стеке, на котором будет размещен данный клип. И тот, и другой па-

раметр должны быть уникальны, чтобы в результате дублирования не получить несколько клипов с одинаковыми именами и чтобы каждый клип был помещен на отдельный уровень, поскольку на одном уровне стека программно генерируемых клипов может находиться только один объект. Имена клипов генерируются при помощи выражения "`mc"+i+"_"+j`". Таким образом, имя клипа состоит из постоянного префикса `mc` и номера его строки `i` и столбца `j`, разделенных нижним подчеркиванием. Номер уровня рассчитывается при помощи выражения `i * nCol + j`, которое после каждой итерации возвращает значение, на единицу большее предыдущего.

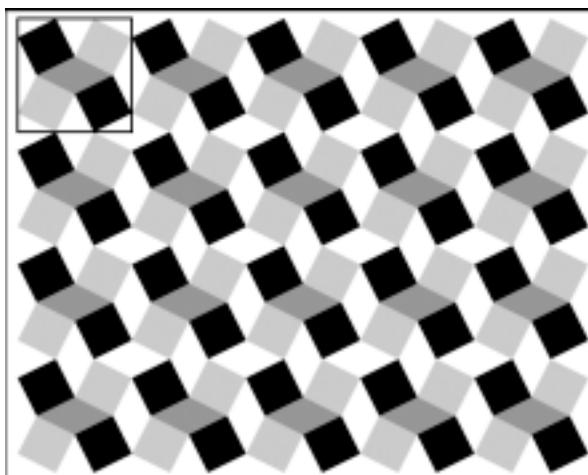


Рис. 20.1. Создание регулярного узорного заполнения путем дублирование образца

При размещении клипов по горизонтали координата каждого последующего клипа определяется как сумма двух слагаемых. Первое слагаемое представляет собой произведение порядкового номера клипа по строке (номер столбца) за вычетом единицы ($j - 1$) на его ширину (поскольку размеры всех дубликатов одинаковы). В этом произведении используется выражение $(j - 1)$, а не порядковый номер клипа `j`, поскольку первый клип в каждой строке (порядковый номер которого $j = 1$) должен находиться в начале отсчета (т. е. иметь абсциссу, равную нулю). Второе слагаемое корректирует положение клипа на основе заданного значения `initx` с учетом того, что точка регистрации клипа (относительно которой отмеряются его координаты) находится в его геометрическом центре, а не в крайней левой точке.

При размещении клипа по вертикали используется та же логика.

Последняя строка отключает отображение родительского клипа, поскольку он не вписывается в матрицу.

Использование трехкадрового цикла

Рассмотренные выше предложения циклов многократно выполняют определенный набор инструкций, однако в связи с тем, что во время выполнения сценария кадра экран не обновляется, эти предложения не могут выполнять действий, связанных с постепенным визуальным изменением тех или иных параметров объектов. Данные циклы выводят результат только после выполнения всех итераций.

Специфика Flash позволяет реализовать еще один вид цикла, основанного на том, что кадры монтажной линейки воспроизводятся последовательно и циклически. Этот цикл получил название *трехкадрового*. Трехкадровый цикл позволяет программным образом реализовывать визуальные изменения состояния объектов. В общем случае для создания трехкадрового цикла используются два слоя и три кадра. Структура монтажной линейки представлена на рис. 20.2.

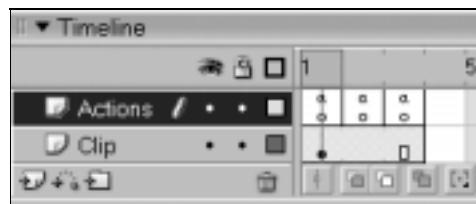


Рис. 20.2. Структура монтажной линейки при создании трехкадрового цикла

На слое **Actions** расположены три пустых ключевых кадра, предназначенные для размещения сценариев. Слой **Clip** содержит экземпляр клипа (или кнопки) с именем `mc`, который должен присутствовать на сцене в течение трех кадров. По аналогии с циклом `for`, трехкадровый цикл включает инициализацию, модификацию и условие, которые реализуются в сценариях соответствующих кадров. Следующий пример перемещает клип `mc` по диагонали, одновременно вращая его. При выходе клипа за пределы рабочей области он начинает движение сначала. Таким образом в данном случае реализуется бесконечный цикл. Бесконечный цикл монтажной линейки не вызывает ошибки в отличие от бесконечного цикла с использованием предложения цикла.

```
//Сценарий первого кадра: инициализация
mc._x=0;
mc._y=0;
mc._rotation=0;
var xMov:Number=10;//Смещение по X
```

```
var yMov:Number=15;//Смещение по Y  
var rot:Number=30;//Поворот  
  
//Сценарий второго кадра: модификация  
mc._x+=xMov;  
mc._y+=yMov;  
mc._rotation-=rot;  
  
//Сценарий третьего кадра:  
//Проверка условия выхода mc за пределы видимости  
if(mc._x>=Stage.width || mc._y>=Stage.height){  
    gotoAndPlay(1); //Инициализация в случае выхода  
}  
else gotoAndPlay(2); //Зацикливание
```

В качестве альтернативы трехкадровому циклу рекомендуется применять обработчик событий клипа `enterFrame`, позволяющий добиться лучшей централизации программного кода. Следующий сценарий позволяет реализовать ту же функциональность, что и сценарий трехкадрового цикла:

```
//Сценарий кадра: инициализация  
mc._x=0;  
mc._y=0;  
mc._rotation=0;  
//Сценарий клипа mc  
onClipEvent (enterFrame) {  
    _x += 10;  
    _y += 15;  
    _rotation -= 30;  
    if (_x>=Stage.width || _y>=Stage.height) {  
        _x = 0;  
        _y = 0;  
        _rotation = 0;  
    }  
}
```

Создание пользовательских функций

Наряду с использованием многочисленных встроенных функций, существует возможность создания пользовательских функций, предназначенных для выполнения различных задач. Пользовательские функции применяются для

выполнения однотипных процедур, которые реализуются с помощью одного и того же базового кода. Функция может принимать параметры или аргументы и возвращать определенное значение или выполнять иное действие. Функция фактически представляет собой "черный ящик". На входе функции передаются определенные данные в виде аргументов, на выходе функция выдает результат, полученный на основе значений входных параметров.

Объявление функции

Процесс создания функции называется ее *объвлением*. Объявление функции осуществляется при помощи предложения `function` и в общем случае имеет следующий формат:

```
function funcName (arg1, arg2,...argN) {  
    statements  
}
```

Ключевое слово `function` указывает интерпретатору, что далее последует объявление функции, а `funcName` — это имя создаваемой функции, при помощи которого данную функцию можно будет вызывать из любого места кода. В круглых скобках указываются аргументы функции `arg1, arg2,...argN`, т. е. данные, которые передаются функции для обработки. Если функция не содержит аргументов, т. е. не принимает параметров, то круглые скобки остаются пустыми (тем не менее, их наличие является *обязательным*). В фигурных скобках находятся предложения `statements`, выполняемые при вызове функции, которые составляют тело функции.

Следующий пример иллюстрирует создание простейшей функции, которая не принимает параметров и при вызове выводит в панель **Output** текстовое сообщение.

```
//Объявление функции greet  
function greet(){  
    trace("Welcome to Macromedia Flash MX 2004")  
}  
  
//Вызов функции greet  
greet(); //Выводит Welcome to Macromedia Flash MX 2004
```

Параметры передаются функции в виде списка идентификаторов, разделенных запятыми. При объявлении функции параметры представляют собой переменные, которые используются в предложениях тела функции. При вызове функции значения параметров задаются в виде литералов или выражений, возвращающих определенное значение. При объявлении функции ActionScript 2.0 позволяет задать тип значения, принимаемого каждым па-

параметром функции. Для этого после имени соответствующей переменной через двоеточие указывается ее тип. Контроль типов принимаемых функцией параметров позволяет избежать ошибок, связанных с тем, что при вызове функции ей передаются данные, тип которых не соответствует типу, заданному при объявлении. В этой ситуации Flash в среде тестирования выведет сообщение о несоответствии типов данных.

Следующий пример создает функцию, вычисляющую произведение своих аргументов, являющихся числами.

```
function multiply(x:Number, y:Number):Number{  
    x*y;  
}  
trace(multiply(5, 10)); //Выводит undefined
```

Приведенная функция формально является корректной, однако в таком виде она не имеет практического применения, поскольку вычисляет произведение, но не возвращает результат вычисления. Поэтому `trace()` возвращает значение `undefined`. Для того чтобы функция явно возвращала вычисленное значение, необходимо использовать предложение `return`, которое имеет следующий формат:

```
return expression
```

Предложение `return` возвращает результат вычисления выражения `expression` в качестве возвращаемого значения функции. Данное предложение приводит к прекращению выполнения функции и замене функции возвращаемым ею значением. В сочетании с контролем типов значений, принимаемых параметрами функции, можно также указать тип значения, возвращаемого функцией. Для этого при объявлении функции после закрывающей круглой скобки через двоеточие указывается тип возвращаемого значения. Если функция не возвращает значения, используется тип `Void`. Контроль возвращаемого функцией значения позволяет избежать ошибок, связанных с тем, что тип возвращаемого значения отличен от заданного. При несоответствии типа возвращаемого значения Flash выведет сообщение об ошибке. С учетом сказанного выше приведенный пример можно переписать следующим образом:

```
function multiply(x:Number, y:Number):Number{  
    return x*y;  
}  
trace(multiply(5, 10)) //Выводит 50
```

Параметры функции могут иметь различный тип. В следующем примере создается функция, позволяющая переводить рубли в доллары и наоборот. Функция принимает три параметра: `amount` — денежная сумма, которая

должна быть переведена, `rate` — курс, определяющий количество рублей за один доллар, `dir` — направление перевода. Последний параметр может принимать два значения (`true` — в рубли, `false` — в доллары).

```
function convert (amount:Number, rate:Number, dir:Boolean):Number{
    if(dir) return amount*rate;
    else return amount/rate;
}
trace(convert (1000, 29.7, true)); //Выводит 29700
}
```

Создание литерала функции

Язык ActionScript предоставляет возможность создания литерала функции. Эта возможность применяется при вызове метода обработчика событий (см. гл. 21), а также в ситуациях, когда существует необходимость использования функции там, где ожидается выражение. Создание литерала функции имеет следующий формат:

```
function (arg1, arg2,...argN){
    statements
};
```

При создании литерала функции ее имя не указывается, и таким образом функция перестает существовать сразу после окончания своей работы. После закрывающей скобки литерала функции ставится точка с запятой. Если литерал функции требуется использовать в другой части кода, его можно сохранить в переменной как объект типа `Function`.

В приведенном ниже сценарии создается литерал функции, запускающей основную монтажную линейку со следующего кадра по отношению к текущему кадру. Эта функция сохраняется в переменной `playNextFrame`:

```
var playNextFrame:Function=function(){
    _root.gotoAndPlay(_currentframe+1);
};
```

В отличие от встроенной функции `nextFrame()`, останавливающей воспроизведение после перехода к следующему кадру, приведенная функция продолжает воспроизведение со следующего кадра. Поскольку данная функция сохранена в переменной, ее можно использовать в дальнейшем. Вызов этой функции осуществляется следующим образом:

```
playNextFrame();
```

Использование локальных переменных

Функция обладает собственной, локальной областью видимости. В связи с этим внутри функции могут создаваться локальные переменные, т. е. переменные, доступ к которым имеют только предложения, находящиеся в теле функции. Локальные переменные существуют лишь на протяжении времени выполнения функции. Для создания локальных переменных используется уже знакомое предложение `var`. Любая переменная, созданная внутри функции при помощи данного предложения, является локальной и, значит, недоступной предложениям, находящимся за пределами функции. Внутри функции также можно создавать или изменять значение нелокальных (глобальных) переменных. В этом случае предложение `var` не используется. Если в теле функции встречается ссылка на переменную, интерпретатор в первую очередь ищет ее в локальной области видимости функции. Следующий пример иллюстрирует использование локальных и нелокальных переменных:

```
var x:Number=2;  
function myFunc () {  
    x=10; /*Изменяет значение нелокальной переменной x*/  
    var y:Number=3; /*Создание локальной переменной y*/  
    var x:String="Dima"; /*Создание локальной переменной x*/  
    return x;  
}  
trace(myFunc()); /*Выводит Dima - значение локальной переменной x*/  
trace(x); /*Выводит 10 - измененное значение нелокальной переменной x*/  
trace(y); /*Выводит undefined, т. к. локальная переменная у существует  
только внутри функции*/
```

Доступ к функции

После создания функция может быть использована любым сценарием фильма до тех пор, пока существует монтажная линейка, к которой эта функция прикреплена. Если функция определена на монтажной линейке клипа, то при удалении клипа со сцены данная функция также удаляется. Доступ к пользовательской функции из других сценариев фильма осуществляется так же, как и доступ к переменным при помощи точечного синтаксиса. Функция доступна непосредственно только из сценария кадра монтажной линейки, к которой она прикреплена, или из сценария кнопки, расположенной на этой монтажной линейке. Во всех остальных случаях при вызове пользовательской функции необходимо использовать адресацию. Так, например, если в кадре основной монтажной линейки определена

функция `myFunc()`, то из клипа `mc`, расположенного на основной монтажной линейке, эту функцию можно вызвать одним из следующих способов:

```
_root.myFunc(); //Абсолютная адресация  
_parent.myFunc(); //Относительная адресация
```

Функция, будучи объектом, может быть сохранена в переменной. Доступ к такой функции осуществляется по имени содержащей ее переменной. Это позволяет использовать более компактную форму ссылки на функцию:

```
gs=gotoAndStop;  
gs("Intro", "start"); //Переход к кадру с меткой start сцены Intro
```

Рекурсия

Рекурсивной функцией является функция, которая вызывает саму себя. Иными словами, рекурсия имеет место, когда в теле функции содержится ее вызов. Классический пример рекурсивной функции — вычисление факториала. Листинг 20.3 содержит функцию, вычисляющую факториал некоторого числа. Факториал числа 0 равен 1 — этот факт учитывается в условном предложении. Поскольку факториал вычисляется только для неотрицательных целых чисел, проводится проверка того, является ли аргумент функции допустимым значением. Для проверки дробности используется функция `indexOf()`, позволяющая определить наличие оператора десятичной точки. Если точка отсутствует, функция `indexOf()` возвращает значение `-1`.

Листинг 20.3. Использование рекурсивной функции для вычисления факториала числа

```
function factorial(n:Number) {  
    /*Если аргумент равен нулю, факториал равен 1  
    */  
    if(n==0) {  
        return 1;  
    }  
    /*Если аргумент дробное (содержит десятичную точку) или  
    отрицательное число, факториал не вычисляется*/  
    else if (String(n).indexOf(".") != -1 || n<0) {  
        return undefined;  
    }  
    /*Если аргумент положительное целое число, не меньше 1,  
    осуществляется вычисление с использованием рекурсии*/  
    else if (n>=1) {  
        return n*factorial(n-1);  
    }  
    trace(factorial(5)); //Выводит 120
```

Массивы

Массив представляет собой упорядоченную структуру данных, состоящую из индексированных элементов. Массивы используются для хранения и обработки группы значений данных. Значения, хранящиеся в массиве, называются элементами массива. Каждый элемент массива имеет свой уникальный цифровой индекс, при помощи которого можно осуществлять обращение к этому элементу. Элементами массива могут быть данные любого типа, например, числа или строки; кроме того, массив может содержать данные разных типов. Массив может быть использован для хранения координат набора точек, списка клиентов фирмы, набора ссылок на клипы основной монтажной линейки и т. п. Применение массивов чрезвычайно удобно при работе с группами однотипных взаимосвязанных данных.

Массив является контейнером, который содержит набор элементов. Каждый элемент массива имеет свой уникальный цифровой индекс, при помощи которого можно осуществлять обращение к этому элементу. Нумерация элементов массива начинается с нуля, т. е. первый элемент всегда имеет индекс 0, второй — 1, третий — 2 и т. д. Используя индекс элемента, можно извлекать его значение, присваивать значение данному элементу либо выполнять с ним определенные действия. Так, например, если массив содержит ссылки на клипы, то, обращаясь к его элементам, можно изменять местоположение или другие параметры соответствующих клипов. Это дает мощные средства разработки интерактивности или создания программной анимации.

Создание массива

Существуют два способа создания массива: при помощи литерала массива или с использованием функции конструктора массива `Array()`.

Литерал массива

Литерал массива представляет собой буквальное описание массива и имеет следующий формат:

```
[e11, e12, ..., e1N]
```

Квадратные скобки указывают на то, что это массив, а `e11`, `e12`, ..., `e1N` — это список элементов массива, которые могут быть представлены в виде литералов данных или выражений. Ниже представлены различные варианты литералов массивов:

```
[1, 2, 3, 4, 5] //Массив из пяти числовых элементов
```

```
["Dmitri", "Elena", "Mark"] //Массив из трех строковых элементов
```

```
var e11:Number=1;
```

```
var el2:Number=2;  
var el3:Number=3;  
[el1, el2+el1, el3+el1+el2]/*Выражения с использованием переменных в  
качестве элементов массива*/
```

Как и любой объект, массив может быть присвоен переменной, что позволяет обращаться к нему в дальнейшем, используя имя этой переменной:

```
var myArray:Array=["Dmitri", "Elena", "Mark"]/*Сохранение массива в  
переменной*/
```

Конструктор массива

Другой способ создания массива предполагает использование оператора `new`

и конструктора `Array()`. При этом используется следующий формат записи:

```
new Array(parameters);
```

Результат использования конструктора `Array()` зависит от передаваемых ему параметров `parameters`, которые указываются в круглых скобках. Если параметры отсутствуют, то конструктор создает пустой массив (не содержащий элементов). В качестве параметров может быть использована длина массива (т. е. количество его элементов), либо непосредственное перечисление элементов. При задании длины массива в качестве параметра передается одно числовое значение. Элементы массива перечисляются через запятую. При создании массив присваивается переменной, для того чтобы к его элементам можно было обращаться в дальнейшем.

```
var myArray1:Array=new Array();/*Присваивание переменной пустого  
массива*/
```

```
var myArray2:Array=new Array(25);/*Присваивание переменной массива из  
25 пустых элементов*/
```

```
var myArray3:Array=new Array("Dmitri", "Elena", "Mark");/*Присваивание  
переменной массива, содержащего три элемента*/
```

При создании массива заданной длины его элементы содержат значение `undefined` до тех пор, пока им не будет присвоено иное значение.

Обращение к элементам и заполнение массива

После создания и сохранения массива в переменной, доступ к его элементам осуществляется при помощи квадратных скобок, в которых указывается индекс элемента. Индекс может быть задан в виде любого выражения, возвращающего числовое значение. Это позволяет динамически генерировать индекс при обращении к требуемому элементу. Следует иметь в виду, что нумерация элементов начинается с нуля, а не с единицы.

```
var myArray:Array=new Array("Dmitri", "Elena", "Mark");  
trace(myArray[1]);//Выводит Elena
```

```
trace(myArray[1+1]); //Выводит Mark  
var x=0;  
trace(myArray[x]); //Выводит Dmitri
```

Присваивание значения элементу массива осуществляется так же, как и присваивание значения переменной. Слева от оператора присваивания находится элемент массива, справа — присваиваемое ему значение в виде литерала или выражения. Если значение присваивается уже существующему элементу массива, его прежнее значение будет переопределено, в противном случае будет создан новый элемент с заданным индексом. Продолжая предыдущий пример, добавим в массив, сохраненный в переменной myArray, три новых элемента:

```
myArray[3] = "Oleg"; /*Создан четвертый элемент со значением Oleg*/  
myArray[4] = myArray[0] + " " + myArray[1]; /*Создан пятый элемент Dmitri & Elena*/  
myArray[19] = 3.1415926; /*Создан 20-й элемент с числовым значением  
3.1415926*/
```

Обратите внимание, что в приведенном коде после пятого элемента (с индексом 4) сразу создается 20-й элемент (с индексом 19). Это абсолютно законная операция, в результате которой индексы массива с пятого по 18-й остаются незанятыми. При попытке извлечь элемент массива, индекс которого находится в этом диапазоне, будет получено значение `undefined`. Поскольку различные элементы массива могут содержать данные различных типов, 20-й элемент массива, в отличие от остальных, содержит число.

При передаче массива в качестве аргумента функции `trace()` в панель **Output** будет выведен перечень всех элементов массива, разделенных запятыми и расположенных в строке в соответствии с их порядковым номером (индексом).

```
trace([1, 2, 3, 4, 5]); //Выводит 1, 2, 3, 4, 5  
var myArray:Array=[1, 2, 3, 4, 5];  
trace (myArray); //Выводит 1, 2, 3, 4, 5
```

При работе с массивами часто используются циклы, позволяющие последовательно перебрать все элементы массива. Так, в следующем сценарии (листинг 20.4) создается пустой массив, заполняемый в цикле прописными буквами латинского алфавита (десятичный код которых лежит в диапазоне от 65 (A) до 90 (Z)). Все элементы массива последовательно выводятся в панель **Output**.

Листинг 20.4. Заполнение массива в цикле

```
var alphabeth:Array=new Array();  
for (i=0;i<26;i++){
```

```

alphabeth[i]=String.fromCharCode(65+i);
trace(alphabeth[i]);
}

```

Некоторые методы работы с массивами

Свойство *length*

Для того чтобы узнать, сколько элементов содержится в массиве (включая пустые элементы), можно применить встроенное свойство *length*. Значение, возвращаемое свойством *length*, всегда на единицу больше индекса последнего элемента массива. Свойство *length* удобно применять при переборе элементов массива. Следующий сценарий обнуляет значение всех элементов массива *score*:

```

for (k=0;k<score.length;k++) {
    score[k]=0;
}

```

При помощи свойства *length* можно изменять размер массива, добавляя новые пустые элементы или удаляя существующие, отсекая элементы, расположенные в конце массива:

```

var myArray:Array=[1, 2, 3, 4, 5];
myArray.length=6;
trace (myArray); //Выводит 1,2,3,4,5,undefined
myArray.length=3;
trace (myArray); //Выводит 1,2,3

```

Метод *push()*

Метод *push()* добавляет новые элементы в конец массива и возвращает измененную длину массива. Для вызова данного метода используется следующий формат:

```
array.push(value1, value2, ..., valueN)
```

В качестве параметров *value1*, *value2*, ..., *valueN* методу передаются любые выражения, значения которых присваиваются добавляемым элементам. Если при вызове данного метода параметры отсутствуют, в конец массива будет добавлен один пустой элемент.

```

var myArray:Array=[1, 2, 3, 4, 5];
var newItem1:Number=6;
var newItem2:Number=7;
trace(myArray.push(newItem1,newItem1+newItem1,8)) /*Выводит 8 - новый
размер массива*/
trace (myArray) //Выводит 1,2,3,4,5,6,12,8

```

Метод *unshift()*

Метод *unshift()* действует так же, как и метод *push()*, но добавляет новые элементы в начало массива. В результате индексы существующих элементов увеличиваются, элементы как бы продвигаются к концу массива. Формат вызова данного метода аналогичен формату вызова метода *push()*:

```
array.unshift (value1, value2, ..., valueN)
```

Если в предыдущем примере в предпоследнем предложении заменить *push()* на *unshift()*, получим:

```
trace(myArray.unshift(newItem1,newItem1+newItem1,8)) /*Выводит 8, новый  
размер массива*/  
trace (myArray) //Выводит 6,12,8,1,2,3,4,5
```

Метод *pop()*

Метод *pop()* удаляет последний элемент в массиве и возвращает его значение. Данный метод не имеет параметров и вызывается следующим образом:

```
array.pop()
```

Следующий пример иллюстрирует использование метода *pop()*:

```
var favouriteMusic:Array=["jazz", "classical", "rock", "pop"];  
trace(favouriteMusic.pop()); //Выводит pop  
trace(favouriteMusic); //Выводит jazz,classical,rock
```

Метод *shift()*

Метод *shift()* удаляет элемент из начала массива и возвращает его значение. В результате индексы всех остальных элементов уменьшаются на единицу, т. е. элементы как бы смешаются в начало массива. Как и метод *pop()*, данный метод не принимает параметров и имеет аналогичный формат:

```
array.shift()
```

Следующий пример иллюстрирует использование метода *shift()*:

```
var favouriteMusic:Array=[ "pop", "jazz", "classical", "rock"];  
trace(favouriteMusic.shift()); //Выводит pop  
trace(favouriteMusic); //Выводит jazz,classical,rock
```

Метод *join()*

Метод *join()* преобразует элементы массива в строки, конкатенирует ("склеивает") их и возвращает в виде одной строки. В этой строке строковые значения элементов разделяются заданным символом (или набором символов). Метод *join()* использует следующий формат:

```
array.join(separator)
```

Здесь *separator* — это строка, используемая в качестве разделителя элементов. Если данный параметр опущен при вызове метода `join()`, то в качестве разделителя будет использована запятая.

```
var favouriteMusic:Array=["jazz", "classical", "rock"];
trace(favouriteMusic.join()); //Выводит jazz;classical;rock
trace(favouriteMusic.join(" ")); //Выводит jazz classical rock
trace(favouriteMusic.join()); //Выводит jazz,classical,rock
```

Следует иметь в виду, что метод `join()` не изменяет элементы массива, а просто объединяет их и возвращает в виде строки.

