ГЛАВА

17

Классы TextField и Selection

В этой главе...

Различные виды текста

Создание текста в среде авторских работ и программным образом в процессе выполнения

Работа с объектами TextField с помощью ActionScript

Применение HTML при работе с объектами TextField

Прокрутка текста

Установка и получение фокуса

Применение класса Selection при работе с объектами TextField

Текст является неотъемлемой частью большинства приложений Flash. В главе рассматриваются различные аспекты процессов по обработке текста во Flash. Вы узнаете, каким образом формируется текст, который контролируется с помощью ActionScript, а также ознакомитесь с методиками применения кода ActionScript при внесении в текст изменений. Вы можете отображать HTML-код, выполнять прокрутку текста, внедрять шрифты, а также реализовать большое число других действий.

Непосредственное отношение к тексту имеет класс `Selection`. Именно этот класс применяется для установки и получения *фокуса*. Активный объект в рабочем поле, реагирующий на перемещения мыши и работу с клавиатурой, имеет фокус. Также можно применять класс `Selection` для получения и установки выделенных областей в пределах класса `TextField`. Речь идет о том, что можно устанавливать и вставлять точки для вводимого текста, выбирать выделенный пользователем текст. После ознакомления с этой главой вы узнаете все эти детали, а также получите много другой полезной информации.

Типы текста

Во Flash можно обрабатывать три типа текста: статический (static), динамический (dynamic) и вводимый (input). Перед изучением ActionScript обычно ограничиваются использованием статического текста. Несмотря на то, что статический текст имеет важное значение, его возможностей не всегда достаточно. Статический текст не может контролироваться и обрабатываться с помощью сценариев столь же успешно, как динамический или вводный текст. Поэтому рассмотрим более подробно эти два типа текста, с которыми можно успешно работать при помощи ActionScript.

Динамический текст

Динамический текст является базовым типом текста, который может обрабатываться с помощью кода ActionScript. При обращении к динамическому тексту можно исключительно программным способом выполнять отображаение, скроллинг, форматирование, изменять размеры и даже создавать новый текст. Все эти действия можно реализовать при разработке Flash-приложений, включающих большие фрагменты динамического текста. К примеру, если необходимо отображать обновленные новости или свойства, желательно использовать динамический текст. Любой текст, обновляемый в процессе выполнения приложения, независимо от того, выполняется обновление на основе загруженных данных, при взаимодействии с пользователем либо по другой причине, должен быть динамическим.

Вводимый текст

Как и следует из названия, именно этот вид текста может вводиться пользователем. При этом доступен практически тот же спектр функциональных возможностей, что и в случае с динамическим текстом, но также доступны дополнительные функции. В данном случае пользователь может вводить текстовые значения. Поля вводимого текста предоставляют пользователю широкий круг возможностей: от ввода имени пользователя и пароля до внесения сопроводительной информации в приложение по ведению электронной коммерции.

Создание объектов TextField

При создании динамического или вводимого текста на самом деле формируется объект `TextField`. Это важно, поскольку поля динамического и вводного текста являются экземплярами класса `TextField`. Это означает, что разрешено применение всех встроенных свойств и методов класса `TextField`, с помощью кода ActionScript реализующих контроль над данными экземплярами.

Ясно, что перед выполнением каких-либо манипуляций с экземпляром `TextField`, необходимо его сформировать. В следующих разделах показаны различные возможности, приводящие к формировнию объектов `TextField`. Важно отметить, что предложеннные способы можно подразделить на возможности по созданию в среде авторских работ либо во время выполнения. Независимо от того, формируется объект `TextField` в среде авторских работ или в период выполнения, данный объект остается объектом `TextField`, управлять которым можно так же, как и любым иным объектом `TextField`. Поэтому в основном формирование объекта `TextField` определяется методикой, наиболее уместной в данном сценарии. Рассмотрим эти вопросы более подробно.

Формирование текста в среде авторских работ

Формирование текста в среде авторских работ означает, что текст создается с помощью инструмента Text (Текст), а объект `TextField` отображается в рабочем поле. В определенных ситуациях создание текста в среде авторских работ обеспечивает ряд преимуществ (по сравнению с процессом формирования текста во время выполнения).

- Для склонного к визуальному восприятию пользователя удобнее выполнять компоновку текста в среде авторских работ, чем формировать текст в течение рабочего цикла.
- При создании текста в среде авторских работ можно применять инспектор свойств для установки многих свойств объекта `TextField`, поэтому нет необходимости в

создании всего кода ActionScript для инициализации данного объекта с помощью этих значений.

Если вы уже сформировали статический текст, это означает, что вам известны основные принципы создания динамического или вводного объекта `TextField` в среде авторских работ. Дело в том, что при формированнии динамического или вводимого текста, точно так же, как и в случае со статическим текстом, выбирают инструмент Text и рисуют форму нового объекта `TextField`. Основным отличием между процессом создания статического и динамического либо вводимого текста является тот факт, что корректный тип следует выделять перед отображением текста в рабочем поле. Поэтому после выделения инструмента Text, но перед созданием объекта `TextField`, необходимо из меню видов Text (Текст) в окне инспектора свойств выделить соответствующий тип, а именно Dynamic Text (Динамический текст) — для динамического объекта `TextField` либо Input Text — для поля вводимого текста, что показано на рис. 17.1.

Тип текста для объекта `TextField` можно изменять после его создания. Установка типа текста перед его отображением рекомендуется в качестве общего технологического совета. Принцип подобного подхода аналогичен тому, который применяется в процессе рисования формы в рабочем поле. Можно изначально выделить цвет заливки либо выделить форму после ее создания и заменить цвет заливки.

Несмотря на то, что можно изменять тип объекта `TextField` после его создания, во многих случаях важно, чтобы либо параметр Dynamic Text, либо Input Text выбирался *перед* рисованием объекта. Довольно часто при создании динамического и/или вводимого текста, нет необходимости добавлять в него какие-либо значения в среде авторских работ. Вместо этого иногда возникает потребность добавить значение в период выполнения (в случае динамического текста) или же разрешить пользователю ввести значение (в случае вводимого текста). Тем не менее, если не введено значение для статического текста, Flash удалит его из рабочего поля. С другой стороны, динамический и вводимый текст позволяют рисовать объект и не поддерживать значение.

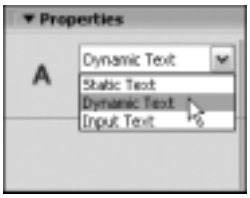

Рис. 17.1. Выделите из меню типов Text параметр Dynamic Text либо Input Text

После создания объекта `TextField` в рабочем поле следует предоставить объекту название экземпляра. Этот шаг имеет определяющее значение, поскольку название экземпляра определяет способ упрвления объектом с помощью кода ActionScript. Можно добавить название экземпляра с помощью инспектора свойств. Если объект `TextField` выделен в рабочем поле, откройте инспектор свойств и введите значение в поле Instance name (Имя экземпляра), как это выполняется при работе с объектами `Button` или `MovieClip`. Название экземпляра должно соответствовать тем же правилам наименования, что и любой другой объект. Если вам необходимо уточнить эти правила наименования, вернитесь назад, к главе 5; правила наименования для переменных и для объектов аналогичны.

Если вы редко имеете дело с Flash, можете применять методику Flash 5 по присваиванию тексту названия переменной. Хотя Flash MX 2004 по-прежнему поддерживает подобный подход для авторизации приложений, совместимых в обратном порядке. Причем не имеет значения, направляется ли в текст вместо названия экземпляра название переменной, если авторизация проходит при помощи содержимого Flash 6 либо Flash 7. Если работа с текстом проходит с помощью названия переменной вместо названия экземпляра, вы не можете получить доступ к свойствам и методам, наследуемым из класса `TextField`. На рис. 17.2 показан вид инспектора свойств, в окне которого отображается объект `TextField`. Обратите внимание, что слева находится поле <Instance Name>. Именно здесь корректно расположить присваи-

ваемое объекту название экземпляра. Справа можно заметить поле с меткой Var: в окне, которое находится справа от него, можно присваивать тексту название переменной, но только в случае авторизации содержимого для более ранних версий плейера. Если же используется последняя версия плейера, этого делать не рекомендуется.

Рис. 17.2. Вид инспектора свойств для объекта `TextField`

При создании объекта `TextField` в среде авторских работ с помощью инспектора свойств также можно устанавливать большое число свойств. Многие из них либо очевидны, либо хорошо известны пользователям. К примеру, вряд ли вызовет затруднение выделение вида шрифта, размера, цвета или выравнивания. Но некоторые из применяемых параметров могут быть незнакомы пользователю, поэтому в данном разделе они рассматриваются более подробно. То есть изучаются настройки типа строки, способности к выделению, поддержки HTML, формирования границы и ограничения максимального количества символов. Ознакомившись с материалом этого раздела, вы узнаете, каким образом эти настройки могут изменяться с помощью инспектора свойств. Соответствующие свойства ActionScript обсуждаются далее в этой главе.

Меню Line type (Тип строки) включает следующие параметры.

- **Single line (Одна строка)**. Эта заданная по умолчанию настройка ограничивает ввод в поле только одной текстовой строкой. Однако, если все же обстоятельства принуждают вас присваивать текст этому полю с помощью выражений ActionScript, можно применить оператор `newline` (либо обратную косую черту `\n`, применение которой подробно рассматривалось ранее) для принудительного ввода в поле символа возврата каретки.
- **Multiline (Несколько строк)**. Данная настройка позволяет вводить в поле несколько строк текста и автоматически переносить каждую строку текста.
- **Multiline no wrap (Несколько строк с запретом переноса)**. Эта настройка разрешает пользователю вводить в поле несколько строк текста, но только при нажатии клавиши <Enter> возможен ввод новой строки текста. Текст автоматически не переносится, если пользователь выполняет ввод, превышая длину текстового поля.
- **Password (Пароль)**. Используя этот параметр, вполне возможно скрывать введенную пользователем информацию, не допуская ее отображения. Каждый вводимый символ отображается в виде звездочки (*), подобно тому, как это происходит для большинства полей с паролями в других общеизвестных пользовательских интерфейсах. Однако, если вы получите доступ к значению поля в выражениях ActionScript, реально введенные символы окажутся доступными.

Справа от меню Line type можно увидеть другие параметры, контролирующие процесс форматирования для редактируемого текстового поля. Первый параметр, Selectable (Способность к выделению), в случае его выбора позволяет пользователю выделять текст в пределах объекта `TextField`, если необходимо скопировать его и вставить в другое поле (или внешний документ). Справа от параметра Selectable находится параметр Enable HTML (Поддержка HTML), благодаря которому в пределах редактируемого объекта `TextField` можно применять дескрипторы форматирования HTML, присваивая значение с помощью

ActionScript. Правее параметра Enable HTML находится параметр Show border (Показать границу). При выборе этого параметра поле вводимого текста автоматически форматируется с помощью фона белого цвета и черной волосяной линии границы. Если параметр Show border не выбран, Flash Player отображает объект `TextField` в период авторского времени с помощью штрихового контура. Этот контур будет невидим при публикации и просмотре фильма с помощью Flash Player.

Удостоверьтесь в том, что цвет текста четко отличается от цвета фона. Довольно распространена ошибка, когда текст белого цвета располагается на белом же фоне. В этом случае текст будет невидимым. Если вы заметили, что текст не отображается, обратите внимание на корректность выбора его цвета.

Возле правой границы окна инспектора свойств находится поле с меткой Maximum Characters (Максимальное количество символов). Этот параметр устанавливает предельное количество символов, вводимых в текстовое поле. По умолчанию значением этого параметра является нуль (0). Это значит, что в поле можно вводить неограниченное количество символов.

Формирование текста во время выполнения

Текст, создаваемый во время выполнения, связан с объектом `TextField`, который формируется с помощью кода ActionScript. Если же текст создается в среде авторских работ, для этого применяется инструмент Text. Несмотря на то, что текст, создаваемый в среде авторских работ, обладает рядом преимуществ, он не позволяет поддерживать программный контроль на том уровне, который можно обеспечить, обращаясь к тексту в течение выполнения. Текст, создаваемый во время выполнения, обеспечивает следующие преимущества.

- Текст, создаваемый во время выполнения, можно формировать на основе содержимого, которое загружается из внешних источников, таких как XML-файлы и базы данных.
- Текст, формируемый в период выполнения, можно создавать при взаимодействии с пользователем.
- Если необходимо создать поледовательности текстовых блоков, именно текст, формируемый в период выполнения, облегчает эту работу.

Для создания текста в период выполнения существует только одна методика, которая применяется независимо от того, будет ли данный текст использоваться как вводимый либо динамический. Метод `createTextField()` может вызываться из любого объекта `MovieClip` для создания нового внедренного в него объекта `TextField`. В методе `createTextField()` применяются шесть параметров: название экземпляра (instance name), глубина (depth), x, y (координаты), ширина (width) и высота (height). Например, для создания нового объекта `TextField` под названием `tLabel` в пределах текущего объекта `MovieClip` можно воспользоваться следующим кодом:

```
this.createTextField("tLabel", this.↵
getNextHighestDepth(), 0, 0, 100, 20);
```

Предыдущий код формирует новый объект `TextField`, используя наибольшие значения глубины текста. Объект помещается в точке с координатами 0,0, а его размеры равны 100 на 20 пикселей.

Метод `createTextField()` *не* возвращает ссылку на заново созданный экземпляр `TextFIeld` несмотря на то, что иногда это предполагается исходя из принципов функционирования методов `attachMovie()` и `duplicateMovieClip()`.

По умолчанию при формировании нового объекта TextField в период выполнения, получим следующие его свойства:

```
autoSize = "none"
background = false
BackgroundColor = 0xFFFFFF
border = false
borderColor = 0x000000
condenseWhite = false
embedFonts = false
html = false
maxChars = null
multiline = false
password = false
restrict = null
selectable = true
textColor = 0x000000
type = "dynamic"
variable = null
wordWrap = false
```

В дальнейшем эти свойства рассматриваются более подробно.

Основные принципы работы с объектами TextField

Создание объекта TextField представляет собой только первый шаг. После этого необходимо приступать к работе с объектом, присваивать свойствам новые значения и вызывать методы. В следующих разделах детально рассматриваются операции, которые можно реализовать программными методами с помощью объектв TextField.

Основные свойства и методы класса TextField

Класс TextField располагает теми же основными свойствами и методами, что и классы Button и MovieClip. Поэтому в данной главе они рассматриваться не будут. Если у вас возникнут какие-либо вопросы относительно этих свойств и методов, обратитесь к главе 9, где содержится более детальное их описание.

Общими для перечисленных классов являются следующие свойства и методы: _accProps, _alpha, _focusrect, _height, _name, _parent, _rotation, _target, _url, _visible, _x, _xmouse, _xscale, _y, _ymouse, _yscale, tabEnabled, tabIndex и getDepth().

Многие из этих свойств могут применяться при выборе способа визуализации экземпляра. Многие из них используются так же, как и в случае с экземплярами объектов MovieClip или Button. К примеру, при перемещении объекта TextField под названием tLabel в позицию с координатами 100,100, код ActionScript примет следующий вид:

```
tLabel._x = 100;
tLabel._y = 100;
```

Однако имеется ряд свойств, которые не реализуются аналогичным образом с заданными по умолчанию установками для объекта TextField. Свойства _alpha и _rotation не могут применяться к объекту TextField при отсутствии внедренного шрифта. Если же попытаться установить одно из таких свойств для объекта TextField, не располагающего

внедренным шрифтом, экземпляр станет невидимым. Обратитесь к разделу “Внедренные шрифты” далее в этой главе для получения дополнительной информации.

