В этой главе:

- Кодирование
- Компоненты
- Полноэкранный режим отображения видео
- Субтитры
- Написание собственного кода для воспроизведения видео

12

Видео

Повышенный интерес к Flash в течение последних нескольких лет во многом вызван появлением в нем возможности воспроизведения видео. И это понятно: Flash не только предлагает простой способ доставки видеоконтента с помощью компонентов (предварительно созданных коллекций ресурсов пользовательского интерфейса и ActionScript), но обеспечивает также возможность всестороннего контроля практически всех аспектов воспроизведения видео. Такая простота и при этом управляемость в сочетании с популярностью Flash Player и тем качеством воспроизведения видео, который он обеспечивает, сделали Flash-видео (Flash video, FLV) одним из наиболее привлекательных форматов для работы с видеоданными.

Мы не сможем охватить детально каждый аспект разработки с использованием FLV и поэтому сосредоточимся на разнообразных способах представления видео, а также некоторых ключевых новых возможностях, появившихся во Flash CS3. В этой главе мы рассмотрим:

- **Кодирование.** Объем и тематический охват нашей книги не позволяют углубиться в подробное рассмотрение кодирования видео во Flash, однако небольшого обзора основных идей будет достаточно, чтобы вы могли начать работать с видео. Эти сведения пригодятся нам также при обсуждении создания титров с использованием контрольных точек – меток времени, которые вставляются в видео во время кодирования и могут содержать дополнительные сведения.
- **Компоненты.** Использование компонента FLVPlayback существенно упрощает работу с видео во Flash. Мы рассмотрим компоненты более подробно при обсуждении полноэкранного видео и субтитров.
- **Полноэкранный режим отображения видео.** Специалисты Flash сделали полноэкранный режим отображения видео очень простым

в обращении эффектом. Мы обсудим те шаги, которые необходимы, чтобы представить видео в истинно полноэкранном режиме – режиме, когда видео заполняет весь экран, а не только окно броузера.

- **Субтитры.** Появившийся во Flash CS3 новый компонент со вполне соответствующим именем FLVPlaybackCaptioning дополняет возможности FLVPlayback, упрощая работу с субтитрами и создание субтитров на многих языках. Мы обсудим также некоторые ограничения и способы их преодоления путем реализации субтитров с использованием контрольных точек.
- **Написание кода на ActionScript.** Хотя компоненты являются очень ценными инструментами, мы хотим продемонстрировать также возможность создания аналогичной функциональности с использованием исключительно кода. Отказ от компонентов означает работу с видеоконтентом без каких-либо внутренних графических ресурсов, что сокращает размера SWF-файла.

Начнем с создания ресурсов, которые будут использоваться в примерах этой главы. Предполагается, что в вашем распоряжении имеются видеофайлы – в формате QuickTime, AVI или даже, возможно, оцифрованное видео в формате DV с собственной видеокамеры. В качестве исходных материалов для работы понадобится пара коротких клипов. Чтобы получить оптимальные результаты в полноэкранном режиме, мы рекомендуем начать с видео высокого качества и выполнять оцифровку с максимальным доступным размером кадра.

Примечание

Если у вас уже есть опыт кодирования FLV-файлов, возможно, вы захотите пропустить следующий раздел. Однако во Flash Video Encoder добавлено несколько улучшений, которые будут использоваться в последующих темах, поэтому вам, возможно, стоит хотя бы бегло просмотреть его.

Кодирование

Сегодня на рынке уже доступны несколько кодировщиков FLV – и их появляется все больше и больше. Три ведущих приложения в этой области – Flash Video Encoder от Adobe, Flix Pro от On2 и Squeeze производства компании Sorenson. Наше изложение опирается на использование Flash Video Encoder от Adobe, поскольку он поставляется бесплатно вместе с Flash CS3. Однако на сопроводительном веб-сайте нашей книги вы можете найти дополнительную информацию о других продуктах, а также о решениях для потоковой передачи и кодирования видео в режиме реального времени, предлагаемых компаниями Adobe, On2 и другими.

Начнем с базовых сведений о Flash Video Encoder. Это приложение имеет довольно простой интерфейс. Первый шаг в использовании кодировщка Flash Video Encoder – добавление своего исходного материа-

ла в очередь кодирования. Файлы можно добавить методом drag-and-drop или с помощью кнопки Add (Добавить).

Следующий шаг – выбор параметров кодирования видео- и аудиоданных. Для этого щелкаем по кнопке Settings (Настройки), в результате чего раскрываются основные элементы интерфейса, с которыми мы будем работать. Для краткости воспользуемся оптимизированными настройками по умолчанию, заданными в приложении изначально и представленными на рис. 12.1. Поскольку в дальнейшем нам предстоит работать с полноэкранным видео, мы воспользуемся вариантом «Flash 8 – High Quality (700kbps)», как показано на рис. 12.1, то есть будет применен видеокодек On2 VP6 с качеством кодирования 700 кбит/с и аудиоформат MP3 с качеством кодирования 128 кбит/с, стереозвук.

