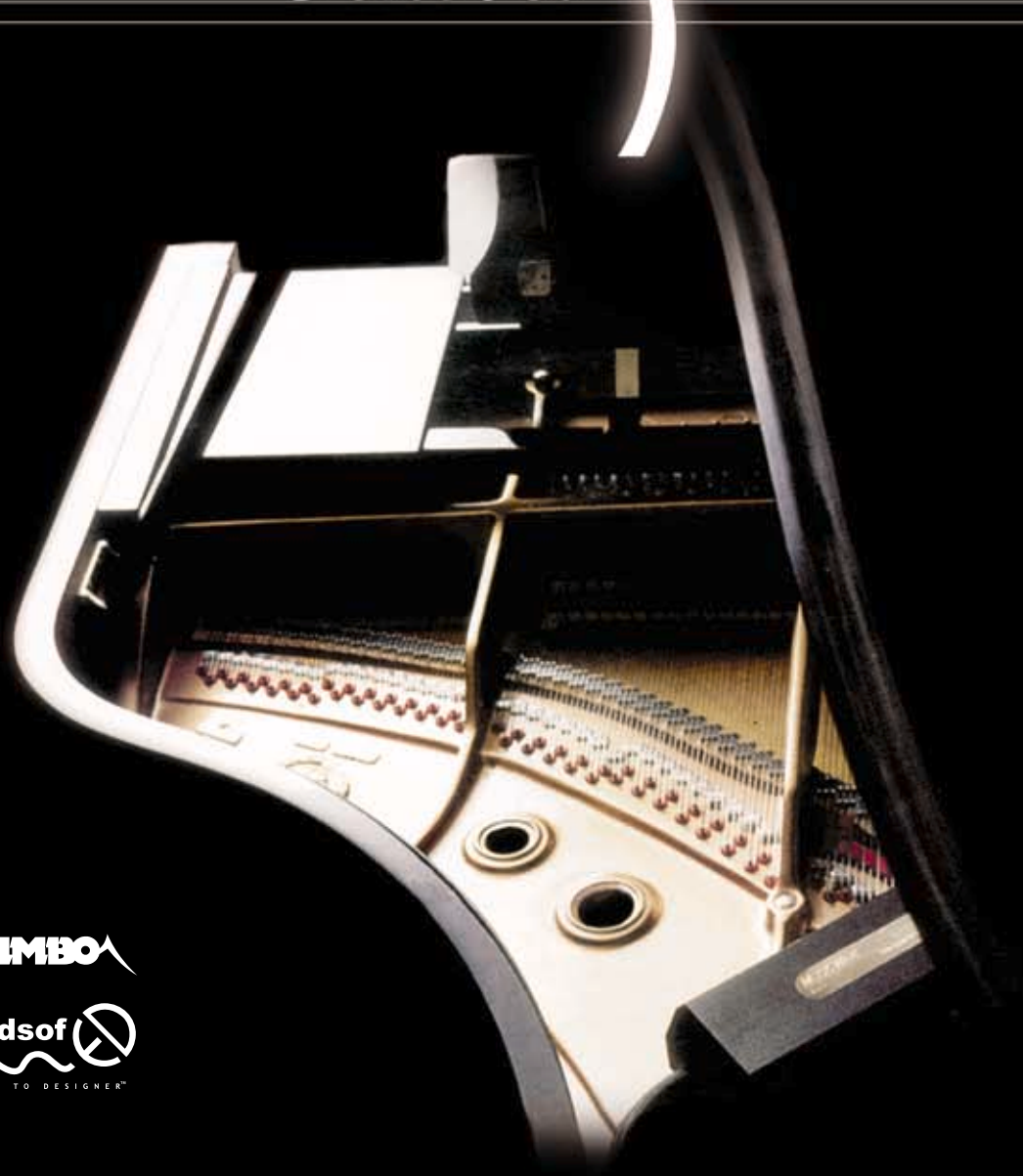


ШЭМ БАНГАЛ

ACTIONSCRIPT

Основы



Foundation ActionScript

Sham Bhangal

ActionScript ОСНОВЫ

Шэм Бангал



*Санкт-Петербург
2002*

Шэм Бангал

ActionScript. Основы

Перевод И. Асеева

| | |
|------------------|---------------------|
| Главный редактор | <i>А. Галунов</i> |
| Зав. редакцией | <i>Н. Макарова</i> |
| Научный редактор | <i>А. Михайлов</i> |
| Редактор | <i>Р. Павлов</i> |
| Художник | <i>С. Борин</i> |
| Корректурa | <i>С. Беляева</i> |
| Верстка | <i>А. Дорошенко</i> |

Бангал III.

ActionScript. Основы. – Пер. с англ. – СПб: Символ-Плюс, 2002. – 480 с., ил.
ISBN 5-93286-031-6

К настоящему времени Flash 5 стал стандартом разработки динамических приложений в Сети, а язык ActionScript из интересной возможности превратился в дисциплину, обязательную при освоении Flash. Внесенные во Flash 5 усовершенствования сделали ActionScript более удобным для дизайнеров за счет множества выпадающих меню и встроенных функций; стандартных, принятых в JavaScript обозначений с точкой, делающих команды легко запоминаемыми; новых способов включения звукового сопровождения и более развитых средств тестирования и отладки. С обеспечиваемой ActionScript интерактивностью веб-дизайн внезапно вышел на новый уровень.

Книга «ActionScript. Основы» написана опытным веб-дизайнером для своих коллег – веб-дизайнеров, уже владеющих основами Flash, но не имеющих опыта в программировании. Понимая, насколько сложно освоить премудрости программирования человеку творческому, далекому от этого занятия, автор последовательно, шаг за шагом знакомит читателя с основными понятиями программирования и их связью с дизайном, после чего основное внимание уделяется применению ActionScript в реальных веб-проектах, обсуждаются планирование больших ActionScript-проектов и динамическая анимация для интерактивных игр. Изучив материал, вы сможете превратить простую линейную анимацию в презентацию с богатыми интерактивными возможностями.

ISBN 5-93286-031-6

ISBN 1-903450-32-2 (англ)

© Издательство Символ-Плюс, 2002

Authorized translation of the English edition © 2000 friends of ED. This translation is published and sold by permission of friends of ED, the owner of all rights to publish and sell the same.

Все права на данное издание защищены Законодательством РФ, включая право на полное или частичное воспроизведение в любой форме. Все товарные знаки или зарегистрированные товарные знаки, упоминаемые в настоящем издании, являются собственностью соответствующих фирм.

Издательство «Символ-Плюс». 193148, Санкт-Петербург, ул. Пинегина, 4,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.

Налоговая льгота – общероссийский классификатор продукции

ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 14.05.2002. Формат 70х100¹/₁₆.

Печать офсетная. Объем 30 печ. л. Тираж 3000 экз. Заказ N

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

Оглавление

| | |
|---|-----|
| Введение | 9 |
| Добро пожаловать | 9 |
| Загружаемые файлы | 12 |
| Что вам необходимо знать | 12 |
| Movie Explorer | 12 |
| Поддержка: мы готовы помочь вам | 15 |
| 1. Начинаем | 17 |
| События и как с ними обращаться | 17 |
| Комментарии | 27 |
| Команды, аргументы и свойства | 29 |
| Инструкции | 31 |
| Синтаксис | 32 |
| Резюме | 45 |
| 2. Строим планы | 46 |
| Планирование | 47 |
| Построение сценариев ActionScript | 52 |
| Веб-сайт stun:design | 62 |
| Резюме | 66 |
| 3. Действия временной диаграммы | 67 |
| Пути временных диаграмм | 68 |
| Добавление основных действий на временную диаграмму | 74 |
| Добавление меток на временную диаграмму | 82 |
| Веб-сайт stun:design | 96 |
| Резюме | 100 |
| 4. Основы интерактивности | 101 |
| Интерактивность и клипы | 109 |
| Создание интерактивности при помощи событий кнопок | 119 |
| События кнопок и навигация | 121 |
| Резюме | 134 |

| | |
|---|------------|
| 5. Трюки с навигацией | 135 |
| Невидимые кнопки | 135 |
| Структура сайта | 140 |
| Интерфейс stun:design | 144 |
| Перетаскиваемые окна | 147 |
| Резюме | 159 |
| 6. Переменные | 161 |
| Переменные | 162 |
| Ввод и вывод | 166 |
| Именованная переменная | 173 |
| Хранение чисел | 175 |
| Логические значения | 178 |
| Объекты, методы и свойства в ActionScript | 181 |
| Резюме | 187 |
| 7. Циклы и принятие решений | 188 |
| Принятие решений | 189 |
| Действие if | 192 |
| Действие else if | 198 |
| Действие else | 202 |
| Циклы | 205 |
| Массивы | 219 |
| Игра в виселицу | 225 |
| Веб-сайт stun:design | 236 |
| Резюме | 241 |
| 8. Звук | 242 |
| Объект sound | 242 |
| ActionScript и стриминг | 259 |
| Веб-сайт stun:design | 260 |
| Резюме | 273 |
| 9. Интерактивность в развитии | 274 |
| И снова об экземплярах | 275 |
| Классы | 276 |
| Определение объектов | 282 |
| Интерактивность и свойства объектов | 283 |
| Галерея stun:design | 297 |
| Веб-сайт stun:design | 305 |
| Создание структуры | 306 |
| Резюме | 320 |

| | |
|---|------------|
| 10. Модульный ActionScript | 321 |
| Модульный ActionScript | 322 |
| Управляющие клипы | 333 |
| Инерционное поведение | 343 |
| Эффект рояния | 347 |
| Настраиваемые клипы | 353 |
| Веб-сайт stun:design | 356 |
| Скрытие страниц | 363 |
| Резюме | 372 |
| 11. Спрайты | 373 |
| Что такое спрайт? | 373 |
| Управление | 375 |
| Движение | 378 |
| Столкновения | 378 |
| Планирование игры stun:zapper | 383 |
| Мир игры (глобальный уровень) | 385 |
| Игрок: объект ship (корабль) и его интерфейсы | 390 |
| Роящийся пришелец: объект alien1 и его интерфейсы | 393 |
| Пуля: объект bullet и его интерфейсы | 398 |
| Веб-сайт stun:design | 401 |
| Страница 1: Заставка | 401 |
| Страница 2: Введение | 406 |
| Страница 3: Основное содержимое | 413 |
| Графика окна и кнопки | 417 |
| Клип содержимого | 423 |
| Управляющий клип окна | 424 |
| Интеграция | 428 |
| Две последние кнопки | 430 |
| Последнее напутствие | 433 |
| Приложение А. Словарь | 435 |
| Приложение В. События | 440 |
| Приложение С. Свойства | 445 |
| Приложение D. Приоритеты операций | 452 |
| Приложение Е. Совместимость Flash 5 | 458 |
| Приложение F. Отладка | 467 |
| Алфавитный указатель | 473 |

Об авторе

В начале карьеры Шэм Бангал работал инженером и специализировался в области промышленных компьютерных систем отображения и управления. Свой досуг он посвящал веб-дизайну, который постепенно занимал все больше и больше времени и вытеснил, в конце концов, инженерную деятельность.