При использовании данного метода с многомерными массивами (т. е. массивами, элементами которых также являются массивы), элементы вложенных массивов всегда разделяются запятыми вне зависимости от заданного разделителя.

Многомерные массивы

Многомерный массив — это массив, элементы которого, в свою очередь, также являются массивами. Такие массивы иногда называются вложенными (nested arrays). Рассмотрим пример простейшего двумерного массива:

```
var pers1:Array=["Steve",57]
var pers2:Array=["Eileen",55]
var pers3:Array=["Heather",32]
var pers4:Array=["Sean",29];
var family:Array=[pers1,pers2,pers3,pers4];
```

Массив `family` содержит четыре элемента, каждый из которых представляет собой массив, содержащий данные об имени и возрасте члена семьи. Таким образом, элементы массива `family` условно можно представить в виде матрицы размером 4×2 , первая строка которой содержит имена членов семьи, а вторая — их возраст.

Steve	Eileen	Heather	Sean
57	55	32	29

Для обращения к элементу двумерного массива необходимо использовать два индекса: первый указывает порядковый номер элемента внешнего массива, а второй — порядковый номер элемента вложенного массива (нумерация элементов начинается с нуля!):

```
//Выводит Имя: Steve; Возраст: 57
trace("Имя: "+family[0][0]+"; Возраст: "+family[0][1]);
```

Для того чтобы вывести отчет обо всех членах семьи, необходимо выполнить последовательный перебор всех элементов массива по столбцам, т. е. по элементам вложенных массивов. Для этого воспользуемся циклом `for`:

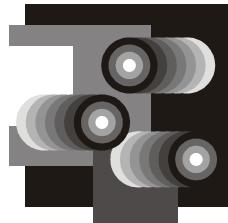
```
for(i=0;i<family.length;i++) {  
    trace("Имя: "+family[i][0]+"; Возраст: "+family[i][1]);  
}
```

В окно **Output** будет выведен следующий результат:

```
Имя: Steve; Возраст: 57  
Имя: Eileen; Возраст: 55  
Имя: Heather; Возраст: 32  
Имя: Sean; Возраст: 29
```

Двумерные массивы удобно использовать для хранения координат точек. В следующем примере создается двумерный массив `coords`, который в цикле заполняется координатами точек, размещенных случайным образом в пределах окна фильма. Для выбора координат случайным образом применяется метод `random()` объекта `Math`. Подробнее о его использовании говорится в следующей главе. Здесь массив `coords` содержит два массива: `xCor` и `yCor`. Элементы массива `xCor` содержат координаты точек по горизонтали, а элементы `yCor` соответствующие им координаты по вертикали:

```
var maxPoints:Number=50;//Число точек  
var xCor:Array=new Array();  
var yCor:Array=new Array();  
var coords:Array=new Array(xCor, yCor);  
for (i=0;i<maxPoints;i++){  
    coords[0][i]=Math.round(Math.random()*Stage.width);  
    coords[1][i]=Math.round(Math.random()*Stage.height);  
    trace("x:"+coords[0][i]+" y:"+coords[1][i]);  
}
```



Глава 21

Применение сценариев

Где оказывался бессильным ум, там часто помогало время.

Сенека Луций Анней Младший

Использование методов обработчиков событий клипов и кнопок

Для прикрепления сценария к экземпляру клипа или кнопки используются обработчики событий (см. гл. 19). Однако размещение различных фрагментов программного кода в разных частях фильма приводит к его рассредоточению и децентрализации. Зачастую бывает значительно удобнее сосредоточить программный код в одном месте, т. е. централизовать его. Для этих целей ActionScript предлагает использование методов обработчиков событий.

Метод обработчика событий вызывается объектом (например, экземпляром клипа или кнопки) с использованием точечного синтаксиса, однако в отличие от других методов, вызов метода обработчика событий осуществляется не непосредственно, а только при наступлении соответствующего события. Для того чтобы при наступлении определенного события вызов метода обработчика инициировал определенные действия, в качестве значения ему должна быть присвоена функция. Данная функция будет выполняться всякий раз при наступлении соответствующего события. В общем случае формат использования метода обработчика событий имеет следующий вид:

```
object.eventMethod = function () {  
    statements  
};
```

Здесь:

- *object* — это имя объекта (экземпляра), вызывающего метод обработчика;
- *eventMethod* — имя метода обработчика;
- *statements* — набор инструкций, выполняемых в результате наступления соответствующего события.

При присваивании методу обработчика функции может быть использовано ее непосредственное описание, т. е. литерал, либо указана ссылка на встро-

енную функцию или функцию, определенную в другой части сценария. В последнем случае важно иметь в виду, что в качестве присваиваемого значения используется не возвращаемое функцией значение, а именно ссылка на нее, т. е. круглые скобки опускаются. Листинг 21.1 иллюстрирует оба варианта присваивания функции и содержит сценарий кадра, который запускает монтажную линейку клипа mc при наведении на него курсора и останавливает ее при уходе курсора за пределы клипа (в первый момент анимация клипа остановлена).

Листинг 21.1. Управление воспроизведением клипа с монтажной линейки

```
mc.stop(); //Остановка анимации клипа  
mc.onRollOver = function() { //Присваивание литерала функции  
this.play(); //Запуск монтажной линейки mc  
};  
mc.onRollOut=stop; //Присваивание ссылки настроенную функцию stop()
```

Следует иметь в виду, что областью видимости предложений, заключенных в фигурные скобки литерала функции, присваиваемого методу обработчика события, является монтажная линейка, содержащая объявление этой функции, а не объект, вызывающий метод. Для того чтобы в теле функции сослаться на сам объект, можно использовать ключевое слово this (как это сделано в листинге 21.1) или непосредственно указать имя целевого объекта. Обработчики событий клипов и кнопок, используемые в качестве методов, позволяют перехватывать события, описанные в гл. 19. Их названия начинаются с префикса on и совпадают с названиями обрабатываемых ими событий. Перечень обработчиков событий клипов и кнопок, используемых в качестве методов, представлен в табл. 21.1.

Таблица 21.1. Обработчики событий клипов и кнопок

Обработчик событий	Применение	
	К кнопкам	К клипами
onPress	+	+
onRelease	+	+
onRollOver	+	+
onRollOut	+	+
onDragOut	+	+
onReleaseOutside	+	+
onDragOver	+	+

Таблица 21.1 (окончание)

Обработчик событий	Применение	
	К кнопкам	К клипами
onEnterFrame	—	+
onMouseMove	—	+
onKeyDown	+	+
onKeyUp	+	+
onMouseDown	—	+
onMouseUp	—	+
onLoad	—	+
onUnload	—	+
onData	—	+
onSetFocus ()	+	+
onKillFocus (newFocus)	+	+

В приведенной таблице имеются только два новых обработчика: `onSetFocus (oldFocus)` и `onKillFocus (newFocus)`. Обработчик `onSetFocus (oldFocus)` вызывается, когда объекту, к которому он прикреплен, передается фокус. Фокус может быть получен, например, при нажатии клавиши `<Tab>`, перемещающей фокус ввода из текстового блока на кнопку или клип. При вызове обработчика присвоенной ему функции передается параметр `oldFocus`, указывающий на предыдущий объект, потерявший фокус.

Обработчик `onKillFocus (newFocus)` вызывается, когда фокус переходит с объекта, к которому прикреплен данный обработчик, на другой объект. Это может произойти при щелчке по текстовому полю ввода (Input text). При вызове обработчика присвоенной ему функции передается параметр `newFocus`, указывающий на следующий объект, которому передается фокус.

Следующий сценарий кадра выводит в панель **Output** сообщение об объекте, получившем фокус, в случае, если он переходит на кнопку с именем `but` и в случае, когда кнопка `but` теряет фокус. Для оценки результата можно создать несколько блоков пользовательского текста, снабдив их произвольными именами экземпляров (Instance Name), и кнопку, экземпляру которой нужно присвоить имя `but`. После этого в среде тестирования при помощи клавиши `<Tab>` можно переместить фокус на кнопку, а щелкнув на текстовом поле, передать фокус ему.

```
but.onSetFocus=function(obj) {
    trace("Фокус перешел с "+obj+" на кнопку");
};

but.onKillFocus=function(obj) {
    trace("Фокус перешел с "+obj+" на текстовое поле");
};
```

```
but.onKillFocus=function(obj){  
    trace("Фокус перешел с кнопки на "+obj);  
};
```

Следует отметить, что кроме методов обработчиков событий, определенных для классов MovieClip и Button, также существует ряд обработчиков, определенных для других встроенных классов ActionScript (Stage, TextField и др.). Перечень всех методов обработки событий для соответствующих классов содержится в разделе **Action Toolbox** панели **Actions**.

Понятие об объектах *Listeners*

Любое событие ActionScript генерируется объектами определенных классов. Ряд классов содержит определение методов, позволяющих обнаруживать различные события. Так, например, для класса MovieClip определен метод onEnterFrame, для класса Stage — метод onResize, для класса TextField — метод onChanged и т. д. Некоторые объекты имеют общие методы обработки событий, например, и в классе MovieClip, и в классе Button определен метод onPress. Однако многие события могут быть обнаружены только объектами определенного класса. Так, например, событие onMouseWheel генерируется только объектом Mouse при вращении колеса прокрутки мыши.

ActionScript предоставляет возможность передачи сообщения о событии, сгенерированном одним объектом, называемым *broadcaster* (источник), другому объекту, называемому *listener* (приемник). В результате этого при обнаружении события, сгенерированного объектом-источником (*broadcaster*), объект-приемник (*listener*) может выполнять определенные действия. Для того чтобы объект-приемник получил возможность принятия сообщений о событиях, генерируемых объектом-источником, его необходимо "зарегистрировать" на получение соответствующих сообщений от соответствующего объекта. Для этого выполняются следующие действия. Вначале должен быть создан объект *listener*, который будет принимать сообщение о событии. После этого свойству данного объекта, название которого должно совпадать с событием, генерируемым объектом-источником, присваивается функция, которая будет выполняться всякий раз при получении сообщения об обнаружении события. Далее необходимо воспользоваться встроенным методом *addListener()* объекта-источника и передать ему в качестве аргумента имя объекта-приемника. Эта процедура имеет следующий формат:

```
listenerObject.eventName = function(){  
    statements  
};  
broadcastObject.addListener(listenerObject);
```

Здесь:

- *listenerObject* — имя объекта-приемника (*listener*);
- *eventName* — название события, генерируемого источником;
- *broadcastObject* — имя объекта-источника, генерирующего событие.

Следует иметь в виду, что можно создать несколько приемников (*listeners*), получающих сообщения о событиях от одного источника, и можно создать один объект-приемник, получающий сообщения от нескольких разных источников.

Листинг 21.2 содержит сценарий, который изменяет размеры клипа *mc* пропорционально изменению размеров окна автономного проигрывателя (при этом остальные объекты сцены не масштабируются). В качестве приемника здесь используется клип *mc*, а источником выступает объект *Stage*, генерирующий событие *onResize* при изменении размеров окна проигрывателя. Первая строчка сценария задает режим масштабирования фильма при помощи встроенного свойства *scaleMode* объекта *Stage*.

Листинг 21.2. Изменение размеров клипа пропорционально изменению окна проигрывателя

```
Stage.scaleMode = "noScale"/*Отключение масштабирования фильма при изменении размеров окна проигрывателя*/  
//Первоначальное соотношение размеров mc и окна  
var xRatio:Number=mc._width/Stage.width;  
var yRatio:Number=mc._height/Stage.height;  
//Присваивание функции приемнику mc  
mc.onResize=function(){  
    this._width=xRatio*Stage.width;  
    this._height=yRatio*Stage.height;  
}  
//Регистрация mc на прием сообщений от Stage  
Stage.addListener(mc);
```

Для того чтобы отключить передачу сообщений об обнаружении события, необходимо удалить объект *listener* (при этом он, естественно, продолжает оставаться на сцене) при помощи метода *removeListener()*, имеющего следующий формат:

```
broadcastObject.addListener(listenerObject);
```

В результате этого действия объект-приемник *listenerObject* перестает реагировать на событие, генерируемое объектом-источником *broadcast-Object*.

Использование динамического и пользовательского текста

Применение динамического (Dynamic) текста позволяет программным образом выводить текстовую информацию в процессе воспроизведения фильма. Использование редактируемого или пользовательского (Input) текста обеспечивает возможность ввода текстовой информации пользователем и ее программной обработки непосредственно в процессе воспроизведения фильма. Динамический или пользовательский текстовый блок может быть создан либо вручную в рабочей среде, либо динамически при помощи ActionScript, однако обработка его содержимого осуществляется исключительно программным образом.

Создание и параметры форматирования динамического и пользовательского текстовых блоков в рабочей среде описываются в гл. 8.

Данные, помещенные в динамический или пользовательский текстовый блок, имеют строковый тип. Для того чтобы получить возможность обращаться к текстовому полю из любого сценария фильма, каждому динамическому или пользовательскому текстовому блоку можно сопоставить уникальную переменную, значением которой является строка, представляющая содержимое этого текстового блока. Для сопоставления переменной текстовому блоку в рабочей среде необходимо выделить соответствующий блок на сцене и задать имя переменной в поле **Var** Инспектора свойств. Для того чтобы программным образом вывести сообщение в динамическое или пользовательское поле, достаточно присвоить сопоставленной ему переменной требуемое значение. Когда пользователь вводит текст в редактируемый текстовый блок, введенная строка автоматически присваивается в виде значения сопоставленной этому блоку переменной. Переменная, соответствующая текстовому полю, содержит значение `undefined` до тех пор, пока ей не будет присвоено иное значение.

Для демонстрации принципа использования динамического и пользовательского текста можно создать на сцене два текстовых блока и одному из них назначить тип **Input Text**, другому — **Dynamic Text**. После этого редактируемому текстовому блоку нужно сопоставить переменную `name`, а динамическому текстовому блоку переменную `greeting` (задав эти имена в поле **Var** Инспектора свойств, для каждого текстового блока соответственно). Для того чтобы выделить текстовое поле черной рамкой и установить белый фон, нужно воспользоваться кнопкой **Show border around text**. Затем на сцену можно поместить кнопку, назначив ей следующий сценарий:

```
on(press) {  
    greeting = "Hello " + name + "!";  
}
```

После выполнения всех указанных действий можно опубликовать фильм или перейти в среду тестирования. Теперь, если в поле `name` ввести свое имя, например `Dmitri`, то при нажатии кнопки в поле `greeting` будет выведено приветствие `Hello Dmitri!`.

Важно отметить, что редактируемое поле может быть использовано не только для ввода данных пользователем, но и для программного вывода информации. Для того чтобы текст отображался и мог быть введен в текстовое поле, необходимо встроить очертания тех символов, которые используются для отображения требуемой информации. Очертания символов можно не встраивать, если имеется уверенность в том, что используемый шрифт установлен у пользователя (*информация о встраивании шрифтов содержится в гл. 8*). Следует иметь в виду, что если размер текстового блока меньше, чем область, занимаемая его содержимым, то часть текста будет срезаться.

Динамический текст может быть использован для вывода различных сведений, связанных с ходом загрузки, процессом воспроизведения фильма или действиями, выполняемыми пользователем. Так, например, следующий сценарий кадра выводит в динамическое поле, которому сопоставлена переменная `coords`, информацию о текущих координатах курсора мыши:

```
_root.onMouseMove=function () {  
    coords="X: "+Math.round(_xmouse)+"      Y: "+Math.round(_ymouse);  
}
```

Редактируемый текст применяется в ситуациях, предполагающих непосредственный ввод пользователем информации, используемой при воспроизведении фильма. Следующий сценарий клипа при нажатии на нем поместит клип в точку, заданную при помощи редактируемых текстовых полей, которым сопоставлены переменные `x` и `y`.

```
on(press) {  
    _x=_root.x;  
    _y=_root.y;  
}
```

В листинге 21.3 приведен сценарий простого калькулятора, позволяющего выполнять арифметические действия с двумя числами (операндами). Для его реализации необходимо создать два пользовательских текстовых блока, в которые будут вводиться числа, сопоставив им переменные `op1` и `op2` соответственно. Поскольку эти поля предназначены для ввода числовых значений, нужно встроить для них только очертания символов числительных (Numerals) и оператора десятичной точки. Затем нужно добавить динамический текстовый блок для вывода результата, сопоставив ему переменную `result`. После этого на сцену нужно поместить пять кнопок, четыре из которых будут выполнять соответствующие операции с введенными числами и выводить результат в поле `result`, а пятая кнопка будет использована для очистки полей. Экземплярам кнопок необходимо присвоить следующие

имена: plusBut — сложение, minusBut — вычитание, multiplyBut — умножение, divideBut — деление, clearBut — сброс.

Листинг 21.3. Калькулятор

```
plusBut.onPress=function(){//Сложение
    result=Number(op1)+Number(op2);
}
minusBut.onPress=function(){//Вычитание
    result=op1-op2;
}
multiplyBut.onPress=function(){//Умножение
    result=op1*op2;
}
divideBut.onPress=function(){//Деление
    if(op2==0)result="ОШИБКА!!!"; //Исключение деления на 0
    else result=op1/op2;
}
clearBut.onPress=function(){//Сброс
    op1=""; //Присваивание пустой строки
    op2="";
    result="";
}
```

При выполнении сложения значения переменных op1 и op2 должны быть явно преобразованы из строкового в числовой тип с помощью функции `Number()`. В противном случае вместо математического сложения будет выполняться операция конкатенации $2 + 2 = 22$. Во всех остальных случаях результат вычислений будет корректным, поскольку интерпретатор автоматически выполнит преобразование значений op1 и op2 в числовой тип.

Классы *TextField* и *TextFormat*

Любой динамический или пользовательский текстовый блок, используемый в фильме, является экземпляром встроенного класса `TextField`. Экземпляр текстового блока может быть включен в фильм в рабочей среде вручную или создан программным образом при помощи метода `createTextField()`. Имя экземпляра текстового блока, созданного в рабочей среде, задается в поле **Instance Name** Инспектора свойств и используется для обращения к текстовому блоку из сценариев ActionScript. Важно понимать разницу между именем экземпляра текстового блока (**Instance Name**) и переменной, сопостав-

ленной текстовому блоку (**Var**). Имя экземпляра используется для обращения к его свойствам, вызова методов и применения обработчиков событий. Переменная же представляет собой ссылку на содержимое текстового блока. ActionScript предоставляет широкий набор возможностей манипулирования динамическими и пользовательскими текстовыми блоками в качестве экземпляров класса `TextField`, включая их создание, форматирование, обработку содержимого и удаление.

Основные свойства класса `TextField`

Встроенные свойства класса `TextField` позволяют программным образом управлять такими параметрами текстовых блоков, как прозрачность, положение, размер, цвет фона и рамки и т. д. Класс `TextField` предоставляет многие свойства, которые также имеются у клипов и кнопок. К их числу относятся: `_x`, `_y`, `_width`, `_height`, `_xscale`, `_yscale`, `_alpha`, `_rotation`, `_visible`, `_name`, `_parent`, `_url`, `_target`. *Описание этих свойств можно найти в разд. "Встроенные свойства клипов и кнопок" гл. 19.*

В табл. 21.2 содержится перечень основных свойств класса `TextField`. Для получения информации об остальных свойствах можно обратиться к справочным материалам, поставляемым вместе с Flash MX 2004.