По щелчку кнопки OK выполняется регистрация выбранных настроек, диалоговое окно закрывается – и вы вновь возвращаетесь в окно очере-

Рис. 12.1. *Если не требуется выполнять специальные настройки для конкретного проекта, начните с одной из стандартных конфигураций Flash 8; в данном случае используется кодек On2 VP6*

ди. Если никакая дальнейшая настройка не нужна, можно начинать кодирование. Для этого мы просто щелкаем по кнопке Start Queue (Начать обработку очереди).

На рис. 12.2 представлен Flash Video Encoder, в интерфейсе которого отображается информация о выбранных настройках и состоянии процесса кодирования.

По умолчанию новый FLV-файл по завершении кодирования будет размещен в одной папке с исходным файлом. В настоящее время при использовании Flash Video Encoder от Adobe нет простого способа просмотра FLV-файлов сразу после кодирования. Однако ситуация быстро меняется: сейчас предлагается ряд FLV-проигрывателей сторонних производителей, и запланированный Adobe Media Player должен будет выйти уже к тому моменту, когда вы будете читать эти строки.[1] Если

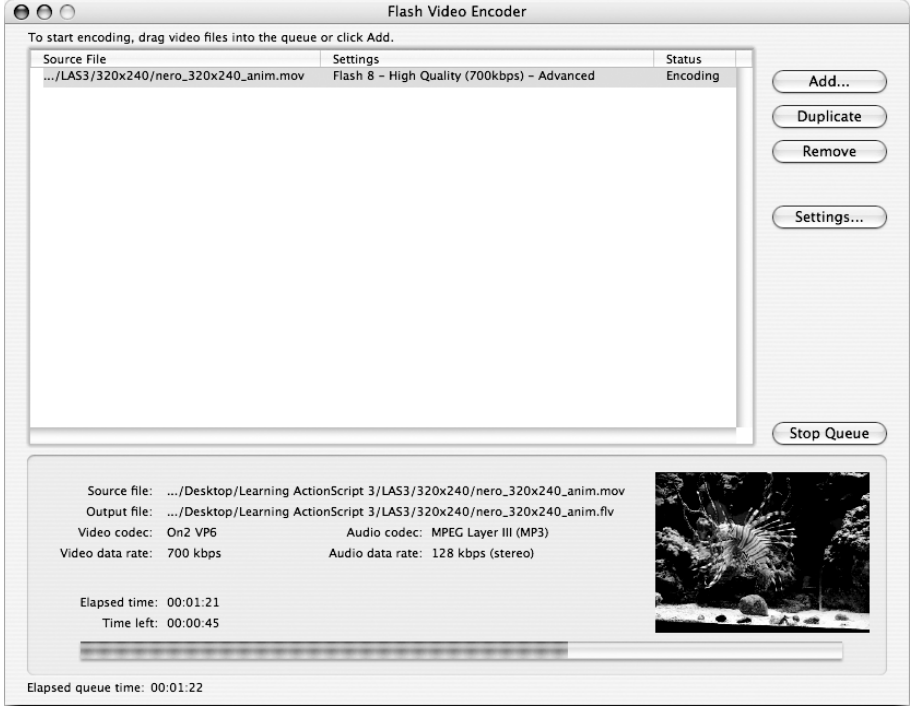

Рис. 12.2. *Во время кодирования Flash Video Encoder отображает индикатор состояния с оценкой прошедшего и оставшегося времени и небольшим окошком предварительного просмотра видео*

[1] Как и предполагали авторы, Adobe Media Player уже выпущен в виде кроссплатформенного AIR-приложения, которое доступно бесплатно по ссылке: *http://www.adobe.com/products/mediaplayer/. – Примеч. науч. ред.*

вы используете Flash CS3 как часть установки Creative Suite, то можете воспроизводить FLV-файлы также с помощью Bridge CS3.

В любом случае, теперь в вашем распоряжении должен быть новый видеофайл Flash с расширением *.flv*. Если вы решили использовать кодек

Поддержка дополнительных видеоформатов

Компания Adobe объявила о планах выпуска следующей версии Flash Player под предварительным названием Flash Player 9 Update 3 (на момент написания данной книги была доступна версия Beta 1). В этом анонсе было заявлено об ограниченной поддержке следующих возможностей: MPEG-4 с кодированным по стандарту H.264 видео и кодированным по стандарту AAC аудио, формат 3GP и формат QuickTime Movie; синхронизированный текст в формате 3GPP (стандартизированный формат субтитров для 3GP-файлов); частоты дискретизации от 8 кГц до 96 кГц (это означает, что вы больше не ограничены только частотами 11,025 кГц, 22,05 кГц и 44,1 кГц во избежание повторных паразитных перекодировок); (нешифрованные) маркеры разделов аудиоданных; и – самое приятное нововведение – подобные ID3 развернутые метаданные, создаваемые iTunes.[1]

Даже если перечисленные возможности, которые уже функционируют в бета-версии Flash Player, сохранятся в окончательной версии лишь частично, это обновление окажет колоссальное влияние на распространение веб-видео. Ожидается, что оно обусловит расширение доступных способов создания веб-видео, устранив необходимость кодировать конечный файл в формат FLV – а значит, уже существующие в перечисленных форматах ресурсы не придется подвергать дополнительным преобразованиям.