Добавление текста

После создания объекта `TextField`, независимо от того, формируется ли он в среде авторских работ или во время рабочего цикла, вам необходимо выполнить следующее. Нужно предоставить для него новое текстовое значение, чтобы оно отобразилось в приложении. Для этого соответствующему свойству `text` нужно присвоить новое строковое значение. К примеру, если создан объект `TextField` под названием `tLabel`, можно добавить текст, который отображается следующим образом:

```
tLabel.text = "This is a label.";
```

Затем замените отображенный в объекте `TextField` текст путем присваивания нового значения свойству `text`. Например:

```
tLabel.text = "This is another label.";
```

Стандарт Unicode и объекты `TextField`

При работе с источниками текстовых данных, загружаемых во Flash-фильм в среде авторских работ или в период выполнения, следует выбрать кодировку UTF-8, UTF-16LE или UTF-16BE. Также среди значений для свойств объекта `TextField` можно использовать управляющие последовательности `\u`, например для таких свойств, как `text` и `restrict`. Если требуется просмотреть управляющую последовательность символов, откройте приложение Character Map в Microsoft Windows (для Windows XP можно выбрать следующие команды Start⇨Programs⇨Accessories⇨System Tools⇨Character Map (Пуск⇨Все программы⇨Стандартные⇨Служебные⇨Карта символов)).

При работе с ActionScript в свойстве `text` текстового поля можно указать следующий символ авторского права (`U+00A9`), что и показано в примере данного кода:

```
tArticle.text = "\u00A9";
```

Если вы не желаете применять управляющие последовательности, текст в кодировке UTF-8 можно указать в отдельном AS-файле, на который имеется ссылка в директиве ActionScript `#include`. Следующий текст можно найти в файле `title_multilanguage.as`, размещенном в папке `ch18` на прилагаемом к книге компакт-диске. Этот AS-файл сохранен из Notepad в Windows XP. Notepad, подобно другим текстовым редакторам (включая TextEdit из Mac OS 9), может сохранять текст в кодировке UTF-8.

В первой строке включенного AS-файла необходимо привести описание //!--UTF8, что позволит Flash MX правильно интерпретировать закодированные символы. В документе Flash к файлу `title_multilanguage.as` присоединяются следующие действия, текстовому полю `output` присваивается значение `title_english`:

```
#include "title_multilanguage.as"
this.createTextField("tOutput", this.getNextHighestDepth(),↵
25, 25, 150, 18);
tUutput.text = title_english;
```

Кодировку UTF-8, UTF-16LE и UTF-16BE также можно использовать с источниками динамических данных, которые загружаются во Flash-фильм в период выполнения с помощью классов `LoadVars` или `XML`.

При экспериментировании с текстом в формате Unicode можно обратиться к следующим сайтам, где содержится соответствующая информация:

```
babelfish.altavista.com
tarjim.ajeeb.com/ajeeb/default.asp?lang=1
```

Более подробная информация содержится в исчерпывающих учебных пособиях Macromedia по использованию Unicode для Flash MX (обратитесь к следующему Web-сайту):

`www.macromedia.com/support/flash/languages/unicode_in_flmx/`

Если в операционной системе используется несколько языков, необходимо удостовериться в установке необходимых языковых пакетов. К примеру, символы японского языка не будут отображаться в фильме Flash, если пакет японского языка не инсталлирован в вашей системе.

Управление многострочным текстом

Объекты `TextField` могут отображать содержимое, представленное одной или несколькими строками. При работе с динамическим текстом можно отображать содержимое с помощью одной или нескольких строк, независимо от значения, присущего объекту свойства `multiline`. Но при обращении к вводимому тексту имеются некоторые отличия. По умолчанию объект `TextField` представлен единственной строкой. Если же необходимо ввести несколько строк текста, следует присвоить свойству `multiline` данного объекта значение `true` (по умолчанию оно имеет значение `false`).

```
tLabel.multiline = true;
```

Для переноса слов рекомендуется иной подход. При обработке многострочного текста желательно, чтобы после достижения границы окна текст переносился на следующую строку. Для достижения этого эффекта с помощью ActionScript, необходимо свойству `wordWrap` присвоить значение `true`:

```
tLabel.wordWrap = true;
```

С другой стороны, если свойство `wordWrap` имеет значение `false`, текст будет продолжаться и после выхода за пределы ограничивающего окна, пока не отобразится символ прерывания строки.

Изменение размеров объекта TextField

Текст в объекте `TextField` претерпевает динамические изменения, поэтому нет необходимости знать его точные размеры в среде авторских работ. Вместо этого можно изменять размеры исходя из его содержания.

При обсуждении процедуры изменения размеров объекта `TextField` следует отличать изменение размеров от масштабирования. Если для объекта `TextField` устанавливается свойство `_xscale` и/или `_yscale`, то изменяется и масштаб отображаемого текста. Если необходимо изменить размер объекта `TextField`, речь идет об изменении размеров ограничивающего его окна наряду с сохранением размеров шрифта в данном тексте.

Самым простым вариантом по изменению размеров служит указание Flash по поводу автоматического изменения размеров `TextField` в соответствии с присваивамым текстом. Для этого необходимо присвоить подходящее значение свойству `autoSize` объекта. Ниже приводится перечень возможных значений.

- **none:** эта настройка задана по умолчанию и означает, что объект `TextField` не изменяет размеры в автоматическом режиме.
- **left:** означает, что объект автоматически изменяет размеры в соответствии с содержимым, используя в качестве фиксированной точки левый верхний угол экземпляра объекта `TextField`.

- **right:** означает, что объект автоматически изменяет размеры, как это происходит и в случае выбора параметра `left`, но в качестве отправной точки применяется правый верхний угол соответствующего экземпляра объекта.
- **center:** объект автоматически изменяет размеры, как это происходит в случае выбора параметров `left` либо `right`, в качестве отправной точки используется область сверху от центра.

Если свойству `autoSize` присвоить булево значение, оно станет "работоспособным". Значение `true` является синонимом значения `left`, а значение `false` служит синонимом значения `none`. Однако желательно применять корректные строковые значения, особенно учитывая тот момент, что ActionScript все больше ориентируется в сторону строгой типизации.

Для того чтобы представить, каким образом свойство `autoSize` влияет на изменение размеров текста, удобно обратиться к реальному примеру. Благодаря следующему коду формируются три объекта `TextField`, к ним добавляется текст, а также рассматриваются различные возможности по изменению его размеров:

```
this.createTextField("tLabel1", this.getNextHighestDepth(), ↵
0, 0, 0, 0);
tLabel1.text = "Label one";
tLabel1.autoSize = "left";
this.createTextField("tLabel2", this.getNextHighestDepth(), ↵
0, 25, 0, 0);
tLabel2.text = "Label two";
tLabel2.autoSize = "right";
this.createTextField("tLabel3", this.getNextHighestDepth(), ↵
0, 50, 0, 0);
tLabel3.text = "Label three";
tLabel3.autoSize = "center";
```

На рис. 17.3 показано, что именно отображается на экране при тестировании предыдущего кода. Вертикальная линия добавляется, если значением `x` является 0, предоставляя в ваше распоряжение точку отсчета.

Рис. 17.3. Примеры трех объектов `TextField` с различными значениями параметра `autoSize`

Конечно, также можно присвоить новые значения свойствам `_width` и `_height`. Тогда ограничивающее окно изменит размеры по ширине с учетом определенного числа пикселей. Свойства `textWidth` и `textHeight` возвращают значения ширины и длины текста, включенного в объект `TextField` (в пикселях), что довольно удобно при реализации некоторых несложных эффектов по изменению размеров. При этом Flash размещает вокруг текста в пределах объекта `TextField` границу шириной в два пикселя. При использовании свойств `textWidth` и/или `textHeight` обратите внимание на этот момент, если вам понадобится добавить к этим значениям еще четыре пикстеля. Следующий код формирует объект `TextField`, размеры котрого соответствуют ширине текста, но для высоты вводятся фиксированные 20 пикселей:

```
this.createTextField("tSample", this.getNextHighestDepth(),↵
0, 0, 0, 0);
tSample.text = "This is\nsome\nsample\ntext.";
```

```
tSample._height = 20;
tSample._width = tSample.textWidth + 4;
```

Предыдущий пример может быть полезен, если необходимо создать объект TextField, имеющий определенное число пикселей на одной из сторон и четко соответствующий данному тексту на другой стороне. Затем с помощью свойств прокрутки можно выполнить скроллинг текста, как показано в разделе "Скроллинг текста" далее в этой главе.

Свойства textWidth и textHeight поддерживают в пределах объекта TextField довольно упрощенную информацию о размерах текста. Для обработки более сложной информации можно обратиться к методу getTextExtent() из объекта TextFormat, как показано в главе 18.

Как сделать текст невыделяемым

По умолчанию предполагается, что текст можно выделять. Речь идет о том, что пользователь может с помощью курсора мыши выделять и копировать текст, а также размещать курсор в пределах экземпляра для ввода (в случае вводимого текста). В некоторых ситуациях подобная установка оказывается предпочтительной. Например, вводимый текст должен быть выделяемым. Если необходимо, чтобы пользователь копировал текст или производил вставки, следует удостовериться в том, что текст выделяемый. Однако иногда текст должен быть невыделяемым. Если текст применяется в пределах кнопки, он должен быть невыделяемым, чтобы на него не оказывала влияние регистрация состояния кнопки.

Состояние, связанное с возможностью выделять объект TextField, регулируется свойством selectable. Заданным по умолчанию значением является true, т.е. пользователь может выделять текст. При установке значения false экземпляр становится невыделяемым.

```
tLabel.selectable = false;
```

Установка границы и фона

По умолчанию объект TextField отображается без использования границ и фона. Однако можно программными методами контролировать целесообразность его отображения.

Свойство border установлено по умолчанию равным false. Если выбрать значение true, Flash отобразит вокруг экземпляра объекта TextField границу в виде волосяной линии:

```
tLabel.border = true;
```

Аналогично, свойство background по умолчанию имеет значение false, но при установке этого свойства к значению true, фоновая заливка отобразится позади текста в пределах объекта TextField:

```
tLabel.background = true;
```

По умолчанию граница окрашена черным цветом, а фон — белым. Цвета можно изменять программными методами с помощью свойств borderColor и backgroundColor. В этих свойствах используются числовые значения, представляющие тот цвет, который необходимо присвоить границе и/или фону. Ниже в экземпляре под названием tLabel выполняется присваивание границе красного цвета, а фону — желтого цвета:

```
tLabel.borderColor = 0xFF0000;
tLabel.backgroundColor = 0xFFFF00;
```

Создание вводимого текста

Как упоминалось ранее в главе, и динамический текст, и вводный являются объектами TextField. Flash различает эти виды текста с помощью единственного свойства: type.

Свойство type имеет два значения: dynamic либо input. Заданным по умолчанию значением является dynamic. При выборе значения input объект превращается во вводимый объект TextField, а пользователь получает возможность вводить или вставлять в него текст. Например, ниже формируется новый объект под названием tUsername, свойству type присваивается значение input, что позволяет реализовать ввод данных пользователем:

```
this.createTextField("tUsername", this.getNextHighestDepth(), ↵
0, 0, 100, 20);
tUsername.type = "input";
```

Обычно при создании вводимого текста необходимо удостовериться в том, что граница отображается, иначе пользователь не будет знать, куда следует вводить текст:

```
tUsername.border = true;
```

Контроль ввода

При работе с вводимым текстом вполне можно контролировать процесс ввода. Для выполнения этой задачи ActionScript предлагает две возможности. Можно присвоить максимальное число символов, которые допускается вводить, либо определить, какие символы запрещены для ввода.

Свойство maxChars контролирует максимальное число символов, которое может вводить пользователь. По умолчанию это свойство имеет значение null, что свидетельствует о неограниченном числе символов, которые можно ввести в поле. Независимо от значения этого свойства, код ActionScript располагает неограниченным доступом для добавления содержимого. Это значение строго контролируется количеством текста, разрешенного пользователем. В следующем примере для экземпляра под названием tUsername значением 10 ограничивается количество вводимых символов:

```
tUsername.maxChars = 10;
```

Свойство restrict контролирует допустимость символов для определенного объекта. Как следует из названия, это свойство можно применять для ограничения диапазона символов, вводимых пользователем в поле. Значением этого свойства является строка, определяющая допустимые и/или недопустимые символы (или диапазоны символов).

Даже в том случае, когда поле ограничено, с помощью ActionScript можно вставлять столько текста, сколько это необходимо. Свойство restrict может воспрепятствовать пользователю ввести в поле нежелательные символы.

Для указания разрешенных символов просто введите эти символы в строковое значение. Данные величины существенно зависят от регистра. Следующий код указывает полю tArticle на доступность только символов A, B, C или D:

```
tArticle.restrict = "ABCD";
```

Для указания диапазона символов также можно применять дефис (-). Следующий код представляет иной способ установления символов от A до D:

```
tArticle.restrict = "A-D";
```

В одной строке можно указать несколько диапазонов. Следующий код разрешает пользоваться символами, от A до D, символом пробела и цифрами от 1 до 4:

```
tArticle.restrict = "A-D 1-4";
```

Для пропуска нежелательных символов или их указания необходимо предварить данный символ (либо диапазон символов) символом ^. Следующий код предлагает для текстового поля под названием `article` в качестве разрешенных все символы, кроме 0–9:

```
tArticle.restrict = "^0-9";
```

Для указания набора разрешенных и запрещенных символов можно пользоваться всеми предложенными синтаксическими элементами. Следующий код разрешает применять все буквы (обязательно в верхнем регистре), исключая при этом применение букв в нижнем регистре.

```
tArticle.restrict = "A-Z^a-z";
```

Заметим, что в предыдущей строке кода не разрешен символ пробела. Символ пробела необходимо указать непосредственно в ограничительном значении.

Для указания одного из синтаксических операторов, которые применяются для обозначения диапазонов применимости (-) или пропусков (^), предварите оператор обратной косой чертой. Следующий код воспрепятствует пользователю ввести знак минуса (-) в текстовое поле `tArticle`. Также символ обратной косой черты может применяться в паре, что означает указание на одинарный символ обратной косой черты, т.е. отмена использования символа (\) отображается как \\:

```
tArticle.restrict = "^\-";
```

Символы Unicode также можно включать и исключать из текстовых полей с помощью управляющих последовательностей `\u`. Обратитесь к примечанию "Стандарт Unicode и объекты TextField", которое содержится в этой главе.

Создание текста пароля

Если поля ввода создаются для потенциально чувствительных к различным изменениями данных, например, для паролей, можно превратить поле ввода в поле для пароля. Это значит, что вместо отображения вводимого пользователем текста, Flash отобразит символы "звездочка". Тогда другие пользователи не смогут прочесть эти данные. Любое поле вводимого текста можно преобразовать в поле для пароля, устанавливая значение `true` для соответствующего свойства `password`:

```
tPassword.password = true;
```

Важно понимать, что поле для ввода пароля не позволяет выполнять какие-либо действия, только представляет способ отображения содержимого в приложении. Вводимые пользователем данные остаются доступными посредством кода ActionScript. Это значит, что применение поля для пароля не позволяет надеяться на реализацию в приложении дополнительных мер безопасности либо возможностей по кодировке. Следует тщательно следить, чтобы секретные данные не пересылались из приложения Flash без шифрования. Обычно в данном случае применяется технология SSL, которая поддерживается Flash.