Таблица 21.2. Основные свойства класса `TextField`

Название свойства	Описание свойства
<code>text</code>	Содержит текст, выводимый в текстовом блоке. Строки разделяются символом перевода каретки <code>\r</code> (ASCII 13)
<code>variable</code>	Имя переменной, сопоставленной данному текстовому блоку. По умолчанию устанавливается значение <code>null</code>
<code>type</code>	Указывает тип текстового блока. Принимает строковое значение <code>dynamic</code> или <code>input</code> . По умолчанию устанавливается значение <code>dynamic</code>
<code>embedFonts</code>	Указывает, используются ли для отображения содержимого текстового блока системные (<code>device fonts</code>) или встроенные шрифты. Принимает логическое значение <code>true</code> (использовать встроенный шрифт) или <code>false</code> (использовать машино-независимый шрифт). По умолчанию используется значение <code>false</code> . Если текстовый блок создается программным образом, для отображения его содержимого с использованием встроенного шрифта данный шрифт необходимо экспорттировать в конечный фильм в качестве шрифтовой символа (см. разд. "Использование общих шрифтовых ресурсов" гл. 8). При этом в настройках параметров связи Linkage Properties необходимо установить флагок Export for ActionScript . Заданное здесь же название шрифтового символа должно использоваться при указании шрифта в свойстве <code>font</code> объекта <code>TextFormat</code>

Таблица 21.2 (продолжение)

Название свойства	Описание свойства
selectable	Указывает, может ли содержимое текстового блока быть выделено или нет. Принимает значение <code>true</code> или <code>false</code>
restrict	Задает набор символов, которые могут быть введены в текстовое поле. Принимает значение в виде строки. Значение <code>null</code> соответствует свободному вводу (любые символы), пустая строка ("") запрещает ввод любых символов. Диапазоны символов задаются при помощи тире (A-Z). Символ "^" позволяет включить набор предшествующих ему символов и исключить символы, находящиеся после него. Если строка начинается с "^", разрешается ввод любых символов, за исключением следующих после "^". Если "^" находится в середине строки, то разрешается ввод символов, предшествующих "^", а символы, находящиеся после "^" исключаются
maxChars	Задает максимальное число символов, которые пользователь сможет ввести в данное поле (это ограничение не распространяется на программный вывод текста). Значение <code>null</code> соответствует неограниченному числу символов. По умолчанию устанавливается значение <code>null</code>
password	Определяет, используется ли для текстового блока режим Password (замена всех символов звездочками). Принимает логическое значение <code>true</code> или <code>false</code> . По умолчанию устанавливается значение <code>false</code>
html	Определяет, используется ли HTML-форматирование для содержимого текстового блока. Принимает значение <code>true</code> или <code>false</code> . По умолчанию устанавливается значение <code>false</code>
htmlText	Содержит описание содержимого тестового блока в HTML-представлении
multiline	Определяет, содержит ли текстовое поле одну или несколько строк. Принимает значение <code>true</code> или <code>false</code> . По умолчанию устанавливается значение <code>false</code>
wordWrap	Определяет, выполняется ли автоматический перевод строки. Принимает логическое значение <code>true</code> или <code>false</code>
autoSize	Позволяет динамически изменять размер текстового блока в зависимости от объема размещаемого в нем текста. Данное свойство принимает одно из четырех строковых значений: <code>none</code> , <code>center</code> , <code>left</code> , <code>right</code> . Результат его действия зависит также от значений свойств <code>multiline</code> и <code>wordWrap</code> . Значение <code>none</code> используется по умолчанию, при этом размер текстового блока строго фиксирован, вследствие чего часть текста может быть срезана. Значение <code>center</code> изменяет размер текстового блока в соответствии с его содержимым, причем, если свойство <code>wordWrap</code> принимает значение <code>false</code> , то размер текстового блока изменяется за счет масштабирования от центра, а если свойство <code>wordWrap</code> принимает значение <code>true</code> , то размер

Таблица 21.2 (окончание)

Название свойства	Описание свойства
	текстового блока изменяется за счет смещения его правой и нижней границ. Значение <code>left</code> изменяет размер текстового блока в соответствии с его содержимым за счет смещения правой и нижней границ в зависимости от значения свойства <code>wordWrap</code> . Значение <code>right</code> изменяет размер текстового блока в соответствии с его содержимым за счет смещения левой и нижней границ в зависимости от значения свойства <code>wordWrap</code> . Любое значение данного свойства, за исключением <code>none</code> , может привести к тому, что реальный размер текстового поля будет отличаться от размера, заданного при его создании
<code>textColor</code>	Задает цвет помещаемого в текстовое поле текста в виде шестнадцатеричного значения
<code>background</code>	Определяет, имеет ли текстовый блок фон. Принимает логическое значение <code>true</code> или <code>false</code> . По умолчанию устанавливается значение <code>false</code>
<code>background-</code> <code>Color</code>	Задает цвет фона текстового блока в виде шестнадцатеричного числа, например, <code>0xD0E0FO</code>
<code>border</code>	Определяет, имеет ли текстовый блок рамку. Принимает значение <code>true</code> или <code>false</code> . По умолчанию устанавливается значение <code>false</code>
<code>borderColor</code>	Задает цвет рамки текстового блока в виде шестнадцатеричного числа
<code>mouseWheel-</code> <code>Enabled</code>	Определяет возможность прокрутки содержимого текстового блока при помощи колеса мыши. Принимает значение <code>true</code> или <code>false</code> . По умолчанию используется значение <code>true</code>
<code>scroll</code>	Определяет индекс строки, находящейся в верхней части видимой области текстового блока. Изменяя значение данного свойства, можно осуществлять построчную прокрутку текста
<code>length</code>	Возвращает количество символов в текстовом блоке. Используется только для чтения
<code>textHeight</code>	Возвращает суммарную высоту всех строк текста, содержащегося в текстовом блоке, в пикселях. Используется только для чтения
<code>textWidth</code>	Возвращает значение ширины, занимаемой текстом, в пикселях. Используется только для чтения

Динамическое создание и задание параметров текста

Для создания динамического или пользовательского текстового блока средствами ActionScript используется метод `createTextField()`. Новый текстовый

блок помещается в стек объекта, вызвавшего `createTextField()`, над всем его содержимым. Данный метод имеет следующий формат:

```
clip.createTextField(instanceName, depth, x, y, width, height)
```

Здесь:

- `clip` — родительский объект (имя экземпляра клипа или `_root`);
- `instanceName` — имя экземпляра создаваемого текстового блока;
- `depth` — z-индекс, т. е. уровень стека родительского объекта, на который будет помещен созданный текстовый блок, `x` и `y` — координаты левого верхнего угла текстового блока относительно начала координат родительского объекта;
- `width` и `height` — ширина и высота текстового блока.

Следующий сценарий создает динамический текстовый блок размером 105×20 пикселов с цветным фоном и рамкой в стеке основной монтажной линейки фильма.

```
_root.createTextField("myText", 1, 0, 0, 105, 20);
myText.text="Динамический текст";
myText.autoSize="none";
myText.textColor=0x336699;
myText.background=true;
myText.backgroundColor=0xB3CBE3;
myText.border=true;
myText.borderColor=0x3399CC;
```

Для удаления текстового блока, созданного с использованием метода `createTextField()`, применяется метод `removeTextField()`, имеющий следующий формат:

```
textField.removeTextField();
```

Здесь `textField` — это имя экземпляра удаляемого текстового блока. Следует иметь в виду, что действие данного метода не распространяется на текстовые блоки, созданные в рабочей среде вручную.

Обработчики событий объектов `TextField`

Класс `TextField` обладает несколькими обработчиками событий, позволяющими реагировать на манипуляции с экземпляром текстового блока. К их числу относятся:

- `onChanged()` — применяется для обнаружения изменения содержимого редактируемого текстового блока пользователем. Данный обработчик не может быть использован для обнаружения изменений, выполняемых программно.

Применение обработчика имеет следующий формат:

```
textField.onChanged = function() {  
    statements  
}
```

- `onSetFocus(oldFocus)` — применяется для обнаружения передачи фокуса текстовому полю при щелчке на нем или в результате использования клавиши `<Tab>`. При обнаружении передачи фокуса функции, присвоенной данному обработчику, в качестве параметра `oldFocus` передается ссылка на предыдущий объект, удерживающий фокус. Если такого объекта не существует, передается значение `null`. Применение данного обработчика имеет следующий формат:

```
textField.onSetFocus = function(oldFocus) {  
    statements  
}
```

- `onScroller(textFieldInstance)` — применяется для обнаружения факта прокрутки теста в текстовом блоке. Принимает параметр `textFieldInstance`, представляющий собой ссылку на экземпляр прокручиваемого текстового блока. Применение данного обработчика имеет следующий формат:

```
textField.onScroller = function(textFieldInstance) {  
    statements  
}
```

- `onKillFocus(newFocus)` — применяется для обнаружения утраты фокуса экземпляра текстового блока. При этом функции, присвоенной данному обработчику, в качестве параметра `newFocus` передается ссылка на объект, к которому переходит фокус. Если такого объекта не существует, передается значение `null`. Применение данного обработчика имеет следующий формат:

```
textField.onKillFocus = function(newFocus) {  
    statements  
}
```

Динамическое форматирование текста. Класс `TextFormat`

Класс `TextFormat` предоставляет возможность форматирования содержимого динамических и пользовательских текстовых полей программным образом. Использование встроенных свойств класса `TextFormat` позволяет программно задавать параметры форматирования символов текста, такие как шрифт,

его размер, начертание и т. д., и параметры форматирования абзаца, такие как выключка, размеры полей, интерлиньяж и т. д. В отличие от объекта `TextField`, имеющего визуальное выражение, объект `TextFormat` представляет собой набор данных, описывающих параметры форматирования текста, и не имеет самостоятельного визуального выражения. Для того чтобы программным образом отформатировать текст, созданный вручную или с использованием `createTextField()`, вначале необходимо создать экземпляр объекта `TextFormat` и задать все необходимые параметры форматирования при помощи его свойств. Для применения этих параметров к экземпляру текстового блока нужно воспользоваться методами класса `TextField`: `setTextFormat()` или `setNewTextFormat()`.

Для создания объекта `TextFormat` используется оператор `new` и функция конструктора `TextFormat()`. Следующая строка кода создает новый пустой экземпляр класса `TextFormat` с именем `textStyle`:

```
textStyle = new TextFormat();
```

В табл. 21.3 представлен перечень основных свойств класса `TextFormat`. Для получения информации об остальных свойствах можно обратиться к справочным материалам, поставляемым вместе с Flash MX 2004.

Таблица 21.3. Основные свойства класса `TextFormat`

Название свойства	Описание свойства
<code>font</code>	Задает название шрифта, используемого для отображения текста. Принимает строковое значение. При использовании шрифтового символа (<code>font symbol</code>) его название указывается в качестве значения данного свойства
<code>size</code>	Задает размер шрифта в пунктах (кегль)
<code>color</code>	Задает цвет текста в виде шестнадцатеричного числа
<code>bold</code>	Задает жирное начертание. Принимает логическое значение <code>true</code> или <code>false</code>
<code>italic</code>	Задает курсивное начертание
<code>underline</code>	Позволяет выводить текст с подчеркиванием. Принимает значение <code>true</code> или <code>false</code>
<code>align</code>	Задает выравнивание абзаца. Принимает одно из трех значений, задаваемых в виде строки: <code>left</code> , <code>right</code> , <code>center</code> . По умолчанию используется значение <code>null</code> , указывающее на то, что значение свойства не определено
<code>bullet</code>	Позволяет выводить текст как маркированный список. Принимает значение <code>true</code> или <code>false</code>

Таблица 21.3 (окончание)

Название свойства	Описание свойства
blockIndent	Задает отступ абзаца для всех строк текстового блока
indent	Задает отступ красной строки абзаца
leading	Задает значение интерлиньяжа (межстрочного расстояния)
leftMargin, rightMargin	Задают значения в пунктах для левого и правого полей текста соответственно
url	Позволяет обеспечить функционирование текста как гиперссылки. В качестве значения необходимо указать строку, содержащую адрес требуемого сетевого ресурса. Для того чтобы гиперссылка действовала, свойству <code>html</code> объекта <code>TextField</code> должно быть присвоено значение <code>true</code>
target	Определяет, в каком окне будет открыт сетевой ресурс, указанный в качестве значения свойства <code>url</code> . По умолчанию используется значение <code>_self</code> . Стандартными значениями данного свойства являются: <code>_self, _blank, _top, _parent</code>

После создания объекта `TextFormat` и задания требуемых значений его свойств данный объект может быть применен для форматирования всего содержимого текстового блока, фрагмента текста или отдельных символов. Для форматирования текста, помещенного в текстовый блок *программным образом*, необходимо использовать метод `setTextFormat()`. В зависимости от того, к какой части содержимого будет применено форматирование, данный метод может принимать различное число параметров. Для того чтобы отформатировать все содержимое текстового блока, используется следующий формат:

```
textField.setTextFormat (textFormat)
```

Здесь:

- `textField` — имя экземпляра текстового блока;
- `textFormat` — имя объекта `TextFormat`, содержащего информацию о параметрах форматирования.

Для того чтобы отформатировать фрагмент текста, применяется следующая запись:

```
textField.setTextFormat (beginIndex, endIndex, textFormat)
```

Здесь:

- `beginIndex` — индекс первого символа строки, входящего в форматируемый фрагмент;

- ❑ `endIndex` — индекс первого символа, следующего за форматируемым фрагментом. Следует иметь в виду, что индекс первого символа равен нулю (см. разд. "Строковый тип данных" гл. 20).

И наконец, для того чтобы применить форматирование к определенному символу текста, необходимо использовать следующий синтаксис:

```
textField.setTextFormat (index, textFormat)
```

Здесь `index` — это индекс форматируемого символа.

Для того чтобы форматирование автоматически применялось к тексту, вводимому пользователем (`Input text`) в процессе воспроизведения фильма, нужно использовать метод `setNewTextFormat()`. Применение данного метода аналогично использованию `setTextFormat()` и выглядит следующим образом:

```
textField.setNewTextFormat (textFormat)
```

Листинг 21.4 иллюстрирует принцип динамического создания и форматирования текста.

Листинг 21.4. Динамическое создание и форматирование текста

```
//Создание динамического текстового блока
_root.createTextField("myText", 1, 0, 0, 100, 50);
with(myText) {
text="Отправить сообщение";
autoSize="center";
html=true;
}
//Создание объекта TextFormat
myStyle=new TextFormat();
with(myStyle) {
font="Pragmatica";
size=24;
color=0xFF0000;
italic=true;
underline=true;
align="center";
url="mailto:dmitri@avalon.ru";
}
//Применение форматирования
myText.setTextFormat (myStyle);
```

Работа с клипами и кнопками

Перетаскивание объектов

Для перетаскивания (буксировки) объектов при помощи курсора мыши можно воспользоваться функцией `startDrag()`. Вызов этой функции приводит к тому, что целевой объект начинает следовать за курсором в пределах окна проигрывателя или заданной прямоугольной области, называемой областью перетаскивания. Данная функция вызывается как метод клипа или как глобальная функция. В зависимости от этого ее формат имеет следующий вид соответственно:

```
movieClip.startDrag(lock, left, top, right, bottom)  
startDrag(target, lock, left, top, right, bottom)
```

Вызов `startDrag()` в качестве метода клипа вызывает перемещение клипа `movieClip`, осуществившего вызов. Если `startDrag()` вызывается как глобальная функция, то в качестве ее первого аргумента `target` необходимо указать имя целевого объекта (*Instance Name*), который будет подвергнут перетаскиванию. Во втором случае в качестве целевого (т. е. перетаскиваемого) объекта может быть использован не только клип, но и кнопка и даже динамический или пользовательский текстовый блок. Параметр `lock` является необязательным и принимает одно из двух логических значений: `true` или `false`, определяющих, будет ли начало координат объекта совпадать с курсором мыши (`true`) при перетаскивании или нет (`false`). По умолчанию для данного параметра устанавливается значение `false`. Остальные параметры также являются необязательными и позволяют задать координаты левого верхнего (`left` и `top`) и правого нижнего (`right` и `bottom`) углов прямоугольной области перетаскивания. Если эти параметры опущены, то в качестве области перетаскивания будет использоваться все окно фильма.

Функция `stopDrag()` прекращает перетаскивание всех объектов, инициированное вызовом функции `startDrag()`. Функция `stopDrag()` не принимает параметров и может вызываться как метод клипа или как глобальная функция.

Следующий сценарий клипа вызывает его перемещение вслед за курсором при нажатии на данный клип и отключает перетаскивание при отпускании левой кнопки мыши:

```
//Сценарий клипа  
on(press){  
    this.startDrag(true);  
}  
on(release){  
    this.stopDrag();  
}
```

Этот простой сценарий может, тем не менее, быть использован для получения достаточно интересных эффектов, если содержащий его клип поместить на слой-маску.

Примечание

Важно понимать, что при использовании ключевого слова `this` в сценарии кнопки установка параметра функции `startDrag()` в значение `true` может привести к неожиданному (на первый взгляд) эффекту, когда при щелчке на кнопке она отскакивает в сторону, в связи с чем событие `release` уже не может быть выполнено. Такое поведение связано с тем, что областью видимости обработчика событий `on()` для кнопки является родительская монтажная линейка, и при вызове функции `startDrag()` начинает перемещаться все ее содержимое, а не только сама кнопка. По этой же причине при перемещении к курсору приклеивается не точка регистрации кнопки, а начало координат родительской линейки. В связи с этим в кнопке нужно использовать явную ссылку на себя с использованием имени экземпляра, например `startDrag(but,true)`.

Перемещение объекта вслед за курсором может быть также реализовано с использованием свойств `_xmouse` и `_ymouse`, как это было продемонстрировано в листинге 19.4 гл. 19.

В листинге 21.5 приведен пример создания функции `moveToBase()`, позволяющей реализовать простой вариант интерфейса, связанного с перетаскиванием объектов (drag-and-drop). Предполагается, что в кадре основной монтажной линейки имеется набор клипов, каждый из которых при нажатии мыши начинает перемещаться вслед за курсором, наподобие фрагментов головоломки (puzzle), и статичные клипы, изображающие участки, в которые должны быть помещены перетаскиваемые элементы. Если перетаскиваемый элемент при отпускании кнопки мыши "сбрасывается" на свою ответную часть (цель), то он автоматически притягивается к ее центру. Если объект отпущен за пределами цели, он возвращается в исходное положение. Функция принимает четыре параметра: `clip` — перетаскиваемый фрагмент, `base` — его ответная часть, `x` и `y` — начальные координаты перетаскиваемого объекта.

О назначении используемых здесь свойстве `_droptarget` и функции `eval()` говорится в гл. 19 и 20 соответственно.

Листинг 21.5. Функция `moveToBase()`

```
function moveToBase (clip:MovieClip, base:MovieClip, x:Number, y:Number) {  
    clip._x=x; /*Задает начальное положение клипа по горизонтали*/  
    clip._y=y; /*Задает начальное положение клипа по вертикали*/  
    clip.onPress=function() {/*При нажатии клип начинает перемещаться*/
```

```

        startDrag(this);
    }

clip.onRelease=function(){
    stopDrag(); /*При отпускании клип прекращает перемещение и
выполняется проверка попадания. Если клип "сброшен" на требуемый объект,
он помещается в центр цели, т. е. приклеивается к ней*/
    if(eval(clip._droptarget)==eval("_root."+base)) {
        clip._x=base._x;
        clip._y=base._y;
    }
/*Если клип "сброшен" мимо цели, он помещается в начальную точку*/
    else {
        clip._x=x;
        clip._y=y;
    }
}

/*Вызов созданной функции. Клип mc1 изначально находится в точке 0.0 и
при попадании на клип base1 будет к нему "приклеиваться"*/
moveToBase(mc1, base1, 0, 0);

```

Для того чтобы аналогичным образом обрабатывать несколько объектов, необходимо добавить на сцену новые перетаскиваемые клипы (clip) и их ответные части (base), присвоив имена их экземпляра. Затем необходимо вызвать функцию `moveToBase()` для каждой пары "клип — ответная часть", передав ей все необходимые параметры (если число объектов велико, это удобно сделать в цикле). Подобным образом можно создать простую обучающую детскую игру (например, сопоставить цвет и его название), головоломку и т. д.

Динамическое создание экземпляров

В рабочей среде Flash создание экземпляров клипов и кнопок осуществляется вручную, однако ActionScript предоставляет широкие возможности по созданию экземпляров программным образом в процессе воспроизведения фильма. Существуют три способа динамического создания экземпляров:

- дублирование имеющихся экземпляров при помощи метода или функции `duplicateMovieClip()`;
- размещение в фильме экземпляра клипа или кнопки, находящегося в библиотеке, при помощи метода или функции `attachMovie()`;
- создание пустого клипа при помощи метода `createEmptyMovieClip()`.

Иерархия объектов.

Размещение экземпляров в стеке

При создании экземпляров клипов и кнопок в рабочей среде они размещаются в стеке слоев, причем порядком их взаимного размещения можно непосредственно управлять посредством соответствующих команд главного меню **Modify>Arrange**. При создании экземпляров программным образом они помещаются в так называемый стек программно генерируемых клипов, где их взаимное расположение определяется z-индексом или уровнем (дословно "глубиной" — depth), также задаваемым программно. Фактически стек программно генерируемых клипов представляет собой аналог палитры слоев, предназначенных для размещения объектов, созданных программным образом.

Стек программно генерируемых клипов представляет собой пространство, предназначенное для размещения экземпляров, созданных программным образом. Стек программно генерируемых клипов основной монтажной линейки (`_root`) расположен над всеми слоями основной монтажной линейки. Таким образом, объект, программным образом помещенный в `_root`, будет перекрывать все объекты, помещенные на сцену в рабочей среде.

Вместе с тем каждый клип обладает собственным стеком программно генерируемых клипов, который, в свою очередь, находится над всеми слоями монтажной линейки данного клипа. Таким образом, объект, программным образом помещенный на монтажную линейку клипа (называемого "родительским"), будет перекрывать все его содержимое, созданное в рабочей среде, однако может находиться под элементами, расположенными в вышележащих (по отношению к слою, содержащему родительский клип) слоях монтажной линейки.

Порядок размещения экземпляров в стеке программно генерируемых клипов определяется его z-индексом, представляющим собой любое целое число. Экземпляры с большими значениями z-индекса размещаются над экземплярами с меньшими значениями z-индекса. При этом на каждом уровне стека может находиться только один экземпляр. Если z-индекс вновь создаваемого экземпляра совпадает с номером уровня стека, на котором уже имеется экземпляр, то новый экземпляр заместит собой предыдущий. Нумерация уровней стека не обязательно должна быть последовательной. Так, один экземпляр может быть помещен на уровень 5, а другой сразу на уровень 500. Значение имеет только взаимное расположение уровней друг относительно друга.

Наконец, помимо стека слоев и стека программно генерируемых клипов, существует еще один стек, называемый *стеком документов*. В стек документов загружаются не отдельные клипы, а целые Flash-фильмы (файлы SWF), которые также размещаются друг над другом, причем вышележащие документы перекрывают нижележащие. Структура стеков Flash Player схематично изображена на рис. 21.1.

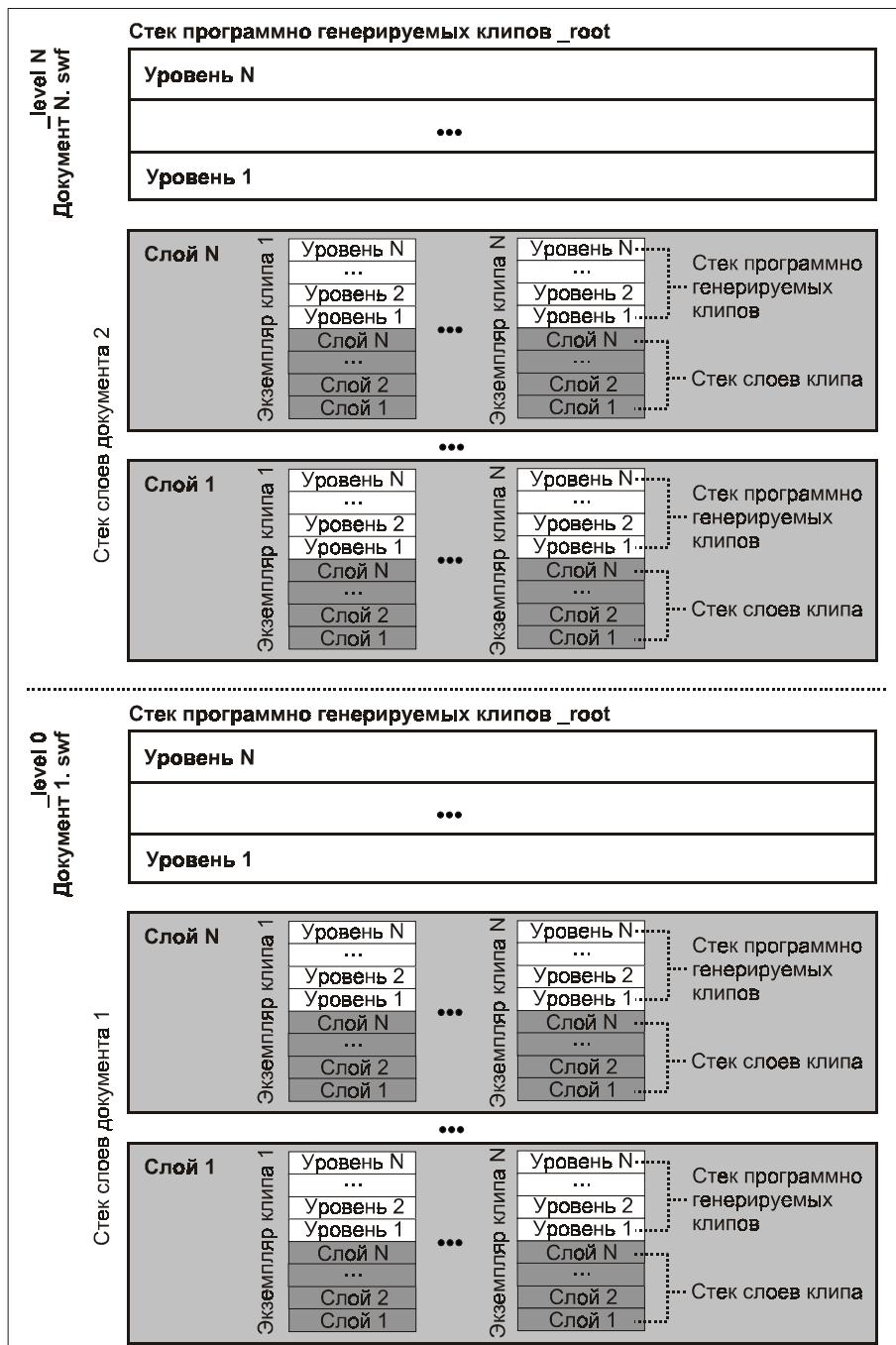


Рис. 21.1. Структура стеков Flash Player

Дублирование экземпляров

Программное дублирование экземпляров осуществляется при помощи функции `duplicateMovieClip()` или одноименного метода клипа.