Более того, хотя на момент написания данной книги официальные комментарии довольно осторожны, похоже на то, что расширенные возможности, обеспечиваемые этими форматами, такие как многодорожечные аудио- или видеофайлы и многоканальные AAC-файлы (упоминаемая бета-версия сводит данные в два канала и понижает частоту дискретизации до 44,1 кГц), также будут поддерживаться в будущем.

Браво команде разработчиков Flash Player! Это огромный шаг вперед к распространению веб-видео с использованием самого популярного в мире средства воспроизведения.

---

[1] Перечисленные возможности, а также поддержка видео высокого разрешения доступны начиная с версии Flash Player 9.0.115.0 – *Примеч. науч. ред.*

On2 VP6, выбрав при кодировании файла один из предлагаемых стандартных вариантов настроек Flash 8, то для воспроизведения файла вам понадобится Flash Player 8 или более поздняя версия. Мы для своих задач используем Flash CS3, поэтому у вас не должно возникнуть никаких проблем.

Компоненты

Самым быстрым способом добавить видео в Flash-приложение является использование компонента FLVPlayback (рис. 12.3). Он берет на себя большую часть работы, избавляя разработчика от необходимости писать много кода и выполнять какие-либо иные операции.

Те, кто имеет достаточный опыт работы с Flash (или взаимодействия с сообществом Flash), знают, что у Flash-разработчиков сформировалось амбивалентное отношение к компонентам. Безусловно, у компонентов есть свои плюсы и минусы. Очевидное преимущество состоит в том, что вы избавлены от необходимости изобретать велосипед каждый раз при решении задачи, с которой может справиться компонент. Это делает их популярными среди новичков в ActionScript. С другой стороны, внешнего вида и функциональности готового компонента может оказаться недостаточно. Нередко компоненты содержат также программные ошибки. Однако, вероятно, самая большая проблема – это то, что использование компонентов с гарантией влечет за собой увеличение размера файла, а иногда приводит также к падению производительности (на уровне компонента либо в приложении в целом).

Тем, кто избегает применения компонентов, мы покажем, как работать с видео исключительно средствами ActionScript. Если же вы от-

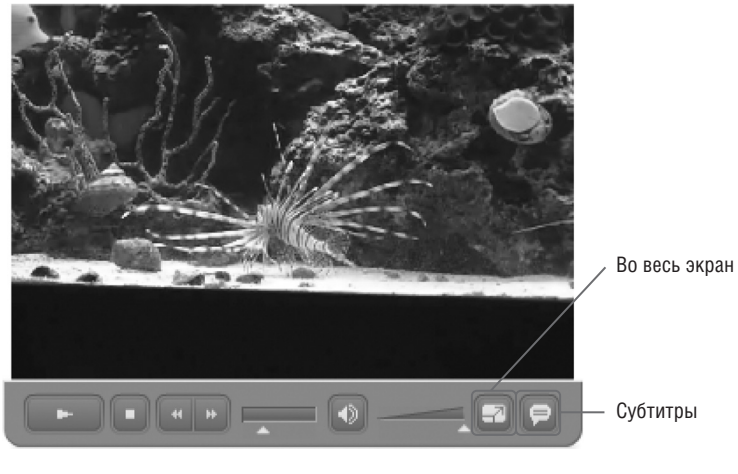

Рис. 12.3. *Компонент FLVPlayback упрощает добавление видео в большинство проектов Flash*

крыты к использованию компонентов, рекомендуем обратить внимание на то, что компонент FLVPlayback, в частности, имеет дополнительные преимущества, сглаживающие некоторые негативные последствия применения компонентов.

Во-первых, у вас есть выбор из нескольких предварительно настроенных контроллеров, или графических тем, но при этом вы можете относительно легко создавать собственные контроллеры. Это позволяет добиваться нужного внешнего вида и функциональности приложения. Во-вторых, компонент может применяться вообще без всякой графической темы. Это позволяет применять готовый код ActionScript для отображения видео, но при этом управлять воспроизведением с помощью собственного кода. Наконец, компонент FLVPlayback обладает рядом возможностей, которые упрощают работу с видео в определенных ситуациях.