Более подробные сведения по поводу SSL можно найти на Web-сайте по адресу `www.netscape.com/eng/security/`.

Изменение цвета текста

Для изменения цвета текста имеется ряд возможностей:

- присваивание нового значения свойству `textColor` объекта `TextField`;
- применение HTML и дескриптора `<font>`;
- использование объекта `TextFormat`;
- применение каскадных таблиц стилей.

Свойство `textColor` может использоваться для применения единственного цвета ко всему тексту в объекте `TextField`. Для этого необходимо лишь присвоить свойству числовое значение. К примеру, следующий код приведет к окрашиванию в красный цвет всего текста в объекте `TextField` под названием `tLabel`:

```
tLabel.textColor = 0xFF0000;
```

Код HTML, класс `TextFormat` и параметры CSS позволяют реализовать более полный контроль над цветом (речь идет о применении различных цветов к разным текстам) в пределах объекта `TextField`. Решение, связанное с HTML, обсуждается более подробно в следующей части данной главы. Класс `TextFormat` и параметры CSS рассматриваются в главе 18.

Применение строковых шестнадцатеричных значений для кодирования свойств цвета

Многие разработчики Flash и студенты, изучающие возможности ActionScript, часто интересуются, можно ли преобразовать строковое значение, содержащее шестнадцатеричное значение, в число, которое бы распознавалось объектными методами и свойствами ActionScript, основанными на использовании числовых типов данных, а не строк. Подобные методики предусматривают использование метода `setRGB()` из объекта `Color`, а также многочисленных свойств цвета из класса `TextField`. Например, следующий код не будет корректным в ActionScript:

```
tArticle.textColor = "0xFFFFFF";
```

Свойство `textColor` использует числовой тип данных. Суть возникшей проблемы достаточно понятна. Можно преобразовать строковое значение, использующее шестнадцатеричное число, в числовой тип данных с помощью функции `parseInt()`. Следующий код преобразовывает строковую переменную под названием `nHexColor` в числовой тип данных для свойства `textColor` объекта `TextField`:

```
nHexColor = "FF0000";
article.textColor = parseInt("0x" + nHexColor);
```

Важность подобного подхода может быть очевидной не сразу. Но что можно предложить для поддержки вводного текстового поля, чтобы пользователь имел возможность указывать шестнадцатеричное значение для элемента пользовательского интерфейса? Результат ввода пользователя в текстовое поле воспринимается как строковые данные, которые следует преобразовать в числовые данные. И снова, эту работу можно выполнить с помощью метода `parseInt()`. Другим вариантом задействования этого процесса служит преобразование любых текстовых значений, загружаемых из внешнего источника данных. К примеру, пусть имеется база данных для заказных цветов, применяемых Flash-фильмом. Если фильм загружает эти данные, они вводятся в виде строки. С помощью функции `parseInt()` можно преобразовать в число количество загружаемых переменных.

Удаление текста

Объекты `TextField` можно удалять программными методами только в том случае, если они созданы аналогичным образом. Можно вызвать метод `removeTextField()` из объекта `TextField`. Если объект создавался с помощью метода `createTextField()`, данный объект будет удален из рабочего поля:

```
tLabel.removeTextField();
```

Создание простого приложения для хранения заметок

В этом упражнении вы получите возможность применить на практике большое число понятий и приемов, рассмотренных в этой главе. Создается очень простое приложение для хранения заметок, куда пользователь может войти для просмотра либо внесения заметок. Причем сами заметки хранятся в общедоступном локальном объекте. Более подробная информация о локальных общедоступных объектах содержится в главе 25.

Данное приложение поддерживает несложный процесс входа перед тем, как пользователь получит возможность просмотра и/или обновления заметок. Основная цель состоит в иллюстрации различных основных свойств объекта `TextField` в контексте. Поэтому процесс регистрации в данном случае является небезопасным. Имя пользователя и пароль сохраняются в SWF-файле, их может легко просмотреть любой пользователь. Комбинации имени пользователя и пароля для приложений, где реализуются меры безопасности, должны сохраняться во внешних хранилищах, например, в базе данных.

1. Откройте файл `notes_starter.fla`, находящийся на прилагаемом к книге компакт-диске, и сохраните его на локальном диске под именем `notes001.fla`.
2. Откройте библиотеку, чтобы можно было видеть два символа Movie Clip, добавленные к FLA-файлу. Если просмотреть настройки связывания для этих двух символов, можно заметить, что они установлены для экспорта в ActionScript.
3. Добавьте в первый кадр основной временной шкалы код ActionScript, показанный в листинге 17.1.

Листинг 17.1. Функция `createLoginScreen()`

```
function createLoginScreen():Void {

   // Создайте объекты TextField и MovieClip.
   this.createTextField("tUsername", ⏎
   this.getNextHighestDepth(), 100, 100, 200, 20);
   this.createTextField("tPassword", ⏎
   this.getNextHighestDepth(), 100, 140, 200, 20);
   this.createTextField("tMessage", ⏎
   this.getNextHighestDepth(), 100, 60, 200, 20);
   this.attachMovie("LoginButtonSymbol", "mcLoginButton", ⏎
   this.getNextHighestDepth());
   // Установите свойства объектов TextField.
   tUsername.border = true;
```

```
    tPassword.border = true;
    tUsername.type = "input";
    tPassword.type = "input";
    tPassword.password = true;
    tMessage.textColor = 0xFF0000;

    // Поместите кнопку.
    mcLoginButton._x = 100;
    mcLoginButton._y = 180;

    mcLoginButton.onRelease = function():Void {

        // Проверьте, правильное ли имя и пароль ввел
        // пользователь.Если это так, вызовите функцию
        // login().
        // В противном случае отобразите сообщение
        // пользователю и удалите значения из полей
        // регистрации объектов TextField.
        if(tUsername.text == "admin" && ↵
        tPassword.text == "admin") {
            login();
        }
        else {
            tMessage.text = "Try again.";
            tUsername.text = "";
            tPassword.text = "";
        }
    };
}

function login():Void {

    // Удалите объекты TextField и MovieClip, которые
    // образуют регистрационный экран.
    tUsername.removeTextField();
    tPassword.removeTextField();
    mcLoginButton.removeMovieClip();

    // Создайте объекты TextField и MovieClip для
    // экрана заметок
    this.createTextField("tNotes", ↵
    this.getNextHighestDepth(), 100, 100, 350, 200);
    this.attachMovie("SaveButtonSymbol", "mcSaveButton", ↵
    this.getNextHighestDepth());

    // Установите свойства объекта TextField.
    tNotes.border = true;
    tNotes.type = "input";

    // Поместите кнопку.
    mcSaveButton._x = 100;
    mcSaveButton._y = 320;

    // Откройте локально общедоступный объект.
    var lsoNotes:SharedObject = SharedObject.↵
    getLocal("notes");

    // Присвойте сохраняемый текст, если таковой имеется.
```

```
    tNotes.text = (lsoNotes.data.notes == ↵
    undefined) ? "" : lsoNotes.data.notes;

    // Когда пользователь выполнит щелчок и отпустит
    // кнопку, сохраните текущие примечания
    // в общедоступном объекте.
    mcSaveButton.onRelease = function():Void {
        lsoNotes.data.notes = tNotes.text;
        lsoNotes.flush();
    }
}

createLoginScreen();
```

4. Протестируйте фильм.

Если попытаться зарегистрироваться с помощью некорректной комбинации имени пользователя и пароля, появится сообщение, отображаемое в виде текста красного цвета. С другой стороны, если регистрироваться с помощью комбинации admin/admin имя пользователя/пароль, вы получите доступ к экрану заметок.

Применение кода HTML совместно с объектами TextField

Объекты `TextField` не только отображают обычный текст, они также могут отобразить HTML-код. Flash поддерживает только подмножество дескрипторов HTML, представленных ниже.

- **<a>**: Flash поддерживает только атрибуты `href` и `target` для дескриптора `anchor`. То есть для создания гиперссылок в тексте можно применять `<a>`.
- **
**: дескриптор `<br>` формирует разрыв строки в тексте.
- **<font>**: для этого дескриптора поддерживаются атрибуты цвета, гарнитуры и размера. Именно `<font>` удобно применить при выполнении простого форматирования частей текста.
- **<p>**: для этого дескриптора поддерживаются атрибуты выравнивания и класса (`align` и `class`). Атрибут выравнивания позволяет выровнять текст слева, справа и по центру. Атрибут класса применяется с каскадными таблицами стилей. Эта новая возможность во Flash обсуждается более подробно в главе 18.
- ****: для этого дескриптора поддерживается атрибут `class`, поэтому он применяется только вместе с каскадными таблицами стилей (см. главу 18).
- **<b>**, **<u>** и **<i>**: Эти дескрипторы приводят к тому, что текст отображается с применением полужирного шрифта, подчеркнутым либо курсивным соответственно.
- **<li>**: данный дескриптор создает элемент списка. Элементы списка отображаются с отступом и с применением слева от них символов буллитов.

Также Flash обеспечивает поддержку дескрипторов `<img>` и `<textformat>`. Поддержка дескриптора `<img>` подробно рассматривается в разделе “Встроенное в текст содержимое”. Дескриптор `<textformat>` является уникальным для Flash, обычно он широко используется с объектом `TextFormat`. Более подробная информация относительно объектов `TextFormat` представлена в главе 18.

Интерпретация HTML-кода в тексте

По умолчанию Flash-объекты `TextField` буквально отображают текст. То есть, если вы запросите `TextField` отобразить следующее:

```
<font color="#FF0000">Red Text</font>
```

Flash отобразит текст буквально вместо визуализации кода HTML и отображения только `Red Text` с примененным к нему красным цветом. Если необходимо, чтобы Flash интерпретировал данный текст в объекте `TextField` в виде HTML-кода, следует указать ему на это. С этой целью свойства объекта `html` устанавливают равными `true`. Например, следующий код указывает Flash на то, что объект `TextField` под названием `tTitle` следует отображать в виде HTML-кода:

```
tTitle.html = true;
```

Если объект установлен для отображения HTML-кода, необходимо все HTML-содержимое присвоить свойству `htmlText` объекта. Даже если объект `TextField` может быть и установлен для отображения HTML-кода, при присвоении значения свойству `text`, он по-прежнему отображается в буквальном смысле. Ниже показан пример присвоения HTML содержания свойству `htmlText` из `tTitle`:

```
tTitle.htmlText = "<b>ActionScript Bible</b>";
```

Хотя свойство `html` для объекта `TextField` имеет значение `true`, по-прежнему отображается разделитель строк (newline), множественные последовательные пробелы, а такж символы дескриптора. Как вам наверняка известно, обычно HTML-код отображается иным образом. Например, в HTML символы разделителя строк не отображаются. Символы разрывов строк отображаются только тогда, когда указан дескриптор `<br>`. Иногда удобно воспользоваться одним из вариантов, иногда же предпочтителен другой вариант. Flash может поддерживать отображение символов служебных пробелов (пробелов, символов разделителя строк и т.д.) либо не отображать их, если в объекте `TextField` включена поддержка HTML-кода. Заданная по умолчанию установка поддерживает отображение подобных символов. При установке значения свойства `condenseWhite` равным `false`, Flash "игнорирует" символы пробела (отличные от одиночных пробелов), поскольку HTML-код обычно интерпетируется в Web-браузере.

Вставка специальных символов в поля HTML

Довольно часто возникает необходимость в использовании в текстовых полях HTML текстовых символов, не включенных в обычный буквенно-числовой набор (т.е. алфавит и числа). И хотя большинство символов, не включенных в этот набор, можно выбрать с помощью клавиатуры, применяя комбинации с клавишей <Shift> (например, комбинация <Shift+2> приводит к символу @), необходимо кодировать некоторые символы (которые иногда невозможно ввести с помощью клавиатуры) для правильного их использования в пределах текстовых полей HTML. К примеру, если в текстовом поле HTML следует отобразить знак "больше чем" (>) или знак "меньше чем" (<), вы не увидите их при простом вводе данных символов в выражение ActionScript. ActionScript интерпретирует знаки "меньше чем" и "больше чем" как открывающиеся и закрывающиеся символы (соответственно) в дескрипторах HTML. Поэтому при вводе в строковое значение, используемое в поле HTML символа < или символа >, ActionScript воспримет окружающий текст как часть дескриптора HTML и не отобразит его. Это можно проверить с помощью следующего кода:

```
this.createTextField("tInfo", this.getNextHighestDepth(),↵
0, 0, 0, 0);
```

```
tInfo.autoSize = true;
tInfo.html = true;
tInfo.htmlText = "< is a less than sign";
```

Если предыдущий код поместить в первом кадре нового документа Flash и протестировать фильм, вы увидите, что текст не отобразится.

Итак, если необходимо указать ActionScript на обязательное буквальное отображение специальных символов, следует кодировать данный символ в виде *имени сущности* или *десятичной символьной ссылки*. Возможно, вы встречали имена сущностей в обычном HTML. Именем сущности для знака "меньше чем" будет `<`. Если это имя вставить в выражение ActionScript, Flash Player верно отобразит символ <:

```
tInfo.htmlText = "&lt; is a less than sign";
```

При тестировании фильма с именем сущности вместо литерала <, в фильме отобразится < `is the less than sign`. Большое число сущностых имен HTML, таких как `&copy` для символа авторского права, *не* функционируют во Flash ActionScript. Для таких символов применяется десятичная символьная ссылка, например:

```
tInfo.htmlText = "&#169; is the copyright sign.";
```

Список распространенных символьных имен сущностей (либо десятичных символьных ссылок) содержится в табл. 17.1.

Таблица 17.1: Специальные символы для текстовых полей HTML

Символ	Название	Значение ActionScript	Имя Unicode
<	Знак меньше чем	`<`	`\u003C`
>	Знак больше чем	`>`	`\u003E`
"	Двойные кавычки	`&quot;`	`\u0022`
&	Амперсанд	`&`	`\u0026`
•	Буллит	`&#149;`	`\u2022`
¶	Знак Пилкроу	`&#182;`	`\u00B6`
©	Знак авторского права	`&#169;`	`\u00A9`
™	Знак регистрации	`&#174;`	`\u00AE`
™	Знак торговой марки	`&#153;`	`\u2122`
ε	Знак фунта	`&#163`	`\u00A3`
¢	Знак цента	`&#162;`	`\u00A2`
•	Знак степени	`&#176;`	`\u00B0`
:	Знак деления	`&#247;`	`\u00F7`

Некоторые из приведенных выше ASCII-символов будут неправильно отображаться во Flash Player 6 или поздних версиях, если новому свойству `System.codePage` не присвоить корректное значение. Более подробная информация содержится в главе 21. Если вы не желаете применять другое значение `codePage`, также можно воспользоваться значением Unicode (как показано в табл. 17.1) для непротиворечивого отображения во Flash Player 6 или более поздних версиях.