Функция `duplicateMovieClip()` позволяет выполнять дублирование клипов, кнопок, динамических или пользовательских текстовых блоков и имеет следующий формат:

```
duplicateMovieClip(target, newname, depth)
```

Здесь:

- `target` — это строка, выражение, возвращающее строку или ссылка на объект (экземпляр), который будет дублирован;
- `newname` — строка или выражение, возвращающее строку, задающая имя дубликата объекта;
- `depth` — уровень или z-индекс дубликата в стеке.

Следует иметь в виду, что при вызове данной функции с основной монтажной линейки дубликат будет помещен в стек над `_root` (т. е. над всем содержимым основной монтажной линейки). При вызове этой функции с монтажной линейки клипа дубликат будет помещен в стек данного клипа (т. е. над всем его содержимым).

При вызове `duplicateMovieClip()` в качестве метода клипа выполняется дублирование данного экземпляра. Вызов метода `duplicateMovieClip()` имеет следующий формат:

```
clip.duplicateMovieClip(newname, depth, initObject)
```

Здесь:

- `clip` — имя экземпляра клипа, который будет дублирован, т. е. клипа-оригинала;
- `newname` — имя дубликата;
- `depth` — номер уровня в стеке, на который будет помещен дубликат (z-индекс);
- `initObject` — необязательный параметр, представляющий собой объект (или ссылку на него), все свойства которого будут скопированы в новый экземпляр.

Следует иметь в виду, что при вызове `duplicateMovieClip()` в качестве метода клипа приняты следующие правила.

- Если клип-оригинал был создан вручную, то его дубликат будет помещен в стек программно генерируемых клипов, расположенный над монтажной линейкой, содержащей клип-оригинал. Так, если родительский клип-оригинал создан вручную и находится на основной монтажной линейке, то его дубликат будет помещен в стек над `_root`.

- Если клип-оригинал был сгенерирован программно при помощи `attachMovie()`, то его дубликат будет помещен в стек данного клипа над всем его содержимым.

Следующий сценарий кадра основной монтажной линейки создаст дубликат клипа `mc` с именем `mc1` и поместит его на первый уровень стека программно генерируемых клипов фильма (т. е. над всеми слоями `_root`) на 50 пикселов правей и ниже оригинала:

```
mc.duplicateMovieClip("mc1",1);
mc1._x+=50;
mc1._y+=50;
```

Для того чтобы удалить программно созданный экземпляр, можно воспользоваться функцией `removeMovieClip()` или одноименным методом клипа. При вызове `removeMovieClip()` в качестве функции используется следующий формат:

```
removeMovieClip(target)
```

Здесь `target` — это строка или выражение, возвращающее строку, или ссылка на объект, который должен быть удален.

При вызове `removeMovieClip()` в качестве метода клипа используется следующий синтаксис:

```
clip.removeMovieClip()
```

Здесь `clip` — это имя удаляемого экземпляра клипа.

Необходимо отметить что `removeMovieClip()` используется только для удаления экземпляров, созданных программным образом; действие этого метода (или функции) не распространяется на экземпляры, созданные в рабочей среде (т. е. вручную).

В результате дублирования клипа его дубликаты наследуют внутреннюю структуру (включая вложенные символы и анимацию) и все параметры оригинала (размеры, угол поворота, прозрачность и т. д.) а также сценарии, прикрепленные к оригиналу. Это дает достаточно мощные возможности создания нескольких объектов, обладающих одинаковой функциональностью.

В качестве примера можно привести сценарий, формирующий шлейф, который перемещается вслед за курсором мыши. Принцип создания шлейфа основан на дублировании клипа, следующего за курсором. Таким образом, шлейф состоит из программно сгенерированных дубликатов экземпляра мультиклипа, размещенного на сцене в рабочей среде. Для того чтобы шлейф постепенно растворялся по мере удаления от курсора на монтажной линейке образующего клипа, можно создать простую анимацию движения, изменяющую степень его прозрачности от 100% до 0%. Каждый клип, полученный в результате дублирования, также будет содержать эту анимацию (рис. 21.2). Длительность анимации монтажной линейки образующего клипа влияет на время видимости шлейфа.



Рис. 21.2. Создание шлейфа мыши

Листинг 21.6 содержит пример сценария, формирующего шлейф. Для выполнения этой задачи необходимо создать клип и поместить на сцену его экземпляр, присвоив ему имя mc. На монтажной линейке mc нужно создать анимацию изменения его прозрачности (длительность анимации может быть порядка 1/3 секунды, т. е. 4 кадра при скорости воспроизведения 12 fps). В последний кадр монтажной линейки клипа mc нужно поместить следующий сценарий:

```
stop();  
this.removeMovieClip();
```

Первое предложение этого сценария позволяет избежать зацикливания монтажной линейки mc (в противном случае он будет постоянно пульсировать). Второе предложение будет удалять все дубликаты mc, после того как они станут невидимы (при этом сам оригинал, т. е. mc будет присутствовать на сцене, поскольку он был создан вручную).

Листинг 21.6. Создание шлейфа мыши

```
/*Создание переменной-счетчика, используемой для генерации имен  
дубликатов и задания их z-индекса*/  
var counter:Number = 0;  
  
//Задание начального положения mc в соответствии с положением курсора  
mc._x = _xmouse;  
mc._y = _ymouse;  
mc.onMouseMove = function() {  
    //Следование mc за курсором при движении мыши  
    this._x = _xmouse;  
    this._y = _ymouse;  
    updateAfterEvent();  
};  
  
//Приращение счетчика после каждого обнаружения перемещения мыши  
counter++;  
  
//Дублирование mc при перемещении мыши  
this.duplicateMovieClip("mc"+counter, counter);
```

```
//Обнуление счетчика при достижении максимального значения
if (counter>=50) {
    counter= 0;
}
};
```

Поскольку каждый дубликат должен быть размещен на отдельном уровне в стеке и иметь уникальное имя, здесь вводится переменная `counter`, значение которой увеличивается на единицу с частотой, равной частоте генерирования события `onMouseMove`. Таким образом, используя значение этой переменной в строковом контексте, можно указать уникальное значение z-индекса и имя для каждого дубликата. Обнуление счетчика выполняется для того, чтобы избежать бесконечного приращения значения `counter`. Максимальное значение счетчика должно быть задано таким образом, чтобы при существующей длительности анимации экземпляры становились невидимыми до того, как будет достигнуто максимальное значение z-индекса. После того как на 50-й уровень стека будет загружен клип `mc50`, его заполнение начнется снова с первого уровня (который к этому времени должен быть освобожден). Если загрузка на определенный уровень будет выполняться до того, как находящийся на нем клип исчезнет из виду и будет удален, то новый клип заместит предыдущий (чтобы наглядно в этом убедиться, можно поместить `stop()` в первый кадр монтажной линейки `mc` и просмотреть результат).

Динамическое присоединение экземпляров

Для того чтобы программным образом поместить в фильм экземпляр символа (клипа или кнопки), находящегося в библиотеке, нужно воспользоваться методом `attachMovie()`. Созданный или "присоединенный" таким образом экземпляр помещается в стек программно генерируемых клипов родительского, т. е. вызвавшего данный метод объекта (клипа или основной монтажной линейки). Для того чтобы получить возможность динамического присоединения экземпляров в рабочей среде, символу в библиотеке нужно задать параметры связи (`linkage`), экспортав его для ActionScript. Для этого необходимо выделить данный символ в библиотеке и выполнить команду его контекстного меню или контекстного меню библиотеки `Linkage`. В появившемся диалоговом окне **Linkage Properties** необходимо ввести уникальный идентификатор в поле **Identifier** и установить флажок **Export for ActionScript**. По умолчанию все экспортруемые для ActionScript символы загружаются до начала загрузки содержимого первого кадра. Это может вызвать задержку перед началом воспроизведения первого кадра фильма. Снятие флажка **Export in first frame** предотвращает загрузку экспортруемого символа перед первым кадром, однако при этом необходимо вручную поместить его экземпляр в любой кадр фильма. В противном случае символ не будет экспортован в конечный фильм. Если флажок **Export in first frame** снят, то динамическое присоединение может быть выполнено только после

загрузки экземпляра экспортируемого символа, помещенного в фильм вручную. Вызов метода `attachMovie()` имеет следующий формат:

```
clip.attachMovie(idName, newName, depth, initObject)
```

Здесь:

- `clip` — это имя объекта, в стек которого будет помещен создаваемый экземпляр;
- `idName` — строка, содержащая уникальный идентификатор символа библиотеки, заданный в окне **Linkage Properties** (его наличие обязательно), `newName` — строка, задающая имя нового экземпляра, помещаемого в стек;
- `depth` — число, указывающее уровень стека, на который будет помещен новый экземпляр;
- `initObject` — необязательный параметр, указывающий объект, свойства которого будут ассоциированы с новым экземпляром. В качестве объекта, вызывающего метод, может использоваться экземпляр клипа или `_root`.

Если экземпляр помещается в стек клипа, то его точка регистрации совпадает с точкой регистрации родительского клипа, если же экземпляр помещается в стек `_root`, то его точка регистрации располагается в начале координат, т. е. в левом верхнем углу рабочей области.

Динамически присоединенные экземпляры могут дублироваться и принимать в свой стек программно генерируемых клипов другие экземпляры.

В листинге 21.7 приведен пример сценария кадра, который создает заданное число экземпляров символа, находящегося в библиотеке документа, с частотой, равной скорости воспроизведения монтажной линейки фильма, и размещает их на сцене случайным образом, задавая каждому случайное значение прозрачности. В результате имеет место эффект мерцания. Экспортируемому символу библиотеки присвоен идентификатор связи `mc`. Динамически созданные экземпляры помещаются в стек программно генерируемых клипов основной монтажной линейки над всем ее содержимым.

Листинг 21.7. Динамическое присоединение экземпляров. Эффект мерцания

```
//Задание максимального числа экземпляров
var max:Number=25;
_root.onEnterFrame=function() {
//Создание экземпляров
for(i=0;i<=max;i++) {
_root.attachMovie("mc","mc"+i,i);
//Задание случайных значений координат в пределах окна фильма
this["mc"+i]._x=Math.random()*Stage.width;
this["mc"+i]._y=Math.random()*Stage.height;
```

```
//Задание случайного значения прозрачности от 0 до 100
this["mc"+i]._alpha=Math.random()*100;
}
}
```

Создание пустого клипа

Наряду с дублированием и присоединением экземпляров символа, созданного вручную, ActionScript предоставляет возможность создания пустого клипа, т. е. клипа без содержимого. Пустой клип, как и клип, созданный в рабочей среде, обладает собственным стеком программно генерируемых клипов и предоставляет пространство для размещения динамически присоединенных экземпляров клипов и текстовых блоков. Пустой клип также может быть использован для загрузки внешних файлов или вызова встроенных методов программного рисования (*см. далее в этой главе*). Для создания пустого клипа используется метод клипа `createEmptyMovieClip()`, имеющий следующий формат:

```
clip.createEmptyMovieClip(instanceName, depth)
```

Здесь:

- *clip* — родительский объект (экземпляр клипа или `_root`), в стек которого будет помещен созданный пустой клип;
- *instanceName* — имя экземпляра созданного клипа;
- *depth* — его z-индекс. Точка регистрации пустого клипа находится в его левом верхнем углу.

Для обращения к клипу, созданному с помощью `createEmptyMovieClip()` используется его имя *instanceName*.

Класс *Math*

Класс `Math` предоставляет целый набор свойств и методов, позволяющих осуществлять математические вычисления. Следует отметить, что все свойства и методы класса `Math` являются статическими, т. е. доступ к ним осуществляется непосредственно из класса и не требует создания экземпляра с помощью конструктора. Методы класса представляют собой функции, позволяющие выполнять различные вычисления и принимающие параметры, а свойства предоставляют доступ к математическим константам. Для вызова свойств или методов класса `Math` используется следующий синтаксис:

```
Math.method(parameter) //Вызов метода
Math.constant //Обращение к свойству
```

Здесь:

- *parameter* — параметр, передаваемый методу;
- *method* — имя свойства, содержащего константу.

В качестве параметра, передаваемого методу, может использоваться число, ссылка на числовую переменную или любое выражение, возвращающее число.

Свойства класса Math

Свойства класса Math (константы) дают возможность использования математических констант в расчетах.

Таблица 21.4. Свойства класса Math

Свойства	Описание
Math.E	Число Эйлера (≈ 2.718)
Math.LN2	Натуральный логарифм 2 (≈ 0.693)
Math.LOG2E	Логарифм e по основанию 2 (≈ 1.442)
Math.LN10	Натуральный логарифм 10 (≈ 2.302)
Math.LOG10E	Десятичный логарифм e (≈ 0.434)
Math.PI	Число π (≈ 3.14159)
Math.SQRT1_2	Квадратный корень 1/2 (≈ 0.707)
Math.SQRT2	Квадратный корень 2 (≈ 1.414)

Основные методы класса Math

Округление значений

Класс Math предоставляет три метода, используемых для округления значений:

- Math.round(*parameter*) — выполняет обычное математическое округление своего аргумента *parameter*. Числа, дробная часть которых больше или равна 0,5, округляются в сторону большего целого, а числа, дробная часть которых меньше или равна 0,5, округляются в сторону меньшего целого;
- Math.ceil(*parameter*) — округляет свой аргумент *parameter* в сторону большего целого (от англ. *ceiling* — потолок);
- Math.floor(*parameter*) — округляет свой аргумент *parameter* в сторону меньшего целого (от англ. *floor* — пол).

Следующие примеры иллюстрируют использование данных методов:

```
Math.round(3.5); //Возвращает 4  
Math.round(-3.5); //Возвращает -3  
Math.ceil(3.00001); //Возвращает 4  
Math.ceil(-3.00001); //Возвращает -3  
Math.floor(3.99999); //Возвращает 3  
Math.floor(-3.99999); //Возвращает -4
```

Использование генератора случайных чисел

Метод `Math.random()` генерирует случайное число, лежащее в интервале $[0, 1)$. Таким образом, результатом вызова данного метода является случайное число от 0 до 1 (причем 1 не входит в интервал). Для того чтобы изменить правую границу интервала появления случайных чисел, можно ввести масштабирующий коэффициент:

```
Math.random()*10; /*Возвращает число в диапазоне от 0 до 10
(исключая 10)*/
```

Для того чтобы случайное число, возвращаемое методом `random()`, было целым, его можно округлить, используя любой из описанных выше методов.

```
Math.round(Math.random()*10);/*Возвращает целое число в диапазоне от 0 до 10*/
```

В следующем сценарии создается функция `randomIntValue`, возвращающая целое число в заданном интервале:

```
function randomIntValue (minValue:Number,maxValue:Number):Number{
    return minValue+Math.round(Math.random() * (maxValue-minValue));
}
trace(randomIntValue(5, 15)); //Выводит целое число в диапазоне от 5 до 15
```

Тригонометрические функции

Методы, используемые для вычисления тригонометрических функций, представлены в табл. 21.5.

Таблица 21.5. Методы класса `Math` для вычисления тригонометрических функций

Метод	Аргументы	Назначение
<code>Math.sin(α)</code>	α — угол в радианах	Возращает результат вычисления синуса угла, заданного в качестве аргумента
<code>Math.cos(α)</code>	α — угол в радианах	Возращает результат вычисления косинуса угла, заданного в качестве аргумента
<code>Math.tan(α)</code>	α — угол в радианах	Возращает результат вычисления тангенса угла, заданного в качестве аргумента
<code>Math.acos(x)</code>	x — число от -1 до 1	Возращает угол в радианах, косинус которого задан в качестве аргумента. Если аргумент лежит за пределами отрезка $[-1, 1]$, возвращается значение NaN
<code>Math.asin(x)</code>	x — число от -1 до 1	Возращает угол в радианах, синус которого задан в качестве аргумента. Если аргумент лежит за пределами отрезка $[-1, 1]$, возвращается значение NaN
<code>Math.atan(x)</code>	x — число от $-\text{Infinity}$ до Infinity	Возращает угол в радианах, тангенс которого задан в качестве аргумента

Таблица 21.5 (окончание)

Метод	Аргументы	Назначение
Math.atan2 (y, x)	y, x – координаты точки относительно центра круга	Возвращает угол поворота точки с координатами y и x, заданными относительно центра круга. Угол измеряется в радианах относительно положительной горизонтальной полуоси круга (рис. 21.3)

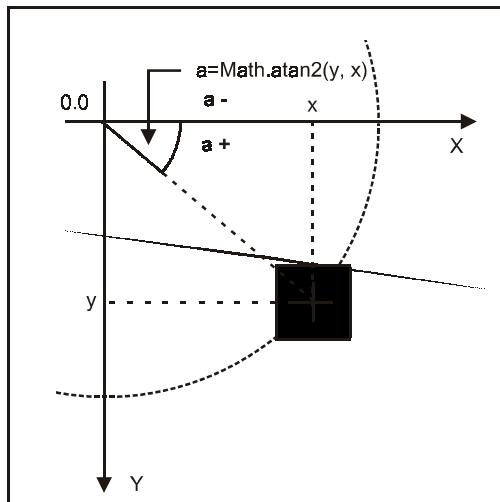


Рис. 21.3. Принцип использования метода Math.atan2(y, x)

Поскольку при работе с тригонометрическими функциями значения углов задаются и возвращаются в радианах, для их преобразования в градусы можно использовать следующую формулу:

$$\text{радианы} = \text{градусы} \cdot \pi / 180.$$

Тригонометрические функции находят широкое применение при необходимости реализации циклических процессов (например, непрерывного перемещения объекта по заданной траектории). Листинг 21.8 содержит сценарий, реализующий перемещение клипа mc по эллиптической орбите, причем клип, перемещаясь, оставляет за собой шлейф из собственных дубликатов.

Для реализации данной задачи необходимо вспомнить параметрическое уравнение эллипса, которое имеет следующий вид:

$$x = a \cdot \cos \theta;$$

$$y = b \cdot \sin \theta \quad (0 \leq \theta \leq 2\pi).$$

Если X-координата объекта (`mc`) будет изменяться по закону косинуса, а Y-координата — по закону синуса, то постепенное приращение аргумента θ (`theta`) будет вызывать движение объекта по эллиптической траектории. С увеличением шага приращения аргумента (`speed`) увеличивается смещение объекта, т. е. перемещение выполняется быстрее. Если одновременно с перемещением объекта выполнять его дублирование, то дубликаты будут формировать шлейф, наподобие хвоста кометы. Ограничение на число дубликатов не позволит объекту "столкнуться" со своим "хвостом", поскольку при загрузке нового дубликата на занятый уровень стека предыдущий экземпляр, расположенный на этом уровне, будет выгружен. Максимальное число дубликатов (`maxOnion`) определяет длину шлейфа, оставляемого объектом. Начальное значение `theta` выбирается случайным образом, так что всякий раз при запуске фильма движение будет начинаться с разных точек.

Листинг 21.8. Программная анимация. Движение по эллиптической орбите

```
var w:Number = 550;//Ширина орбиты
var h:Number = 400;//Высота орбиты
var speed:Number = 0.05;//Приращение аргумента theta
var maxOnion:Number = 10;//Длина шлейфа
var theta:Number = Math.random()*Math.PI*2; /*Движение из произвольной
точки*/
var d:Number = 0;//Инициализация счетчика дубликатов
mc.onEnterFrame = function() {
mc._x = Math.sin(theta)*(0.5*w-.5*mc._width)+0.5*w;
mc._y = Math.cos(theta)*(0.5*h-.5*mc._height)+0.5*h;
theta += speed;
if (theta>=Math.PI*2) { //Ограничение на значение аргумента
theta = 0;
}
//—————ДУБЛИРОВАНИЕ (формирование шлейфа)—————
d++;
if (d>=maxOnion) { //Ограничение на максимальное число дубликатов
d = 0;
}
mc.duplicateMovieClip("mc"+d, d);
};
```

Используя различные функциональные зависимости, можно без труда реализовать перемещение объектов по разнообразным траекториям. Так, например, заменив восьмую и девятую строки в тексте листинга 21.8 приве-

денными ниже строками, можно заставить mc перемещаться по траектории, описываемой функцией *астроиды* (здесь применяется операция возвведения в степень, описываемая ниже):

```
mc._x=Math.pow(Math.sin(theta),3)*(0.5*w-0.5*mc._width)+0.5*w;  
mc._y=Math.pow(Math.cos(theta),3)*(0.5*h-0.5*mc._height)+0.5*h;
```

В качестве иллюстрации применения метода `Math.atan2()` приводится листинг 21.9, содержащий сценарий клипа, заставляющий клип "следить" за перемещением курсора. Клип изменяет угол поворота, ориентируясь в направлении курсора мыши. Назначив данный сценарий клипу, можно скопировать его в рабочей среде или программным образом и добиться того, что все дубликаты также будут отслеживать перемещение курсора. Этот прием можно применить, например, при создании панели навигации, "приводящей взглядом" курсор. Для того чтобы сценарий выполнялся корректно, начальный угол поворота объекта в рабочей среде должен быть равен нулю. Для этого содержимое клипа должно быть позиционировано горизонтально в направлении положительной полуоси X (слева направо), либо в четвертой строке значение угла `angle` должно быть скорректировано на угол между положительной полуосью X и осью объекта.

Листинг 21.9. Следение за мышью

```
onClipEvent (mouseMove) {  
/*Сохраняем расстояние между курсором и клипом по осям X и Y*/  
var x=_parent._xmouse-this._x;  
var y=_parent._ymouse-this._y;  
/*Вычисляем угол линии, соединяющей точку регистрации клипа и курсор, и  
переводим его в градусы*/  
angle=Math.atan2(y,x)/(Math.PI/180);  
trace(Math.atan2(y,x));  
//Поворачиваем объект вслед за курсором  
this._rotation=angle;  
updateAfterEvent();  
}
```

Другие операции

Методы `Math.max()` и `Math.min()` принимают два аргумента и возвращают значение большего или меньшего из них соответственно. Данные методы имеют следующий формат:

```
Math.max(x, y) //Возвращает больший из своих аргументов  
Math.min(x, y) //Возвращает меньший из своих аргументов
```

Метод `Math.pow()` возвращает результат возведения одного аргумента в степень, заданную вторым аргументом, и использует следующий синтаксис:

`Math.pow(base, exponent)`

Здесь:

- `base` — основание (т. е. число, возводимое в степень);
- `exponent` — показатель степени, который может быть положительным, отрицательным или дробным (что позволяет использовать данный метод для вычисления корней).