Сейчас Шэм занимается тем, что пишет книги для издательства friends of ED, и это оставляет ему все меньше времени на веб-дизайн... Забавно, как жизнь ходит по кругу!



Шэм живет со своей подругой Карен в Манчестере в Великобритании.

Благодарю Андерса и Джейн за их вклад в проект stun:design. Надеюсь, я не слишком изнурил их работой.

Я изучал веб-дизайн около года, спрятавшись на принадлежащей Роджеру и Энни Вильямс маленькой ферме в глубине графства Сомерсет. Эта книга посвящается вам обоим. Благодарю за то, что вы приютили меня вместе с моей странной привычкой сочетать дневную работу с сетевыми забавами. Прошу прощения за трехсотваттную аудиосистему, которая помогала по ночам бодрствовать мне (и на верное, всей деревне).

Особенная благодарность Роджеру за пиво и жареные трупы животных в жаркие солнечные дни. Для мужчины, у которого подруга вегетарианка, мясо подобно глотку воды в пустыне.

Введение

Добро пожаловать

Интерактивность, которую предоставляет ActionScript для Flash 5, устанавливает новый стандарт в веб-дизайне. ActionScript – это самое значительное усовершенствование Flash 5 по сравнению с Flash 4. Он превратился из простого языка подготовки сценариев в полноценную объектно-ориентированную среду программирования, с улучшенным интерфейсом для ввода и редактирования текста, с новыми командами, обогатившими набор действий, и многими новыми операциями, применяемыми к фильмам. Долгожданные обозначения с точкой позволяют легче запоминать команды, и написание программ стало более удобным по сравнению с Flash 4.

В общем, ActionScript стал теперь значительно более мощным и профессиональным инструментом, который позволяет включать в ваши веб-сайты умопомрачительные эффекты, оставляя размер файлов в разумных пределах, и дает возможность повысить уровень интерактивности ваших разработок.

Сочетание всех этих и многих других усовершенствований предоставляет новые средства и возможности управления, преимуществами которых пользуются лучшие Flash-дизайнеры при создании знакомых вам впечатляющих эффектов. Они не просто следуют правилам, они переопределяют их и учат Flash действовать совершенно по-новому. Это то, что сможете делать вы, прочитав эту книгу.

Мы стремимся дать вам твердые основы, которыми нужно овладеть, чтобы держаться в русле событий. Вы начнете с изучения того, как планировать ActionScript-проект, изучите действия, обеспечивающие простое управление временной диаграммой, совершите полный тур по кодам, допускающим повторное использование, узнаете, как работать со звуком, и создадите свою первую интерактивную игру. ActionScript может показаться какой-то совершенно новой игрой, но эта книга докажет вам, что ее правила не так уж сильно отличаются от старых и что конечный результат оказывается более эффектным, чем все то, что вы до сих пор видели!

Одна оговорка – обучение ActionScript вовсе не запрещает вам оставаться дизайнером. Эта книга открывает перед вами новый путь и ве-

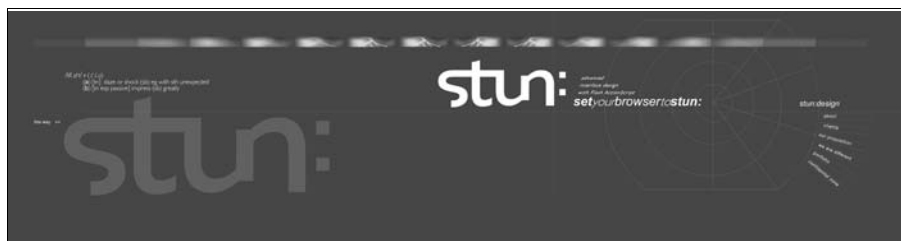
дет вас от линейной анимации к созданию современных динамических Flash веб-сайтов, ни на минуту не упуская из виду, что для веб-дизайнера важнее всего, какие новые средства для реализации его собственных замыслов может дать ActionScript. Мы не хотим сделать из вас программистов, рассуждающих в терминах нулей и единиц (будто больше нет ничего на свете). Идеи остаются самым дорогим товаром, и мы думаем, что облегчить путь от воображения к реальности точно так же важно, как и научить вас всем этим скобкам и жутким символам, в которых легко запутаться, словно в паутине.

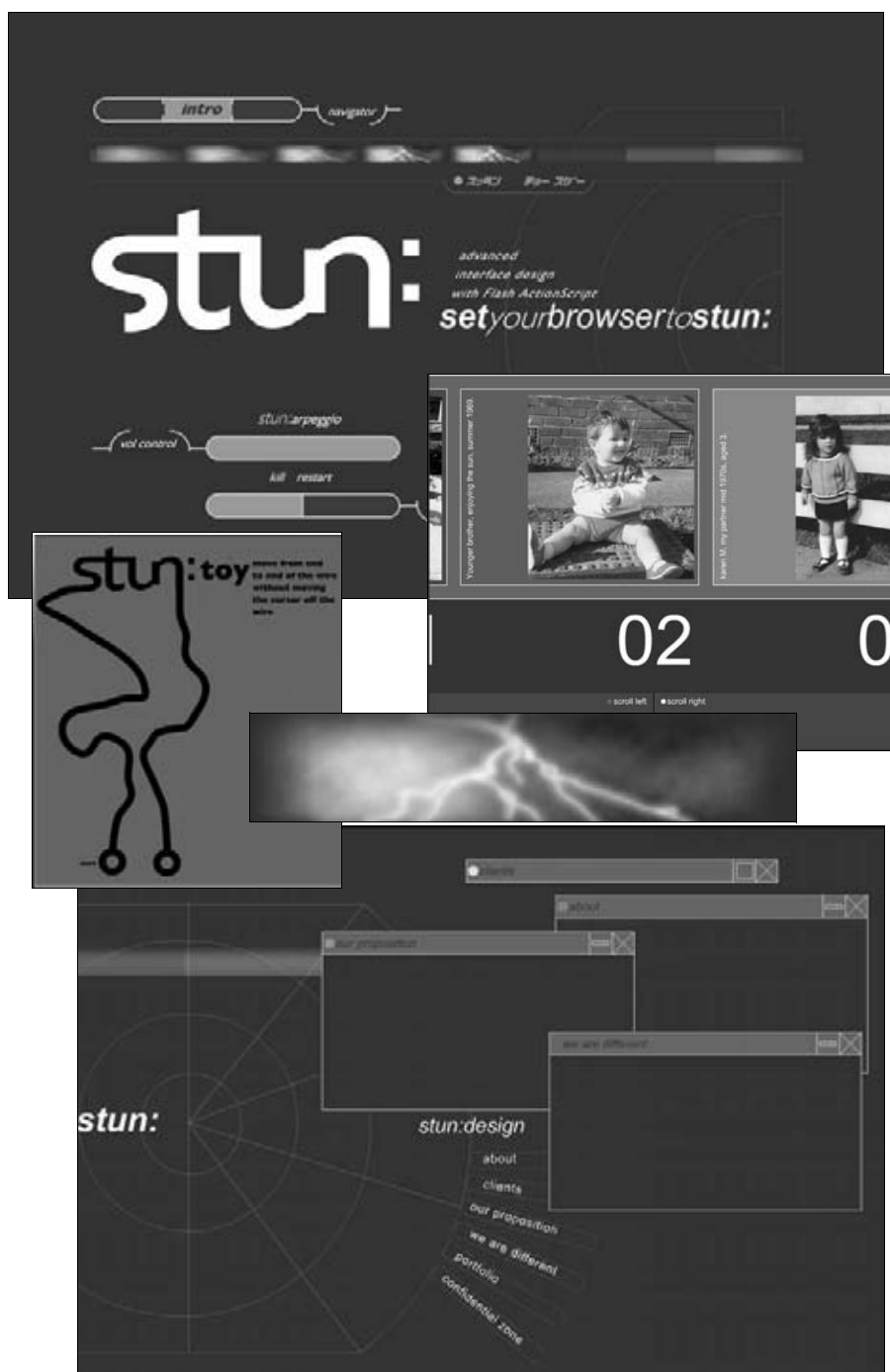
Задачи и философия этой книги

Нельзя не признать, что на 500 страницах невозможно научить всему, что следует знать об ActionScript, но мы собираемся дать вам твердые основы, обеспечивающие глубокое знание ключевых понятий ActionScript. Мы намерены с самого начала книги использовать такие приемы, как обозначения с точкой, которые часто называют «продвинутыми», но которые предлагают простой путь к овладению неизмеримой мощью Flash 5.

Мы не верим тем книгам, где основные принципы кодирования демонстрируются на примитивных примерах, не обладающих коммерческой ценностью и не применимых в реальной жизни. Обучение на теоретических примерах – это не для нас. Мы решили, что всю силу ActionScript, которую вам предстоит применять в своих разработках, вы сможете лучше оценить, если связать изучаемое с **реальным** веб-сайтом. В процессе проработки этой книги мы будем строить **stun:design**, профессиональный веб-сайт, который полностью создан на ActionScript и существует как коммерческий проект. Это поможет закрепить полученные знания путем применения их в ситуации из реальной жизни, где сроки поджимают, клиенты жалуются и все идет не так, как надо.