Устранение проблем, связанных со свойством htmlText

Даже при удачном вызове свойства htmlText для данного объекта TextField могут возникнуть проблемы. Если HTML-отформатированный текст добавить к существующему в текстовом поле HTML тексту, перед новым добавлением отобразится символ разрыва строки, поскольку дескриптор <P> автоматически вставляется вокруг новых добавленных фрагментов текста. Например, следующий код формирует две отдельные строки текста в многострочном текстовом поле даже в том случае, когда не указаны дескрипторы <BR>, <P> или символы \r:

```
this.createTextField("tArticle", this.getNextHighestDepth(), ↵
25, 35, 300, 100);
tArticle.html = true;
tArticle.multiline = true;
tArticle.wordWrap = true;
tArticle.htmlText = "<B>Bold text</B>";
tArticle.htmlText += "<I>Italic text</I>";
```

Чтобы избежать нежелательных разрывов строк, можно сразу установить свойство htmlText или сохранять текущее содержимое в новой переменной String, либо устанавливать свойство htmlText таким образом, чтобы его значение соответствовало значению новой переменной. Последнее решение и получило отражение в следующем коде:

```
this.createTextField("tArticle", this.getNextHighestDepth(), ↵
25, 35, 300, 100);
tArticle.html = true;
tArticle.multiline = true;
tArticle.wordWrap = true;
var sTempHTML:String = "<B>Bold text</B>";
sTempHTML += "<I>Italic text</I>";
tArticle.htmlText = sTempHTML;
```

Со свойством htmlText можно связать и другую проблему, возникающую в том случае, когда оно используется в циклическом коде, таком как цикл for или while. Если требуется выполнить несколько операций вставки в текстовое поле, Flash Player может более эффективно визуализировать текст в виде HTML-кода при условии, что к свойству htmlText обращались только один раз. Ниже приводится примерный код, который потенциально замедлит работу Flash Player. Текстовое поле вставляется в цикле for, который выполняется 50 раз, при каждом проходе добавляя HTML-форматированный текст:

```
this.createTextField("tArticle", this.getNextHighestDepth(), ↵
25, 35, 300, 100);
tArticle.html = true;
tArticle.multiline = true;
tArticle.wordWrap = true;
trace(getTimer());
for(var i:Number = 0; i < 100; i++) {
   tArticle.htmlText += "<b>item</b><br>";
}
trace(getTimer());
```

Данный код вставляется в пустой документ Flash, второе действие trace() полностью выполняется в течение некоторого времени, вследствие чего в панели Output отображается некий результат. Несмотря на то, что будет изменяться реальное отличие между значением оператора for, сообщаемым до этого, и значением, сообщаемым после; тесты показывают значения, находящиеся в диапазоне от 3 до 7 сек. Затем рассмотрим, каким образом можно, лишь немного изменив код, резко сократить время исполнения:

```
this.createTextField("tArticle", this.getNextHighestDepth(),↵
25, 35, 300, 100);
tArticle.html = true;
tArticle.multiline = true;
tArticle.wordWrap = true;
```

```
trace(getTimer());
var sTempHTML:String = "";
for(var i:Number = 0; i < 100; i++) {
sTempHTML += "<b>item</b><br>";
}
tArticle.htmlText = sTempHTML;
trace(getTimer());
```

Предыдущий код будет выполняться не более 150 мс на достаточно быстром компьютере. Вот в чем отличие!

Добавление гиперссылок в текст

Дескриптор <a> можно применять для добавления гиперссылок к тексту. Например, ниже показано, каким образом формируется гиперссылка в объекте TextField под названием tContent:

```
tContent.html = true;
tContent.htmlText = "<a
href='http://www.person13.com'>www.person13.com</a>";
```

Удостоверьтесь, что знаки кавычек применяются правильно. Обратите внимание, что в предыдущем примере внутренние знаки кавычек являются одинарными, что устраняет возможный конфликт с внешними знаками кавычек. Если вы предпочитаете применять только двойные или же только одинарные знаки кавычек, тогда обязательно отделите (to escape) внутренние кавычки символом обратной косой черты. Более подробные сведения по этому вопросу содержатся в главе 15.

Гиперссылки в тексте Flash могут вести себя иначе, чем стандартные гиперссылки в HTML. Обратите особое внимание на цель, с которой открываются новые ссылки. По умолчанию целью открытия гиперссылки является текущий кадр браузера. Это значит, что если приложение Flash выполняется в окне браузера, то текущий кадр браузера заменяется новым содержанием. Если необходимо обратиться к другой цели, можно воспользоваться целевым атрибутом в дескрипторе <a>. К примеру, ниже показано, как открывается ссылка в новом окне браузера:

```
tContent.htmlText = "<a href='http://www.person13.com'
target='_blank'>www.person13.com</a>";
```

Также отметим, что, в отличие от большинства Web-браузеров, Flash не визуализирует гиперссылки так, чтобы они отличались от оставшейся части текста. Если вы привыкли обращаться к Web-браузерам при просмотре HTML-кода, тогда, возможно, вам известно, что гиперссылки обычно подчеркиваются и выделяются синим цветом (или пурпурным, если они просматривались). Поскольку Flash не различает гипертекст с помощью каких-либо визуальных методик, можно добавить несложное форматирование, с целью уточнить, является ли текст связанным. К примеру, включение дескрипторов <font> и <u> поможет форматировать связанный текст так, что его можно отобразить в окне Web-браузера:

```
tContent.htmlText = "<font color='#0000FF'><u>
<a href='http://www.person13.com'
target='_blank'>www.person13.com</a></u></font>";
```

Добавление почтовых ссылок

Среда Flash поддерживает несколько других директив href, включая директиву mailto. Директива mailto позволяет открывать новое электронное сообщение с указанным электронным адресом.

Директива `mailto` в тексте Flash играет ту же роль, что и директива `mailto` для обычной Web-страницы в HTML. Когда пользователь щелкает на ссылке, в заданном по умолчанию клиенте электронной почты, например в Microsoft Outlook (или Outlook Express) открывается новое электронное сообщение. Пользователь заинтересован в реальной пересылке электронного сообщения; директива `mailto` просто открывает новое окно для электронных сообщений с предварительно указанным полем `to` (также возможно указание полей `subject` (тема) и `body` (тело)).

Для этого просто воспользуйтесь дескриптором `<a>`, как показано в предыдущем разделе, но примените директиву `mailto`, следом за которой укажите двоеточие и электронный адрес, куда необходимо переслать новое сообщение. Например:

```
tContent.htmlText = "<a href="mailto:joey@person13.com">↵
send email</a>";
```

Вызов функций JavaScript

Если Flash-приложение выполняется в окне Web-браузера, который поддерживает JavaScript, функции JavaScript можно вызвать с помощью директивы `javascript` в составе дескриптора `<a>`. Рекумендуем также заключить все вызовы функций JavaScript в пределы оператора `void()`, чтобы не открывались другие окна. Ниже показано, каким образом можно добавить простую ссылку, которая приводит к появлению всплывающего окна с предупреждением JavaScript:

```
tContent.htmlText = "<a href=\"javascript:void(alert(
'This is a message from Flash'));\">click this text</a>";
```

Конечно, также можно воспользоваться директивой `javascript` для вызова настраиваемых функций JavaScript, определенных в пределах главной HTML-страницы.

Более подробную информацию о поддержке со строны браузеров и их функциональных возможностях можно найти на Web-странице по адресу `www.macromedia.com/support/flash/ts/documents/browser_support_matrix.htm`.

Вызов из текста функций ActionScript

Одна из малоизвестных, но довольно полезных возможностей состоит в вызове из текста функций ActionScript. Директива `asfunction` может применяться в пределах дескриптора `<a>` для вызова функции ActionScript. Эта функциональная возможность разрешает вносить в текст дополнительные интерактивные свойства.

Синтаксис `asfunction` имеет следующий вид:

`asfunction:`*`имя_функции`* `[,`*`парам1`*`,…,`*`парамN`*`]`

Между директивой, двоеточием, названием функции, запятыми или параметрами не должно быть пробелов. В синтакисе разрешены пробелы только в пределах параметрических значений.

Ниже приводится пример, где вызывается оператор `trace()` и ему передается значение `a message`:

```
tContent.htmlText = "<a href='asfunction:↵
trace,a message'>click this text</a>";
```

Также директива `asfunction` может применяться для вызова заказных функций. Например:

```
tContent.htmlText = "<a href='asfunction:changeTextColor'>↵
   click to change color</a>";
function changeTextColor():Void {
   tContent.textColor = Math.random() * 255 * 255 * 255;
}
```

Важно отметить, что директиву `asfunction` можно использовать со статическим текстом. Директиву `asfunction` можно вводить в поле URL link (URL-ссылка) в окне инспектора свойств для статического текста. Например:

```
asfunction:trace,a message from static text
```

Внедрение содержимого в текст

Одной из новых возможностей Flash MX 2004 является внедрение содержимого в поддерживающие HTML объекты `TextField`. Данное содержимое может представлять собой внешний файл в формате JPG (не прогрессивный JPEG) или SWF-файл, либо символ Movie Clip с идентификатором связывания. До обсуждения этой возможности заметим, что сегодня подобный подход трудно признать оптимальным. Несмотря на то, что к этой возможности обращаются довольно часто, следует учесть, что тестирование не дает однозначного перечня обстоятельств, влияющих на реализацию данной возможности. Если попытаться внедрить в данный текст несколько содержательных компонентов, они накладываются друг на друга таким образом, что некоторые из них не просматриваются вовсе. Если создавать объекты `TextField` программными методами и попытаться добавлять внедренное содержание в имеющийся текст, окажется, что данная возможность просто не работает. Для ознакомления с обновлениями плейера следует обратиться к Web-сайту Macromedia Web, где отражены подобные вопросы.

Несмотря на то, что данная возможность связана с некоторыми проблемами, Flash Player 7 поддерживает форматы JPEG, SWF и содержимое Movie Clip внедряетсяя в объекты `TextField`.

Исходя из вышесказанного обсудим, как применить эту возможность на практике. Дескриптор HTML `<img>` "разрешает" внедрять изображение в стандартный HTML-документ. Flash несколько расширяет данную функциональную возможность, поддерживая применение дескриптора `<img>` в пределах поддерживающего HTML объекта `TextField`, что позволит отображать не только изображения, но также SWF-содержание и содержимое символа Movie Clip. К примеру, для отображения некоторого изображения в объекте `TextField` можно применить следующий код:

```
this.createTextField("tContent",
this.getNextHighestDepth(), 0, 0, 200, 200);
tContent.border = true;
tContent.html = true;
tContent.htmlText = "A picture of a lake: <img width='180' ↵
   height='120' src='http://www.person13.com/asb/image2.jpg'>";
```

Заметим из предыдущего примера, что поддержка Flash для дескриптора `<img>` включает атрибуты `width` и `height`. Ниже приводится полный перечень поддерживаемых атрибутов.

- **width:** ширина в пикселях, которая используется при отображении содержания.
- **height:** высота в пикселях, которая используется при отображении содержания.
- **src:** URL-ссылка, или идентификатор связывания для содержимого. При указании URL можно использовать абсолютную или относительную URL-ссылку. В качестве альтер-

нативы также можно отобразить содержимое MovieClip, добавляемое из символа в библиотеке с помощью идентификатора связывания.

- **align:** горизонтальное выравнивание содержимого в пределах объекта TextField. Можно выравнивать слева (по умолчанию), справа или от центра.
- **hspace:** количество пикселей, которые отделяют содержимое в вертикальном направлении. По умолчанию это значение равно 8.
- **vspace:** количество пикселей, которые отделяют содержимое в горизонтальном направлении. По умолчанию это значение равно 8.
- **id:** идентификатор для содержимого, который может применяться для целевого направления встроенного содержимого с помощью кода ActionScript.

Заметим, что при загрузке графического содержимого в объект TextField, оно загружается асинхронно. Это значит, что TextField инициализируется на базе текстового содержимого. Поэтому однократно загруженное графическое содержимое не вызывает повторной инициализации TextField. Например, если при загрузке изображения в объект TextField свойству autoSize присваивается значение true, объект не изменяет соответствующим образом размеры с учетом загружаемого изображения (хотя тестирование и показывает, что размеры изменяются в вертикальном направлении). Кроме того, заметим, что дескриптор <img> не может играть роль первого содержимого в пределах объекта TextField, сформированного в период выполнения. В подобных случаях дескриптор <img> очевидным образом игнорируется. Данный вопрос трактуется иначе для объектов, сформированных в среде авторских работ.

Любая из приведенных выше инструкций истинна лишь на момент написания книги. Рекомендуем обратиться к обновлениям плейера, где могут быть устранены некоторые проблемы, связанные с графическим содержимым, встраиваемым в объекты TextField.

В списке атрибутов, поддерживаемых дескриптором <img>, можно отметить поддержку Flash-атрибута под названием id. Данный атрибут может применяться в качестве целевого для графического содержимого, которое встроено в объект TextField. Данное содержимое загружается во внедренный объект MovieClip с помощью названия экземпляра, определенного с помощью атрибута id. Это значит, что все свойства и методы класса MovieClip можно применять ко внедренному содержимому. К примеру, можно инструктировать загруженный SWF-файл по поводу остановки или продолжения воспроизведения. Например, можно проверять процесс загрузки для изображения и указывать TextField на необходимость изменения размеров после загрузки содержимого и выполнения инициализации. Например:

```
this.createTextField("tContent", this.getNextHighestDepth(), ↵
0, 0, 200, 200);
tContent.html = true;
tContent.htmlText = "A picture of a lake:
<img id='mcImage' width='180' height='120' ↵
align='center' vspace='0' hspace='0'↵
   src='http://www.person13.com/asb/image2.jpg'>";

// Определите функцию, которая контролирует процесс
// загрузки изображения.
function checkLoad():Void {
   var nLoaded:Number = tContent.mcImage.getBytesLoaded();
   var nTotal:Number = tContent.mcImage.getBytesTotal();
```

```
    // Если содержимое загружено и инициализировано, укажите
    // TextField на необходимость изменения размеров и
    // очистки интервала.
    if(nLoaded/nTotal >= 1 && tContent.mcImage._width > 0) {
        tContent.autoSize = "left";
        clearInterval(nInterval);
    }
}

// Установите интервал, где вызывается функция
// checkLoad() приблизительно к 100 миллисекундным
// интервалам.
var nInterval:Number = setInterval(checkLoad, 100);
```

При реализации предыдущего кода необходимо обратить внимание на следующие моменты.

- При отображении границы в объекте `TextField`, изменение ее размеров реализовано некорректно.
- Следует сформировать объект `TextField` таким образом, чтобы его начальные размеры превышали те, что необходимы для соответствия загружаемому содержимому. В противном случае загружаемое содержимое будет обрезано.

Создание программы просмотра HTML-информации

В этом разделе некоторые из HTML-понятий, рассмотренные в предыдущих разделах, применяются для создания приложения. Это приложение разрешет пользователю выделить тему в предметном указателе и просмотреть дополнительную информацию по этой теме в объекте `TextField`, поддерживающем HTML.

1. Откройте новый Flash-документ и сохраните его под именем `informationViewer001.fla`.
2. Добавьте в первый кадр основной временной шкалы код, показанный в листинге 17.2.