Ниже демонстрируются примеры использования метода `Math.pow()`.

```
trace(Math.pow(2,3)); //Возвращает 8
trace(Math.pow(2,-2)); //Возвращает 0.25
trace(Math.pow(27,1/3)); //Возвращает 3
```

Класс `Stage`

Класс `Stage` используется для получения информации о границах окна Flash Player и задания параметров выравнивания и масштабирования фильма. Свойства и методы класса `Stage` являются статическими, т. е. доступ к ним осуществляется непосредственно из класса и не требует создания экземпляра с помощью конструктора. Для вызова свойств `Stage` используется следующий синтаксис:

`Stage.property`

Здесь `property` — название свойства.

В табл. 21.6 приведен перечень свойств класса `Stage` и их описание.

Таблица 21.6. Свойства класса `Stage`

Свойство	Назначение
<code>Stage.scaleMode</code>	Устанавливает режим масштабирования фильма в пределах окна Flash Player. Данное свойство принимает одно из следующих строковых значений: "showAll", "exactFit", "noScale" и действует аналогично применению параметра/атрибута тегов <code><object></code> и <code><embed></code> (см. гл. 18). Действие данного свойства распространяется также на автономный проигрыватель (Projector)
<code>Stage.align</code>	Выравнивание фильма относительно окна Flash Player. Данное свойство принимает одно из следующих строковых значений: "T", "B", "L", "R", "TL", "TR", "BL", "BR". Результат применения того или иного значения эквивалентен применению одноименного параметра/ атрибута тегов <code><object></code> и <code><embed></code> (см. гл. 18). Действие данного свойства распространяется также на автономный проигрыватель (Projector)

Таблица 21.6 (окончание)

Свойство	Назначение
Stage.height	Возвращает значение высоты фильма или окна Flash Player, исходя из установленного значения свойства scaleMode. Если scaleMode принимает значение "noScale", то height возвращает значение ширины окна Flash Player. В этом случае изменение размеров окна проигрывателя приведет к изменению значения свойства height. Если scaleMode принимает значение, отличное от "noScale", то свойство height возвращает значение высоты рабочей области фильма, заданное в рабочей среде документа. Используется только для чтения
Stage.width	Возвращает значение ширины фильма или окна Flash Player, исходя из установленного значения свойства scaleMode. Действует аналогично свойству height (см. выше). Используется только для чтения
Stage.showMenu	Позволяет скрыть все пункты контекстного меню проигрывателя Flash Player, за исключением About Macromedia Flash Player 7 . Принимает логическое значение true или false

Следует иметь в виду, что при размещении Flash-фильма на HTML-странице значения свойств Stage.scaleMode и Stage.align могут быть переопределены соответствующими параметрами тегов <object> и <embed>, созданными в результате публикации в формате HTML. Для того чтобы этого избежать, нужно удалить соответствующие фрагменты из HTML-кода.

Класс Stage содержит обработчик событий onResize, позволяющий обнаруживать факт изменения размеров фильма, при условии, что свойству Stage.scaleMode присвоено значение "noScale". Для того чтобы получить возможность выполнения определенных действий при изменении размера фильма, необходимо зарегистрировать объект-приемник (listener) на прием сообщений об обнаружении данного события от объекта Stage. Для этого используется метод класса Stage addListener():

```
listenerObject.onResize = function() {
statements
};
```

Пример, иллюстрирующий применение обработчика onResize, приведен в листинге 21.2.

Класс Mouse

Класс Mouse предоставляет возможность манипулирования стандартным курсором мыши, а также передавать объектам сообщения о событиях, связанных с мышью.

ных с мышью. Доступ ко всем методам класса Mouse осуществляется непосредственно из класса и не требует создания его экземпляра с помощью конструктора.

Методы класса *Mouse*

Метод `Mouse.hide()` позволяет скрыть изображение стандартного курсора мыши.

Метод `Mouse.show()` позволяет включить отображение стандартного курсора мыши.

Листинг 21.10 демонстрирует пример сценария, позволяющего сымитировать пользовательский курсор. Данный сценарий должен быть назначен экземпляру клипа, используемому в качестве курсора.

Листинг 21.10. Создание простого пользовательского курсора

```
onClipEvent(mouseMove) {  
    Mouse.hide(); //Отключение отображения стандартного курсора  
    _x=_parent._xmouse; //Следование за курсором  
    _y=_parent._ymouse;  
    updateAfterEvent(); //Обновление экрана  
}  
  
on(press) { //Увеличение размеров при нажатии кнопки мыши  
    _xscale=150;  
    _yscale=150;  
}  
  
on(release) { //Возврат к исходным размерам  
    _xscale=100;  
    _yscale=100;  
}
```

Метод `Mouse.addListener()` позволяет зарегистрировать объект-приемник (`listener`), которому будет передаваться сообщение о событиях `onMouseUp`, `onMouseDown` и `onMouseWheel` от объекта-источника `Mouse`. Метод `Mouse.removeListener()` отключает передачу сообщений объекту-приемнику (см. разд. "Понятие об объектах *Listeners*" данной главы). События `onMouseUp` и `onMouseDown` описываются в разд. "Обработчики событий клипов" гл. 19.

Событие `onMouseWheel` генерируется при прокрутке колеса мыши. Для того чтобы получить возможность выполнять определенные действия, связанные с прокруткой колеса мыши, необходимо зарегистрировать объект-приемник (`listener`) при помощи метода `Mouse.addListener()`.

Для обработки события `onMouseWheel` используется следующий синтаксис:

```
listenerObject.onMouseWheel = function (delta , scrollTarget) {  
statements  
};
```

Здесь:

- `delta` — необязательный параметр, определяющий, сколько строк прокручивается за каждое смещение колеса мыши. При прокрутке вверх значения `delta` положительные, при прокрутке вниз — отрицательные;
- `scrollTarget` — это необязательный параметр, содержащий ссылку на самый верхний клип, над которым находится курсор мыши в момент прокрутки.

Следующий сценарий перемещает клип `mc` по вертикали при прокрутке колеса мыши (для его работы фокус должен быть передан окну Flash Player, для чего можно щелкнуть внутри окна):

```
mc.onMouseWheel=function(delta) {  
    this._y+=delta;//Перемещение mc с шагом равным шагу прокрутки  
}  
Mouse.addListener(mc);//Регистрация mc на принятие сообщений
```

Класс Key

Класс `Key` предоставляет набор свойств и методов, позволяющих осуществлять программный контроль клавиатуры. Эта возможность широко используется при разработке интерактивности, предполагающей управление объектами при помощи клавиатурных клавиш. Все свойства и методы класса `Key` являются статическими, т. е. доступ к ним осуществляется непосредственно из класса и не требует создания экземпляра с помощью конструктора.

Методы класса Key

Метод `Key.isDown()` осуществляет проверку нажатия клавиши, указанной в качестве его аргумента, и возвращает логическое значение `true`, если клавиша нажата, и `false` — если клавиша не нажата. Вызов данного метода осуществляется следующим образом:

```
Key.isDown(keycode)
```

Здесь `keycode` — виртуальный код клавиши, являющийся числом, которое соответствует клавише клавиатуры. Виртуальный код клавиши отвечает физическому местоположению клавиши клавиатуры, а не соответствующему ей символу и не зависит от регистра и используемого языка. Обычно коды кла-

виш от A до Z соответствуют кодовым позициям этих букв в кодировке ASCII (65–90). Аналогичным образом дело обстоит и с цифровыми клавишами основной клавиатуры. Однако коды остальных клавиш не соответствуют кодам их символов в кодировке ASCII. Для получения доступа к некоторым специальным клавишам не обязательно помнить их код, для этого достаточно воспользоваться встроенными свойствами класса Key. Далее в этом разделе будет представлен сценарий, позволяющий определить код любой клавиши.

Метод Key.isDown() часто используется в проверочных выражениях условных предложений для проверки условия нажатия клавиши. Так, следующий сценарий выводит в панель **Output** сообщение на протяжении всего времени, пока клавиша <S>, виртуальный код которой равен 83, остается нажатой.

```
_root.onEnterFrame = function() {  
    if (Key.isDown(83)) {  
        trace("Нажата клавиша <S>");  
    }  
};
```

Поскольку проверка нажатия должна выполняться непрерывно, здесь используется обработчик событий onEnterFrame, осуществляющий проверку с частотой, равной скорости воспроизведения монтажной линейки.

Важной особенностью метода Key.isDown() является способность одновременно контролировать нажатие нескольких клавиш. Например, выражение Key.isDown(65) && Key.isDown(90) возвратит значение true только при одновременном нажатии клавиш <A> и <Z> (или <Ф> и <Я>).

Метод Key.getASCII() не принимает аргументов и возвращает целое число, соответствующее ASCII-коду последней нажатой клавиши. Данный метод различает буквы верхнего и нижнего регистра и не различает клавиши с одинаковыми значениями ASCII-кода (например, цифры на основной и дополнительной клавиатуре). Следует иметь в виду, что метод Key.getCode() не принимает аргументов и возвращает целое число, соответствующее виртуальному коду последней нажатой клавиши. Поскольку виртуальный код описывает физическое расположение клавиши, то буквам "ф", "Ф", "а" и "А" будет соответствовать один и тот же виртуальный код (65), т. к. они находятся на одной и той же клавише. В то же время цифра 1 основной клавиатуры и цифра 1 дополнительной клавиатуры имеют разные виртуальные коды (49 и 97), т. к. находятся на разных клавишиах.

Следующий сценарий клипа, приведенный в листинге 21.11, позволит определить виртуальный код и ASCII-код любой нажимаемой клавиши ("\\r" используется в качестве оператора перевода каретки).

Листинг 21.11. Определение кода клавиши (сценарий назначается экземпляру клипа)

```
onClipEvent (keyDown) {  
    trace(String.fromCharCode(Key.getAscii()) + ":\r" + "Виртуальный код " +  
        Key.getCode() + "\rASCII код " + Key.getAscii())  
}
```

Метод `Key.isToggled()` позволяет контролировать специальные клавиши `<Caps Lock>` и `<Num Lock>`, возвращая логическое значение `true` или `false` в зависимости от того, включен ли соответствующий режим или нет. Вызов данного метода осуществляется следующим образом:

```
Key.isToggled(keycode)
```

Здесь `keycode` — это код клавиши `<Caps Lock>` (20) или `<Num Lock>` (144). Следующий сценарий иллюстрирует применение данного метода:

```
onClipEvent (keyDown) {  
    if(Key.isToggled(20))trace("Режим Caps Lock включен");  
    else trace ("Режим Caps Lock отключен")  
}
```

Метод `Key.addListener()` позволяет зарегистрировать объект-приемник (`listener`), которому будет передаваться сообщение о событиях `onKeyUp` и `onKeyDown` от объекта-источника `Key`. Метод `Key.removeListener()` отключает передачу сообщений объекту-приемнику (см. разд. "Понятие об объектах *Listeners*" данной главы).

Свойства класса Key

Свойства класса `Key` предоставляют коды специальных клавиш (таких как `<Enter>`, `<Left>`, `<Home>`, `<Escape>` и т. д.). Таким образом, для контроля этих клавиш необязательно помнить наизусть их коды. Перечень всех свойств можно увидеть во всплывающей подсказке, которая автоматически появляется вслед за вводом оператора точки после указания имени класса `Key`. Все свойства записываются прописными буквами, например:

`Key.LEFT`, `Key.ENTER`, `Key.HOME`

и т. д.

Управление объектом при помощи клавиатуры

Использование методов объекта `Key` позволяет осуществлять управление объектами при помощи клавиш клавиатуры. Листинг 21.12 содержит сценарий, позволяющий осуществлять простое перемещение клипа `mc` при помо-

щи клавиш стрелок клавиатуры (для перемещения по диагонали можно нажать две клавиши).

Листинг 21.12. Простое управление объектом при помощи клавиатуры

```
var step:Number=5;//Шаг смещения
mc.onEnterFrame=function() {
if(Key.isDown(Key.LEFT)) _x-=step;
if(Key.isDown(Key.RIGHT)) _x+=step;
if(Key.isDown(Key.UP)) _y-=step;
if(Key.isDown(Key.DOWN)) _y+=step;
};
```

Сценарий, представленный в листинге 21.13, позволяет реализовать более сложный характер движения управляемого объекта, учитываящий инерционность и возможность придания ускорения движению. Данный сценарий должен быть размещен в кадре монтажной линейки, содержащем экземпляр клипа с именем `mc`. При нажатии и удерживании клавиши управления курсором клип начинает перемещаться, автоматически разворачиваясь в требуемом направлении (для этого используется прием, аналогичный представленному в листинге 21.9). При нажатии клавиши `<Space>` клип получает ускорение, однако скорость может возрастать только до заданного максимального значения. При отпускании клавиши или нажатии другой клавиши клип еще некоторое время движется по инерции. Для корректной работы в рабочей среде содержимое клипа должно быть сориентировано, как показано на рис. 21.4.

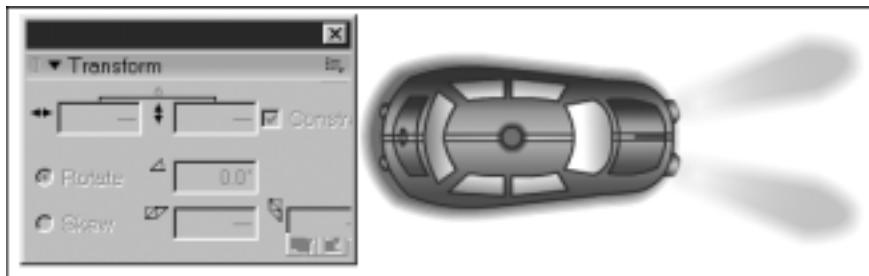


Рис. 21.4. Начальное положение клипа

**Листинг 21.13. Управление объектом при помощи клавиатуры.
Инерция и ускорение**

```
var xSpeed = 0;//Начальное смещение по горизонтали
var ySpeed = 0;//Начальное смещение по вертикали
```

```
var speedUp = 0.5;//Шаг смещения
var accelerate = 1;//Начальное значение ускорения
var decay=0.9;//Константа затухания (определяет инерционность)
var maxSpeed=25;//Максимально допустимая скорость
mc._rotation=0;//Начальный угол поворота клипа
//-----УСКОРЕНИЕ-----
mc.onEnterFrame=function() {
    //При нажатии SPACE – ускорение возрастает
    if(Key.isDown(Key.SPACE)) accelerate+=1;
    /*Если SPACE не нажата – ускорение затухает, достигая значения 1, при
    котором ускорение отсутствует, т. е. клип движется равномерно или
    покоятся*/
    else accelerate*=1-decay;
    if (accelerate<1) accelerate=1;
//-----ВЫЧИСЛЕНИЕ СМЕЩЕНИЙ-----
/*Если нажата клавиша LEFT или RIGHT, клип движется в соответствующем
направлении с текущим значением ускорения; в противном случае смещение
по горизонтали затухает*/
if(Key.isDown(Key.LEFT)) xSpeed-=speedUp*accelerate;
else if (Key.isDown(Key.RIGHT)) xSpeed+=speedUp*accelerate;
else xSpeed*=decay;
/*Если нажата клавиша UP или DOWN, клип движется в соответствующем
направлении с текущим значением ускорения; в противном случае смещение
по вертикали по инерции затухает*/
if(Key.isDown(Key.UP)) ySpeed-=speedUp*accelerate;
else if(Key.isDown(Key.DOWN)) ySpeed+=speedUp*accelerate;
else ySpeed*=decay;
//-----КОНТРОЛЬ СКОРОСТИ-----
/*Фактическая скорость вычисляется по теореме Пифагора как квадратный
корень из суммы квадратов смещений по горизонтали и вертикали*/
var speed=Math.sqrt(xSpeed*xSpeed+ySpeed*ySpeed);
/*Если фактическая скорость достигла или превысила предельное значение,
снижаем ее путем уменьшения смещения клипа по горизонтали и вертикали
пропорционально превышению скорости*/
if (speed>=maxSpeed) {
    trace(Math.sqrt(xSpeed*xSpeed+ySpeed*ySpeed));
    xSpeed*=maxSpeed/speed;
    ySpeed*=maxSpeed/speed;
}
//-----ПЕРЕМЕЩЕНИЕ КЛИПА-----
//Перемещение клипа
mc._x+=xSpeed;
```

```

mc._y+=ySpeed;
//Поворот клипа по направлению движения
mc._rotation = Math.atan2(ySpeed, xSpeed) / (Math.PI/180)
//Ограничение выхода клипа за пределы рабочей области
if(mc._x>Stage.width) mc._x=0;
if(mc._x<0) mc._x=550;
if(mc._y>Stage.height) mc._y=.0;
if(mc._y<0) mc._y=400;
};

```

Контроль времени и даты

Программный контроль времени и даты осуществляется при помощи методов встроенного класса `Date`. Для получения доступа к методам класса `Date` необходимо создать экземпляр класса и сохранить его в переменной. Для создания экземпляра используется функция конструктора `Date()`. В общем виде эта процедура имеет следующий вид:

```
var myDate:Date = new Date();
```

После того как экземпляр класса создан и помещен в переменную, вызов методов осуществляется с использованием имени данной переменной и оператора точки:

```
myDate.method()
```

В табл. 21.7. представлены некоторые методы класса `Date` и дано их описание.

Таблица 21.7. Некоторые методы класса `Date`

Метод	Описание
<code>getDate()</code>	Возвращает день месяца в виде числа от 1 до 31 в соответствии с местным временем
<code>getDay()</code>	Возвращает номер дня недели в виде числа от 0 (воскресенье) до 6 (понедельник) в соответствии с местным временем
<code>getFullYear()</code>	Возвращает год в виде четырехзначного числа в соответствии с местным временем
<code>getHours()</code>	Возвращает часы в соответствии с местным временем
<code>getMilliseconds()</code>	Возвращает миллисекунды в соответствии с местным временем
<code>getMinutes()</code>	Возвращает минуты в соответствии с местным временем
<code>getMonth()</code>	Возвращает номер месяца, начиная с 0 (0 — январь, 1 — февраль и т. д.) в соответствии с местным временем

Таблица 21.7 (окончание)

Метод	Описание
getSeconds()	Возвращает секунды в соответствии с местным временем
toString()	Возвращает строку, содержащую время и дату

Листинг 21.14 содержит сценарий, выполняющий сравнение текущей даты и заданной даты истечения испытательного срока. Подобный подход можно реализовать для создания простой защиты демонстрационной версии фильма, которая представляется заказчику на предварительном этапе. В случае если дата испытательного срока еще не истекла, можно, например, переместить воспроизводящую головку на сцену, содержащую начало фильма, или загрузить в проигрыватель внешний SWF-файл.

Листинг 21.14. Проверка окончания испытательного срока

```
var date=new Date();
//Сохраняем текущую дату
var day = date.getDate(); //День
var month = date.getMonth()+1; //Месяц (начиная с 1)
var year = date.getFullYear(); //Год
//Задаем дату окончания испытательного срока
var expiryDay = 13; //День
var expiryMonth = 4; //Месяц
var expiryYear = 2003; //Год
//Сообщение, выводимое при окончании испытательного срока
var expiryMessage = "Испытательный срок истек " + expiryDay + "." +
    expiryMonth + "." + expiryYear
//Вывод текущей даты
trace("Сегодня "+day+"."+month+"."+year);
//Проверка окончания испытательного срока
if(year>expiryYear){
    trace(expiryMessage);
}
else if (month>expiryMonth&&year==expiryYear) {
    trace(expiryMessage);
}
else if (day>expiryDay&&month==expiryMonth&&year==expiryYear) {
    trace(expiryMessage);
}
else trace("Испытательный срок продолжается...");
```

Методы класса `Date` позволяют без труда создать часы со стрелками, показывающие системное время. Сценарий, реализующий эту задачу, представлен в листинге 21.15. Данный сценарий должен быть помещен в кадр основной монтажной линейки, при этом на сцене необходимо разместить три экземпляра символов типа мюви-клип, изображающих стрелки часов. Этим экземплярам необходимо присвоить имена `hours` (часовая стрелка), `minutes` (минутная стрелка), `seconds` (секундная стрелка). Важно, чтобы центр регистрации каждого символа, изображающего стрелку, совпадал с точкой, относительно которой будет выполняться вращение. В рабочей среде стрелки должны располагаться в вертикальном направлении, соответствующем 12-ти часам (при этом угол поворота каждого экземпляра должен быть равен нулю, как показано на рис. 21.5). Принцип действия сценария заключается в том, что текущее значение времени связывается с углом поворота соответствующей стрелки. Поскольку дуга окружности составляет 360 градусов, минутные и секундные деления размещаются на ней через каждые 6 градусов ($360 / 60 = 6$), а часовые деления — через 30 градусов ($360 / 12 = 30$). Для корректной работы часов важно точно разметить циферблат.

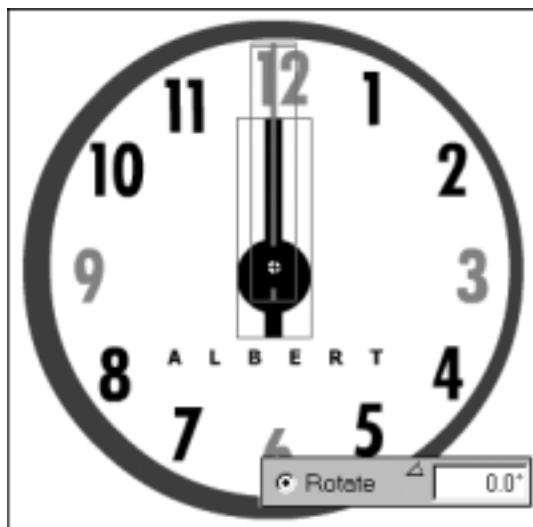


Рис. 21.5. Расположение стрелок часов в рабочей среде

Листинг 21.15. Часы

```
_root.onEnterFrame=function() {  
var time:Date=new Date();
```

```
//Сохранение текущего времени  
var sec=time.getSeconds();  
var min=time.getMinutes();  
var hour=time.getHours();  
  
//Вращение стрелок  
_root.seconds._rotation=sec*6;  
_root.minutes._rotation=(min+sec/60)*6;  
_root.hours._rotation=((hour+min/60)*30);  
}
```

Загрузка внешних фильмов и изображений

Возможности ActionScript позволяют осуществлять динамическую загрузку в текущий фильм внешних файлов SWF и растровых изображений в формате JPEG. Внешние файлы могут быть загружены в предназначенный для этого клип либо на один из уровней глобального стека документов Flash Player. Загрузка внешних файлов SWF позволяет реализовать еще один способ организации навигации по проекту (наряду с использованием функции `getURL()`), состоящему из нескольких файлов SWF. При этом используется единственная HTML-страница, а переход от одного раздела к другому осуществляется путем загрузки соответствующего файла непосредственно в текущий документ, размещенный на данной странице. Динамическая загрузка изображений бывает весьма полезна при необходимости включения в проект большого числа растровых изображений, например для создания слайд-шоу или демонстрации каталога художественных работ. Встраивание большого числа растровых изображений в документ приведет к существенному увеличению его объема и, соответственно, времени загрузки. Динамическая загрузка изображений позволит организовать процесс загрузки более эффективно.

Для выполнения загрузки внешних фильмов или изображений можно воспользоваться любым из трех существующих способов.

- Метод `loadClip()` встроенного класса `MovieClipLoader`.
- Метод или функция `loadMovie()`.
- Функция `loadMovieNum()`.