Делая акцент скорее на практику, нежели на теорию, в каждой главе мы следуем основной модели и вводим новую тему, проиллюстрированную примерами, которые постепенно усложняются и наконец складываются вместе в работающий веб-сайт, полный трюков и хитростей ActionScript. Зайдите на *stundesign.com* – и вы увидите, что будете уметь, прочитав эту книгу.





Загружаемые файлы

Мы также подготовили полностью законченные FLA-файлы для каждого упражнения. Их можно в готовом виде, главу за главой, скачать с *friendsofed.com*. Пользуйтесь ими в собственных разработках – или не пользуйтесь, как вам угодно, но они определенно могут пригодиться для того, чтобы сравнить сделанное вами с законченным продуктом Шэма, когда вдруг окажется, что вы совсем перестали понимать происходящее.

Что вам необходимо знать

Вы взялись за эту книгу, так что мы думаем, что вам знакомы основы Flash. Мы предполагаем, что вы уже создавали обычные фильмы, монтировали кадры на временной диаграмме, но при этом слышали, что Flash 5 дает гораздо больше возможностей для творчества.

Вы, может быть, даже прочли книгу «Foundation Flash 5» издательства «friends of ED», которая дала вам основательные познания во Flash и пробудила желание продолжить путешествие.

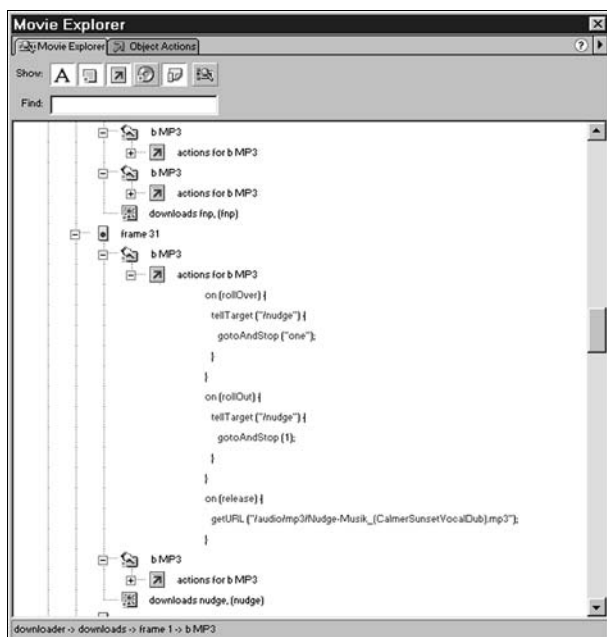
ActionScript для дизайнера – как электродрель, заменившая старый коловорот, который был, конечно, полезен, но слабоват. Как и всякий электроинструмент, получающий энергию по проводам, эта электродрель несет с собой опасность наделать много бед, если с ней неправильно обращаться. Чтобы этого не случилось, прочтите, пожалуйста, следующие краткие инструкции по технике безопасности. Мы соберемся пройти по небольшому списку общих установок Flash 5, которые необходимо сделать перед началом работы, чтобы приступить к созданию сценариев прямо с первой главы.

Работая с Flash, вы научитесь превращать объекты в символы (при помощи клавиши <F8> или Insert ► Convert to Symbol) – хороший прием, позволяющий сократить время загрузки. Если вы вытягиваете два экземпляра символа из библиотеки, ActionScript столкнется с проблемой, как различать их. Поэтому следует снабжать объекты **именами**, чтобы ActionScript знал, где находится каждый экземпляр, и мог ими управлять. Не беспокойтесь, мы покажем, как это делается, когда придет время. Вы умеете давать имена клипам (movieclips), а в этой книге мы встретимся с множеством символов клипов!


Movie Explorer


По мере изучения этой книги ваши FLA будут все сложнее, и может случиться так, что создав фильм или фрагмент сценария, вы затем потеряете его в безграничном пространстве Flash. Эту проблему решит Movie Explorer, с помощью которого легко отыскивать символы, фильмы

и сценарии. В любой момент можно вызвать Movie Explorer при помощи Window ► Movie Explorer.



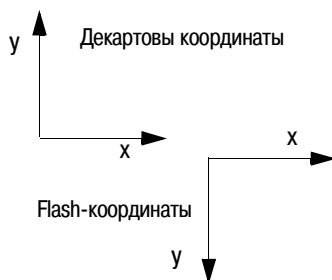
Так выглядит окно Movie Explorer – вашего проводника по просторам Flash

Поиграв немного с Movie Explorer, вы обнаружите, что с его помощью можно сделать несколько вещей. Если распечатать его содержимое, это поможет понять, что происходит, когда мы начнем создавать во Flash необычные конструкции, вроде вкладывания одного фильма в другой. Вы сможете найти все объекты на временной диаграмме, нажав кнопку с двумя квадратами .

Действительно полезно и поле Find:, с помощью которого можно найти все строки ActionScript, управляющие данным объектом. Если вы заметили, что объект ведет себя странно и вопреки ожиданиям, то это позволит разобраться с его поведением. Можно пройти от библиотечного шаблона ко всем экземплярам соответствующего библиотечного символа и проследить, что меняется при изменении библиотечной версии этого символа. Войдите в тексты сценариев ActionScript, нажав на кнопку со стрелкой , и Movie Explorer позволит вам распечатать все коды разом, причем это единственное место во Flash, где можно это сделать. Возможно, вы пока не видите смысла в некоторых из этих функций, но все равно подружитесь с Movie Explorer – в трудную минуту он вас выручит.

Рабочая область

Если вы включите линейки при помощи View ► Rulers, то заметите, что цифровые деления на линейках начинаются с верхнего левого угла, и их значения растут слева направо (вдоль оси X) и от левого верхнего угла вниз (вдоль оси Y). Такие направления осей приняты на печатных прессах и у дизайнеров, но они не совпадают с направлениями, принятыми в геометрии Декарта и у математиков, которые обычно направляют ось Y вверх, так что начало координат находится в левом нижнем углу.



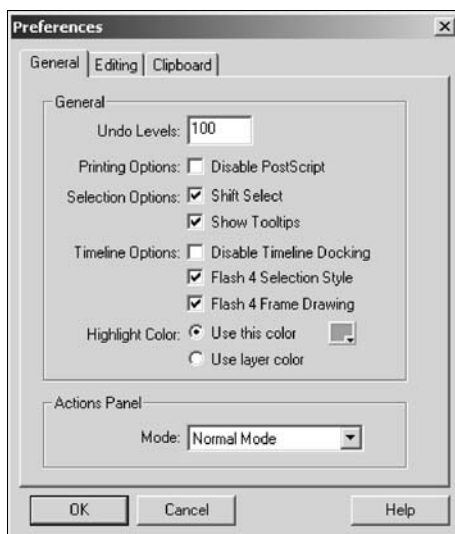
Особенно это может смутить тех, кто имеет математическую подготовку – приходится увеличивать координату Y, чтобы опустить объект. Будьте внимательны!

Установки Flash

Мы собираемся начать программировать прямо с первой главы (не пугайтесь, это гораздо проще, чем вы думаете!), поэтому важно согласовать наши Preferences (общие установки) до начала работы, чтобы Flash у всех нас работал одинаково. Вызовите окно Preferences во Flash, выбрав Edit ► Preferences.

Все установки, которые необходимо сделать, находятся на вкладке General. В группе Selection Options выберите Shift Select. Так вы определите способ множественного выбора объектов во Flash. Во многих графических программах (таких как Adobe PhotoShop) для выбора нескольких объектов или добавления объекта к выделенному набору требуется нажимать клавишу <Shift>. Если выделить что-нибудь при ненажатой клавише <Shift>, все предыдущие выделения теряются. Чтобы заставить Flash работать именно таким образом, следует установить флажок Shift Select. Если вы его сбросите, Flash позволит осуществлять множественный выбор простым щелчком мышью. Затем выделение можно будет сбросить, щелкнув в пустой области. Такой способ действий может вызвать затруднения у тех, кто привык к методу Shift Select, стандартному для большинства графических и 3D-программ.

Также следует обратить внимание на группу Timeline Options. Flash 5 предлагает слегка отличающийся от Flash 4 способ выбора кадров. Методы Flash 5 – это шаг вперед в анимации, основанной на временной диаграмме, но они могут вызывать затруднения при желании выделить отдельные кадры для редактирования ActionScript, так что поправим это. Установите флажки Flash 4 Selection Style и Flash 4 Frame Drawing. Flash 4 Selection Style делает рисование кадров в стиле Flash 5 излишним.



Пока вы находитесь в диалоговом окне Preferences, убедитесь, что установлен флажок Show Tooltips, так как во всплывающих подсказках часто содержится ценная информация. Окно Preferences теперь должно выглядеть примерно так.

Поддержка: мы готовы помочь вам

Если у вас есть вопросы по этой книге или относительно издательства friends of ED, заходите на наш веб-сайт, где можно найти много e-mail-адресов или просто воспользоваться адресом feedback@frindsofed.com.

На этом веб-сайте вы найдете также массу других интересных вещей: интервью со знаменитыми дизайнерами, отрывки из других наших книг, дискуссии, в которых можно принять участие, а можно просто посидеть и посмотреть, о чем говорят между собой другие дизайнеры. Так что если у вас есть замечания или вопросы, пишите нам – именно для этого мы там и находимся. Мы любим читать ваши отклики.

Об оформлении этой книги

Мы как только могли старались сделать эту книгу ясной и легкой для чтения, а поэтому использовали всего несколько стилей:

- Когда вы впервые встретите важное слово, оно будет выделено **полужирным** шрифтом, а впоследствии будет набираться обычным шрифтом.
- Особым шрифтом выделяются технические термины, фразы, появляющиеся на экране, и коды.
- Команды меню мы будем записывать в таком виде: Menu ► Sub-menu ► Sub-menu.
- Когда речь пойдет о том, что действительно важно, эта информация будет выделена так:

Примечание

Это очень важный материал – не пропустите его!

- Прорабатываемые упражнения выглядят примерно так:
 1. Откройте Flash.
 2. Создайте новый файл фильма и сохраните его как TestMovie.fl.
 3. И так далее.

PC и Mac

Чтобы книга была возможно более легкой для чтения, мы по умолчанию используем команды PC, так что всякий раз, когда речь пойдет о командах с мышью, вам не придется читать что-нибудь вроде: «щелкните правой кнопкой мыши на PC или левой кнопкой при нажатой клавише <Ctrl> на Mac».