Листинг 17.2. Функции `createTextFields()` и `viewSection()`

```
function createTextFields():Void {

    // Создайте код два объекта TextField.
    this.createTextField("tIndex",
this.getNextHighestDepth(),50,100,100,200);
    this.createTextField("tContent",
this.getNextHighestDepth(),200,100,200,200);

    // Установите свойства tIndex.
    tIndex.selectable = false;
    tIndex.border = true;
    tIndex.html = true;
    tIndex.multiline = true;
    tIndex.wordWrap = true;

    // Создайте HTML для индекса и затем присвойте его
```

```
   // свойству htmlText объекта.
   var sIndex:String = "Click on one of the
   following links:<br>";
   sIndex += "<li><a
href='asfunction:viewSection,text'>text</a></li>";
   sIndex += "<li><a
href='asfunction:viewSection,html'>html</a></li>";
   sIndex += "<li><a
href='asfunction:viewSection,scrolling'>scrolling</a></li>";
    tIndex.htmlText = sIndex;

   // Установите свойства для tContent.
   tContent.selectable = false;
   tContent.border = true;
   tContent.html = true;
   tContent.multiline = true;
   tContent.wordWrap = true;
}

function viewSection(sSection:String):Void {

   // Уточните, какое содержимое отображается на базе той
   // выделенной области, на которой произведен щелчок.
   switch (sSection) {
      case "text":
         tContent.htmlText = "This is
         the section about text.";
         break;
      case "html":
         tContent.htmlText = "HTML allows you to <font
         color='#FF0000'>colorize</font> text";
         break;
      case "scrolling":
         tContent.htmlText = "Read more about scrolling
         in the next part of the chapter.";
   }
}

createTextFields();
```

3. Протестируйте фильм.

Скроллинг текста

Текст в пределах текстового поля может прокручиваться с помощью кода ActionScript как в горизонтальном, так и в вертикальном направлениях. Свойства скроллинга из класса `TextField` вступают в силу всякий раз, когда объем присвоенного текстовому полю текста превышает реальное пространство, которое доступно в просматриваемой части поля. К примеру, при наличии десяти строк текста и текстового поля, высота которого составляет только пять строк, следует настроить свойство скроллинга, что позволит просмотреть оставшиеся пять строк текста. Каждая строка текста в текстовом поле имеет индексное число. Данное индексное число основано на 1, а значит, на первой строке находится индекс 1. На рис. 17.4 проиллюстрирована работа программного скроллинга на примере объектов `TextField`.

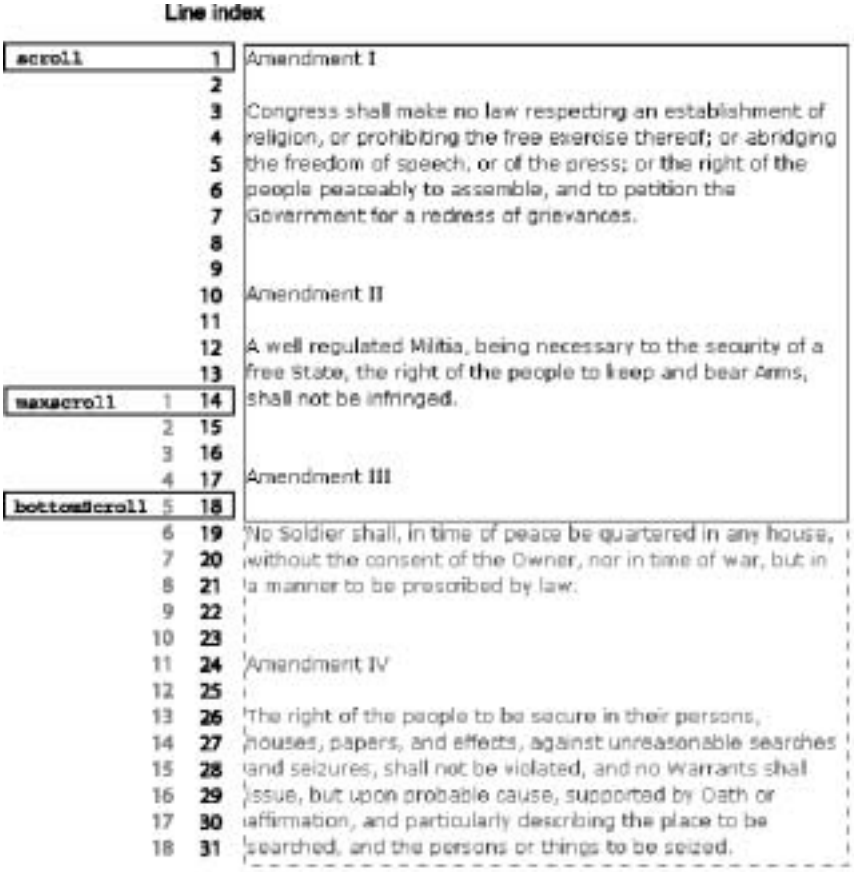

Рис. 17.4. В данном текстовом поле выполнено присваивание содержимого, имеющего 31 строку, в то же время просматриваемая часть поля представлена 18 строками

Скроллинг текста в вертикальном направлении

Свойство `scroll` из текстового поля контролирует, какой линейный индекс содержимого текстового поля находится в данный момент в верхней реально просматриваемой части поля. Данное свойство предназначено "только для чтения-записи". То есть вполне можно присваивать ему новое значение, что приведет к скроллингу текста в вертикальном направлении. Если текстовое значение присваивается объекту `TextField` с помощью кода ActionScript, содержимое текста автоматически индексируется при заполнении текстового поля. В примере, показанном на рис. 17.4, имеется 31 строка содержимого текста, но поле для отображения содержимого имеет только 18 просматриваемых строк. Тем не менее, число строк содержания *не* эквивалентно числу значений `scroll`. Причина этого раскрывается при описании свойства `maxscroll`.

Для прокрутки содержимого поля в направлении вверх воспользуйтесь свойством `scroll`. Следующий код показывает, как можно перемещать текст в пределах текстового поля под названием `tArticle` по одной строке при каждом щелчке на свойстве `mcScrollUp`:

```
mcScrollUp.onRelease = function():Void {
   tArticle.scroll++;
}
```

Для прокрутки содержимого поля в направлении вниз (текст перемещается сверху вниз), вычтите значение из свойства `scroll`. Благодаря следующему коду текст отступает на одну строку при каждом щелчке на кнопке:

```
mcScrollDown.onRelease = function():Void {
      tArticle.scroll--;
}
```

Если текст смещается в пределах поля, свойство `scroll` обновляется для отражения индексного значения, видимого в настоящий момент в верхней части поля.

Свойство `scroll` можно применять в текстовых полях Flash 4 и 5. Для применения свойства `scroll` обратитесь к названию `Var` для редактируемого текстового поля.

Свойство `bottomScroll` возвращает индекс для текста, который отображается на последней просматриваемой строке объекта `TextField`. В отличие от свойства `scroll`, свойство `bottomScroll` предназначено "только для просмотра"; с помощью `bottomScroll` нельзя изменять расположение текстового содержания. Для показанного на рис. 17.4 примера диапазон возможных значений для свойства `bottomScroll` составляет от 18 до 31.

Свойство `maxscroll`, как следует из названия, представляет максимальное значение для значения `scroll`, которое может возвратить объект `TextField`. Данное свойство предназначено "только для просмотра", и если динамически не добавлять или не удалять из объекта `TextField` реальный текст, значение `maxscroll` является фиксированным. Нельзя производить прокрутку на расстояние, превышающее значение `maxscroll`, также не выполняется прокрутка при указании в пределах текстовых полей отрицательных значений.

При добавлении значения к свойству `scroll` текстовое поле остановит продвижение текста, когда последняя строка текстового содержимого отобразится внизу текстового поля. В показанном на рис. 17.4 примере последней строкой текста является строка 31, а текстовое поле имеет высоту 18 строк. Если удалиться на 18 строк от строки 31, получим строку 14. Эта строка представляет собой максимальное значение свойства `scroll` для данного содержимого в определенном текстовом поле. Это значение аналогично значению, которое свойство `maxscroll` возвращает для данного текстового поля. Значение свойства `maxscroll` можно вычислить вручную следующим образом:

```
maxscroll = общее количество строк - количество просматриваемых
строк в текстовом поле + 1
```

Свойство `maxscroll` доступно по названию `Var` для редактируемых текстовых полей в фильмах Flash 4 и 5.

Скроллинг текста в горизонтальном направлении

В ActionScript можно выполнять скроллинг содержимого текстового поля в горизонтальном направлении. В отличие от предыдущих методов скроллинга, методы `hscroll` и `maxhscroll` используют пиксельные единицы измерения, а не номера индексных строк.

Свойство hscroll предназначено для просмотра и записи, оно выбирает либо устанавливает для объекта TextField текущую горизонтальную позицию для скроллинга (в пикселях). Если объект TextField инициализируется с помощью заданных по умолчанию значений, значением свойства hscroll является 0. Если увеличить значение hscroll, текст в пределах объекта TextField перемещается справа налево. Если уменьшить значение hscroll, текст переместится слева направо. Значение свойства hscroll не может быть установлено меньшим нуля. Следующий размещенный на основной временной шкале код создает объект TextField, включающий текст, который прокручивается справа налево на 10 пикселей, независимо от того, где произведен щелчок в рабочем поле Flash-фильма:

```
_this.createTextField("tDisplay", 1, 25, ↵
25, 30, 300);
tDisplay.text = "Hello, how are you? This
text scrolls with each mouse click.";

this.onMouseDown = function(){
   tDisplay.hscroll += 10;
};
```

Предназначенное "только для просмотра" свойство maxhscroll возвращает максимальное значение hscroll для текстового поля. Подобно свойству maxscroll, это свойство возвращает значение максимального смещения теста в пределах текстового поля. Значение hscroll не может устанавливаться меньшим значения maxhscroll. Значение maxhscroll зависит от объема текста, который не может отображаться в пределах видимой области текстового поля. При добавлении текста к индивидуальной линии в пределах поля, значение свойства maxhscroll увеличивается.

Применение событий для текстовых полей

Язык ActionScript располагает довольно универсальными обработчиками событий для редактируемых текстовых полей. Эти обработчики событий способны выполнять код независимо от того, где в фильме пользователь взаимодействует с текстовым полем. Для каждого из этих обработчиков можно указать анонимную или именованную функцию.

Обнаружение изменений в тексте

Метод обработчика событий onChanged() обнаруживает любые изменения в тексте, инициированные пользователем в пределах объекта. Если пользователь добавляет текст к полю с помощью клавиатуры, этот обработчик вызывается при нажатии клавиши пользователем, а не в момент, когда клавиша отпускается.

Метод обработчика onChanged() не обнаруживает изменений в текстовом поле, произведенных с помощью кода ActionScript. Другими словами, если изменить содержимое объекта TextField, используя свойство text либо htmlText, обработчик события onChanged() не вызывается.

Следующий код пересылает сообщение trace() в панель Output, где в режиме Test Movie (Тестировать фильм) текст должен был добавиться к объекту TextField под названием tComments:

```
this.createTextField("tComments", 1, 25, ⏎
25, 100, 20);
tComments.border = true;
tComments.type = "input";
tComments.onChanged = function(){
   trace("Text within comments field = " + ⏎
tComments.text);
};
```

Если необходимо определить несколько функций, выполняемых с помощью определенного метода обработчика `onChanged()` объекта `TextField`, можно сформировать для данного объекта `TextField` несколько объектов-слушателей. Каждый объект-слушатель может располагать собственным обработчиком `onChanged()`. О слушателях для объекта `TextField` можно узнать в разделе "Дообавление слушателей к объектам `TextField`" далее в этой главе.

Обнаружение изменений фокуса

Если фокус перемещается к объекту `TextField` в результате вмешательства пользователя (т.е. пользователь щелкает на объекте `TextField` либо перемещает фокус с помощью клавиши <Tab>), Flash вызывает метод обработчика событий `onSetFocus()` для данного объекта. Этот метод передает параметр, указывающий на предыдущий объект, который удерживал фокус (если таковой существует). Следующий код отображает действие `trace()` в панели Output, когда фильм находится в режиме Test Movie. Если пользователь щелкает на одной из двух полей, действие `trace()` указывает, какое поле получило фокус. Если предыдущее поле сфокусировано, метод `onSetFocus()` также сообщает название объекта:

```
this.createTextField("tMessageOne", ⏎
this.getNextHighestDepth(), 25,25,200,20);
this.createTextField("tMessageTwo", ⏎
this.getNextHighestDepth(), 250,25,200,20);
tMessageOne.border = true;
tMessageTwo.border = true;
tMessageOne.text = "message one";
tMessageTwo.text = "message two";

tMessageOne.onSetFocus = function(oPrevFocus:Object):Void {
   trace(this._name + " is now focused.");
   if(tPrevFocus._name != null){
      trace(tPrevFocus._name + " no longer has focus.");
   }
};

tMessageTwo.onSetFocus = function(oPrevFocus:Object):Void {
   trace(this._name + " is now focused.");
   if(tPrevFocus._name != null){
      trace(tPrevFocus._name + " no longer has focus.");
   }
};
```

С другой стороны, если фокус "убран" с объекта `TextField` вследствие вмешательства пользователя, Flash вызывает метод `onKillFocus()` для данного объекта. Этот обработчик передает ссылку на объект, который получает новый фокус. Если фокус покидает поле и не направляется в новый объект, этот параметр равен `null`. Следующий код пересылает действие `trace()` в панель Output в режиме Test Movie при вызове метода обработчика

onKillFocus(). Если щелкнуть внутри текстового поля tMessage, а затем щелкнуть вне него, вызывается обработчик. Аргумент obj представляет объект, который получает новый фокус:

```
this.createTextField("tMessage", this.getNextHighestDepth(), ↵
25, 25, 200, 20);
tMessage.text = "You have a message waiting.";
tMessage.border = true;
tMessage.onKillFocus = function(oPrevFocus:Object):Void {
trace(this._name + " is no longer focused.");
   trace(oPrevFocus._name + " is now focused.");
};
```

Обнаружение скроллинга

Метод обработчика onScroller() обнаруживает, когда изменяются свойства скроллинга объекта TextField. Среди других функций, этот обработчик может усовершенствовать свойство scroll из TextField, позволяя удостовериться, что последняя строка текста отображается в нижней части поля. Следующий код формирует слушателя мыши, который добавляет текст в поле сообщения при щелчке пользователя и последовательном освобождении мыши. Если длина текста в поле сообщения превышает видимое в поле число строк, вызывается метод обработчика onScroller() и устанавливается свойство scroll, которго равна значению свойства maxscroll:

```
this.createTextField("tMessage", this.getNextHighestDepth(), ↵
25, 25, 200, 100);
tMessage.border = true;
tMessage.multiline = true;
tMessage.wordWrap = true;

var nCount:Number = 0;

tMessage.text = "You have " + nCount + " ↵
message(s) waiting.\n";

tMessage.onScroller = function(){
   this.scroll = this.maxscroll;
};

oAddTextListener = new Object();
oAddTextListener.onMouseUp = function(){
   nCount++;
   tMessage.text += "You have " + nCount + ↵
" message(s) waiting.\n";;
};
Mouse.addListener(oAddTextListener);
```

Программа просмотра текста со скроллингом

В этом упражнении создается простое приложение, где используются связанные со скроллингом понятия, рассмотренных в предыдущих разделах. Данное приложение создает

несколько генерируемых программными методами объектов TextField. Один из этих объектов предназначен для отображения текста журнала, а два объекта, внедренные в объекты MovieClip, применяются в качестве кнопок для прокрутки текста истории. Чтобы получить в текстовом поле достаточный для целей прокрутки объем текста, применяется класс LoadVars. Если вы незнакомы с особенностями класса LoadVars, для получения более подробной информации обратитесь к главам 26 и 35.