Встроенный класс `MovieClipLoader` является одним из новшеств, представленных с выходом FlashPlayer 7.0 и Flash MX 2004. С его помощью можно выполнять загрузку внешних файлов SWF и JPEG и осуществлять контроль за ходом загрузки. Для того чтобы получить доступ к методам данного клас-

са, необходимо создать его экземпляр при помощи функции конструктора `MovieClipLoader()` и сохранить его в переменной:

```
var myLoader = new MovieClipLoader()
```

Для выполнения загрузки используется метод `loadClip()`, имеющий следующий синтаксис:

```
myLoader.loadClip("url", target)
```

Здесь:

- `myLoader` — имя экземпляра объекта `MovieClipLoader`;
- `url` — путь к загружаемому файлу в виде строки или выражения, возвращающего строку (следует иметь в виду, что при размещении Flash-фильма, осуществляющего загрузку внешних файлов на HTML-странице, путь `url` к загружаемым файлам задается относительно данной страницы);
- `target` — ссылка на целевой клип, в который будет осуществлена загрузка внешнего файла, или целое число, задающее номер уровня стека документов.

При загрузке внешнего файла в клип он замещает собой все содержимое целевого клипа, причем левый верхний угол загружаемого фильма или изображения размещается в точке регистрации целевого клипа. Для получения аккуратного результата необходимо добиться соответствия размеров целевого клипа и загружаемого фильма или изображения. Чтобы в результате загрузки положение загруженного объекта четко совпадало с положением целевого клипа, нужно убедиться в том, что точка регистрации целевого клипа находится в его левом верхнем углу. Загрузка внешнего файла в клип приводит к замещению его содержимого, однако сам клип продолжает существовать и к нему по-прежнему можно обратиться по имени экземпляра. Изменение свойств целевого клипа оказывает влияние на его содержимое (т. е. на загруженный фильм или изображение).

Если в качестве параметра `target` используется целое число, то загружаемый объект будет помещен не в клип, а на указанный уровень стека документов, при этом его левый верхний угол будет совмещен с левым верхним углом рабочей области. Flash Player может одновременно осуществлять воспроизведение нескольких документов, при этом они размещаются друг над другом в так называемом *стеке документов*. Каждый документ SWF или загруженное растровое изображение занимает свой уровень стека документов. Все уровни пронумерованы, причем документы, расположенные на вышележащих уровнях (имеющих больший номер), перекрывают документы, находящиеся на нижележащих уровнях (имеющих меньший номер). Каждый уровень может содержать только один документ. При загрузке нового документа на уровень, содержащий документ, предыдущий документ выгружается. Нумерация уровней начинается с нуля и не обязательно должна быть после-

довательной, т. е. можно загрузить один документ на уровень 1, а следующий — сразу на уровень 100. На нулевом уровне стека документов всегда размещается основной фильм. Если внешний документ загружается на нулевой уровень, то он замещает собой содержимое основной монтажной линейки фильма. Непосредственное обращение к уровням стека документов осуществляется с помощью свойства `_level`, после которого следует указание номера уровня.

Следующий сценарий демонстрирует применение метода `loadClip()`:

```
//Создание объекта MovieClipLoader и сохранение его в переменной myLoader  
var myLoader:MovieClipLoader=new MovieClipLoader();  
//Загрузка изображения в клип holder  
myLoader.loadClip("Image1.jpg",holder);  
//Загрузка документа slideShow.swf на 10-й уровень стека (_level10);  
myLoader.loadClip("slideShow.swf",10);
```

Примечание

Проигрыватель Flash Player не поддерживает загрузку изображений в формате JPEG, использующих режим **Progressive**. При сохранении изображения (например, в Photoshop), предназначенного для загрузки в Flash, убедитесь в том, что опция **Progressive** отключена. В противном случае изображение просто не будет загружено!

Для того чтобы удалить загруженный документ, можно воспользоваться методом `unloadClip()`, который вызывается следующим образом:

```
myLoader.unloadClip(target)
```

Здесь `target` — строка, указывающая имя целевого клипа (в который был загружен внешний файл) или целое число, указывающее уровень стека документов, содержащий загруженный файл. Вызов данного метода удаляет указанный в качестве его аргумента клип или очищает соответствующий уровень стека.

Следующий сценарий удаляет клип `holder` из предыдущего фрагмента кода при нажатии на кнопку `but`:

```
but.onPress=function() {  
    myLoader.unloadClip("holder");  
}
```

В качестве альтернативы применению методов объекта `MovieClipLoader` для динамической загрузки внешних фильмов SWF и растровых изображений JPEG можно воспользоваться методом клипа `loadMovie()` или функцией `loadMovieNum()`.

Метод `loadMovie()` действует аналогично методу `loadClip()` и позволяет осуществить загрузку внешнего файла в клип. Метод `loadMovie()` имеет следующий синтаксис:

```
clip.loadMovie("url", method)
```

Здесь:

- `clip` — целевой клип, в который будет осуществлена загрузка внешнего файла;
- `url` — путь к загружаемому файлу в виде строки или выражения, возвращающего строку;
- `method` — необязательный параметр, задающий метод передачи переменных внешнему сценарию (`GET` или `POST`). `loadMovie()` также может вызываться в качестве глобальной функции. При этом используется следующий синтаксис:

```
loadMovie("url", target, method)
```

Здесь `target` — это строка, содержащая ссылку на целевой клип или уровень стека документов (в виде `"_leveln"`, где `n` — номер уровня). При вызове `loadMovie()` в качестве функции важно, чтобы ссылка `target` была задана правильно. Если данный параметр принимает значение `undefined`, то внешний файл будет загружен в стек текущего объекта (т. е. над содержимым его монтажной линейки).

Функция `loadMovieNum()` аналогична функции `loadMovie()`, однако в отличие от последней загружает внешний документ не в клип, а на заданный уровень стека документов:

```
loadMovie("url", level, method)
```

Здесь `level` — целое число, указывающее уровень стека, на который будет осуществлена загрузка внешнего документа.

Для выгрузки (удаления) документов, загруженных с помощью функций `loadMovie()` и `loadMovieNum()`, можно использовать функции `unloadMovie()` и `unloadMovieNum()` соответственно. Данные функции вызываются следующим образом:

```
unloadMovie(target)  
unloadMovieNum(level)
```

Здесь:

- `target` — строка, содержащая имя выгружаемого клипа;
- `level` — целое число, указывающее уровень стека, на котором содержится загруженный документ.

В результате загрузки внешнего SWF-файла, в котором используется абсолютная адресация, в клип, все ссылки, реализуемые в загруженном фильме с применением свойства `_root`, по умолчанию разрешаются относительно основной монтажной линейки документа, содержащего родительский клип, а не относительно основной монтажной линейки загруженного фильма. Это может явиться причиной неполадок в работе сценария. Для того чтобы указать, каким образом будут разрешаться ссылки на `_root`, используемые в загруженном в клип SWF-файле, можно воспользоваться новым свойством `_lockroot`. Для того чтобы ссылки с использованием свойства `_root` в загружаемом документе всегда разрешались относительно его основной монтажной линейки, необходимо присвоить свойству `_lockroot` значение `true`. Присвоение может быть осуществлено как в самом загружаемом документе, так и в документе, содержащем родительский клип, в который осуществляется загрузка. В первом случае в кадр основной монтажной линейки загружаемого документа достаточно добавить следующее предложение:

```
this._lockroot=true;
```

Если по тем или иным причинам доступ к исходному документу загружаемого файла отсутствует, то в документе, содержащем клип, в который осуществляется загрузка, можно поместить код следующего вида:

```
var myLoader:MovieClipLoader=new MovieClipLoader();
myLoader.loadClip("movie.swf", mc);
mc._lockroot=true;
```

Здесь `mc` — клип, в который осуществляется загрузка файла `movie.swf`.

Создание предзагрузчика

Загрузка любого Flash-фильма из сети не происходит мгновенно, при этом, если его объем достаточно велик, то время загрузки может быть весьма продолжительным. Среда тестирования позволяет оценить реальный процесс загрузки при помощи команды **View>Simulate Download** (см. разд. "Эмуляция загрузки" гл. 16). Для того чтобы избежать остановок воспроизведения, возникающих при загрузке, и сделать ее процесс более предсказуемым и понятным для зрителя, необходимо снабдить фильм предзагрузчиком. Принцип действия предзагрузчика (Preloader) заключается в том, что он не позволяет начинать воспроизведение фильма до полного окончания его загрузки. На протяжении загрузки зритель может видеть сообщение о количестве загруженных байт, которое может быть проиллюстрировано увеличивающейся полосой состояния, простой анимацией или иным образом.

Для создания предзагрузчика можно применить два метода класса `MovieClip`: `getBytesLoaded()` и `getBytesTotal()`.

Метод `getBytesLoaded()` возвращает количество загруженных байт определенного клипа или всего фильма и использует следующий формат:

```
clip.getBytesLoaded()
```

Метод `getBytesTotal()` вызывается аналогично и возвращает размер определенного клипа или всего фильма в байтах:

```
clip.getBytesTotal()
```

Здесь `clip` — загружаемый клип или `_root`.

Листинг 21.16 демонстрирует создание простого графического предзагрузчика, который выводит в динамическое текстовое поле состояние загрузки в процентах, и отображает процесс загрузки графически в виде изменяющейся в размерах (по горизонтали) полосы состояния. Для создания данного предзагрузчика необходимо выполнить следующие действия:

1. Создать новый ключевой кадр и сделать его первым, поместив перед всеми остальными кадрами основной монтажной линейки.
2. В первом кадре создать динамический текстовый блок и сопоставить ему переменную `counter`, введя ее имя в поле **Var** Инспектора свойств.
3. В первом кадре создать муви-клип, содержащий изображение полосы состояния загрузки. Точка регистрации клипа должна совпадать с левой гранью полосы состояния. Далее необходимо присвоить этому клипу имя `progressBar`, задав его в поле **Instance Name** Инспектора свойств. Затем поверх этого клипа нужно нарисовать рамку, совпадающую с его границами (рис. 21.6).

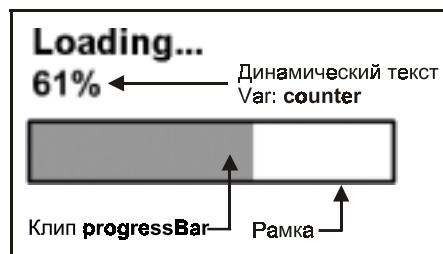


Рис. 21.6. Простой графический предзагрузчик

4. В первый кадр основной монтажной линейки поместить сценарий, представленный в листинге 21.16.

Листинг 21.16. Графический предзагрузчик

```
// Сохранение размера фильма в байтах в переменной total
var total=_root.getBytesTotal();
```

```
progressBar.onEnterFrame=function() {
/*Сохранение текущих значений загруженных байт в переменной loaded*/
var loaded=_root.getBytesLoaded();
//Если загрузка продолжается...
if (loaded<total) {
//Остановка воспроизведения
stop();
//Сохранение текущего состояния загрузки в процентах в переменной
var loadProgress=Math.round(loaded/total*100);
/*Вывод состояния загрузки в процентах в динамическое текстовое поле
counter*/
counter=loadProgress+"%";
/*Изменение ширины полосы состояния в соответствии с процентом
загруженных байт*/
progressBar._xscale=loadProgress;
}
//Когда загрузка окончена – запуск воспроизведения
else play();
}
```

5. Создать пустой ключевой кадр после последнего кадра основной монтажной линейки и поместить в него следующий сценарий:

```
/*Сценарий кадра, следующего за последним кадром анимации основной
монтажной линейки*/
gotoAndPlay(2);
```

Это нужно сделать для того, чтобы при зацикливании воспроизведения основной монтажной линейки после окончания загрузки не возвращаться в первый кадр.

Если на основной монтажной линейке имеется только один кадр (за исключением кадра предзагрузчика), то в него нужно просто поместить `stop();`

Создание слайд-шоу с динамической загрузкой изображений

В гл. 13 описывался процесс создания слайд-шоу с использованием автоматической анимации. Такой подход требует внедрения всех используемых растровых изображений в документ, что приводит к существенному увеличению его размера. В данном разделе описывается более приемлемый для сети Интернет подход, связанный с загрузкой изображений в фильм из внешних файлов в формате JPEG.

В листинге 21.17 представлен сценарий интерактивного слайд-шоу, обладающего следующей функциональностью. Наличие кнопок "Вперед" и "Назад" позволяет последовательно просматривать изображения, загружая их из внешнего файла. Первое изображение будет загружено автоматически. При переходе к следующему или возврату к предыдущему изображению текущее изображение постепенно становится прозрачным, после чего начинается загрузка нового изображения. В процессе загрузки на сцене появляется текстовое поле, содержащее информацию о состоянии загрузки изображения в процентах. После того как изображение полностью загрузится, текстовое поле исчезает и новое изображение постепенно "проявляется", становясь непрозрачным. До тех пор, пока новое изображение полностью не загрузится и не станет полностью непрозрачным, переход к следующему изображению невозможен.

Для реализации данной задачи необходимо выполнить следующие действия:

1. Создать две кнопки, по нажатию на которые будет выполняться переход к предыдущему или следующему изображению. Экземплярам этих кнопок необходимо присвоить имена (Instance Name) *prevPic* и *nextPic* соответственно.
2. Создать мультиклип, исполняющий роль контейнера, в который будет осуществляться загрузка изображений. Наполнение клипа-контейнера не играет роли, поскольку загружаемые изображения заменят содержимое его монтажной линейки. Экземпляр этого клипа нужно назвать *holder*. При позиционировании *holder* на сцене следует иметь в виду, что левый верхний угол загружаемого изображения будет находиться в точке регистрации данного клипа.
3. Все изображения, включенные в слайд-шоу, должны быть размещены в том же каталоге, где будет находиться HTML-страница, содержащая Flash-фильм, и названы *Image1.jpg*, *Image2.jpg*, *Image3.jpg* и т. д. Убедитесь в том, что данные изображения не используют режим **Progressive!** Для того чтобы все изображения занимали одинаковую область в клипе *holder*, они должны иметь одинаковый размер (для приведения размеров можно воспользоваться любым растровым редактором, например, Fireworks или Photoshop).
4. В кадр, содержащий кнопки *prevPic* и *nextPic* и клип *holder*, поместить сценарий, представленный в листинге 21.17 (комментарии можно опустить).
5. Для корректной работы слайд-шоу переменной *maxPic* в начале сценария необходимо присвоить значение, соответствующее количеству изображений (в данном примере их 12). Читатель также может варьировать значение переменной *alphaStep* для регулировки скорости изменения прозрачности изображения.

Листинг 21.17. Интерактивное слайд-шоу с динамической загрузкой изображений

```
//Создание нового объекта MovieClipLoader
var picLoader:MovieClipLoader = new MovieClipLoader();
//Начальный номер загружаемого изображения
var pic:Number = 1;
//Максимальное число загружаемых изображений
var maxPic:Number = 12;
//Шаг изменения прозрачности изображения
var alphaStep:Number = 10
/*Начальное значение прозрачности клипа holder для автоматической
загрузки первого изображения*/
holder._alpha = alphaStep;
/*Создание динамического текстового блока для вывода в него состояния
загрузки*/
_root.createTextField("display",1,275,200,0,0);
display.autoSize="center";
display.textColor=0xff0000;
/*Переход к следующему изображению только при условии, что текущее
изображение полностью загрузилось и стало непрозрачным*/
//————КНОПКИ "ВПЕРЕД" И "НАЗАД"————
nextPic.onPress = function() {
    if (pic<maxPic && !fadeIn && !fadeOut && loadProgress == 100) {
        pic++;
        fadeIn = true;
    }
};
/*Переход к предыдущему изображению только при условии, что текущее
изображение полностью загрузилось и стало непрозрачным*/
prevPic.onPress = function() {
    if (pic>1 && !fadeIn && !fadeOut && loadProgress == 100) {
        pic--;
        fadeIn = true;
    }
};
//————
//Сценарий, выполняемый с частотой, равной скорости воспроизведения
фильма
_root.onEnterFrame = function() {
```

```
//————ПРЕДЗАГРУЗЧИК————
loadProgress =
Math.round(holder.getBytesLoaded()/holder.getBytesTotal()*100);
    if (loadProgress<100) {
/*Если изображение еще не загрузилось, выводим состояние загрузки*/
        display._visible=true;
        display.text=loadProgress+"%";
    }
/*Если изображение загрузилось, отключаем текстовое поле*/
    else display._visible=false;
//————КОНЕЦ ПРЕДЗАГРУЗЧИКА————
/*Если непрозрачность holder продолжает уменьшаться (т. е. _alpha
уменьшается), но остается больше минимального значения, равного
alphaStep, уменьшаем значение _alpha с шагом alphaStep*/
    if (holder._alpha>=alphaStep && fadeOut) {
        holder._alpha -= alphaStep;
    }
/*Если непрозрачность holder достигла минимального значения
(alphaStep), загружаем в holder требуемое изображение, переключаем
флаги*/
    if (holder._alpha<=alphaStep) {
        picLoader.loadClip("Image"+pic+".jpg", holder);
        fadeOut = false;//Прекращаем уменьшать непрозрачность
        fadeIn = true;//Начинаем увеличивать непрозрачность
    }
/*Если непрозрачность holder продолжает увеличиваться (т. е. _alpha
растет), но остается меньше 100, увеличиваем _alpha с шагом alphaStep
только при условии, что изображение полностью загружено*/
    if (holder._alpha<=100 && fadeIn && !fadeOut) {
/*Проверка окончания загрузки изображения, если true, то уменьшаем
прозрачность*/
        if(loadProgress>=100)holder._alpha += alphaStep;
    }
/*Если ни одно из предыдущих условий не выполняется, то изображение
стало полностью непрозрачным, поэтому прекращаем увеличивать
непрозрачность*/
    else {
        fadeIn = false;//Прекращаем увеличивать непрозрачность
    }
};
```

6. Протестировать фильм в среде тестирования в режиме **View>Simulate Download**.

Дополнительные пояснения

Для обеспечения изменения прозрачности клипа, содержащего загруженное изображение, в данном сценарии, используются две переменные флага, *fadeIn* и *fadeOut*. Каждая из них в определенный момент принимает логическое значение *true* или *false*. Так, переменная *fadeIn* принимает значение *true*, если непрозрачность клипа *holder* увеличивается, а переменная *fadeOut* принимает значение *true*, если непрозрачность клипа *holder* уменьшается. Во избежание путаницы следует пояснить, что увеличение непрозрачности означает увеличение значения свойства *_alpha* (при этом объект становится менее прозрачным), соответственно уменьшение непрозрачности означает уменьшение значения свойства *_alpha* (при этом объект становится более прозрачным). Таким образом, максимальное значение непрозрачности соответствует *_alpha = 100*. В качестве минимального значения непрозрачности (или максимальной прозрачности) в данном сценарии используется значение *alphaStep*. Читатель может изменить это минимальное значение, заменив *alphaStep* в условиях проверочных предложений произвольным фиксированным значением.

Динамическое рисование при помощи ActionScript

ActionScript располагает набором средств, позволяющих программным образом рисовать контуры, формируя их из отрезков прямых и парабол, и создавать цветовое заполнение замкнутых контуров. Для динамического рисования используются методы класса *MovieClip*. Методы рисования можно применять к экземпляру клипа или к *_root*. Если методы рисования применяются к клипу, то динамически нарисованные формы будут располагаться под всем содержимым монтажной линейки этого клипа, созданным в рабочей среде. Если методы рисования применяются к основной монтажной линейке (*_root*), то динамически нарисованные формы будут располагаться под всем содержимым основной монтажной линейки, созданным в рабочей среде, т. е. на заднем плане. Методы рисования также могут быть применены к пустому клипу, созданному при помощи метода *createEmptyMovieClip()*. Таким образом, перед тем как приступить к рисованию, необходимо создать (или определить) объект, к которому будут применены методы программного рисования.

Создание форм, состоящих из прямолинейных сегментов

Для того чтобы нарисовать произвольную форму, границы которой представляют собой прямолинейные отрезки, необходимо создать программный код, содержащий последовательность команд, описанную ниже.

Перед началом рисования вызывается метод `lineStyle()`, позволяющий задать толщину, цвет и прозрачность рисуемого контура. Для вызова метода `lineStyle()` используется следующий формат:

```
clip.lineStyle(thickness, RGB, alpha)
```

Здесь:

- `Clip` — это целевой объект, в котором будет размещена нарисованная форма (под всем содержимым этого объекта, созданным в рабочей среде);
- `thickness` — толщина линии, измеряемая в точках (`points`); значение толщины может изменяться в диапазоне от 0 до 255, значение 0 соответствует линии типа `hairline` (если данный параметр опущен или его значение не определено, то линия не будет нарисована);
- `RGB` — цвет линии, заданный в шестнадцатеричном формате (например, `0xFF0000`) (если значение цвета не указано, в качестве цвета линии по умолчанию будет использован черный цвет);
- `alpha` — прозрачность линии, задаваемая в диапазоне от 0 до 100 (если значение прозрачности не задано, будет использовано значение 100).

Для того чтобы задать цвет заливки создаваемого контура, можно воспользоваться методом `beginFill()`, имеющим следующий синтаксис:

```
clip.beginFill(RGB, alpha)
```

Здесь:

- `RGB` — цвет заливки, заданный в шестнадцатеричном формате (если данный параметр не задан, заливка не будет создана);
- `alpha` — прозрачность заливки, задаваемая в диапазоне от 0 до 100 (если значение прозрачности не задано, будет использовано значение 100).

Для задания точки, с которой будет начато рисование, используется метод `moveTo()`, имеющий следующий формат:

```
clip.moveTo(x, y)
```

Аргументы `x` и `y` являются числами, задающими начальную позицию для рисования. Координаты `x` и `y` задаются относительно точки регистрации родительского клипа или левого верхнего угла рабочей области, если рисование осуществляется в `_root`.

Рисование прямолинейных отрезков осуществляется при помощи метода `lineTo()`, который соединяет текущую точку рисования с точкой, заданной его аргументами:

```
clip.lineTo(x, y)
```

Здесь `x` и `y` — координаты конца создаваемого отрезка. В качестве начальной точки используется текущая точка рисования, определяемая методом

`moveTo()` или предыдущим вызовом `lineTo()`. В результате применения данного метода текущая точка рисования перемещается в позицию, указанную `x` и `y`.

Следующий сценарий рисует в `_root` прямоугольник с красной рамкой толщиной 10 и зеленой полупрозрачной заливкой (использование предложения `with` позволяет избежать многократного повторения имени целевого объекта (см. гл. 19)).

```
with(_root) {  
    lineStyle(10,0xFF0000, 100); //Параметры линии  
    beginFill(0x00FF00,50); //Параметры заливки  
    moveTo(0,0); //Начальная точка  
    lineTo(100,0);  
    lineTo(100,100);  
    lineTo(0,100);  
    lineTo(0,0); //Замыкание контура  
    endFill();
```

Метод `endFill()` используется для того, чтобы залить существующий контур заливкой, определенной методом `beginFill()`. Если при вызове данного метода контур не замкнут, то он автоматически замкнется и будет залит. Метод `endFill()` вызывается следующим образом:

```
clip.endFill();
```

Для того чтобы удалить формы, нарисованные программным образом, нужно воспользоваться методом `clear()`, имеющим следующий формат:

```
clip.clear()
```

В результате вызова `clear()` из родительского объекта удаляются все динамически нарисованные формы, а также стиль линий, заданный методом `lineStyle()`.