Например, компонент умеет автоматически определять необходимость потоковой передачи видео с сервера потоковой передачи, просто выполняя синтаксический разбор URL-адреса источника видеоконтента. Если потоковая передача нужна, он выполнит необходимые начальные запросы к серверу потоковой передачи, так что разработчику не придется обрабатывать эти соединения в сценарии самостоятельно.

На решение о том, использовать компоненты или нет, оказывают влияние все перечисленные и многие другие факторы. Однако в любом случае будет нелишним знать, как компоненты работают, что они делают хорошо и в чем их недостатки. Это подготовит вас к работе с заказчиками и коллегами, предпочитающими компоненты, а также поможет при выборе того, каким образом реализовывать воспроизведение видео в каждой конкретной ситуации.

Работа с компонентом FLVPlayback

Привлекательность компонентов для многих пользователей определяется в первую очередь тем, что для работы с ними не обязательно хорошо знать ActionScript. Практически всю необходимую настройку компонентов можно выполнить через интерфейс Flash и его панели Parameters (Параметры) или Components Inspector (Инспектор компонентов). Однако мы хотим сосредоточиться здесь на ActionScript, поэтому будем создавать экземпляры и конфигурировать компоненты динамически с помощью кода.

Для этого компоненты должны быть расположены в библиотеке вашего файла (или, для опытных пользователей, в Shared Library (Общая библиотека), к которой ваш файл имеет доступ без каких-либо ограничений, связанных с безопасностью или кроссдоменным взаимодействием). Мы будем работать с одним файлом, поэтому вам необходимо только создать новый файл *.fla* и переместить методом drag-and-drop компонент FLVPlayback в библиотеку. В качестве альтернативы вы можете перетащить его на сцену и затем удалить с нее. В обоих случаях

компоненты окажутся в вашей библиотеке. После этого все готово к тому, чтобы приступить к работе.

Поскольку позже в этой главе мы будем рассматривать создание субтитров, в рамках учебнго примера вы можете сразу сделать то же самое для компонента FLVPlaybackCaptioning. Однако когда вы выпускаете свой продукт, желательно не раздувать файл неиспользуемыми компонентами. Итак, если вы не собираетесь снабжать свои файлы *.flv* субтитрами, не предпринимайте никаких дополнительных действий.

В целях обучения мы для простоты начнем с кода, использующего временную шкалу, и добавим в первый кадр файла следующие строки:

```
1  import fl.video.*;
2
3  var vid:FLVPlayback;
4
5  vid = new FLVPlayback();
6  vid.source = "nero_320x240_cp.flv";
7  addChild(vid);
```

В первой строке располагается хорошо знакомый оператор `import`, который неоднократно использовался в предыдущих главах. Он обеспечивает доступ к классу `FLVPlayback`. В строке 3 указывается тип переменной экземпляра, а в строке 5 создается экземпляр компонента FLVPlayback. В строке 6 с помощью переменной экземпляра задается свойство `source` компонента, определяющее видеофайл для воспроизведения. Наконец, в строке 7 экземпляр компонента добавляется в список отображения, который обеспечивает его отображение на сцене.

Этот код проще, чем обычно бывает, поскольку здесь в существенной мере используются значения параметров по умолчанию, в том числе `autoplay` (автоматическое воспроизведение), исходное расположение в точке (0, 0) и отсутствие графической темы. Однако при таких настройках видеофрагмент будет автоматически воспроизведен лишь один раз, после чего останется на экране, не давая никакой возможности управлять им.

Чтобы добавить существующую графическую тему, необходимо выбрать один из доступных вариантов. Вкратце, существуют две основные категории поставляемых тем: располагающиеся поверх или «над» видео (рядом с нижним краем картинки) и располагающиеся вне или «под» видео (сразу под картинкой). Каждая из этих основных групп графических тем содержит богатый набор различных конфигураций, что позволяет выбирать необходимые функции из множества комбинаций: воспроизведение, пауза, остановка, перемотка, выключение звука, громкость, переход в полноэкранный режим и включение субтитров.

Список тем можно увидеть, взглянув на параметр `skin` на панели `Component Inspector`, представленной на рис. 12.4. Для предварительного просмотра тем, поставляемых с Flash, щелкните параметр `skin` на панели `Component Inspector`.