Обе инструкции мы пишем только там, где имеется отличие от стандартного соответствия команд Mac командам на PC и соответствующая команда необходима. Когда мы просто пишем: «щелкните мышью», мы подразумеваем щелчок *левой* кнопкой мыши на PC и просто *щелчок* на Mac. Обычное соответствие команд таково:

| PC | Mac |
|---|---|
| Щелчок правой кнопкой | Щелчок левой кнопкой при нажатой клавише <Ctrl> |
| Щелчок левой кнопкой при нажатой клавише <Ctrl> | Щелчок левой кнопкой при нажатой клавише <Apple> |
| Нажатие клавиши <Z> при нажатой клавише <Ctrl> (Отмена) | Нажатие клавиши <Z> при нажатой клавише <Apple> |
| Нажатие клавиши <Enter> (Ввод) при нажатой клавише <Ctrl> | Нажатие клавиши <Enter> при нажатой клавише <Apple> |

Теперь мы готовы приняться за дело. Начнем!

- *Основы теории программирования, дающие представление о блоках, из которых построен ActionScript*
- *Как добавить действие в ключевой кадр*
- *Основы синтаксиса программирования ActionScript*
- *Как с помощью одной строки ActionScript превратить линейную анимацию в интерактивную*

1

Начинаем

В первой главе мы вкратце познакомимся с теорией программирования. Не пугайтесь – ровно настолько, насколько это необходимо, чтобы с самого начала понимать важные термины. Заканчивается глава демонстрацией того, как всего две строки ActionScript могут обеспечить основы интерактивности. Кодировать уже с первой главы? Да, и это не так страшно, как вы думаете...

Вы думаете об ActionScript и интерактивности, вы думаете о том, как пользователь отреагирует на ваш сайт и как он станет *инициировать события*. Вот почему мы вместе, не так ли? Чтобы внести интерактивность в ваши разработки. С точки зрения Flash эта интерактивность состоит из **событий**, с одной стороны, и **инструкций**, предписывающих реакцию на эти события, – с другой. События – это краеугольный камень наших программ.

События и как с ними обращаться

В детстве мне подарили часы с кукушкой. Точнее говоря, часами с кукушкой их можно было назвать только условно, поскольку они были сделаны в виде картонного льва, раскачивающийся хвост которого изображал маятник. Когда часам наступала пора бить, лев просыпался, открывал пасть и издавал рык. Этот лев мог служить и будильником, разражаясь в назначенный час могучим ревом. Смысл этого воспоминания в том, что оно объясняет вам один из основных конструктивных элементов ActionScript – **события**.

Как я уже говорил, были две вещи, которые побуждали моего льва к действию:

- Когда оканчивался очередной час, наступало время рычать.
- Когда подходило установленное на будильнике время, наступало время реветь.

Это были два события, которых ожидал лев. Как только происходило любое из них, лев должен был совершить соответствующее **действие**. ActionScript называют языком, который **управляется событиями**, что звучит по-программистски невнятно, пока вы не поймете: подобно льву, он создан так, что ждет, пока что-нибудь не случится. Он отвечает на события – ничего более сложного здесь нет.

Итак, что представляет собой событие во Flash?

Это может быть что-то вполне понятное, что случается у всех на виду, скажем, когда пользователь нажимает кнопку на веб-сайте или на клавиатуре. Событие может быть и менее очевидным, если оно происходит внутри машинного отделения Flash, как, например загрузка фильма или переход к следующему кадру. Когда Flash переходит к очередному кадру, он смотрит, не связаны ли с этим событием какие-либо инструкции, и поступает соответственно. Такие события называются **внутренними**, поскольку Flash порождает их сам.

Имеете ли вы дело с внутренним или с внешним событием, Flash ActionScript действует по следующей схеме:

1. Сценарий ActionScript настраивается на обнаружение определенного события.
2. Как только событие происходит, выполняется обрабатывающий это событие набор инструкций ActionScript.

Сценарий ActionScript, запускаемый на шаге 2, иногда называют **обработчиком события**, который мыслится всегда в связи с главным событием. Отсюда первое правило ActionScript: если есть событие, то **должен быть** обработчик события.

В реальном мире можно встретить множество пар типа «событие/обработчик события». К примеру, электрический чайник ждет, пока закипит вода (событие), и имеет специальную электросхему, которая выключает его, когда это событие происходит (обработчик события). То же и с центральным отоплением – вы устанавливаете свой термостат на определенную температуру. Когда в квартире становится холодно (событие), отопление включается (обработчик события), чтобы стало теплее.

События во Flash

Вернемся в мир Flash и рассмотрим простейшую пару событие/обработчик события, например процесс нажатия кнопки (рис. 1.1–1.3).

Нажми меня, и что-то
произойдет!



Рис. 1.1. Кнопка ожидает действие со стороны пользователя

Rrrrrr!



Рис. 1.2. Пользователь нажимает кнопку – это событие



Рис. 1.3. Пользователь слышит звуковой сигнал

Flash замечает **событие** (нажатие кнопки) и запускает присоединенный сценарий ActionScript – **обработчик события** (рис. 1.4). В результате пользователь слышит звуковой сигнал (рис. 1.3).



Рис. 1.4. Flash запускает обработчик события

На пару событие/обработчик события легко указать, когда дело касается чего-то вроде кнопок, но события не всегда очевидны, когда Flash генерирует их сам в фоновом процессе. Тем не менее, если вы уяснили себе основное – что у события всегда имеется соответствующий обработчик события, – это существенно упростит понимание дальнейших рассуждений в этой книге, когда мы погрузимся в глубины ActionScript.

Новичок ли вы в программировании или уже знакомы с такими языками, как BASIC или Pascal, вряд ли вам все же хорошо известна структура «событие/обработчик события». Вы, может быть, скажете: «Отлично, нам рассказали, что из себя представляет каждый из членов этой пары, но как их применять? Какой от них толк?»

Одно из преимуществ, которые управляемые событиями ActionScript вносит во Flash, состоит в том, что появилась возможность реагировать на неожиданности в **реальном времени** – как только они случаются. Я объясню это несколько подробнее.

События в реальном времени

Такие языки программирования, как BASIC, хорошо справляются со своими задачами, когда имеется четко определенный алгоритм их решения. Программа, суммирующая счета от бакалейщика, совершает всегда один и тот же набор действий в неизменном порядке:

1. Стоимости всех покупок набираются на клавиатуре.
2. Компьютер их суммирует.
3. На экран выводится итоговая сумма.

К третьему шагу можно переходить, только если выполнены первые два.

К несчастью, реальная жизнь не так проста. Когда мы на самом деле приходим в супермаркет и пытаемся купить все, что нам нужно, то часто многое забываем и потому бродим по магазину туда и сюда. Мы забываем, что нужно купить молока, и вспоминаем об этом, когда уже отошли от него на шесть рядов полок, направившись за едой для собаки. Все случается не в том порядке, в каком нам хотелось бы. В том, как мы бродим по супермаркету, нет ни логики ни разума.

В реальной жизни нельзя надеяться, что события будут происходить в желаемом порядке, однако мы можем реагировать на них по мере их наступления. Мы не можем быть уверены, что первым наступит событие А, за которым последуют В и С именно в этом порядке (рис. 1.5). Нужно сказать себе: «**Либо А, либо В, либо С** могут случиться в любой момент. Пока я разбираюсь с В, может произойти А, не дожидаясь, пока я к нему подготовлюсь.»

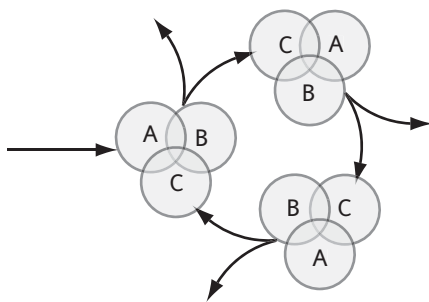


Рис. 1.5. Как будут развиваться события, никто не знает...

Это определяющая характеристика событий в **реальном времени** — они происходят в случайном порядке, поодиночке или группами. Они наступают в порядке, отличном от гладкой последовательности, от схемы, за которой легко проследить (а именно так должны они происходить по мнению BASIC и Pascal) (рис. 1.6).

Поэтому ActionScript должен быть готов среагировать, как только заметит что-то — или еще быстрее! В этом и состоит роль пары

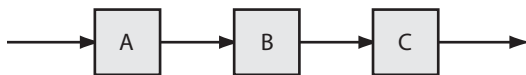


Рис. 1.6. Последовательность действий

«событие/обработчик события». Первая половина (событие) следит за тем, не случилось ли **что-нибудь**, но не знает, когда это «**что-нибудь**»

может случиться. Вторая половина (обработчик события) всегда готова к ответу, готова сделать то, что вы хотите, как только это понадобится.

Таким образом, разница между языками, управляемыми событиями, такими как Flash, и другими, более линейными языками, такими как BASIC, состоит в том, что:

- Управляемый событиями Flash ориентирован на реальное окружение и реальные процессы.
- Линейный BASIC подходит скорее для решения аналитических проблем, когда данные уже собраны и их можно обрабатывать вне связи с текущим состоянием окружающей среды.

Ну вот, мы совершили нашу первую вылазку в теорию ActionScript. Пришло время засучить рукава и присоединить первый кусок программы к фрагменту Flash-фильма.

Сценарий ActionScript можно привязать к трем объектам фильма:

- Ключевому кадру
- Кнопке в рабочей области
- Клипу в рабочей области

Начнем с самого легкого – добавим сценарий ActionScript к кадру.

Присоединение сценария ActionScript к ключевым кадрам

Я покажу здесь, что добавление действия к кадру не включает в себя ничего более сложного, чем ключевой кадр, окно Frame Actions и одна строка кода. Смотрите.

1. В новом фильме выберите первый кадр в первом слое (рис. 1.7).



Рис. 1.7. Первый кадр первого слоя первой сцены

2. Не отменяя выделения кадра, вызовите окно Frame Actions.

Это можно сделать по-разному:

- Щелкнуть по кадру правой кнопкой мыши при нажатой клавише <Ctrl>, а затем в выпадающем меню выбрать Action.
- Дважды щелкнуть по кадру.
- Выбрать Window ► Actions (или применить соответствующую комбинацию клавиш, указанную в меню вслед за именем действия).