Используя динамическое рисование, можно реализовать множество интересных декоративных анимационных эффектов. Так, например, если нарисовать форму, а затем перемещать ее вершины по определенным траекториям, то форма будет принимать разные очертания, плавно перетекая и трансформируясь. Листинг 21.18 демонстрирует пример сценария кадра основной монтажной линейки, реализующего подобный эффект (рис. 21.7).

В данном сценарии создается четыре массива. Два из них (`initx` и `inity`) используются для хранения значений углов, задаваемых случайным образом. Углы будут использоваться для вычисления координат вершин по горизонтали и вертикали соответственно. Два других массива (`x` и `y`) будут использоваться для хранения самих координат вершин. Пользовательская функция `moveTo()` организует перемещение каждой вершины по эллиптической орби-

те, это делается примерно так же как и в листинге 21.8. Элементы массивов initx и inity используются в качестве аргументов для вычисления координат вершин. Варьируя шаг приращения угла, можно изменять скорость перемещения вершины по горизонтали или вертикали. Рисование выполняется по описанному выше принципу. Формы перерисовываются с частотой, равной скорости воспроизведения фильма, причем всякий раз происходит изменение координаты вершин. Очистка экрана необходима для того, чтобы предыдущие формы не оставались на сцене при перемещении вершин и рисовании новых форм.

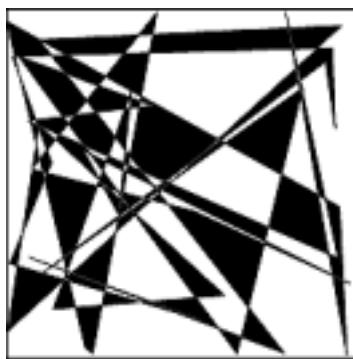


Рис. 21.7. Анимация
с использованием динамического рисования

Листинг 21.18. Анимация с использованием динамического рисования

```
var maxPoints:Number = 35;//Число вершин
var lineColor:Number = 0x3300FF;//Цвет контура
var fillColor:Number = 0xFF9933;//Цвет заливки
/*Массивы для хранения значений углов, используемых в качестве аргументов
функции синуса для задания координат вершин*/
var initx:Array= new Array();//Массив для вычисления X-координат
var inity:Array= new Array();//Массив для вычисления Y-координат
/*Массивы для хранения координат точек вершин*/
var x:Array= new Array();
var y:Array= new Array();
/*Начальные значения углов определяются случайным образом в диапазоне от
0 до 2PI*/
for (k=0; k<=maxPoints-1; k++) {
    initx[k] = Math.random()*2*Math.PI;
```

```
inity[k] = Math.random()*2*Math.PI;  
}  
//————ФУНКЦИЯ, ПЕРЕМЕЩАЮЩАЯ ВЕРШИНЫ————  
function movePoints():Void{  
for (i=0; i<=maxPoints-1; i++) {  
/*Перемещение вершин по эллиптической орбите внутри квадрата размером  
100x100*/  
    x[i] = 100*Math.sin(initx[i])+100;  
    y[i] = 100*Math.sin(inity[i])+100;  
    initx[i] += .06;//Приращение угла для изменения координаты по X  
    inity[i] += .08;//Приращение угла для изменения координаты по Y  
}  
}  
//————РИСОВАНИЕ————  
_root.onEnterFrame = function():Void {  
    clear();//Очистка экрана перед новым перемещением вершин  
    movePoints();//Перемещение вершин  
    moveTo(x[0], y[0]);//Установка начальной точки рисования  
    lineStyle(1, lineColor, 100);//Параметры линий  
    beginFill(fillColor, 100);//Заливка контура  
    for (i=1; i<=maxPoints-1; i++) {  
        lineTo(x[i], y[i]);//Соединение вершин  
    }  
    lineTo(x[0], y[0]);//Замыкание контура  
};
```

Примечание

Для получения дополнительной иллюстрации применения методов динамического рисования можно адресовать читателя к разделу "Creating a simple line drawing tool" справочных материалов по ActionScript (**Help>ActionScript Dictionary**). Здесь описывается простой способ создания рисующего инструмента.

Управление кривыми

Для создания форм, состоящих из криволинейных сегментов, используется метод `curveTo()`. Кривые, создаваемые при помощи этого метода, содержат только одну управляющую точку, положение которой определяет характер кривизны сегмента. Криволинейный сегмент рисуется из точки, заданной методом `moveTo()` или из текущей точки рисования, являющейся конечной точкой предыдущего сегмента (рис. 21.8).

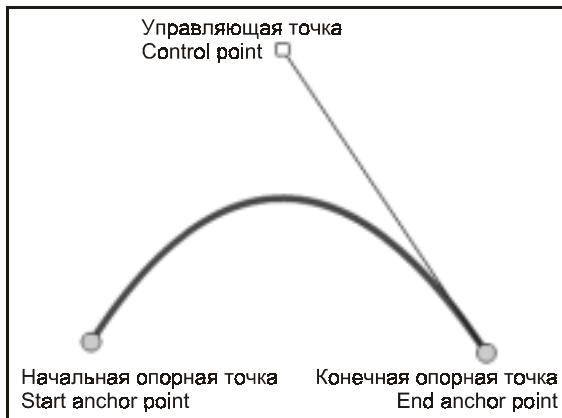


Рис. 21.8. Создание кривой при помощи метода `curveTo()`

Данный метод имеет следующий формат:

```
clip.curveTo(controlX, controlY, anchorX, anchorY)
```

Здесь:

- *controlX* и *controlY* — координаты управляемой (контрольной) точки кривой по горизонтали и вертикали соответственно;
- *anchorX* и *anchorY* — координаты конечной опорной точки сегмента по горизонтали и вертикали соответственно.

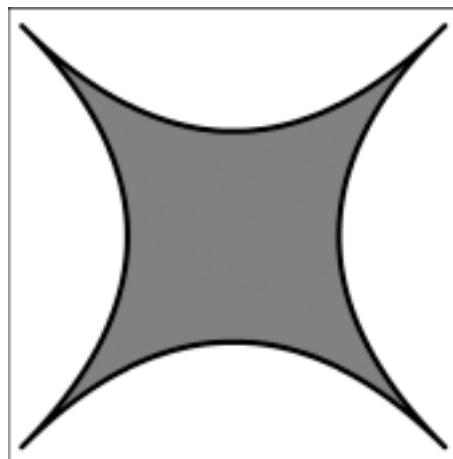


Рис. 21.9. Применение метода `curveTo()`

Следующий сценарий рисует фигуру, изображенную на рис. 21.9.

```
_root.createEmptyMovieClip( "mc", 1 );
with ( _root.mc ){
    lineStyle( 0, 0x00FF00, 100 );
    beginFill( 0xFF0000 );
    moveTo( 0, 0 );
    curveTo( 200, 200, 400, 0 );
    curveTo( 200, 200, 400, 400 );
    curveTo( 200, 200, 0, 400 );
    curveTo( 200, 200, 0, 0 );
    endFill();
}
```

При перемещении опорных (anchor) и управляющих (control) точек происходит изменение формы фигуры. Листинг 21.19 содержит сценарий, позволяющий перемещать эти точки при помощи курсора мыши. Для его реализации необходимо в библиотеке создать два символа типа **Movie Clip**, изображающие опорную и управляющую точки (рис. 21.10). Данные символы будут динамически присоединяться к `_root` при помощи метода `attachMovie()`, поэтому для каждого из них необходимо установить флагок **Export for ActionScript** в диалоговом окне настройки параметров связи **Linkage Properties**. Кроме того, в поле **Identifier** каждому из них необходимо задать идентификатор `anchor` (для символа с изображением опорной точки) и `control` (для символа с изображением управляющей точки).

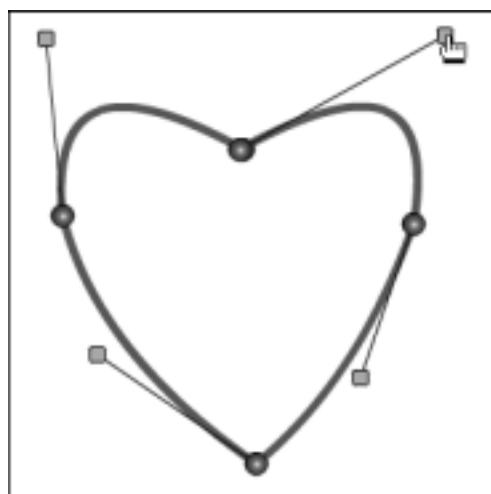


Рис. 21.10. Управление кривыми

Листинг 21.19. Интерактивное управление кривыми

```
var maxPoints=5;//Число опорных точек
//Присоединение экземпляров
for(i=1;i<=maxPoints;i++) {
    attachMovie("anchor","anchor"+i,i);
    attachMovie("control","control"+i,i+maxPoints);
//Позиционирование экземпляров на сцене
this["anchor"+i]._x=(i-.5)*Stage.width/maxPoints;
this["anchor"+i]._y=Stage.height*0.8;
this["control"+i]._x=(i-.5)*Stage.width/maxPoints;
this["control"+i]._y=Stage.height*0.2;
/*Удаление несуществующей контрольной точки (кривая имеет только одну
управляющую точку)*/
removeMovieClip("control1");
/*—ОРГАНИЗАЦИЯ ПЕРЕТАСКИВАНИЯ КЛИПОВ, ИМИТИРУЮЩИХ ТОЧКИ—*/
this["anchor"+i].onPress=function() {
    startDrag(this,false,0,0,550,400);
}
this["anchor"+i].onRelease=function() {
    stopDrag();
}
this["control"+i].onPress=function() {
    startDrag(this,false,0,0,550,400);
}
this["control"+i].onRelease=function() {
    stopDrag();
}
}
//————РИСОВАНИЕ————
_root.onEnterFrame=function() {
    _root.clear();
    moveTo(anchor1._x,anchor1._y);//Первая точка — текущая
    for(i=2;i<=maxPoints;i++) {
        lineStyle(5,0x0000FF,100);
        /*Привязываем координаты опорных и управляемых точек к координатам
изображающих их клипов*/
        curveTo(this["control"+i]._x,this["control"+i]._y,this["anchor"+i]._x,
this["anchor"+i]._y);
```

```
lineStyle(0,0x000000,100);  
//Рисование управляемых линий  
moveTo(this["control"+i]._x,this["control"+i]._y);  
lineTo(this["anchor"+i]._x,this["anchor"+i]._y);  
}  
}
```

Использование динамических масок

Язык сценариев ActionScript предоставляет возможность создания масок на основе экземпляра муви-клипа. Для создания динамической маски используется метод класса MovieClip `setMask()`. Клип, используемый в качестве маски, может содержать любое число кадров и слоев, кроме того, клип может содержать формы, нарисованные при помощи методов динамического рисования. Метод `setMask()` может быть вызван следующим образом:

```
maskedClip.setMask(mask)
```

Здесь:

- `maskedClip` — имя экземпляра маскируемого клипа;
- `mask` — имя экземпляра клипа, используемого в качестве маски.

В качестве упражнения в использовании динамической маски можно выполнить следующие действия:

1. В новом документе создать символ типа **Movie Clip** и поместить в первый кадр его монтажной линейки сценарий, представленный в листинге 21.18. После этого вытащить экземпляр символа на сцену и присвоить ему имя (Instance Name) `mask`. Данный клип будет использоваться в качестве маски.
2. Создать еще один символ и на его монтажную линейку поместить любое изображение или анимацию. Вытащить экземпляр этого символа на сцену и присвоить ему имя `masked`. Данный клип будет использоваться в качестве маскируемого.

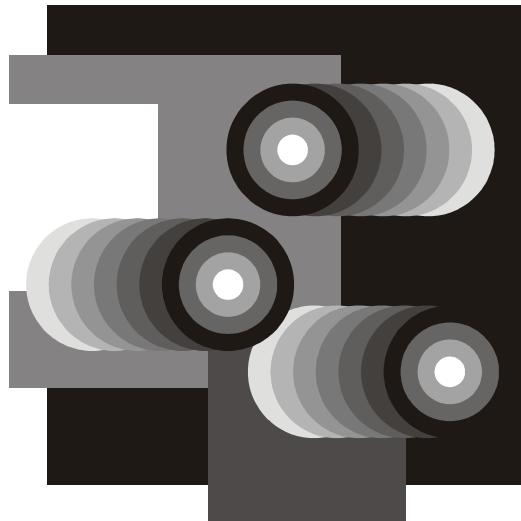
3. В первый кадр основной монтажной линейки поместить следующий сценарий:

```
masked.setMask(mask);
```

4. Просмотреть результат в среде тестирования.

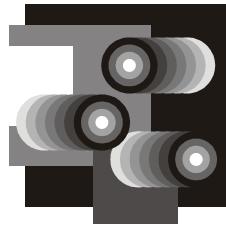
При необходимости отключения программной маски нужно передать методу `setMask()` в качестве аргумента значение `null`, например:

```
masked.setMask(null)
```



ЧАСТЬ V

ПРИЛОЖЕНИЯ



Приложение 1

Горячие клавиши и команды главного меню Flash MX 2004

Операция	Горячие клавиши	Примечание
File>New	<Ctrl>+<N>	Создать новый файл
File>Open	<Ctrl>+<O>	Открыть существующий файл
File>Open Recent		Открыть файл, с которым недавно работали
File>Close	<Ctrl>+<W>	Закрыть текущий файл
File>Close All		Закрыть все файлы
File>Save	<Ctrl>+<S>	Сохранить текущий файл
File>Save and Compact		Компактное сохранение файла
File>Save as	<Ctrl>+<Shift>+<S>	Сохранить файл с новым именем
File>Save as Template		Сохранить файл как шаблон
File>Save All		Сохранить все файлы
File>Revert		Восстановить файл в состояние начала работы
File>Import>Import to Stage	<Ctrl>+<R>	Импортировать файлы, созданные в других приложениях
File>Import>Import to Library		Импортировать файл в библиотеку
File>Import>Open External Library	<Ctrl>+<Shift>+<O>	Открыть библиотеку другого документа
File>Export>Export Image		Экспортировать кадр в выбранный формат
File>Export>Export Movie	<Ctrl>+<Alt>+<Shift>+<S>	Экспортировать ролик как фильм или последовательность кадров

(продолжение)

Операция	Горячие клавиши	Примечание
File>Publish Setting	<Ctrl>+<Shift>+<F12>	Настроить параметры публикации
File>Publish Preview	<F12>	Публикация с просмотром результата
File>Publish	<Shift>+<F12>	Публиковать документ во все указанные форматы
File>Page Setup		Настроить параметры печати страницы
File>Print	<Ctrl>+<P>	Печать кадра
File>Send		Отправить документ по электронной почте
File>Exit	<Ctrl>+<Q>	Выход из программы
Edit>Undo	<Ctrl>+<Z>	Отменить последнюю команду
Edit>Redo	<Ctrl>+<Y>	Восстановить последнюю отмененную команду
Edit>Cut	<Ctrl>+<X>	Вырезать в буфер обмена
Edit>Copy	<Ctrl>+<C>	Скопировать в буфер обмена
Edit>Paste in Center	<Ctrl>+<V>	Вставить из буфера обмена в центр рабочей области
Edit>Paste in Place	<Ctrl>+<Shift>+<V>	Вставить из буфера, сохраняя прежние координаты
Edit>Paste Special		Вставка объекта в специальном виде, как фильм SWF, как растровое изображение, как текст из буфера в формате *.doc (Word), *.txt (ASCII)
Edit>Clear	<Backspace>	Очистить выделение
Edit>Duplicate	<Ctrl>+<D>	Дублировать выделенный объект
Edit>Select All	<Ctrl>+<A>	Выделить все объекты в сцене
Edit>Find and Replace	<Ctrl>+<F>	Найти и заменить
Edit>Find Next	<F3>	Найти далее
Edit>Timeline>Cut Frames	<Ctrl>+<Alt>+<X>	Вырезать кадры в буфер обмена
Edit>Timeline>Copy Frames	<Ctrl>+<Alt>+<C>	Копировать кадры в буфер обмена
Edit>Timeline>Paste Frames	<Ctrl>+<Alt>+<V>	Вставить кадры из буфера обмена

(продолжение)

Операция	Горячие клавиши	Примечание
Edit>Timeline>Remove Frames	<Alt>+<Backspace>	Удалить кадры
Edit>Timeline>Clear Frames	<Shift>+<F5>	Очистить содержимое кадров
Edit>Timeline>Select All Frames	<Ctrl>+<Alt>+<A>	Выделить все кадры в сцене
Edit>Edit Symbols/Edit Document	<Ctrl>+<E>	Редактировать выделенный символ или вернуться на сцену из режима редактирования символа
Edit>Edit Selected		Редактировать группу или символ выделенного объекта
Edit>Edit in Place		Редактировать группу или символ прямо на сцене
Edit>Edit All		Вернуться к редактированию фильма (из режима редактирования группы или символа в контексте фильма)
Edit>Preferences	<Ctrl>+<U>	Настройки рабочей среды фильма
Edit>Customize Tools Panels		Настроить панель инструментов
Edit>Keyboard Shortcuts		Настроить горячие клавиши программы
Edit>Font Mapping		Замена отсутствующего в системе шрифта, используемого в документе, на альтернативный из списка
View>Go to>First	<Home>	Переход к первой сцене фильма
View>Go to>Previous	<Page Up>	Переход к предыдущей сцене
View>Go to>Next	<Page Down>	Переход к следующей сцене
View>Go to>Last	<End>	Переход к последней сцене
View>Go to>имя сцены		Переход к поименованной сцене фильма
View>Zoom In	<Ctrl>+<=>	Увеличение масштаба отображения в 2 раза
View>Zoom Out	<Ctrl>+<->	Уменьшение масштаба отображения в 2 раза
View>Magnification>Fit in Window		Масштаб по размеру окна без полос прокрутки

(продолжение)

Операция	Горячие клавиши	Примечание
View>Magnification>100%	<Ctrl>+<1>	Выбор масштаба отображения от 25 до 800%
View>Magnification>400%	<Ctrl>+<4>	Выбор масштаба отображения 400%
View>Magnification>800%	<Ctrl>+<8>	Выбор масштаба отображения 800%
View>Magnification>Show Frame	<Ctrl>+<2>	Установка масштаба, при котором рабочая область целиком отображается в окне редактирования
View>Magnification>Show All	<Ctrl>+<3>	Установка масштаба, при котором все объекты сцены отображаются в окне редактирования
View>Preview Mode>Outlines	<Ctrl>+<Alt>+<Shift>+<O>	Режим просмотра в контурах
View>Preview Mode>Fast	<Ctrl>+<Alt>+<Shift>+<F>	Отключение сглаживания для всех объектов сцены
View>Preview Mode>Antialias	<Ctrl>+<Alt>+<Shift>+<A>	Отключение сглаживания текста
View>Preview Mode>Antialias Text	<Ctrl>+<Alt>+<Shift>+<T>	Режим сглаживания графики и текста
View>Preview Mode>Full		Полное сглаживание всех объектов
View>Work Area	<Ctrl>+<Shift>+<W>	Отображение рабочей и вспомогательной области. Отключение данного режима приводит к запрету использования вспомогательной области
View>Rulers	<Ctrl>+<Alt>+<Shift>+<R>	Показать линейки
View>Grid>Show Grid	<Ctrl>+<'>	Показать сетку
View>Grid>Edit Grid	<Ctrl>+<Alt>+<G>	Редактировать параметры сетки
View>Guides>Show Guides	<Ctrl>+<;>	Показать направляющие
View>Guides>Lock Guides	<Ctrl>+<Alt>+<;>	Фиксировать направляющие
View>Guides>Edit Guides	<Ctrl>+<Alt>+<Shift>+<;>	Редактировать параметры направляющих

(продолжение)

Операция	Горячие клавиши	Примечание
View>Guides>Clear		Удалить направляющие
View>Snapping> Edit Snap Align		Настройка параметров режима притягивания с выравниванием
View>Snapping> Snap Align		Режим притягивания с выравниванием
View>Snapping> Snap to Grid	<Ctrl>+<Shift>+ +<'>	Притягивать объекты к сетке
View>Snapping> Snap to Guides	<Ctrl>+<Shift>+ +<;>	Притягивать объекты к направляющим
View>Snapping> Snap to Pixels		Притягивать объекты к пиксельной сетке
View>Snapping> Snap to Object	<Ctrl>+<Shift>+ +</>	Притягивать объекты
View>Hide Edges	<Ctrl>+<H>	Скрыть/показать края выделенных объектов
View>Show Shape Hints	<Ctrl>+<Alt>+ <H>	Показать метки подсказки
Insert>New Symbol	<Ctrl>+<F8>	Создать новый символ
Insert>Timeline>Layer		Создать новый слой
Insert>Timeline> Layer Folder		Создать новую папку слоев на монтажной линейке
Insert>Timeline> Motion Guide		Создать слой пути
Insert>Timeline>Frame	<F5>	Создать новый обычный кадр
Insert>Timeline> Keyframe		Создать новый ключевой кадр
Insert>Timeline> Blank Keyframe		Создать пустой ключевой кадр
Insert>Timeline> Create Motion Tween		Создать анимацию движения
Insert>TimelineEffect> Assistans>Copy to Grid		Применить эффекты монтажной линейки
Insert>TimelineEffect> Assistans>Distributes Duplicate		

(продолжение)

Операция	Горячие клавиши	Примечание
Insert>Timeline Effect>Effects>Blur		
Insert>TimelineEffect>Effects>Drop Shadow		
Insert>Timeline Effect>Effects>Expand		
Insert>Timeline Effect>Effects>Explode		
Insert>Timeline Effect>Transform		
Insert>Timeline Effect>Transition		
Insert>Scene		Создать новую сцену
Modify>Document	<Ctrl>+<J>	Редактировать свойства (базовые настройки) фильма
Modify>Convert to Symbol	<F8>	Создать символ из выделенного на сцене объекта
Modify>Break Apart	<Ctrl>+	Разбить выделенный объект
Modify>Bitmap>Swap Bitmap		Заменить один экземпляр растрового изображения другим
Modify>Bitmap>Trace Bitmap		Трассировать выделенное растровое изображение
Modify>Symbol>Swap Symbol		Заменить символ
Modify>Symbol>Duplicate Symbol		Дублировать символ
Modify>Shape>Smooth		Сглаживание кривых
Modify>Shape>Straighten		Спрямление кривых
Modify>Shape>Optimize	<Ctrl>+<Alt>+<Shift>+<C>	Оптимизация кривых
Modify>Shape>Convert Lines to Fills		Преобразовать линию в форму
Modify>Shape>Expand Fills		Расширить/сжать форму по размеру
Modify>Shape>Softens Fills Edges		Смягчить границы форм (эффект растушевки)

(продолжение)

Операция	Горячие клавиши	Примечание
Modify>Shape>Add Shape Hint	<Ctrl>+<H>	Добавить метки подсказки при анимации формы
Modify>Shape>Remove All Hints		Удалить все метки подсказки
Modify>Timeline>Distribute to Layers	<Ctrl>+<Shift>+<D>	Распределить выделенные объекты по слоям
Modify>Timeline>Layer Properties		Редактировать параметры текущего слоя
Modify>Timeline>Reverse Frames		Разместить выделенные кадры в обратном порядке
Modify>Timeline>Synchronize Symbols		Синхронизировать анимацию символов в выделенных кадрах
Modify>Timeline>Convert to Keyframes	<F6>	Преобразовать выделенные кадры в ключевые
Modify>Timeline>Clear Keyframes	<Shift>+<F6>	Очистить выделенные ключевые кадры
Modify>Timeline>Convert to Blank Keyframes	<F7>	Преобразовать выделенные кадры в пустые ключевые
Modify>Timeline Effects>Edit Effect		Редактировать эффект монтажной линейки
Modify>Timeline Effects>Remove Effect		Удалить эффект монтажной линейки
Modify>Transform>Free Transform		Преобразовать форму инструментом Free Transform (Свободное трансформирование)
Modify>Transform>Distort		Преобразовать форму с модификатором Distort (Перспективное искажение)
Modify>Transform>Envelope		Преобразовать форму с модификатором Envelope (Огибающие)
Modify>Transform>Scale		Масштабировать объект
Modify>Transform>Rotate and Skew		Поворот и скос объекта
Modify>Transform>Rotate 90 CW	<Ctrl>+<Shift>+<9>	Поворот влево объекта на 90 градусов