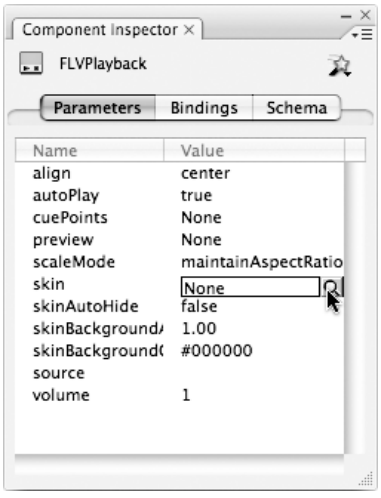

Рис. 12.4. *Инспектор компонентов Flash*

Щелчок по опции skin в Component Inspector открывает диалоговое окно, в котором можно выполнить предварительный просмотр каждой графической темы. Здесь также представлены имена тем, которые можно использовать для быстрой ссылки на тему. На рис. 12.5 показано, как

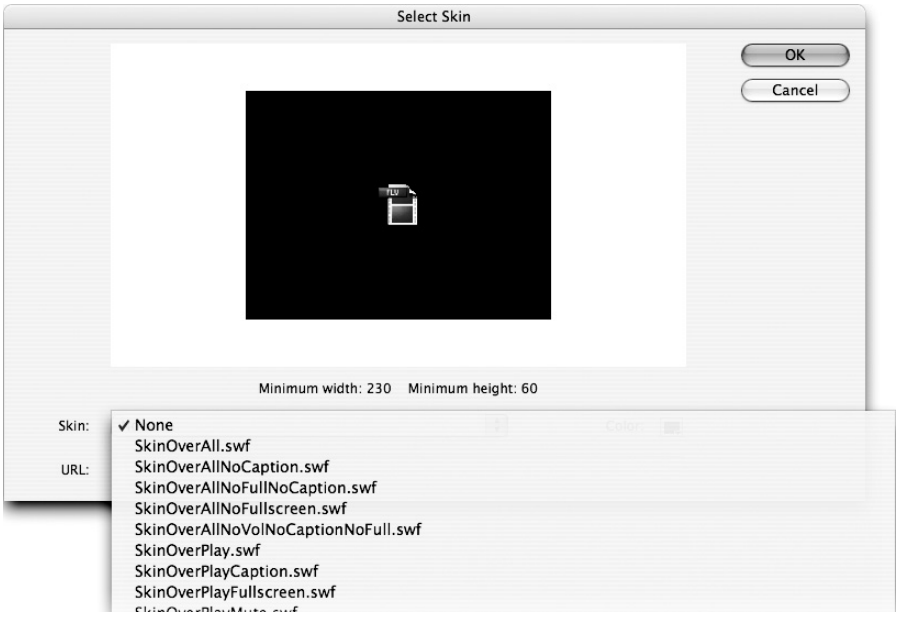

Рис. 12.5. *Графические темы, поставляемые с Flash*

увидеть все темы, которые поставляются с Flash и могут быть использованы в коде на ActionScript.

Выбрав графическую тему, можно определиться с ее цветом и прозрачностью. Эти параметры помогут гармонично сочетать блок управления с видеорядом при использовании графических тем, располагающихся поверх изображения, или с фоном приложения (если он есть) для тем, располагающихся под изображением.

Теперь осталось лишь добавить в существующий сценарий следующие три строки. В строке 7 указывается используемая тема, в строке 8 задается цвет, а в строке 9 – прозрачность.

```
1   import fl.video.*;
2
3   var vid:FLVPlayback;
4
5   vid = new FLVPlayback();
6   vid.source = "nero_320x240_cp.flv";
7   vid.skin = "SkinUnderPlayStopSeekMuteVol.swf";
8   vid.skinBackgroundColor = 0xAEBEFB;
9   vid.skinBackgroundAlpha = 0.5;
10  addChild(vid);
```

Включив эти строки в код, вы добавите к видеофрагменту контроллер, что даст возможность интерактивно управлять воспроизведением. Пример этого сценария с видео, представленным на иллюстрациях к данной главе, можно найти в файле *full_screen_tt_01.fla*.

Примечание

Не забывайте, что в ActionScript 3.0 значения, задаваемые процентными соотношениями (такие как прозрачность), нормированы, то есть диапазоном допустимых значений для них является 0–1, а не 0–100.

Полноэкранный режим отображения видео

Одна из наиболее захватывающих новых возможностей видео во Flash – истинно полноэкранный режим, который доступен начиная с версии Flash Player 9. Термин *истинно полноэкранный режим отображения видео* означает, что изображение само занимает весь экран (это напоминает полноэкранный режим программного DVD-проигрывателя), а не располагается в окне внутри броузера или проигрывателя, развернутого на весь экран.

Flash CS3 предлагает методы доступа к полноэкранному видео через ActionScript (их мы рассмотрим чуть ниже), а также через компонент FLVPlayback. Прежде чем переходить к реализации этого режима, обсудим два предварительных шага, которые обеспечат качественное отображение видео в полноэкранном режиме.

Первый из этих двух шагов – подготовка оптимального исходного материала. Для улучшения кодирования DV-источников Flash Video Encoder теперь поддерживает устранение чересстрочной развертки (deinterlacing) – преобразование двух полей DV-источника (они подобны двум видеокадрам, каждый из которых содержит половину от общего числа строк развертки, отображаемых в два раза быстрее) в кадры, используемые форматом FLV. При работе с чересстрочным исходным материалом чаще всего упоминают о «гребенке», видимой по краям видеоизображений. Устранение чересстрочной развертки исходного видео во время кодирования существенно сглаживает этот эффект.