- Щелкнуть по маленькой пиктограмме с изображением стрелки в правом верхнем углу экрана (рис. 1.8). (Во Flash действия всегда представляются этим символом – стрелкой.)



Рис. 1.8. Именно такой стрелкой (бывают и другие)

Примечание

Если вы выберете один из первых двух вариантов и щелкнете по кадру, не забудьте о том, какие установки вы сделали в *Preferences*. (Если это необходимо, вернитесь к введению и посмотрите, что мы рекомендовали на этот счет.) Если флажок рядом с *Flash 4 Selection Style* сброшен, то, возможно, вы сами создали себе проблемы – если у вас на временной диаграмме имеется несколько ключевых кадров и трансформаций (tweens), будет трудно выделять определенные ключевые кадры и добавлять к ним действия.

Каким бы способом вы ни открыли окно Frame Actions, вы увидите такое же, как на рис. 1.9.

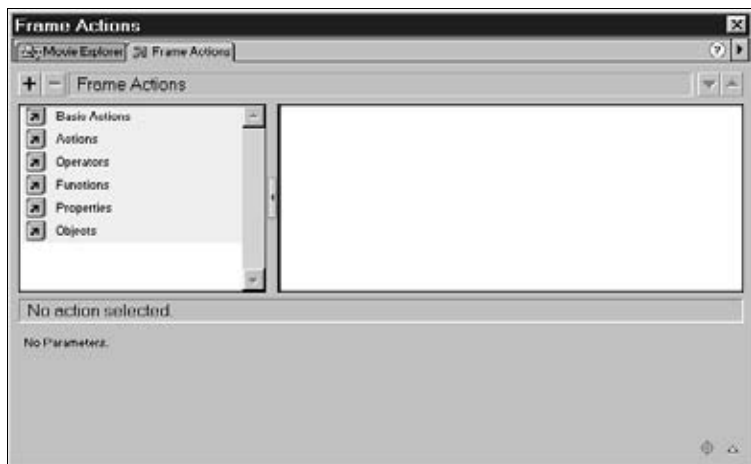


Рис. 1.9. Окно Frame Actions в режиме Normal

Если нижняя часть окна кажется слишком маленькой, и окно выглядит примерно так, как на рис. 1.10, вы, возможно, по ошибке оказались в экспертном режиме (Expert mode).

Чтобы проверить, не так ли это, щелкните мышью по кнопке с маленькой стрелкой в правом верхнем углу окна Frame Actions и выберите обычный режим.

В следующей главе я подробнее объясню, что означают эти два режима, и обещаю, что к концу этой книги мы постоянно будем работать в экспертном режиме. А пока поработаем в режиме Normal.

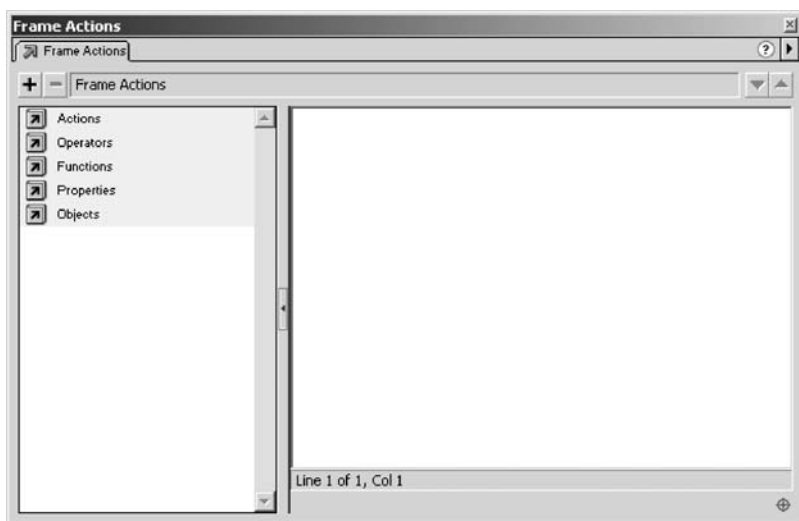


Рис. 1.10. Окно Frame Actions в режиме Expert

Взгляните на нижнюю часть меню и убедитесь, что Colored Syntax и Show Deprecated Syntax помечены флажками, а Font Size установлен в соответствии с вашим монитором и зрением. Я уже старею, поэтому выбираю Large (рис. 1.11). Обсуждение этих опций мы также отложим до будущих времен – не стоит прерывать вашу активную работу.

Ну вот, мы установили основные параметры, вернемся теперь к главному окну.

Окно Frame Actions будет центром нашей деятельности по созданию сценариев ActionScript. В дальнейшем мы изучим его более подробно, а пока я покажу вам ровно столько, сколько необходимо, чтобы создать первый кусочек ActionScript – простое действие.

На левой панели (рис. 1.9) вы видите шесть пиктограмм в виде закрытых книг, на каждой из которых нарисована стрелка «действие».

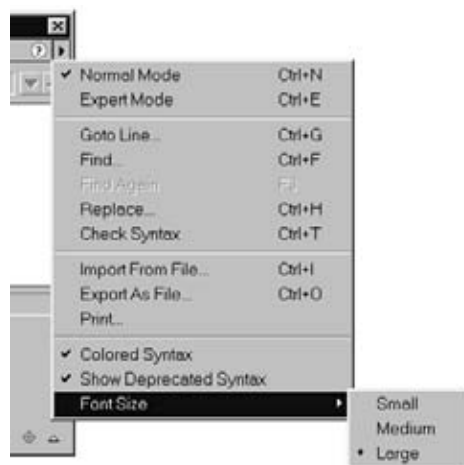


Рис. 1.11. Такая стрелка вызывает подменю

Пиктограммы помечены следующим образом: Основные действия, Действия, Операторы, Функции, Свойства и Объекты (рис. 1.12). Каждая из них содержит готовые к употреблению фрагменты ActionScript, которые мы со временем внимательно изучим. Если вы наведете на пиктограмму указатель мыши, то увидите всплывающую подсказку, в которой дается краткое определение содержимого и его назначения (рис. 1.13).



Рис. 1.12. Пиктограммы, позволяющие вставлять в сценарий готовые фрагменты ActionScript

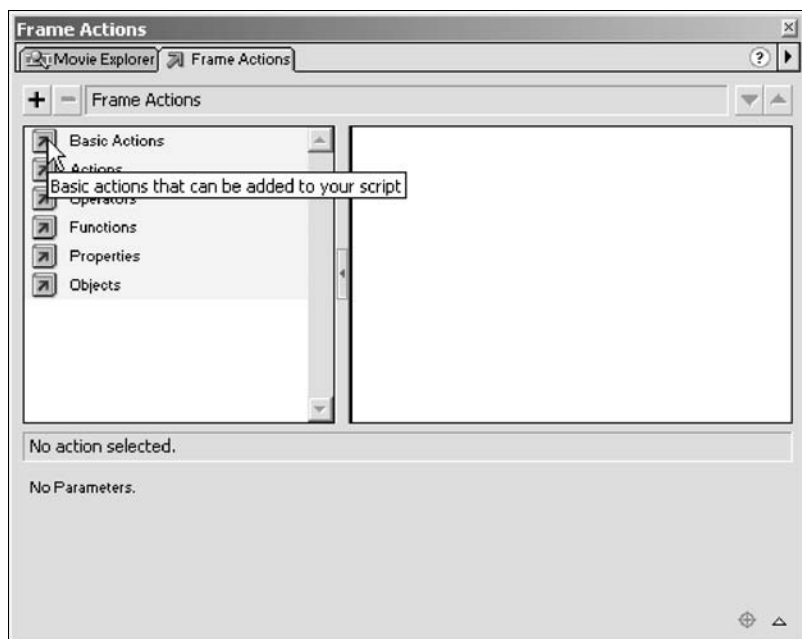


Рис. 1.13. Всплывающая подсказка гласит: «Основные действия, которые могут быть добавлены в ваш сценарий»

3. Чтобы открыть «книгу» Basic Actions, щелкните мышью по соответствующей пиктограмме. Значок преобразится, представив раскрытую книгу (что вполне логично). Появится выпадающее меню, содержащее все заключенные в этой «книге» действия или команды (рис. 1.14). Это те элементарные действия, которые служат кирпичиками при строительстве ActionScript.

И снова вы можете получить дополнительную информацию об этих действиях, подведя к пиктограмме указатель мыши и задержавшись

на ней до появления всплывающей подсказки, которая объяснит их назначение (рис. 1.14).



Рис. 1.14. Подсказка гласит: «Остановить проигрывание фильма»

Примечание

Во время выполнения дальнейших шагов не забывайте о том, что кадр должен оставаться активным до тех пор, пока мы не закончим работу с ActionScript. Если окно Frame Actions внезапно станет пустым, это, вероятно, произойдет оттого, что вы щелкнули мышью где-то за пределами окна, что повлекло за собой отмену выделения кадра. Вы можете вернуть окно Frame Actions, снова выделив кадр, так что не паникуйте и не гадайте, куда подевались все заготовленные действия.

4. Дважды щелкнув мышью по действию Stop, вы увидите, как на правой панели появится соответствующая строка кода (рис. 1.15). Если этого не случилось, вспомните о необходимости проверить, по-прежнему ли выделен кадр.

```
stop ();
```

Рис. 1.15. Это добавленное вами действие

Можно продолжать добавлять действия тем же способом или попробовать что-нибудь новое.

В левом углу строки заголовка окна Frame Actions находится кнопка «плюс» (+) (рис. 1.16).



Рис. 1.16. Кнопка «плюс» в левом верхнем углу окна Frame Actions

5. Установите на нее курсор и дождитесь подсказки Add a new item to the script (Добавить к сценарию новый элемент).

Нажмите ее, и появится раскрывающийся список пиктограмм в виде закрытых книг – от Basic Actions до Objects. Задержите курсор на

Actions и выберите Stop из списка доступных действий, перечисленных в алфавитном порядке. И точно так же, как мы это уже видели, выбранный вами код появится в правой панели окна.

Заметьте, что этот способ выбора имеет то преимущество, что он показывает горячие клавиши для добавления каждого действия, которыми вы, возможно, предпочтете пользоваться, когда ближе познакомитесь со всеми действиями.