1. На прилагаемом к книге компакт-диске находится текстовый файл под названием story.txt. Этот файл включает текст, который загружается в SWF-файл, а значит, он будет находиться в том же каталоге, что и SWF-файл. Поэтому скопируйте этот файл с компакт-диска на свой локальный диск.
2. Откройте новый Flash-документ и сохраните его под именем storyViewer001.fla. Удостоверьтесь, что файл сохранен в том же каталоге, что и story.txt.
3. Добавьте код, показанный в листинге 17.3, первый кадр основной временной шкалы.

Листинг 17.3. Код программы просмотра со скроллингом текста

```
function createContentField():Void {

   // Создайте объект TextField, который применяется для
   // отображения текста журнала.
   this.createTextField("tContent", ↵
   this.getNextHighestDepth(), 100,100,300,300);

   // Установите свойства TextField так, чтобы они
   // корректно отображали содержимое.
   tContent.border = true;
   tContent.selectable = false;
   tContent.multiline = true;
   tContent.wordWrap = true;
}

function createScrollButtons():Void {

   // Создайте объекты MovieClip, применяемые в качестве
   // кнопок для просмотра текста истории.

   this.createEmptyMovieClip("mcScrollDown", ↵
   this.getNextHighestDepth());
   this.createEmptyMovieClip("mcScrollUp", ↵
   this.getNextHighestDepth());

   // Добавьте объекты TextField, внедренные в объекты MovieClip.
   mcScrollUp.createTextField("tLabel",↵
   mcScrollDown.getNextHighestDepth(), ↵
   410, 100, 75, 20);
   mcScrollDown.createTextField("tLabel",↵
   mcScrollUp.getNextHighestDepth(), 410, 380, 75, 20);

   // Установите свойства объектов TextField.
   mcScrollUp.tLabel.border = true;
   mcScrollUp.tLabel.selectable = false;
   mcScrollDown.tLabel.border = true;
   mcScrollDown.tLabel.selectable = false;
```

```
    // Установите отображение текста в объекте TextField.
    mcScrollUp.tLabel.text = "Scroll Up";
    mcScrollDown.tLabel.text = "Scroll Down";

    // Если пользователь щелкнет на экземпляре
    // MovieClip, текст
    // журнала прокручивается вниз по одной странице.
    mcScrollDown.onPress = function():Void {
        tContent.scroll = tContent.bottomScroll + 1;
    };

    // Когда пользователь щелкает на экземпляре MovieClip
    // текст журнала прокручивается вверх по одной странице.
    mcScrollUp.onPress = function():Void {
        tContent.scroll -= (tContent.bottomScroll - ⏎
        tContent.scroll) + 1;
       };
}

function loadText():Void {

    // Создайте новый объект LoadVars.
    var lvText:LoadVars = new LoadVars();

    // При загрузке содержимого присвойте его для
    // отображения текста для ytContent объекта TextField.
    lvText.onData = function(sData:String):Void {
        tContent.text = sData;
    };

    // Укажите Flash на загрузку данных из story.txt.
    lvText.load("story.txt");
}

createContentField();
createScrollButtons();
loadText();
```

4. Протестируйте этот фильм.

При тестировании фильма журнал отображается в поле `tContent`; при щелчке и отпускании кнопок прокрутки содержимое будет прокручиваться постранично.

Добавление слушателей к объектам TextField

Объекты `TextField` могут включать в себя слушателей, которые обнаруживают изменения в содержимом текста или свойствах прокрутки. Методы, определенные для слушателей, практически идентичны методам с теми же названиями, которые обсуждались в предыдущем разделе. Основное отличие между `onChanged()` и `onScroller()` в качестве объектных методов слушателя и методами `TextField` состоит в следующем. Для одного текстового поля можно сформировать несколько объектов-слушателей, причем у каждого слушателя могут быть собственные обработчики `onChanged()` и `onScroller()`.

Обнаружение изменений текста

Метод `onChanged()` объекта-слушателя функционирует идентично методу `onChanged()` для объекта `TextField`. Чтобы воспользоваться методом `onChanged()` объекта-слушателя, сначала определите объект-слушатель. Существует множество типов слушателя, простейшим является объект `Object`. Затем зарегистрируйте слушателя в объекте `TextField`, применив метод `addListener()`.

Следующий код формирует два объекта `TextField`. Первый объект представляет собой поле ввода, разрешающее пользователю вводить текст. Метод слушателя объекта `onChanged()` вызывается всякий раз, когда пользователь обновляет значение в первом поле, обновляется и второе значение для отображения текста в обратном порядке:

```
this.createTextField("tInput", this.getNextHighestDepth(), ↵
25, 25, 200, 200);
this.createTextField("tCopy", this.getNextHighestDepth(), ↵
250, 25, 200, 200);
tInput.border = true;
tCopy.border = true;
tInput.type = "input";
var oListener:Object = new Object();
oListener.onChanged = function():Void {
   var sText:String = tInput.text;
   var aText:Array = sText.split("");
   aText.reverse();
   sText = aText.join("");
   tCopy.text = sText;
};
tInput.addListener(oListener);
```

Обнаружение скроллинга

Метод `onScrolled()` для объекта-слушателя обнаруживает изменения в свойствах прокрутки объекта `TextField`. Аналогично методу `onChanged()`, следует определить метод для объекта-слушателя и затем зарегистрировать слушателя с помощью `TextField`, применяя метод `addListener()`.

Работа со шрифтами

В следующих разделах описываются некоторые способы работы со шрифтами во Flash.

Внедренные шрифты

Одной из замечательных особенностей фильмов Flash является возможность постоянного просмотра в Web художественного изображения и шрифтов с помощью любой версии Flash Player. Внедрение шрифтов обычно является наиболее контролируемым оптимизационным шагом в процессе развертывания Flash. Но если вы будете невнимательны, при встраивании полных шрифтов можно сильно "раздуть" Flash-фильмы (SWF файлы).

Кроме очевидной выгоды от внедренных шрифтов, а именно представления текста в удобном для пользователя виде, есть еще одно преимущество. Внедренные шрифты реально активизируют объекты `TextField` для реализации некоторых заданий, которые в против-

ном случае не были бы выполнены. Например, как уже упоминалось, объект `TextField`, применяющий аппаратные (невстроенные) шрифты, нельзя ни вращать, ни изменять значение прозрачности. Если же встроить шрифт, обе эти процедуры становятся реальными.

Статический текст

Независимо от способа формирования статического текста в фильме, Flash MX автоматически встраивает определенные, введенные в поле, символы. Поскольку статический текст не может изменяться динамически, Flash встраивает только те символы, которые реально вводятся в поле на сцене. Тем не менее, если для статического текста отключить встраивание шрифтов, можно выбрать параметр Use Device Fonts (Использовать шрифты принтера) в инспекторе свойств, когда статический текст выбран в рабочем поле. Это воспрепятствует встраиванию шрифтовых символов в фильм Flash (SWF-file) и приведет к отображению шрифта, выделенного в инспекторе свойств *только в том случае*, если пользователь располагает этим шрифтом в качестве инсталлированного в своей системе. В противном случае шрифт заменяется родовым системным шрифтом.

Создание динамического и вводимого текста в среде авторских работ

По умолчанию Flash не вкладывает шрифты для динамического и вводимого текста. Поэтому, если требуется удостовериться, что применен подходящий шрифт, следует четко указать Flash на необходимость его внедрения. Для этого выбирается объект `TextField` и выполняется щелчок на кнопке Character (Символ) в окне инспектора свойств. На рис. 17.5 показано диалоговое окно Character Options (Параметры символа).

Рис. 17.5. Диалоговое окно Character Options имеет специфические возможности по встраиванию шрифтов для редактируемых текстовых полей

Как можно заметить, заданная по умолчанию настройка состоит в том, чтобы не вкладывать какие-либо символы. Однако при выборе параметра Specify Ranges (Указать диапазоны) можно выделить любую комбинацию предварительно установленных диапазонов. Либо можно выбрать другой вариант и указать настраиваемый набор символов в поле Include These Characters (Включить эти символы). Чем больше сиимволов включается, тем большим становится размер файла. В следующем списке детализируются некоторые из наиболее широко используемых диапазонов:

- **Uppercase (A–Z).** Этот параметр определяет внедрение только алфавитных символов, находящихся в верхнем регистре. При использовании шрифта Times New Roman подобные символы добавляют около 3 Кбайт дискового пространства. С помощью этого параметра добавляются такие символы:
  `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
- **Lowercase (a–z).** Этот параметр определяет внедрение только алфавитных символов, находящихся в нижнем регистре. При использовании шрифта Verdana эти символы увеличат размер Flash-фильма приблизительно на 3 Кбайт. К файлу SWF добавляются следующие символы:
  `abcdefghijklmnopqrstuvwxyz`
- **Numerals (0–9).** Данным параметром определяется внедрение числовых символов. При использовании шрифта Times New Roman этот параметр увеличит размер файла приблизительно на 1 Кбайт. Во Flash-фильм добавляются такие символы:
  `0123456789`
- **Punctuation (!@#$%...).** Этот параметр включает всю принятую в английском языке пунктуацию. Если для текстового поля применяется Times New Roman, размер файла для фильма Flash увеличивается более чем на 2 Кбайта. С помощью этого параметра добавляются следующие символы:
  `!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~`

Если внедряется только ограниченный набор символов, введите необходимые символы в поле Include These Characters. Заметим, что для некоторых шрифтов нужно встраивать и символ пробела (т.е. вставлять пустое пространство, нажимая в этом поле клавишу пробела).

Многие разработчики Flash-приложений предпочитают включать в поля динамического и вводимого текста URL-ссылки Internet. Удостоверьтесь, что символ `@` (для электронных почтовых сообщений), а также символы `&`, `=`, `+` и `-` (для сценарных запросов) подключены с помощью описанного выше символьного параметра, если вы намерены во Flash-фильме отображать подобные URL-ссылки для пользователя

Ниже перечислены общие принципы работы с некоторыми опциями по внедрению шрифтов.

- Если подключить только один регистр (верхний либо нижний), невнедренный регистр отобразится как внедренный. К примеру, если при внедрении символов в нижнем регистре пользователь введет символ в верхнем регистре `A`, в поле данный символ отобразится с помощью нижнего регистра `a`.
- Если попытаться присвоить невнедренные символы редактируемому текстовому полю посредством выражения ActionScript (либо с помощью названия `Var`, либо путем обращения к свойству `text` объекта `TextField`), данные символы не отобразятся в текстовом поле. Однако переменная или свойство `text` сохранят предписанное в выражении значение до тех пор, пока пользователь фокусирует текст, либо переменная получит новое значение с помощью нового выражения ActionScript.

- Кнопки стилей шрифта из инспектора свойств также оказывают влияние на размер файла и вид текстового поля в фильме Flash (SWF-файл). Первый символ любого редактируемого поля определяет цвет, стиль и размер (в пунктах) всего поля. Если необходимо проконтролировать специфическое форматирование в пределах отдельных разделов одного и того же текстового поля, рекомендуем для редактирования текстовых полей обратиться к параметру HTML. Подобные вопросы (работа с HTML и объектами `TextField`) обсуждаются далее в этой главе.
- Если для встраивания различных подмножеств шрифта с одной и той же гарнитурой подключено несколько различных полей, каждое из этих полей может использовать внедренные символы и другого поля. Например, если в одно редактируемое текстовое поле введены буквы шрифтом Verdana в нижнем регистре, а в другое поле — буквами шрифта Verdana в верхнем регистре, в обоих полях могут применяться данные символы как в верхнем, так и в нижнем регистрах. Для реализации подобного эффекта эти поля *не* обязательно должны сосуществовать в одном кадре (или на основной временной шкале).
- Параметры внедрения следует использовать для любого редактируемого текстового поля, которое маскируется с помощью параметра Mask Layer (Маска слоя) или вкладывается в экземпляр `MovieClip`, трансформированный с помощью настроек для символьных экземпляров в панели Transform (Преобразование) либо в меню Color (Цвет) инспектора свойств. Для должного зрительного эффекта Flash Player визуализирует механизм, который необходим для получения контуров любых символов, встроенных в фильм Flash (SWF-файл).
- Нельзя получить доступ к внедренным шрифтам одного Flash-фильма (SWF-файла) из другого Flash-фильма (SWF-файла). Например, если в объект `TextField` первоначального Flash-фильма были внедрены символы нижнего регистра шрифта Verdana, а в объект `TextField` другого Flash-фильма были внедрены символы верхнего регистра того же шрифта, причем второй фильм загружается в первый фильм, невозможно использовать символы нижнего и верхнего регистров в этих двух фильмах. Чтобы совместно использовать встроенный шрифт в нескольких Flash-фильмах, следует создать шрифтовые символы в общедоступных библиотеках и связать их с текстовыми полями.

Информация относительно общедоступных библиотек и символов шрифтов содержится в главе 34.

Текст, созданный в период выполнения

При создании объекта `TextField` в период выполнения шрифты можно внедрять исключительно программными методами. Для этого есть два способа. Первый способ состоит в формировании экземпляра `TextField` в среде авторских работ (речь идет о создании вводимого ли динамического поля с помощью инструмента Text), который внедряет данный текст, а затем указывает вашим созданным в период выполнения экземплярам на применение этого же шрифта. Второй способ предполагает обращение к шрифтовому символу. Рассмотрим более подробно каждую из этих возможностей.

Применение шрифта объекта TextField, созданного в среде авторских работ

Возможно, наиболее простой способ для программного применения внедренного шрифта заключается во внедрении шрифта перед началом его создания в среде авторских работ (для объекта `TextField`). Вне рабочего поля можно сформировать единый динамический объект `TextField` и внедрять символы, которые желательно применить, как обсуждалось в разделе "Динамический и вводимый текст, созданный в среде авторских работ". Внедрение шрифтов в среде авторских работ для объекта `TextField` не только проще с точки зрения реализации, в этом случае также обеспечивается более строгий контроль перечня внедренных символов. Для программного использования шрифтов, внедренных в среде авторских работ для `TextField`, выполните следующие шаги:

1. Создайте в среде авторских работ объект `TextField` вне рабочего поля. Можно получить экземпляр и назвать его или не делать этого, можно добавить в него текст, а можно и обойтись без этого. Вряд ли объект `TextField` будет применяться для других целей, поэтому эти манипуляции не являются обязательными.
2. Внедрите символы шрифта, который будет применяться, в `TextField` в среде авторских работ, следуя инструкциям из раздела "Создание динамического и вводимого текста в среде авторских работ".
3. Создайте текст во время выполнения при помощи метода `createTextField()`.
4. Присвойте свойству `embedFonts` из объекта `TextField` значение `true`.
5. Создайте объект `TextFormat`.
6. Присвойте имя внедренного шрифта свойству `font` объекта `TextFormat`.
7. Присвойте объекту `TextFormat` объект `TextField`, применяя метод `setTextFormat()`.

Символы шрифта и класс `TextFormat` ранее не рассматривались. Более подробные сведения о них содержатся в главах 34 и 18 соответственно. В этом разделе представим основную справочную информацию.