При выборе одной из новых предустановок DV, имеющихся в Flash CS3 Video Encoder, эта опция будет включена автоматически, а размер кодированного видео будет приведен к стандартному – 640×480 пикселов. Если требуется обеспечить «широкоэкранное» представление, вы можете сохранить размер и пропорции исходного видео, вручную отключив опцию изменения размера. Можно также вручную включить опцию устранения чересстрочной развертки в разделе настроек Video (Видео), как показано на рис. 12.6.

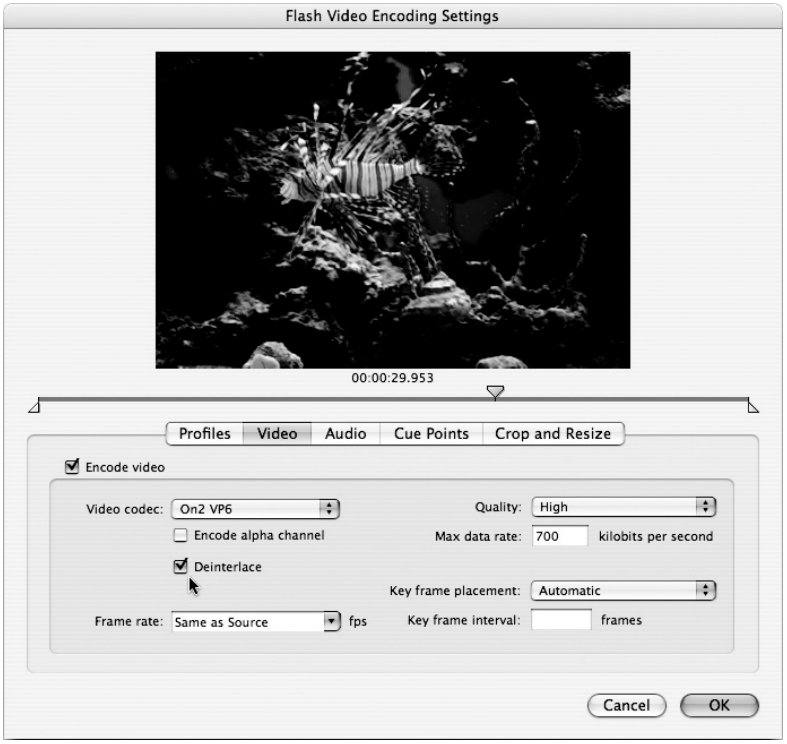

Рис. 12.6. *Устранение чересстрочной развертки доступно в версии Flash Video Encoder, поставляемой с Flash CS3*

Второй необходимый шаг – дать проигрывателю Flash Player указание разрешить переход к полноэкранному отображению. Очевидно, что возможность принимать решение о переключении в полноэкранный режим ни в коем случае не должна быть предоставлена создателям контента – иначе рекламные объявления на Flash выводились бы во весь экран, лишая пользователя всяких средств управления. Эту функцию должен активировать разработчик, предоставляя пользователю возможность переключаться между обычным и полноэкранным режимами.

Чтобы эта возможность стала доступной, в главный HTML-файл проекта необходимо добавить новый параметр `allowFullScreen` со значением `true`. Один из способов сделать это – вручную ввести этот параметр в теги `object` и `embed`, как показано в следующем примере.

```
<object>
  ...
  <param name="allowFullScreen" value="true" />
  <embed ... allowfullscreen="true" />
</object>
```

Другое быстрое и простое решение, очень удобное для тестирования, – выбрать в разделе HTML диалогового окна Publish Settings (Параметры публикации) шаблон публикации Flash Only – Allow Full Screen (Только для Flash – разрешить полноэкранный режим).

Выполнив эти операции, вы можете протестировать файл с помощью команды Publish Preview (Предварительный просмотр публикации) (File→Publish Preview→HTML). Обычно такое значение настроек HTML-шаблона задано в меню Publish Preview по умолчанию, но для выбора доступны и другие варианты настройки. Если вариант HTML недоступен вообще, перейдите в меню File→Publish Settings и добавьте HTML в качестве поддерживаемого формата.

Обеспечив поддержку полноэкранного режима отображения видео в главном HTML-файле, мы подготовились к тому, чтобы предоставить пользователю доступ к опции отображения во весь экран. Чтобы сделать это автоматически в компоненте FLVPlayback, выберите любую графическую тему, которая поддерживает полноэкранный режим или в имени которой есть слово «all» (все), – например, SkinOverPlayFullscreen.swf или SkinUnderAll.swf. Эти и подобные темы добавляют кнопку Full Screen (Во весь экран), как показано на рис. 12.3. Позже в этой главе мы покажем, как можно реализовывать возможность полноэкранного воспроизведения видео в собственном коде на ActionScript.