6. Если вы попробовали применить оба описанных метода, то, возможно, вставили в кадр два действия Stop (рис. 1.17). Нам нужно только одно, поэтому удалим лишнее: сперва щелкните мышью по нему, чтобы его выделить, а затем либо нажмите клавишу <backspace>, либо щелкните мышью по кнопке «минус» (-), которая тоже находится в левой части строки заголовка окна Frame Actions.

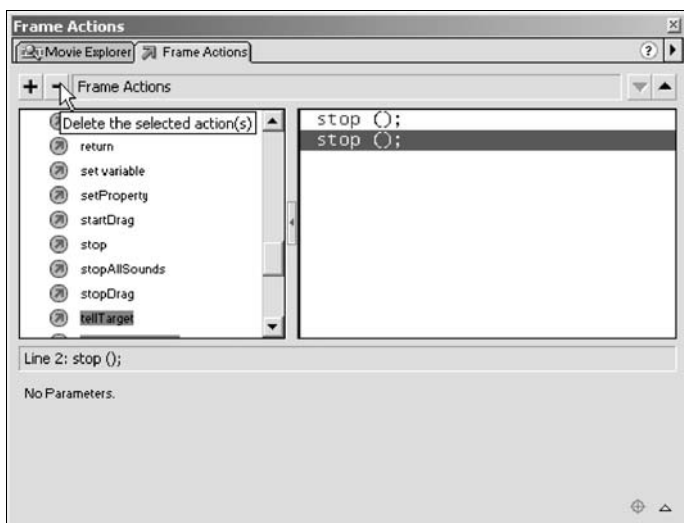


Рис. 1.17. Два действия Stop, одно из которых выделено

Мы вставили в кадр очень простое действие Stop. Нетрудно понять, для чего оно нужно. На этой стадии мы имеем дело с единственной строкой кода, которая сама себя объясняет, но я уже сейчас покажу вам кое-что из того, что поможет вам разбираться в больших фрагментах кода, которые вы будете писать, когда поднаберетесь опыта. Если вы привыкнете этим пользоваться уже сейчас, с самого начала, то сэкономите много часов мучительных размышлений впоследствии...

Мы снова проделаем процесс вставки, но на этот раз будем вставлять нечто другое, а именно **комментарии**.

Комментарии

Считайте комментарии заметками на память, предназначенными для вашего личного употребления, пометками на полях, если хотите. Комментарии не содержат никаких инструкций для Flash, они только выделяют в сценарии ActionScript место, где можно пояснить, что именно в данный момент происходит. Давайте вставим комментарий, и вы все поймете.

Добавление комментария

1. И снова щелкните мышью по кнопке + в окне Frame Actions и подведите курсор к Actions (рис. 1.18). В раскрывающемся списке выберите на этот раз comment.

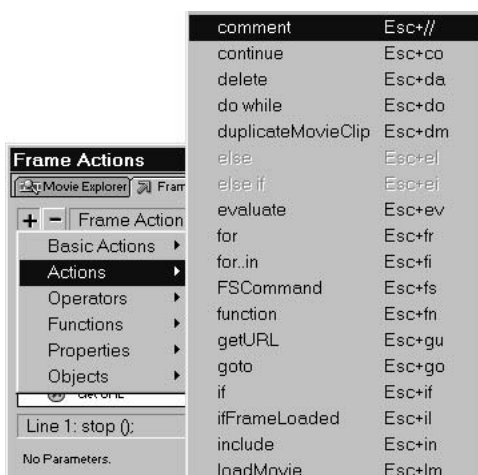


Рис. 1.18. Выбор действия comment

На правой панели появится строка, начинающаяся с //. Этот символ объявляет, что за ним следует комментарий, и приказывает Flash его игнорировать.

При этом несколько изменится вид нижней части окна действий – в ней появится поле ввода текста с меткой Comment (рис. 1.19). Вы можете набрать в нем любой текст, поясняющий сценарий, чтобы по прошествии времени легче было понять последующий код. В хорошо откомментированном фрагменте проще разобраться и, значит, его проще будет отлаживать полгода спустя, когда вы забудете, что за идеи приходили вам в голову при написании этого кода.



Рис. 1.19. Здесь можно написать текст комментария к коду

2. Чтобы добавить комментарий, щелкните мышью в поле Comment и наберите stop the movie. Посмотрите, как ваш текст отобразится в верхней части правой панели окна (рис. 1.20).

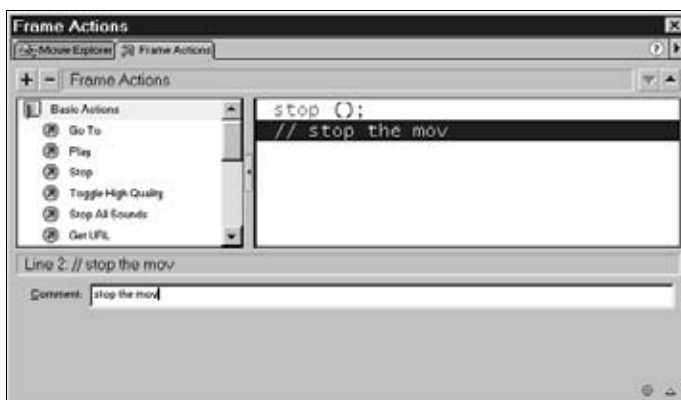


Рис. 1.20. Не самый полезный комментарий, но мы пока только учимся

Вы сделали для себя заметку о том, что делает только что введенное действие.

Вы, может быть, решите, что логичнее было бы разместить поясняющий комментарий перед действием `stop ();`. В процессе разрастания сценария в любой момент можно изменить порядок действий или комментариев при помощи кнопок со стрелками «вверх» и «вниз» в правом верхнем углу окна (рис. 1.21).



Рис. 1.21. Еще стрелки

3. Убедитесь что ваш комментарий все еще выделен (синим или желтым фоном в зависимости от того, работаете ли вы на Мас или на РС). Нажмите теперь стрелку «вверх», и комментарий, поднявшись на одну строку вверх, окажется в первой строке. Если вам захочется, можете опустить его обратно при помощи стрелки «вниз».

Хотя создаваемый сценарий с ростом числа действий выглядит все сложнее, испещряясь скобками и другими знаками, не забывайте, что **Flash пишет его за вас**. Таким образом он помогает вам, если вы программируете в обычном режиме, что мы недавно и делали. Все, что от вас требуется, это выбирать подходящие действия и заполнять предлагаемые поля ввода. Как начинающий вы не обязаны в совершенстве знать все тонкости синтаксиса ActionScript, поскольку Flash будет вам очень помогать.

4. Наконец, закройте окно Frame Actions. Если вы посмотрите на кадр (рис. 1.22), то увидите, что теперь там появилась маленькая буква «а». Она указывает, что к этому кадру присоединены действия.



Рис. 1.22. «а» указывает, что к кадру присоединены действия

Присоединение действий к кнопкам и клипам ничуть не сложнее – выберите в рабочей области нужную кнопку или клип, откройте окно Frame Actions и введите свой сценарий. Подробнее мы это рассмотрим далее в главе 5.

Теперь, поработав в ActionScript вручную, мы должны вернуться к программистской терминологии.

Инструкции в программе должны быть очень точными, поэтому, подобно другим языкам программирования, ActionScript имеет весьма жесткую структуру. Инструкции могут включать в себя три четко определенные части:

- Команды
- Аргументы
- Свойства

Команды, аргументы и свойства

Смысл команд и аргументов станет ясен, если вы мысленно вернетесь к тем временам, когда родители приказывали вам делать уборку в спальне. Речь не о том, что они отдавали вам команду, а вы возражали на это, приводя веские аргументы. Конечно, возможно, все так и было,

но мы сейчас не об этом. Если проводить аналогию с `ActionScript`, то здесь возможны два варианта:

- Вас воспитывали обычные родители, и у вас была только одна спальня. Родителям не было нужды указывать, какую именно спальню надо убрать, поскольку это ясно следовало из команды: «Немедленно! Ступай наверх и приберись в этой чертовой спальне!»
- Вы были из детей миллиардеров, и у вас было по меньшей мере двенадцать спален, а родители в таком случае говорили: «Поднимись сейчас же в спальню номер семь, иначе не получишь трюфелей с ромом!»

В первом случае родителям не нужно сообщать вам какие-то дополнительные подробности, поскольку очевидно, что именно от вас требуется. Вам дается простая команда. Во втором случае родителям буржуя Пети приходится сообщать дополнительную информацию – о том, *какую именно* спальню ему следует прибрать. Этот дополнительный кусок информации, уточняющий исходную команду, и называется **аргументом**.

У команды `stop()`, которую мы только что рассматривали, нет аргументов, поскольку Flash и так знает, что ему делать. Она означает остановку *данной* временной диаграммы. Не нужно уточнять, к какой именно диаграмме эта команда относится, поскольку Flash понимает, что она может относиться только к той, на которой помещена.

Вскоре мы будем использовать команду `gotoAndPlay()`, которая сообщает Flash, чтобы он перешел к новому кадру. Очевидно, мы должны указать Flash, на *какой именно* кадр он должен перейти, поэтому мы добавляем аргумент:

```
gotoAndPlay(10);
```

Flash получает предписание перейти на новый кадр (команда), и этот кадр – 10 (аргумент).

Спускаясь на следующий уровень детализации, вы приходите к **свойствам**.

Свойство вашей спальни – быть либо чистой, либо нет. Свойством звука, который Flash исполняет в вашем фильме, является громкость. Положение объекта на экране определяется двумя свойствами: *x* и *y*.

Список свойств, существующих в мире `ActionScript`, содержится в одной из книжек в окне действий, которая вполне закономерно называется `Properties`. Щелкните мышью по книге `Properties`, а затем задержите мышь над одним из свойств, чтобы увидеть, какими атрибутами оно управляет (рис. 1.23).



Рис. 1.23. Книга свойств. Подсказка: «Горизонтальная координата мыши»

Соедините вместе эти кусочки головоломки (рис. 1.24): команды, аргументы и свойства. Что получилось? Целая инструкция.