(продолжение)

Операция	Горячие клавиши	Примечание
Modify>Transform> Rotate 90 CCW	<Ctrl>+<Shift>+ +<7>	Поворот вправо на 90 градусов
Modify>Transform> Flip Vertical		Зеркальное отображение по вертикали
Modify>Transform> Flip Horizontal		Зеркальное отображение по горизонтали
Modify>Transform> Remove Transform	<Ctrl>+<Shift>+ +<Z>	Отменить преобразования
Монтаж объектов в стопке	<Ctrl>+<Shift>+ +<Up>	Переместить объект на передний план
Modify>Arrange> Bring to Front		
Modify>Arrange> Bring Forward	<Ctrl>+<Up>	Переместить объект на одну позицию в стопке вперед
Modify>Arrange> Send Backward	<Ctrl>+<Down>	Переместить объект назад на одну позицию
Modify>Arrange> Send to Back	<Ctrl>+<Shift>+ +<Down>	Переместить объект на задний план, делая его самым нижним в стопке
Modify>Arrange>Lock	<Ctrl>+<Alt>+ +<L>	Заблокировать выделенный объект (нельзя редактировать и выделять)
Modify>Arrange> Unlock All	<Ctrl>+<Alt>+ +<Shift>+<L>	Отменить блокировку всех объектов
Modify>Align		Выровнять выделенные объекты:
Modify>Align>Left	<Ctrl>+<Alt>+ +<1>	По левому краю
Modify>Align>Horizontal Center	<Ctrl>+<Alt>+ +<2>	По центру по горизонтали
Modify>Align>Right	<Ctrl>+<Alt>+ +<3>	По правому краю
Modify>Align>Top	<Ctrl>+<Alt>+ +<4>	По верхнему краю
Modify>Align>Vertical Center	<Ctrl>+<Alt>+ +<5>	По центру по вертикали
Modify>Align>Bottom	<Ctrl>+<Alt>+ +<6>	По нижнему краю
Modify>Align>Distribute Widths	<Ctrl>+<Alt>+ +<7>	Распределить по ширине

(продолжение)

Операция	Горячие клавиши	Примечание
Modify>Align>Distribute Heights	<Ctrl>+<Alt>+<9>	Распределить по высоте
Modify>Align>Make Same Width	<Ctrl>+<Alt>+<Shift>+<7>	Сделать равными по ширине
Modify>Align>Make Same Height	<Ctrl>+<Alt>+<Shift>+<9>	Сделать равными по высоте
Modify>Align>To Stage	<Ctrl>+<Alt>+<8>	Выравнивание по рабочей сцене
Modify>Group	<Ctrl>+<G>	Сгруппировать выделенные объекты
Modify>Ungroup	<Ctrl>+<Shift>+<G>	Разгруппировать
Text>Font		Выбор шрифта текста
Text>Size		Выбор размера шрифта
Text>Style>Plain	<Ctrl>+<Shift>+<P>	Обычное начертание
Text>Style>Bold	<Ctrl>+<Shift>+	Жирное начертание
Text>Style>Italic	<Ctrl>+<Shift>+<I>	<i>Курсивное начертание</i>
Text>Style>Superscript		Превращает выделенный текст в верхний индекс
Text>Style>Subscript		Превращает выделенный текст в нижний индекс
Text>Align>Align Left	<Ctrl>+<Shift>+<L>	Выравнивание абзаца по левому краю
Text>Align>Align Center	<Ctrl>+<Shift>+<C>	Выравнивание абзаца по центру
Text>Align>Align Right	<Ctrl>+<Shift>+<R>	Выравнивание абзаца по правому краю
Text>Align>Justify	<Ctrl>+<Shift>+<J>	Выравнивание абзаца по формату
Text>Tracking>Increase	<Ctrl>+<Alt>+<Right>	Увеличить межбуквенный интервал на 0,5 пункта
Text>Tracking>Decrease	<Ctrl>+<Alt>+<Left>	Уменьшить межбуквенный интервал на 0,5 пункта

(продолжение)

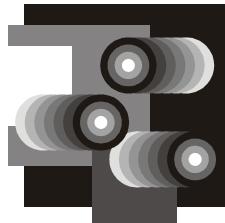
Операция	Горячие клавиши	Примечание
Text>Tracking>Reset	<Ctrl>+<Alt>+<Up>	Восстановить стандартный интервал
Text>Scrollable		Установка режима прокрутки для динамического или пользовательского текста
Text>Check Spelling		Проверка орфографии
Text>Spelling Setup		Настройка параметров проверки орфографии
Commands>Manage Saved Commands		Изменить набор команд
Commands>Get More Commands		Загрузить команды из сети Интернет
Commands>Run Commands		Выполнить набор команд
Commands>No Commands Found		Отсутствие команд
Control>Play	<Enter>	Воспроизвести/остановить текущий ролик
Control>Rewind	<Ctrl>+<Alt>+<R>	Переход в начало фильма
Control>Go to End		Переход в конец фильма
Control>Step Forward One Frame	<,>	Переход к следующему кадру
Control>Step Backward One Frame	<,>	Переход к предыдущему кадру
Control>Test Movie	<Ctrl>+<Enter>	Режим тестирования ролика (предварительный просмотр фильма). Выполнение данной команды приводит к переходу в среду тестирования, открывающуюся в собственном окне. Для выхода из среды тестирования необходимо закрыть ее окно (<Ctrl>+<Q>)
Control>Debug Movie	<Ctrl>+<Shift>+<Enter>	Вызов отладчика фильма Debugger
Control>Test Scene	<Ctrl>+<Alt>+<Enter>	Режим тестирования сцены (предварительный просмотр текущей сцены или объекта в среде тестирования)

(продолжение)

Операция	Горячие клавиши	Примечание
Control>Loop Playback		Зациклить воспроизведение
Control>Play All Scenes		Воспроизвести все сцены
Control>Enable Simple Frame Actions		Разрешить выполнение простых сценариев при воспроизведении фильма в рабочей среде
Control>Enable Simple Buttons	<Ctrl>+<Alt>+	Разрешить активное поведение кнопок в редакторе
Control>Enable Live Preview		Включить режим Live Preview при работе с компонентами в рабочей среде
Control>Mute Sounds		Запретить воспроизведение звуков в фильме
Window>New Window	<Ctrl>+<Alt>+<K>	Открыть новое окно с новым файлом *.FLA
Window>Toolbars>Main		Показать/скрыть основную палитру инструментов
Window>Toolbars>Controller		Показать палитру проигрывателя ролика
Window>Toolbars>Edit Bar		Показать/скрыть строку состояния
Window>Properties	<Ctrl>+<F3>	Показать/скрыть инспектор свойств
Window>Timelines	<Ctrl>+<Alt>+<T>	Показать/скрыть монтажную линейку
Window>Tools	<Ctrl>+<F2>	Показать/скрыть панель инструментов
Window>Library	<F11>,<Ctrl>+<L>	Показать библиотеку фильма
Window>Design Panels>Align	<Ctrl>+<K>	Показать/скрыть панель выравнивания
Window>Design Panels>Color Mixer	<Shift>+<F9>	Показать/скрыть панель синтеза цветов
Window>Design Panels>Color Swatches	<Ctrl>+<F9>	Показать/скрыть панель каталога цветов
Window>Design Panels>Info	<Ctrl>+<I>	Показать/скрыть панель информации
Window>Design Panels>Scene	<Shift>+<F2>	Показать/скрыть панель сцен

(окончание)

Операция	Горячие клавиши	Примечание
Window>Design Panels>Transform	<Ctrl>+<T>	Показать/скрыть панель трансформирования
Window>Development Panels>Actions	<F9>	Показать/скрыть панель редактора ActionScript
Window>Development Panels>Behaviors	<Shift>+<F3>	Показать/скрыть панель Behavior
Window>Development Panels>Components	<Ctrl>+<F7>	Показать/скрыть панель Components
Window>Development Panels>Component Inspector	<Alt>+<F7>	Показать/скрыть панель Component Inspector
Window>Development Panels>Debugger	<Shift>+<F4>	Показать/скрыть панель отладки команд
Window>Development Panels>Output	<F2>	Показать окно вывода
Window>Others Panels>Accessibility	<Alt>+<F2>	Показать/скрыть панель Accessibility
Window>Others Panels>History	<Ctrl>+<10>	Показать/скрыть панель History
Window>Others Panels>Movie Explorer	<Alt>+<F3>	Показать/скрыть панель Проводника
Window>Others Panels>Strings	<Ctrl>+<11>	Показать/скрыть панель Strings
Window>Others Panels>Common Libraries		Показать стандартные библиотеки программы
Window>Hide Panels	<F4>	Показать/скрыть все панели на экране
Window>Panels Sets		Показать/скрыть специально настроенный набор перемещаемых панелей
Window>Save Panel Layout		Сохранение расположения панелей под именем
Window>Cascade		Расположить окна каскадом
Window>Tile		Расположить окна в виде мозаики
Window>список файлов		Перейти к файлу из списка открытых файлов



Приложение 2

Интернет-ресурсы, посвященные Flash

Ресурс	Примечание
www.macromedia.com/support/flash	Содержит регулярно обновляемую документацию в формате PDF, базу данных технических замечаний для разработчиков Flash-проектов и форум, посвященный различным аспектам применения Flash
www.flashkit.com	Крупнейший источник информации по Flash. На сайте представлено все, что может иметь отношение к Flash (примеры, учебники, новости, статьи, чат, галерея и др.)
www.ultrashock.com	Портал для разработчиков Flash. Содержит большое количество учебных примеров и файлов .fla , сосредоточенных на ActionScript
www.flasher.ru	Русскоязычный портал, содержащий большой учебный материал по различным аспектам работы с Flash: статьи, исходные модули, форум и др.
www.flash-ripper.com	Русскоязычный ресурс, предназначен для разработчиков Flash-проектов. Содержит статьи по разным аспектам языка сценариев ActionScript, а также ссылки на другие ресурсы, посвященные Flash
www.swifftools.com	Содержит различные вспомогательные программы, расширяющие возможности Flash
www.multikov.net	Русскоязычный ресурс, целиком посвященный Flash-анимации. Содержит множество образцов работ и учебных материалов по созданию двумерной анимации

Предметный указатель

А

Адресация:

- абсолютная 484
- относительная 485

Анимация:

- автоматическая (Tweened animation) 301
- вращения 310, 319
- встроенные эффекты 337
- градиента 305
- движения (Motion tweening) 309
- по замкнутой траектории 315
- по маршруту 312
- покадровая (Frame by frame) 235
- программная (Program animation) 482
- программная (Program animation) 508, 516, 559, 585
- растровой графики 328
- с использованием вложенных символов 323
- с использованием меток подсказки (Shape hints) 307
- с применением маски 329
- текста 328
- ускорение/замедление 304
- устранение краевого эффекта 320
- фонового изображения 321
- формы (Shape Tweening) 302
- циклическая 319

Б

Библиотека:

- интерфейс 267
- контекстное меню 271
- общая (Shared Library) 199

общая библиотека (Shared Library) 273

стандартные библиотеки (Common Libraries) 273

В

Векторная графика:

- импорт 117
- оптимизация 112
- форматы файлов 118, 426
- экспорт 427

Видео:

- замена (Swap) 169
- импорт 163
- настройка свойств 170
- поиск и замена 374
- публикация 423
- редактирование 163
- сжатие 162, 164
- форматы файлов 162, 426
- экспорт 429

Вспомогательные элементы интерфейса:

- Grid (Сетка) 102
- Guides (Направляющие) 103
- Pixel Grid (Пиксельная сетка) 103
- Rulers (Линейки) 102

Г

Генератор случайных чисел 558

Группа:

- вложенная (nested) 92
- разгруппирование 93
- редактирование 92
- создание 92

Д

Динамическая загрузка изображений и документов 573
Динамическая маска 591
Динамическое рисование 583, 587
Динамическое создание экземпляров 548
Документ:
 отчет о размерах фильма 398
 печать 430
 поиск и замена элементов 369, 377
 публикация 401
 размещение в Интернете 407, 433
 создание 23
 сохранение 25
 тестирование 389, 390
 экспорт 426
 эмуляция загрузки 392
Дублирование экземпляров 551

З

Заливка:
 градиентная (Gradient) 133
 растровая (Bitmap) 154
 расширение/сжатие 115
 редактирование 66, 137, 138, 156
 смягчение краев 115
 создание 69, 72, 584
 сплошная (Solid) 131
 фиксация 139
Звук:
 встроенные эффекты 352
 добавление в кадр 348
 импорт 348
 использование 360
 оптимизация 355
 параметры 346
 поиск и замена 374
 редактирование 353
 тип синхронизации:
 Event 349
 Start 350
 Stop 351

Stream 351
форматы файлов 347, 426
экспорт 362, 430

И

Иерархия объектов:
 монтаж в стопке 93
 наложенный уровень 89
 рабочий уровень 88
 размещение в стеке 549, 574

Импорт:
 видео 163, 168
 графики векторной 117
 графики растровой 143
 звучка 348
 поддерживаемые форматы 118, 142,
 162, 347
 последовательности изображе-
 ний 118
 символов 265
 текста 174
 цветовой палитры 130

Инспектор свойств:
 анимация 303, 309
 документ 51
 звук 348
 кадр 234
 текст 176, 181, 185
 экземпляр символа 255

Инструмент:
 Brush (Кисть) 69
 Eraser (Ластик) 73
 Eyedropper (Пипетка) 73
 Fill Transform (Трансформация
 Заливки) 136
 Free Transform (Свободное
 трансформирование) 74, 79
 Ink Bottle (Чернильница) 71
 Lasso (Лассо) 67
 Line (Линия) 59
 Oval (Овал) 61
 Paint Bucket (Ведро Заливки) 72, 135
 Pen (Перо) 107
 Pencil (Карандаш) 68

Polystar (Многоугольник) 62
 Rectangle (Прямоугольник) 61, 62
 Selection (Выделение) 62
 Subselection (Частичное выделение) 111
 Text (Текст) 172
 Zoom (Лупа) 37

Интерфейс:

- вспомогательная область 28
- главное меню 26
- контроллер (Controller) 35
- полоса редактирования (Edit Bar) 28
- рабочая область 28
- среда тестирования:
 - график покадровый (Frame by Frame) 398
 - график потоковый (Streaming Graph) 396
 - команды меню 391
 - стандартная панель (Main) 27

K

Кадр:

- вставка 230
- выделение интервальное 228
- выделение покадровое 227
- комментарий (comment) 234
- копирование 229
- метка (label) 234
- перемещение 228
- преобразование типа 230
- реверсирование 231
- создание 223, 224, 225
- тип:
 - кадры трансформации 225
 - ключевой (Keyframe) 222, 225
 - простой (Frame) 223, 225
- удаление 231
- якорь (anchor) 234

Класс:

Array:

- методы 526
- свойства 526

Button:
 свойства 478

Date:
 методы 570

Key:
 методы 565
 свойства 567

Math:
 методы 557
 свойства 557

Mouse:
 методы 564

MovieClip:
 свойства 478

MovieClipLoader:
 методы 574

Stage:
 свойства 562

TextField:
 методы 540, 544
 свойства 538

TextFormat:
 свойства 543

Комментарии в коде 453

Контур:

- атрибуты 59
- замыкание 107
- преобразование в заливку 115
- редактирование 66, 110, 111
- создание 107, 108, 583

M

Макрос (Command) 365

Массивы:

- заполнение 525
- многомерные 528
- создание 523, 524

Модификатор:

- Brush Mode (Режим Кисти) 70
- Brush Shape (Форма Кисти) 69
- Brush Size (Размер Кисти) 69
- Distort (Искажение) 78
- Enlarge (Увеличить) 37
- Envelope (Оболочка) 78

Eraser Mode (Режим Стирания) 73
 Eraser Shape (Форма Ластика) 73
 Faucet (Кран) 74
 Gap Size (Величина Зазора) 72
 Lock Fill (Фиксирование заливки) 139
 Magic Wand (Волшебная палочка) 152
 Magic Wand Properties (Свойства Волшебной палочки) 152
 Pencil Mode (Режим Карандаша) 68
 Polygon Mode (Многоугольное лассо) 67
 Reduce (Уменьшить) 37
 Rotate and Skew (Поворот и Скос) 78
 Round Rectangle Radius (Радиус скругления) 61
 Scale (Масштабирование) 78
 Smooth (Сглаживание) 112
 Snap to Objects (Притягивание к объектам) 66
 Straighten (Спрямление) 112
 Use Pressure (Нажим) 70
 Use Tilt (Наклон) 70
 Монтажная линейка:
 контекстное меню 221
 структура 208
 Монтажная линейка Timeline:
 встроенные эффекты 337

H

Навигация:
 при помощи динамической загрузки документов 573
 при помощи монтажной линейки 469
 с использованием гиперссылок 474
 Настройка:
 глобальных параметров фильма 51
 горячих клавиш 49
 масштаба просмотра 37
 рабочей среды (Preferences) 40

O

Обработчик событий:
 on() 463, 466
 onClipEvent() 481
 onResize() 563
 использование в качестве метода 530
 область видимости 466, 467, 531
 объекта Listener 533, 563, 564
 объекта TextField 541
 Операторы:
 арифметические 500
 логические 507
 сравнения 501
 Операции с объектами:
 вставка 65
 выделение 63, 111
 выравнивание 95
 вырезание 65
 копирование 65
 перемещение 64
 приведение размеров 98
 распределение 97
 удаление 66
 установка равных промежутков 99

П

Палитра (служебная панель):
 Actions 454
 Align (Выравнивание) 94
 Behaviors (Поведения) 489
 Color Mixer (Синтез цвета) 126
 Color Swatches (Каталог цветов) 129
 Color (Системная палитра) 128
 History (Истории) 35, 365
 Info 83
 Library (Библиотека) 35, 143
 Movie Explorer (Проводник) 375
 Output 462
 Scene 364
 Strings 193
 Tools (Инструменты) 31
 Transform 82
 закрепление 29
 Инспектор свойств (Properties) 33
 слоев 208

Переменные:

- адресация 496
- глобальные 497
- изменение значения 500
- локальные 521
- область видимости 495
- создание 493
- строгий контроль типов данных 493

Перетаскивание экземпляров 546

Поведения (Behaviors) 490

Подсказки по коду 456, 494

предзагрузчик 577

Предложение:

- `else` 509
- `if` 508
- `return` 519
- `with` 487

Публикация:

- документа 401
- настройка параметров формата:
 - Flash (SWF) 404
 - GIF 416
 - HTML 408
 - JPEG 420
 - PNG 421
 - Projector (EXE) 423
 - Quick Time (MOV) 424
- определение наличия Flash Player 410
- переменные шаблона 443
- профиль 402
- стандартные шаблоны HTML 408

P

Растровая графика:

- анимация 328
- глубина цвета 142
- замена (Swap) 149
- импорт 143
- оптимизация 145
- поиск и замена 374
- публикация 416, 420, 421
- разбиение 149
- разрешение 141

растровая заливка 154

- трассировка:
 - автоматическая 158
 - ручная 160
- форматы файлов 142, 426
- экспорт 428

Режим:

- `Snap Align` 100
- `Snap to Objects` 101
- `Snap to Pixels` 103
- калькирования (Onion skin) 238
- множественного редактирования кадров (Edit multiple frames) 239
- отображения содержимого сцены 39
- редактирования:
 - группы 92
 - символа 247
- фиксации заливки (Lock Fill) 139

C

Символ:

- импорт 265
- начало координат 245
- поведение 242
- пустой 556
- редактирование 247
- синхронизация 311
- создание 245, 247, 249, 250
- тестирование 390
- тип:
 - графический (Graphic) 242
 - кнопка (Button) 243
 - муви-клип (Movie Clip) 242
 - шрифтовой (Font Symbol) 199

Слайд-шоу 331, 580

Слежение за курсором мыши 561

Слой:

- группирование 217, 218
- операции 215
- распределение объектов по слоям 211
- режимы 213
- создание 211

тип:

- Ведомый (Guided) 216, 312
- Маска (Mask) 218
- Маскируемый (Masked) 218, 329
- Направляющий (Guide) 216
- Нормальный (Normal) 216
- Слой пути (Motion Guide) 216, 312
- Слой-маска (Mask Layer) 329
- удаление 214

Сцена:

- дублирование 364
- переход между сценами 365, 468
- создание 364
- тестирование 390
- удаление 365

T

Теги:

- !DOCTYPE 435
- allow-access-from 448
- cross-domain-policy 448
- embed:
 - атрибуты 436
- map 445
- meta 435
- object:
 - атрибуты 436
 - параметры 438
 - param 436

Текст:

- анимация 328
- импорт 174
- кодировка 171
- машино-независимые шрифты 186
- многоязыковая поддержка 192
- поиск и замена 371
- разбиение 190
- сглаживание 178, 187

тип:

- динамический (Dynamic) 180, 535
- пользовательский (Input) 184, 535
- статический (Static) 175
- трансформация 188
- шрифтовой символ 199

Текстовый блок:

- динамическое создание 540
- динамическое форматирова-
ние 542
- имя экземпляра (Instance
Name) 182
- переменная 183, 535
- произвольной длины 172
- фиксированной длины 172

Типы данных:

- undefined 494
- логический (Boolean) 506
- строковый (String) 502
- числовой (Number) 499

Трансформация объектов:

- зеркальное отражение (Flip) 76
- искажение (Distort) 77
- масштабирование (Scale) 76
- поворот (Rotate) 76
- при помощи огибающих
(Envelope) 78
- скос (Skew) 77

Ф

Функции встроенные:

- eval() 506
- fscommand() 487
- getURL() 472
- loadMovie() 576
- trace() 462
- безусловного перехода 462
- воспроизведения и останов-
ки 460
- выбора кадра 461
- для работы со строками 504

Функции пользовательские:

- литерал 520
- область видимости 521
- рекурсивные 522
- создание 518
- строгий контроль типов данных 518

Ц

Цвет:

- блок управления цветом (Color) 57
- заливки 66
- контура 66
- модели цвета:
 - HSB 125
 - RGB 123
- поиск и замена 373
- прозрачность 127
- синтез 131
- сохранение 128
- шестнадцатеричное представление (HEX) 124

Цветовая палитра:

- Web 216 безопасная 131
- импорт 130
- экспорт 131

Цикл:

- for 512
- while 511
- вложенный 513
- трехкадровый 516

Э

Экземпляр символа:

- динамическое присоединение 554
- замена (Swap) 263
- имя (Instance Name) 261
- поведение 262
- поиск и замена 374
- разделение 265
- свойства 259, 261
- создание 246, 255
- трансформация 255
- цветовые эффекты (Color Effects) 256

Экспорт:

- видео 429
- звука 362, 430
- изображения 427
- поддерживаемые форматы 426
- последовательности изображений 429
- фильма 428
- цветовой палитры 131