Субтитры

Субтитры – это текст, отображаемый синхронно с воспроизведением видео. Субтитры очень полезны для перевода звукового ряда на другие языки, что позволяет расширить аудиторию видеоресурса. Они необ-

ходимы также глухим людям и людям с нарушениями слуха как альтернатива звукоряду в диалогах и повествованиях.

Правительство Соединенных Штатов приняло закон, известный как Статья 508 (поскольку является 508-й статьей закона «О реабилитации» от 1973 года), который обозначил определенные требования к доступности контента, предназначенного для использования в государственных учреждениях. Многие частные компании, в том числе работающие на рынке образовательных услуг, также выдвигают требования к доступности контента. По мере все более широкого внедрения этих требований в практику роль субтитров в цифровом видео будет возрастать.

Поддержку отображения субтитров в Flash обеспечивает компонент FLVPlaybackCaptioning, используемый в сочетании с компонентом FLVPlayback. Для использования субтитров необходимо добавить на сцену компонент FLVPlaybackCaptioning во время разработки или динамически с помощью ActionScript на этапе исполнения.

Примечание

Если вы используете компонент FLVPlayback, то для отображения субтитров необходимо выбрать графическую тему, в имени которой есть слово «Caption» (субтитр). Такие темы предоставляют пользователю кнопку, которая позволяет включать и отключать субтитры. Эту кнопку можно увидеть внизу справа на рис. 12.7, а также на рис. 12.3, где она указана выноской.

Самый простой способ вывода субтитров – использовать сам компонент FLVPlayback. Если в конкретный момент времени на сцене присутствует только один экземпляр компонента FLVPlayback, то компонент,

Рис. 12.7. *Субтитры могут отображаться в рамках компонента FLVPlayback*

отвечающий за субтитры, в качестве поведения по умолчанию автоматически выявляет компонент воспроизведения и использует его внутренний текстовый элемент как место для размещения субтитров. Если требуется присутствие на сцене более одного элемента, вы можете вручную указать тот компонент FLVPlayback, контент которого необходимо снабдить субтитрами, а также задать собственную цель для размещения субтитров (в тех случаях, когда требуется использовать другой текстовый элемент – возможно, интегрированный в ваш интерфейс, а не в видеоряд). Результат будет близок к представленному на рис. 12.7.

Создание субтитров с использованием Timed Text

Прежде чем выводить субтитры, необходимо создать их наполнение. Есть два простых способа подготовки субтитров. Предпочтительным является создание XML-файла с использованием формата W3C Timed Text, который формально называют Distribution Format Exchange Profile (DFXP)[1], а между собой – просто TT.

В этой главе мы рассмотрим лишь часть функциональных возможностей Timed Text; получить дополнительную информацию об этом формате вы можете на сайте W3C по адресу *http://www.w3.org/AudioVideo/TT/*. О подмножестве функций этого формата, поддерживаемом в Flash CS3, можно прочитать в разделе «Timed Text Tags»[2] встроенной справочной системы. MAGpie, инструмент создания субтитров, разработанный Национальным центром доступности мультимедиа (National Center for Accessible Media – NCAM) – лидером в этой области, уже поддерживает формат DFXP. Когда эта книга готовилась к печати, Manitu Group также объявила о планах реализовать поддержку DFXP в своем продукте Captionate.

Создать собственный файл Timed Text нетрудно. Пример XML-кода, рассматриваемый в этом разделе, представляет собой немного измененный фрагмент файла, поставляемого с нашим примером видеоматериала *nero_720x480_cp.flv*. Для краткости в этой печатной версии присутствуют только два субтитра, причем стиль второго был изменен, чтобы продемонстрировать функции, представленные в полной версии файла. Мы ограничимся обсуждением этой сокращенной версии, поскольку она включает в себя большинство функций, используемых в типовом проекте с субтитрами. Однако не забывайте, что это не полный исходный файл.

---

[1] Профиль обмена форматами распространения. – *Примеч. перев.*

[2] На русском языке этот раздел справки («Использование субтитров в формате Timed Text») доступен по адресу *http://help.adobe.com/ru_RU/ActionScript/3.0_UsingComponentsAS3/WS5b3ccc516d4fbf351e63e3d118a9c65b32-7ee5.html*. – *Примеч. науч. ред.*