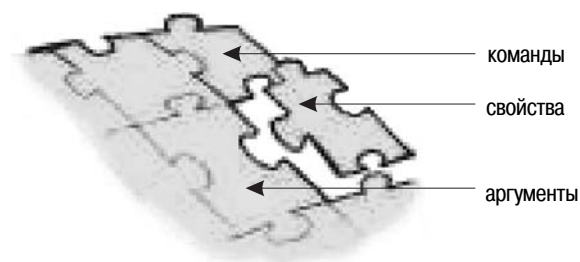


Рис. 1.24. Головоломка

Инструкции

Существует много различных видов инструкций, но все они принадлежат к одной из следующих категорий или общих функциональных типов:

1. Исполнить команду.
2. Исполнить команду, используя данный аргумент.
3. Изменить данное свойство аргумента на другое.
4. Если это утверждение истинно, тогда исполнить данное множество действий.
5. Продолжать исполнять группу инструкций, пока некоторое утверждение истинно.

Если вы никогда прежде не программировали, почувствуйте вкус этого занятия, поразмыслив о том, что вы делаете каждый день – о том, как наполняете стакан водой:

продолжать...
наливать (воду)
пока стакан не наполнится.

Строка наливает (воду); относится ко второму типу. **Команда** – это *наливать*, а **аргумент** – *вода*.

Продолжать... пока стакан не наполнится относится к типу 5 – продолжать наливает воду, пока стакан не наполнится.

Если вы подобным образом разобьете на части свои повседневные действия, то увидите, что программирование – это вовсе не создание сложной последовательности инструкций на иностранном языке. Это просто разбиение каждого процесса на основные блоки. Разложить любой процесс до набора составляющих его скелет инструкций так просто, что мы обычно проделываем это, не задумываясь.

Для того чтобы программировать, не нужно быть умным, нужна только способность думать на правильном уровне. Проблема в том, что этот уровень настолько низко расположен и настолько близок основам мышления, что большинству из нас приходится заставлять себя думать так примитивно! Этот навык приходит с опытом, и он не имеет ничего общего с математикой или сложением больших чисел в уме, как вам, возможно, казалось.

Когда вы пытаетесь выстроить команды программы, даже гораздо более сложные, чем только что рассмотренная инструкция `Stop this movie`, вы существенно облегчите свою задачу, разбив путь к достижению цели на эти простейшие элементы. В уме или на бумаге распланируйте в подобных выражениях то, что вы хотите сделать, дабы прояснить для себя структуру сценария `ActionScript`.

Как бы просты инструкции ни были, они должны иметь понятную структуру, чтобы быть вполне ясными для тех, кому они предназначены – будь то буржуй Петя, которому необходимо точно знать, какую именно спальню ему нужно убрать, или Flash, который должен точно знать, на какой кадр перейти. В хаосе должен присутствовать порядок.

Такую организацию обеспечивает четкая структура команд `ActionScript`, благодаря которой Flash понимает, чего вы от него хотите. Эту структуру описывает синтаксис языка программирования.

Синтаксис

Важной частью любого языка от сербскохорватского до китайского является допустимый порядок построения предложений – **синтаксис**. Независимо от того, на каком языке разговариваем, мы ожидаем ус-

лышать слова, расположенные в определенном порядке, так, чтобы можно было придать им определенное значение и понять, о чем нам говорят. Если кто-то скажет: «Корова пасется на большом зеленом поле», вы знаете, что именно поле большое и зеленое, а не корова.

Существуют также дополнительные правила разделения записанных слов знаками препинания, поэтому и при письме мы можем передать тот же смысл, что и в живой речи. Мы умеем записывать информацию о ритме речи так же, как и ее значение. Синтаксис и пунктуация вместе позволяют нам строить вопросительные, восклицательные и утвердительные предложения – это разнообразие делает наш язык интересным.

Компьютерам нет нужды быть выразительными, они должны быть точными, из чего следует, что изучение языков программирования гораздо легче, чем обучение речи и письму на естественном языке. Сейчас мы увидим, как ActionScript организует и выражает себя.

В отличие от английского языка с его многообразием выразительных форм, ActionScript при помощи своего синтаксиса должен выразить лишь небольшое число смысловых аспектов, которые делятся на три категории:

1. Когда отдельное действие начинается или заканчивается

Это звучит почти так же, как вопрос: «Где предложение начинается и где оно заканчивается?» В английском языке оно может начинаться с новой строки или на той же строке, что и предыдущее предложение. Оно может заканчиваться точкой, вопросительным знаком или восклицательным. Во Flash все гораздо проще. Каждая основная команда начинается с новой строки и завершается точкой с запятой:

```
stop();
```

Исключениями из этого правила являются:

- Комментарии, которые, как мы видели, начинаются с символов `//` и могут заканчиваться чем угодно, поскольку Flash их полностью игнорирует.
- Вложенные команды, в которых добавочные команды заключаются в фигурные скобки `{ }`, чтобы они обрабатывались независимо, подобно тому, как предложения группируются в абзацы, составляя для читателя некое целое. Скоро я объясню это подробнее.

2. Какая часть действия является командой, какая – аргументом, а какая – свойством?

В приведенном примере первая строка является комментарием, описывающим то, что происходит в следующих строках. Как мы уже говорили, Flash не понимает или даже не читает ничего, что следует за символами `//`, поэтому там можно помещать все что угодно:

```

// установить x в 10 и остановить фильм ← комментарий (пурпурный)
x = 10;
действие (голубой) → stop ();
                        ↑ аргумент (черный)

```

Мы имеем здесь два действия: **x=10** и **stop ()**. Оба они начинаются с новой строки и завершаются точкой с запятой. Действия *всегда* следуют этому синтаксису. В английском языке знаки препинания многозначны (символ «.» может означать точку в конце предложения или отделять дробную часть числа). В ActionScript точка с запятой *всегда* означает конец строки. (Не правда ли, это «*всегда*» звучит успокаивающе? Это означает, что однажды это усвоив, мы больше не станем об этом думать и нам никогда не придется говорить: «но в этом особом случае...».)

Определить, какие части действия являются командой, аргументом и свойством, может на деле оказаться весьма непросто, поэтому ActionScript сообщает нам это двумя способами: через **расположение** и через **выделение цветом**. Распознаванию кода по его расположению вы научитесь со временем, когда будете уверенно владеть конкретными командами ActionScript, так что сосредоточимся пока на цветах. Flash отображает коды, используя три цвета, обозначающих три части речи языка ActionScript:

- Пурпурный для комментариев
- Голубой для команд (и некоторых других зарезервированных слов)
- Зеленый для свойств
- Черный или серый для всего остального

Нам, очевидно, трудно изобразить эти цвета в черно-белой книге, поэтому, как можно видеть на предыдущей диаграмме, я просто написал их названия. Когда в следующих главах вы будете создавать сценарии ActionScript, вы убедитесь в справедливости моих слов, а пока я могу только заверить вас в том, что говорю правду.

3. Какие строки должны исполняться, когда выполняется некоторое условие, и какие — когда оно не выполняется. Какие группы действий должны исполняться повторно.

Это как раз то самое «все остальное», которое отображается серым цветом. Мы дойдем до него со временем, но для тех, кто уже знаком с ActionScript для Flash 4 или вообще с программированием, я скажу, что это могут быть:

- Имена переменных
- Пути к переменным или свойствам
- Скобки и операторы (() + - / * и т. д.)

Как вы обнаружите в процессе овладения ActionScript, не всегда нужно, чтобы действия исполнялись строго по очереди, одно за другим. И порой вам захочется, чтобы некоторая определенная последовательность действий выполнялась, только если случилось что-то еще. Это будет наше первое знакомство с условными предложениями, в которых мы просим Flash проверить, истинно или ложно некоторое утверждение, и указываем, что ему следует делать в зависимости от ответа на этот вопрос.

И наконец, рассмотрим, как фигурные скобки `{ }` и отступы используются для предписания желательного порядка действий и как команды вкладываются одна в другую.

Вложение

Я терпеть не могу множества точек, скобок, запятых и всего такого. Как, наверное, большинство людей ненавидят пауков, потому что у них слишком много ног, так я не переношу строчки, в которых слишком много скобок. Они подозрительно похожи на математические выражения, а мне не хочется иметь с ними ничего общего. В случае с ActionScript я избавляюсь от этой болезненной неприязни, убеждая себя, что это не те скобки, что заставляют нас хвататься за калькулятор и думать. Это такие скобки, которые просто объединяют несколько предметов в одну группу, как это делается в следующем предложении:

«Все маленькие собаки (в том числе таксы и терьеры) собираются под деревом слева, а все другие собаки (в том числе гончие и овчарки) — справа.»

Это пугает меня гораздо меньше, чем выражения типа:

$$s = _ (u+v)t + 5(a2+20c) - 1$$

Уф! Паучьи лапки! Я знаю, что большая часть нижеследующего материала похожа на ужасные уравнения, но на самом деле все не так плохо. По большей части все это лишь краткий способ записи предложения про собак и деревья. Когда у нас дело касается какой-либо математики, то она обычно относится к разряду выражений $2=1+1$.

Следующая команда является условным (if) действием:

```

      истинно ли это?
      ↓
if (x<10) {
  y = -10;
  z = 15; ← если да, выполнить это
} else {
  y = 10;
  z = 45; ← если нет, выполнить это
}
```

Не вникайте слишком во все эти *y* и *z*. Я хотел только показать вам синтаксис инструкции, которая приказывает Flash решить, верно ли то или иное утверждение, и поступить соответственно.

И вот что при этом происходит:

1. Flash смотрит на содержимое скобок в строке 1.
2. Если это утверждение истинно, исполняются строки 2 и 3.
3. Если утверждение ложно, исполняются строки, следующие за словом «else».

Строки 2 и 3 заключены в фигурные скобки и снабжены отступами. Строки 5 и 6 – тоже:

```
.....{
y = -10;
z = 15;
}.....
```

Синтаксис говорит нам, что строки 2 и 3 должны исполняться как единая последовательность. Строки 5 и 6 должны исполняться как альтернативная последовательность команд в зависимости от того, является ли утверждение ($x < 10$) *истинным* или *ложным*.