Выше представлены обобщенные инструкции. Теперь рассмотрим конкретный пример.

1. Создайте новый динамический объект `TextField` в среде авторских работ. Разместите объект вне рабочего поля; причем не обязательно предоставлять ему название экземпляра.
2. Откройте диалоговое окно Character Options (Параметры символа) для объекта `TextField` и выберите параметр Uppercase Letters (Символы верхнего регистра). Затем щелкните на кнопке OK.
3. Добавьте следующий код в первый кадр основной временной шкалы:

```
this.createTextField("tLabel", this.getNextHighestDepth(),↵
   100, 100, 100, 20);
tLabel.text = "abcdefg";
```

4. Протестируйте фильм. Текст должен отобразится с помощью заданного по умолчанию шрифта с засечками. Фильм тестируется для уточнения расположения текста, а именно того, что текст отображается перед установкой внедренного шрифта.
5. Присоедините следующий код после кода, добавленного на шаге 3:

```
this.createTextField("tLabel", this.getNextHighestDepth(), ↵
   100, 100, 100, 20);
```

```
tLabel.text = "abcdefg";
tLabel.embedFonts = true;

var tfFormatter:TextFormat = new TextFormat();
tfFormatter.font = "Verdana";

tLabel.setTextFormat(tfFormatter);
```

6. Протестируйте фильм снова. В это время текст остается невидимым, поскольку внедренные символы верхнего регистра относятся только к шрифту Verdana.
7. Откройте диалоговое окно Character Options еще раз в среде авторских работ для объекта `TextField`. Установите внедрение символов нижнего регистра.
8. Протестируйте фильм снова. Теперь текст должен отображаться с применением шрифта Verdana.

Применение символа из набора Font

Шрифт можно внедрять также с помощью символа из набора Font. Символ шрифта позволяет применять общедоступные библиотеки, а также может быть предпочтителен в некоторых ситуациях. С другой стороны, символ шрифта требует внедрения всего набора шрифтов. Поэтому, если применяется лишь несколько символов, можно использовать `TextField` в среде авторских работ `TextField` для внедрения этих определенных символов.

Ниже перечисляются основные шаги по использованию символа Font.

1. Создайте символ шрифта в библиотеке и выберите экспорт этого символа в ActionScript.
2. Присвойте свойству `embedFonts` для объекта `TextField` значение `true`.
3. Создайте объект `TextFormat` и присвойте идентификатор связывания символа шрифта свойству `font` объекта `TextFormat`.
4. Присвойте объект `TextFormat` объекту `TextField`.

Рассмотрим более подробно некоторые выполненные до этого шаги.

Для создания символа шрифта выполним следующие действия.

1. Откройте библиотеку и выберите команду New Font (Новый шрифт) из меню библиотеки.
2. В диалоговом окне Font Symbol Properties (Свойства символа шрифта) назовите символ и выделите шрифт, который необходимо внедрить.
3. После создания символа откройте настройки по связыванию символа, установите флажок Export for ActionScript (Экспорт в ActionScript) и придайте символу индентификатор связывания.

Свойство `embedFonts` для объекта `TextField` имеет заданное по умолчанию значение `false`. При выборе значения `true` для этого свойства Flash получает указание на применение только определенного внедренного шрифта (который определяется на следующем шаге с помощью объекта `TextFormat`). Если Flash не может найти определенный внедренный шрифт, тогда в этом поле текст не отображается:

```
tLabel.embedFonts = true;
```

Для создания объекта `TextFormat`, применяющего внедренный шрифт и присваивающего объект объекту `TextField`, выполните следующие шаги.

1. Создайте новый объект `TextFormat` с помощью конструктора:

```
var tfFormatter:TextFormat = new TextFormat();
```

2. Присвойте название идентификатора связывания свойству `font` объекта `TextFormat`. Например, рассмотрим формирование символа шрифта с помощью идентификатора связывания `ArialEmbeddedFont`:

```
tfFormatter.font = "ArialEmbeddedFont";
```

3. Присвойте объект `TextFormat` объекту `TextField` с помощью метода `setTextFormat()`:

```
tLabel.setTextFormat(tfFormatter);
```

Применение стилей Faux Bold и Italic со внедренными шрифтами

Иногда для внедренного шрифта удобно применять стиль faux bold либо italic, для чего необходимо выбрать любой из соответствующих параметров (или оба параметра) в диалоговом окне Font Symbol Properties (Свойства символа шрифта). Тем не менее, каждый из этих параметров формирует определенные контуры шрифта, которые генерируются и экспортируются совместно с Flash-фильмом (SWF-файлом). К примеру, при установке флажка Bold для шрифта Verdana, который использовался в последнем упражнении, можно применять с текстовыми полями только следующие свойства объекта `TextFormat`. (Заметим, что объект `TextFormat` можно назвать иначе, не обязательно использовать название `siteStyle`.):

```
var siteStyle:TextFormat = new TextFormat();
siteStyle.font = "Verdana";
siteStyle.bold = true;
siteStyle.size = 12;
comments.setTextFormat(siteStyle);
```

Если не указывать свойство `bold`, текст не отобразится в объекте `TextField`. Подобно этому, при установке в диалоговом окне Font Properties обоих флажков, Bold и Italic, этот шрифт можно применять только в том случае, когда свойствам `bold` и `italic` присвоены значения `true`:

```
var siteStyle:TextFormat = new TextFormat();
siteStyle.font = "Verdana";
siteStyle.bold = true;
siteStyle.italic = true;
siteStyle.size = 12;
comments.setTextFormat(siteStyle);
```

Если вы хотите воспользоваться стилями normal, faux bold либо faux italic, следует в панели Library для каждого стиля сформировать три отдельных символа шрифта. Если стили faux bold и faux italic используются вместе, следует добавить четвертый символ шрифта, для которого установлены два флажка, Bold и Italic. Не забывайте о том, каждый символ шрифта должен быть установлен для экспорта с помощью фильма Flash (SWF-файл), чтобы применять его вместе с ActionScript. Внедрение такого большого числа шрифтов может быть непрактично с точки зрения требований к размеру файла для Flash-фильма. Удостоверьтесь в том, что вам эти шрифты действительно необходимы, перед тем как принимать решение об их включении.

Не забывайте также, что эти стили называются *faux* bold и *faux* italic, а это означает, что Flash MX при формировании контуров, экспортируемых с фильмом Flash, должен "сам догадаться", каким должен быть шрифт: полужирным или курсивным. Большинство гарнитур шрифта имеют индивидуальные шрифтовые файлы для этих стилей, например, Futura Oblique ("наклонный" является другим названием курсива) либо Futura Bold. Вместо применения faux bold либо faux italic можно вкладывать оригинальный вид шрифта, разработанный для форматирования, с учетом полужирного шрифта либо курсива.

Для минимизации добавленного размера файла (в байтах) для контуров шрифта в окончательных фильмах Flash, можно прибегнуть к использованию Macromedia Fontographer. Тогда возможно создавать файлы заказных шрифтов, которые включают только те символы, кото-

рые необходимы в текстовых полях вашего проекта. Flash MX не предлагает каких-либо иных опций подмножеств шрифтов для символов шрифтов, что характерно для обычных динамических или вводных текстовых полей в диалоговом окне Character Options (доступ можно получить из инспектора свойств).

Применение шрифтов принтера во Flash

Также можно выбрать вариант использования трех различных гарнитур шрифта с уникальными названиями Flash. Эти шрифты перечислены в самом верху (Windows) или в самом низу (Macintosh) меню Font в окне инспектора свойств. Можно также получить к ним доступ с помощью команды Text⇨Font (Текст⇨Шрифт) в Windows и Macintosh.

- **_sans:** при отображении текста в указанном поле применяется заданный по умолчанию системный шрифт без засечек. Начертание шрифта без засечек состоит в округлении углов для всех символов. Обычно гарнитура шрифта без засечек исключает применение декоративных краев. При работе в Windows заданным по умолчанию шрифтом без засечек является Arial. При работе в Macintosh этот шрифт представлен как Helvetica. Гарнитуры шрифтов без засечек часто называют готическими.
- **_serif:** при отображении текста в указанном поле применяется заданный по умолчанию системный шрифт с засечками. Засечка представляет линию, которая применяется для завершения краев символов. В этой книге применяется гарнтитура шрифта с засечками. Обратите внимание на края таких символов, как *F* и *D*, и сравните с теми же буквами в заголовке раздела, где применяется гарнитура шрифта без засечек. Заданными по умолчанию системными шрифтами с засечками являются Times New Roman (при работе в Windows) и Times (при работе в Macintosh).
- **_typewriter:** применяется заданное по умолчанию системное моноширинное (с одинаковой шириной букв) начертание, при котором для каждого символа определяется одинаковая ширина. Большинство шрифтов не являются моноширинными. Например, ширина, необходимая для буквы *W*, значительно превышает требуемую ширину для буквы *I*. При работе в Windows заданным по умолчанию моноширинным начертанием является Courier New. При работе в Macintosh этим требованиям отвечает шрифт Courier.

При использовании шрифтов принтера вам не понадобятся какие-либо инструментальные средства по внедрению шрифтов из инспектора свойств для инструмента Text. Шрифты принтера обычно применяются в целях сокращения размера файла для фильма Flash (SWF-файлы), если точная гарнитура шрифта не является обязательной для текста в пределах фильма. К примеру, хотя вы заинтересованы в том, чтобы в логотипе компании применялось то же самое начертание шрифта, которое используется в печатных материалах и сигнальных сообщениях, вам не нужно обращаться к этому шрифту в форме Flash, куда пользователи вводят свою контактную информацию.

Шрифты принтера жают текст в редактируемых текстовых полях, которые контролируются масками слоя или получены в результате преобразования. К примеру, если объект `TextField`, включающий шрифт принтера, текст не будет отображаться. Любые изменения в панели Transform либо в меню Color из инспектора свойств для этого экземпляра сделают текст невидимым.

Вставка специальных символов в редактируемые текстовые поля

Некоторые символы нуждаются в использовании специального синтаксиса для отображения в пределах текстового поля, редактируемого с помощью ActionScript. К примеру, если необходимо применить код ActionScript для установки значения объекта `TextField`, следует узнать, каким образом в строчное выражение вставляется символ возврата каретки, кавычки и дескрипторы. Ниже приводится перечень символов специального форматирования, которые можно применять в пределах текста, присвоенного с помощью ActionScript. Большинство из них представляют собой *пары символов обратной косой черты* — это термин, применяемый для специальных символов в нескольких языках по написанию сценариев. Пары символов обратной косой черты всегда записываются как *встроенные*, т.е. они определяются в пределах этой же строки. Пары символов обратной косой черты также называют управляющими последовательностями.

- **newline, \n, \r:** `newline` является специфическим для Flash оператором, который принуждает каретку к возврату в пределах редактируемого текстового поля. Оператор `newline` не применяется в пределах строкового выражения. Тем не менее, пара символов обратной косой черты `\n` либо `\r`, которая всегда вставляет возврат каретки, записывается как часть строчного выражения.
- **\\:** если в редактируемое текстовое поле необходимо с помощью ActionScript вставить символ обратной косой черты (\), то пара символов обратной косой черты предназначена именно для этой цели.
- **\t:** эта пара вставляет в поле символ . Действительно, вам *не следует* в пределах редактируемого текстового поля выполнять это вручную.
- **\":** эта пара вставляет в пределы строки символ двойных кавычек.
- **\':** данная пара вставляет одинарную кавычку в строку, которая применяется для редактируемого текстового поля.

Несмотря на то, что к другим управляющим последовательностям прибегают довольно редко, это вполне возможно. Например, для символа обратной косой черты применяется `\b`, либо для символа прокрутки страницы используется `\f`. Также можно указывать байты в восьмеричном (`\000-\377`), шестнадцатеричном (`\x00-\xFF`) либо 16-битовом символе Unicode, используя шестандцатеричное представление (\u0000-\uFFFF).

Для применения этих специальных последовательностей по форматированию, просто используйте их в строковом выражении, которое записывается для редактируемого текстового поля. К примеру, при создании экземпляра объекта `TextField` под названием `tArticle` и формировании действия в первом кадре текущей основной временно́й шкалы

```
tArticle.text = "Part I" + newline + "This is how it began.";
```

данный код отображает текст в поле следующим образом:

```
Part I
This is how it began.
```

Данный код можно записать с помощью следующего синтаксиса:

```
article.text = "Part I\rThis is how it began.";
```

Даже в том случае, когда данный синтаксис используется при создании одного термина, `I\rThis`, ActionScript корректно интерпретирует пару символов обратной косой черты и отображает текст в поле следующим образом:

```
Part I
This is how it began.
```

Перечисленные последовательности нельзя вводить в реальное редактируемое поле в рабочем поле Flash. Они применяются только в коде ActionScript.

Создание программы отображения случайных букв

В этом упражнении в рабочем поле создается интересный эффект букв, которые то "угасают", то становятся более четкими, причем этот эффект проявляется случайным образом. Поскольку данный эффект опирается на модификацию свойства `_alpha` объектов `TextField`, необходимо внедрять шрифт.

1. Откройте новый Flash-документ и сохраните его под именем `randomLetters001.fla`.
2. Переименуйте заданный по умолчанию слой как `Embedded Font` и добавьте новый слой под названием `actions`.
3. Применяя инструмент Text (Текст), добавьте вне рабочего поля, на слое Embedded Font, динамический объект `TextField` в среде авторских работ.
4. При выделенном объекте `TextField` воспользуйтесь раскрывающимся меню для изменения шрифта на Verdana, затем щелкните на кнопке Character (Символ) из инспектора свойств для открытия диалогового окна Character Options (Параметры символа).
5. Выберите параметр Specify Ranges (Указать диапазоны) и введите в поле Include These Characters (Включить эти символы) значение `abcdef`.
6. Щелкните на кнопке OK для выхода из диалогового окна Character Options.
7. Добавьте код из листинга 17.4 в первый кадр слоя `actions`.

Листинг 17.4. Код программы отображения случайных букв

```
function displayLetter():Void {

   // Создайте случайное целое значение для поля
   // одного из индексов из массива aLetters.
   var nRandomIndex:Number = Math.ceil(Math.random() *
   aLetters.length) - 1;

   // Создайте случайные числа для применения в
   // качестве координат x и y для буквенного TextField.
   var nRandomX:Number = Math.random() * 550;
   var nRandomY:Number = Math.random() * 400;

   // Получите следующее доступное значение глубины;
   var nDepth:Number = this.getNextHighestDepth();

   // Создайте новый объект TextField с применением
   // случайных координат x и y
   this.createTextField("tLetter" + nDepth, nDepth,
```

```
    nRandomX, nRandomY, 0, 0);

    // Присвойте ссылку на переменную, чтобы сделать
    // ее более удобной для работы с объектом TextField.
    var tLetter:TextField = this["tLetter" + nDepth];

    // Установите значения свойства autoSize и текстовых
    // свойств так, чтобы отображалась случайная буква.
    tLetter.autoSize = "left";
    tLetter.text = aLetters[nRandomIndex];

    // Установите заказное свойство под названием
    // fadeDirection, приращение, изменяющее прозрачность.
    // Установите первоначально альфа к 0.
    tLetter.fadeDirection = 5;
    tLetter._alpha = 0;

    // Укажите Flash на встраивание шрифта для TextField.
    tLetter.embedFonts = true;

    // Создайте объект TextFormat, который указывает
    // Flash на использование шрифта Verdana и
    // установите размер, равный 15.
    var tfFormatter:TextFormat = new TextFormat();
    tfFormatter.font = "Verdana";
    tfFormatter.size = 15;

    // Присвойте объект TextFormat объекту TextField.
    tLetter.setTextFormat(tfFormatter);

    // Установите интервал, при котором буква будет
    // исчезать и появляться.
    tLetter.nInterval = setInterval(this, ↵
    "alphaFade", 10, tLetter);
}

function alphaFade(tLetter:TextField):Void {

    // Увеличьте значение прозрачности для буквы
    // в поле TextField.
    tLetter._alpha += tLetter.fadeDirection;

    // Проверьте, полностью ли исчезла буква.
    // Если это так, установите свойство fadeDirection
    // равным -5, чтобы TextField начал исчезать.
    // В противном случае, если буква исчезнет полностью...
    if(tLetter.fadeDirection > 0 && tLetter.↵
    _alpha >= 100) {
        tLetter.fadeDirection = -5;
    }
    else if(tLetter.fadeDirection < 0 && ↵
    tLetter._alpha <= 0) {
        // ... очистите интервал и удалите объект TextField.
        clearInterval(tLetter.nInterval);
        tLetter.removeTextField();
    }

     // Удостоверьтесь, что экран обновлен.
```

```
    updateAfterEvent();
}

// Создайте массив букв.
var aLetters:Array = ["a", "b", "c", "d", "e", "f"];

// Установите интервал, при котором будет отображаться новая буква.
var nDisplayInterval:Number = setInterval(this, "displayLetter", 1);
```

8. Протестируйте фильм.

Понятие о классе Selection

Класс `Selection` помогает управлять фокусом и выделением внутри Flash-приложения. Объекты `TextField`, `MovieClip` и `Button` могут получать фокус, но выделение применяется только по отношению к объектам `TextField`. Более подробно фокус и выделения рассматриваются в следующих разделах.

Методы из класса `Selection` являются статическими. Это означает, что не требуется создавать экземляр класса. Достаточно просто вызвать эти методы из класса.

Работа с фокусом

Фокус — это понятие, используемое во многих языках программирования для описания активной части приложения. Например, в окнах Web-браузеров обычно фокусируется окно переднего плана. Это означает, что такие действия, как ввод с клавиатуры, перехватываются именно этим окном и никаким другим. То же самое можно сказать о Flash-приложениях. Объект `Button`, `MovieClip` или `TextField` может получать фокус в среде Flash. Например, если пользователь щелкает мышью в экземпляре объекта `TextField`, именно здесь устанавливает фокус. Как только объект получает фокус, пользователь может вводить текст.

Поиск фокуса

Метод `getFocus()` возвращает полный путь (в виде строки) для сфокусированных в текущий момент объектов `TextField`, `Button` либо `MovieClip`. Если отсутствует какой-либо объект с фокусом, возвращается значение `null`. Например, если активно текстовое поле под названием `tUsername` на основной временной шкале, `Selection.getFocus()` возвратит в качестве строки `_level0.tUsername`. Обычно данный метод применяется для присваивания значения переменной, которая сохраняет информацию о пути. Это проиллюстрировано в последующих примерах.

Если объекту `TextField` присвоена и переменная, и название экземпляра, `getFocus()` возвращает путь с помощью названия экземпляра. Если указано только название переменной, `getFocus()` возвращает путь с помощью названия переменной.

Установка фокуса

Метод `setFocus()` назначает фокус объекту, который указан в качестве параметра. Этот параметр представляет путь (относительный или абсолютный) для объектов `MovieClip`, `Button` или `TextField`, определенный в виде строки. Если объект, которому передается

фокус, является объектом TextField, тогда Flash также отображает в пределах экземпляра объекта I-образный курсор.

В следующем примере формируется новый объект TextField, которому передается фокус:

```
this.createTextField("tUsername", this.getNextHighestDepth(), ⏎
25, 25, 150, 20);
tUsername.border = true;
tUsername.type = "input";
Selection.setFocus("tUsername");
```

Прослушивание изменений фокуса

К классу Selection можно добавлять объекты-слушатели (listener objects), которые сообщают о том, что произошло изменение фокуса. Для добавления объекта-слушателя примените метод Selection.addListener(). Например, следующий код регистрирует нового слушателя под названием oListener:

```
Selection.addListener(oListener);
```

Для удаления слушателя примените метод Selection.removeListener():

```
Selection.removeListener(oListener);
```

Когда фокус изменился, класс Selection уведомляет всех зарегистрированных слушателей и вызывает собственный метод onSetFocus(). Метод onSetFocus() автоматически передает два параметра, а именно ссылку на объект, который имел фокус накануне, и название объекта, который только что получил фокус. Рассмотрим следующий пример:

```
var oFocusListener:Object = new Object();
oFocusListener.onSetFocus = function(oPrevFocus:Object,
oNewFocus:Object):Void {
 trace("Current focus = " + oNewFocus);
 trace("Previous focus = " + oPrevFocus);
};
Selection.addListener(oFocusListener);
```

Работа с выделенной областью

Выделенная область представляет собой диапазон символов в пределах объекта TextField, которые подсвечиваются. Всякий раз, когда выполняется щелчок и выделяются символы в пределах текста, формируется выделенная область. Также можно сформировать выделенную область программными методами, что и будет показано ниже. Во вводном тексте также можно вставлять I-образный курсор в пределах поля. Эта позиция известна как *знак вставки*. С помощью кода ActionScript и класса Selection можно восстановить текущее значение для местоположения знака вставки, значение текущего положения для начала выделенной области и такое же значение для конца выделенной области. Также можно применять ActionScript для подключения автоматизированных выделенных областей (или подсветок) в пределах текстового поля.

Все методы из класса Selection функционируют с помощью индексов, значения которых отсчитываются от нуля.

Заметим, что все методы класса Selection, которые служат для работы с выделенной областью и знаком вставки, не нуждаются в указании объекта TextField, для которого ус-

танавливается либо выбирается выделенная область. Поскольку за один раз выделяется только одна область, автоматически используется объект TextField, имеющий фокус в текущий момент времени. Как уже упоминалось, объект TextField может получать фокус в результате взаимодействия с пользователем либо программными методами.

Получение текущей выделенной области

Выделенная область имеет начальную и конечную точки, которые указываются соответствующими индексами в пределах сфокусированного TextField. Для восстановления индексов можно применить методы Selection.getBeginIndex() и Selection.getEndIndex().

Метод Selection.getBeginIndex() возвращает начальный индекс выделенной области. Если выделенная область отсутствует, данный метод возвращает -1. Метод Selection.getEndIndex() возвращает данный индекс непосредственно после последнего символа выделеной области. Если выделенная область отсутствует при вызове метода, данный метод возвращает -1.

Эти методы можно протестировать с помощью следующего кода:

```
this.createTextField("tOutput", ⮐
this.getNextHighestDepth(), 100, 100, 200, 200);
tOutput.border = true;
tOutput.multiline = true;
tOutput.wordWrap = true;
tOutput.text = "The Selection class enables you to
retrieve the selected text programmatically.";
this.onMouseUp = function():Void {
 trace(Selection.getBeginIndex() + " " +
 Selection.getEndIndex());
};
```

При тестировании предыдущего кода выделите некоторую часть текста и отпустите кнопку мыши. Начальный и конечный индексы выделенной области отобразятся в панели Output.

Довольно часто желательно использовать выделительные индексы вместе с методами substring() и/или substr() из класса String. К примеру, следующий код является небольшой вариацией (изменения показаны жирным шрифтом) предыдущего кода, в результате применения которого в панели Output отображалась текущая выделенная область, а не только индексы:

```
this.createTextField("tOutput", ⮐
this.getNextHighestDepth(), 100, 100, 200, 200);
tOutput.border = true;
tOutput.multiline = true;
tOutput.wordWrap = true;
tOutput.text = "The Selection class enables ⮐
you to retrieve the selected text programmatically.";
this.onMouseUp = function():Void {
   trace(tOutput.text.substring(Selection.getBeginIndex(),
   Selection.getEndIndex()));
};
```

Установка выделенной области

В дополнение к восстановлению выделенной области, также можно устанавливать выделенную область и программными средствами. Для этого вызывается метод Selection.setSelection(). Данный метод использует два параметра: целые числа, определяющие начальный и конечный индексы. Начальный индекс служит индексом для первого символа в

выделенной области, а конечный индекс представляет собой индекс, следующий непосредственно за последним символом.

Часто метод `Selection.setSelection()` используется вместе с методом `Selection.setFocus()`. Не забывайте, что `Selection.setSelection()` применяется к `TextField`, который в текущий момент времени имеет фокус. Складывается ситуация, при которой желательно программными средствами доставить фокус к первому `TextField`. Следующий код формирует объект `TextField`; если щелкнуть и отпустить кнопку мыши, выделяется слово "set":

```
this.createTextField("tOutput", this.getNextHighestDepth(), ⏎
100, 100, 200, 200);
tOutput.border = true;
tOutput.multiline = true;
tOutput.wordWrap = true;
tOutput.selectable = false;
tOutput.text = "The Selection class also enables you to ⏎
   set the selected text programmatically.";
this.onMouseUp = function():Void {
   Selection.setFocus("tOutput");
   Selection.setSelection(40, 43);
};
```

Работа со знаком вставки

Также можно программными методами получать и устанавливать в пределах объекта `TextField` знак вставки. Метод `getCaretIndex()` возвращает текущий индекс для I-образного курсора. Если такой курсор не активизирован, этот метод возвращает `-1`.

Знак вставки можно установить с помощью метода `setSelection()`. Для этого начальный и конечный индексы устанавливаются равными одному и тому же значению.

Замена выделенного текста

Метод `replaceSel()` из класса `TextField` заменяет активную выделенную область значением, указанным как параметр метода. Поскольку для функционирования данного метода выделенная область должна представлять собой текущий фокус, желательно применять данный метод в комбинации со слушателями, которые присоединяются либо к объекту `Mouse`, либо к объекту `Selection`.

Помните о том, что фокус нельзя установить одновременно в двух местах. Другими словами, метод `replaceSel()` нельзя использовать для одновременного выделения некоторого текста и выполнения щелчка кнопкой. Также невозможен и другой путь: после щелчка пользователя на кнопке другой элемент пользовательского интерфейса изменит текст. В обоих случая выделенная область (или фокус) в текстовом поле будут утеряны.

Следующий код заменяет выделенный текст (или индекс) в пределах объекта `TextField` под названием `tArticle` с помощью содержания поля под названием `tWord`. Замена происходит, если пользователь щелкнет и отпустит мышь, а фокус обнаруживается в поле `tArticle`:

```
this.createTextField("tArticle", ⏎
this.getNextHighestDepth(), 100, 25, 200, 100);
this.createTextField("tWord", ⏎
this.getNextHighestDepth(), 25, 25, 50, 20);
tArticle.border = true;
```

```
tWord.border = true;
tWord.type = "input";
tArticle.text = "This is some text in a text field.";
var oListener:Object = new Object();
oListener.onMouseUp = function(){
 if(Selection.getFocus().indexOf("tArticle") != -1){
  tArticle.replaceSel(tWord.text);
 }
};
Mouse.addListener(oListener);
```

Использование клавиши <Tab>

При работе с объектами TextField следует обратить внимание, каким образом устанавливается фокус посредством клавиши <Tab>. Стандартные вычислительные методики разрешают пользователю изменить фокус с помощью клавиши <Tab>. По умолчанию объекты TextField могут таким образом получать доступ к фокусу. Однако можно подключать либо отключать подобные функциональные возможности, а также указывать порядок получения фокуса объектами TextField при нажатии клавиши <Tab>.

Подключение и отключение фокуса, инициированного с помощью табуляции

Для уточнения, при каких условиях нажатие клавиши <Tab> может передать фокус тексту, применяется свойство tabEnabled. По умолчанию это свойство имеет значение true, а значит, экземпляр может получить фокус, инициированный нажатием клавиши <Tab>. Как правило, вводимый текст должен обладать этой возможностью, то есть клавиша <Tab> должна быть подключена в этом случае. Однако динамический текст обычно не использует эту возможность, т.е. клавиша <Tab> не подключена.

Изменение порядка табуляции

Свойство tabIndex позволяет определять порядок, в соответствии с которым объекты доступны с помощью клавиши <Tab>. В качестве значения свойства tabIndex может использоваться любое положительное целое число. Увеличение значения свидетельствует о порядке доступа: объект, имеющий значение tabIndex, равное 1, является первым объектом, который фокусируется до объекта со значением tabIndex, равным 2. Если присвоить значение tabIndex любому объекту, видимому в рабочем поле в текущий момент времени, порядок табуляции определяется исключительно значениями tabIndex для видимых объектов. Если объект не располагает значением свойства tabIndex, он не включается в эту последовательность.

Следующий код формирует три вводимых объекта TextField. Объекты tEmail и tPostalCode располагают присвоенными значениями свойства tabIndex, в то же время к объекту tComments это не относится. Если протестировать данный код, можно заметить, что клавишу <Tab> можно применять для чередования фокуса между tEmail и tPostalCode. Однако для доставки фокуса к полю tComments вам следует выполнить щелчок в этом поле:

```
this.createTextField("tEmail", 1, 25, 35, 100, 20);
tEmail.border = true;
tEmail.type = "input";
tEmail.tabIndex = 1;
```

```
this.createTextField("tPostalCode", 2, 150, 35, 200, 20);
tPostalCode.border = true;
tPostalCode.type = "input";
tPostalCode.tabIndex = 2;

this.createTextField("tComments", 3, 25, 75, 325, 200);
tComments.border = true;
tComments.type = "input";
```

Если присоединить следующую строку кода к предыдущей, можно заметить, что при установке свойства `tabIndex` для `tComments`, `tComments` включается в последовательность:

```
tComments.tabIndex = 3;
```

Объекты `TextField`, `MovieClip` и `Button` располагают свойством `tabIndex`. Более подробная информация об этом свойстве содержится в главе 9.

Мы хотели бы знать ваше мнение по поводу этой главы. Посетите наш сайт по адресу `www.flashsupport.com/feedback` и заполните интерактивную форму.

Резюме

- Текст во Flash подразделяется на статический, динамический и вводимый. С динамическим и вводимым текстами работают посредством ActionScript.
- Объекты `TextField` можно формировать в среде авторских работ с помощью инструмента Text либо в период рабочего цикла, применяя метод `createTextField()`.
- Используя основные свойства класса `TextField`, можно контролировать такие аспекты объекта, как отображаемый текст, перенос слов, цвет текста и т.п.
- Текст Flash может визуализировать некоторые дескрипторы HTML, позволяя внести форматирование и гиперссылки и даже вложить содержимое.
- Текст можно прокручивать в вертикальном и горизонтальном направлениях, применяя встроенные свойства скроллинга.
- Фокус относится к активной части приложения. Можно получать и устанавливать фокус в пределах приложения Flash, пользуясь методами `Selection.getFocus()` и `Selection.setFocus()`.
- Класс `Selection` также позволяет получать и устанавливать выделенный текст в пределах сфокусированного объекта `TextField`.
- С помощью свойств `tabEnabled` и `tabIndex` можно определять, каким образом клавиша <Tab> влияет на фокус в пределах приложения.