Чтобы запомнить такие вещи, мне приходится прибегать к разным глупым ухищрениям. Этот, пожалуй, слишком сложный синтаксис я представляю в виде ряда блюд, где на каждом блюде находится содержимое пары фигурных скобок { }. Записанный как ряд блюд, синтаксис выглядит примерно так:

```
if (x<10)      {y=-10; z=15;}   else      {y=10; z=45;}
```

или обобщенно:

```
if (this is true)  {do this sausage}   else   {do this sausage}
```

Не пытайтесь сейчас заучить структуру `if...else`, но обратите внимание на то, как отступы и фигурные скобки указывают нам тот порядок, в котором выполняются отдельные действия.

Структура циклов очень похожа, за исключением того, что теперь у нас только одно блюдо, которое повторяется:

```
while (x<10) {
x = x+1;
y = y+5;
z = z-10;
}
```

В моих глупых обозначениях это выглядит так:

```
while (x is less than 10) {keep giving these values to x, y and z}
```

Эту главу мы закончим простым примером. Мы воспользуемся уже знакомой вам линейной анимацией, чтобы дать вам понятие о том, на-

сколько добавление простейшего сценария ActionScript может ее улучшить.

ActionScript добавляет интерактивность

Мы знаем, какая замечательная вещь этот Flash, потому что он избавляет нас от огромной работы, необходимой обычно при создании эффектов анимации. Нам нужно создать только начальный и конечный ключевые кадры, а все промежуточные он изготавит сам. В этом практически все дело и состоит, но мы выйдем за пределы обычной линейной анимации, чтобы увидеть, какой силой обладает введение даже простейших элементов ActionScript.

Прежде чем приступить, давайте убедимся, что мы начинаем новый фильм, иными словами, что ни с одним из кадров, с которыми мы теперь будем работать, не связано действие Stop из предыдущего упражнения.

1. Убедимся, что на рабочем столе видны линейки разметки. Выберем из набора инструментов Circle. Следя за тем, чтобы изображалась сплошная фигура, нарисуем маленький кружок в рабочей области где-нибудь слева (рис. 1.25).
2. Возьмем теперь инструмент Arrow (Стрелка) и выделим кружок вместе с его контуром. Нажмем клавишу <F8> или при помощи меню Insert ► Convert to Symbol вызовем окно Symbol Properties, введем имя символа circle и убедимся, что этому символу назначен тип поведения Graphic (рис. 1.26–1.27).

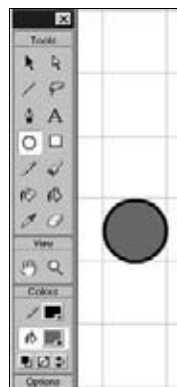


Рис. 1.25. Рисуем кружок

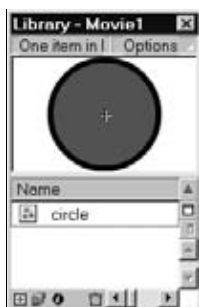


Рис. 1.27. Символ в библиотеке



Рис. 1.26. Придаем кружку значение символа и называем его

Пользователи Flash 4 заметят, что во Flash 5 типы поведения в этом окне перечислены в другом порядке, так что не промахнитесь, действуя на автопилоте!

3. Теперь у нас есть все для создания простой анимации. На временной диаграмме выберите кадр 40 и нажмите <F6>, создавая тем самым новый ключевой кадр (рис. 1.28). Мы хотим создать трансформацию движения (motion tween) между двумя ключевыми кадрами.

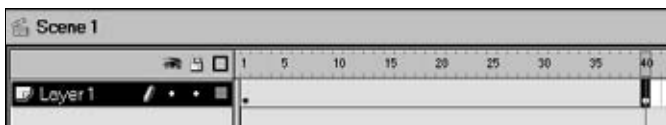
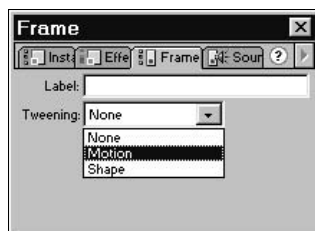


Рис. 1.28. Временная диаграмма

4. Выберите кадр 1 и вызовите панель Frame (Window ► Panels ► Frame). В выпадающем меню Tweening выберите Motion (рис. 1.29).



Вы увидите синюю полосу, протянувшуюся от первого до сорокового кадра (рис. 1.30). Она представляет движение между двумя этими кадрами.

Рис. 1.29. Задаем трансформацию

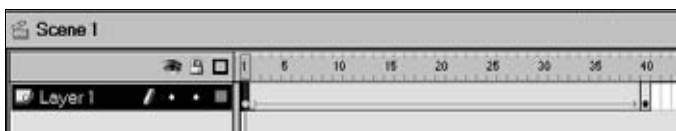


Рис. 1.30. Теперь трансформация задана

5. Между ключевыми кадрами с кружком пока ничего не происходит, так что движением это назвать нельзя. Выберите сороковой кадр и переместите кружок вправо (рис. 1.31).

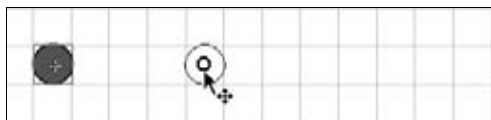


Рис. 1.31. Здесь кружок окажется по окончании движения

6. Теперь, выбрав Control ► Test Movie, проверим, что у нас получилось. Вы увидите, как кружок перемещается слева направо, совершая движение между двумя ключевыми кадрами (рис. 1.32).



Рис. 1.32. Так мы изобразили движение

Я уверен, что с такой Flash-анимацией вы уже знакомы. Кружок всегда перемещается слева направо, движение всегда длится одинаковое количество времени и довольно-таки скоро надоедает. Эта анимация осуждена повторять одно и то же вечно. Ей совершенно все равно, что бы мы при этом ни делали – она не является *интерактивной*.

Попробуем что-нибудь с этим сделать.

Анимация с применением ActionScript

1. Начнем новый фильм, выбрав File ► New. Снова создадим кружок, но разместим его теперь в середине рабочей области и припишем ему тип поведения Movie Clip, а не Graphic (рис. 1.33). Назовем его так же – circle.
2. Сохраняя кружок выделенным, откроем панель Instance (Window ► Panels ► Instance) (рис. 1.34). Назовем кружок ball (шарик).

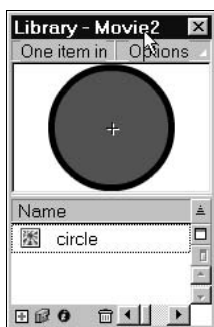


Рис. 1.33. Этот кружок имеет тип поведения «клип»

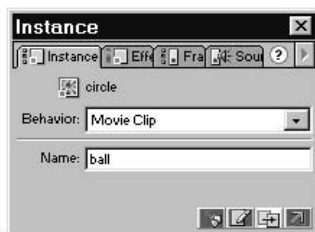


Рис. 1.34. Экземпляр клипа получает собственное имя

Примечание

Если панель *Instance* затенена и не позволяет вам ввести имя, убедитесь, что кружок выбран в *Scene 1* и что вы не щелкнули на нем мышью дважды, так что в левом верхнем углу окна оказался выделенным *Circle*. Щелкните мышью на вкладке *Scene 1*, чтобы кружок снова оказался в квадрате.

Почему мы должны действовать именно так?

Чтобы ActionScript смог управлять клипом, он должен знать его имя. Мы могли бы оставить ему старое имя circle, под которым он известен в библиотеке, но у нас возникнут затруднения, когда мы захотим применить к нему действия ActionScript в каком-нибудь другом месте. Поэтому мы даем *особое* имя именно этому *определенному кружку* – **имя экземпляра**. Вы можете представить себе эту ситуацию так, как если бы вам нужно было различать близнецов. Они, возможно, оба *мальчики*, но каждому из них надо дать свое имя, чтобы можно было обращаться к ним по отдельности.

Примечание

Создание индивидуальных имен экземпляров исключительно важно при программировании. Это пример *конкретизации*, процесса, в котором мы берем элемент общей природы и придаем ему индивидуальность, наделяя его отличительными свойствами, простейшим из которых служит его собственное имя. Конкретизация может заключаться в присвоении новому объекту много большего, чем просто имя; это очень важное понятие, с которым мы ближе познакомимся к концу этой книги, когда начнем изучение объектно-ориентированного программирования.

3. Для управления ball применим ActionScript. Уже сейчас, в самом начале овладения ActionScript, рекомендую вам всегда хранить сценарии ActionScript на отдельном слое. Нажмите кнопку Insert Layer (+), чтобы вставить новый слой (рис. 1.35). Назовите этот слой actions.
4. В слое actions обозначьте второй кадр в качестве ключевого (при помощи <F6>). Вставьте новый кадр в слой 1 (при помощи <F5>). Временная диаграмма теперь выглядит примерно, как на рис. 1.36.

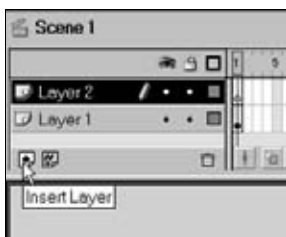


Рис. 1.35. Вставленный слой нужно переименовать



Рис. 1.36. В слое actions два ключевых кадра, а в слое Layer 1 – один

Теперь мы готовы писать наш первый сценарий ActionScript. Возможно, вы еще не совсем поняли, зачем я просил вас проделать все это, но не волнуйтесь. В следующих главах я все подробно объясню, а пока только хочу показать, что можно сделать при помощи всего лишь двух строчек ActionScript.

5. Выделите первый кадр в слое actions. Теперь нам придется работать в окне Frame Actions. Помните, как его вызвать? Сейчас проще всего это сделать, щелкнув по маленькой пиктограмме со стрелкой в правом нижнем углу окна Flash (рис. 1.37).

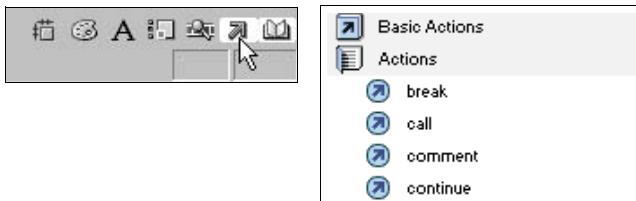


Рис. 1.37. Стрелка позволяет вызвать окно Frame Actions, а в книжке Actions содержится список действий