Внимание

В статье справки Flash «Timed Text Tags» указано, что игнорируются все атрибуты тега `<tt>`, однако если удалить атрибут `xmlns:tts`, то Flash CS3 сформирует ошибки в классах `TimedTextManager`, `EventDispatcher` и `URLLoader`. Если опустить только атрибут `xmlns`, ошибок не возникнет, но к субтитрам не будет применено стилевое оформление. Оба эти атрибута должны рассматриваться как обязательные.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <tt xmlns="http://www.w3.org/2006/04/ttaf1"
3    xmlns:tts="http://www.w3.org/2006/04/ttaf1#styling">
4    <head>
5      <styling>
6        <style id="1"
7          tts:textAlign="center"
8          tts:fontFamily="_sans"
9          tts:fontSize="18"
10         tts:fontWeight="bold"
11         tts:color="#FFFF00FF"/>
12       <style id="2" tts:backgroundColor="#00000000"/>
13       <style id="3" tts:backgroundColor="#FFFFFFFF"/>
14       <style id="trans" style="1 2"/>
15       <style id="opaq" style="1 3"/>
16     </styling>
17   </head>
18   <body>
19     <div>
20       <p begin="00:00:05.00" dur="00:00:04.00" style="opaq">Немо -
         рыба-зебра<br />
21       (<span tts:fontStyle="italic">Pterois volitans</span>),</p>
22       <p begin="00:00:09.00" dur="00:00:02.00" style="trans"> в домашнем
         аквариуме, имитирующем коралловый риф.</p>
23     </div>
24   </body>
25 </tt>
```

В строках с 1 по 3 располагаются два стандартных тега для идентификации файла. Первый тег является xml-объявлением; мы рекомендуем указывать кодировку UTF-8 (как в самом теге, так и при создании файла), чтобы обеспечить поддержку специальных символов. Это особенно важно при создании субтитров на разных языках. Второй тег – корневой тег документа. Обратите внимание на примечание об особенностях использования атрибутов данного тега.

Примечание

Полный список поддерживаемых и неподдерживаемых свойств TT можно найти в разделе «Timed Text Tags» справки Flash. Вот некоторые из них, заслуживающие особого внимания:

- `fontFamily` (семейство шрифта) поддерживает шрифты устройства, как видно из нашего примера.
- `fontSize` (размер шрифта) использует только первый из вертикальных размеров, если их задано несколько. Поддерживает абсолютные и относительные размеры, но не указание размера в процентах.
- `lineHeight` (высота строки), `padding` (внутренние поля) и `overflow` (переполнение), хотя и являются потенциально полезными для оформления субтитров, относятся к неподдерживаемым свойствам.

Существующий в единственном экземпляре парный тег `<head>` (строки 4 и 17) является необязательным, но мы рекомендуем использовать его, поскольку это значительно упрощает стилевое оформление субтитров. Парный тег `<styling>`, который также может быть представлен лишь в единственном экземпляре (строки 5 и 16), тоже является необязательным, но необходим для создания *стилей*. Стили перечислены в строках с 6 по 15 и являются сущностями каскадной таблицы стилей (CSS) для документа в формате Timed Text. Вы можете создать сколько угодно стилей, но каждый из них должен иметь уникальный идентификатор, задаваемый атрибутом `id`. Атрибуты стилей, которые фактически определяют форматирование, очень похожи на свойства CSS, но снабжены префиксом `tts:`.

Стили можно назначать напрямую, используя их идентификаторы, однако имеется также возможность более эффективно управлять форматированием, создавая новые стили, состоящие из существующих стилей. Обратимся к стилям нашего примера. Мы хотим получить два представления субтитров: одно с черным фоном для использования на светлых зонах видео, а другое – с прозрачным фоном, чтобы обеспечить минимальное перекрытие видеоряда текстом.

Примечание

Более подробную информацию о формате #AARRGGBB можно найти в главе 9.

Стиль *1* состоит из всех стилевых атрибутов, общих для обоих представлений, то есть в него не включена прозрачность фона. Стили *2* и *3* описывают только цвет фона и делают его прозрачным или непрозрачным соответственно. Формат Timed Text также использует нотацию цвета #AARRGGBB, но компоненты Flash игнорируют конкретные значения первой пары цифр, определяющей альфа-канал, извлекая из нее только параметр для определения прозрачности и непрозрачности: значение 00 представляет прозрачный фон, а *любое* отличное от нуля значение задает непрозрачный фон. (Мы использовали противоположное нулю значение, #FFFFFFFF, чтобы напомнить, что описывается непрозрачный фон.)

Вы можете непосредственно задать для субтитров несколько стилей (например, *id="1 2"*), но таким же способом можно создать новый

стиль. Это позволяет присвоить ему легко узнаваемое имя, что мы и сделали в строках 14 и 15, определив, что стиль «trans» (сокращение от английского «transparent», то есть «прозрачный») обеспечивает прозрачный фон, поскольку использует стили с идентификаторами *1* и *2*, а «opaq» (сокращение от английского «opaque», то есть «непрозрачный») – непрозрачный, потому что использует стили с идентификаторами *1* и *3*.

Для применения стилей ко всем субтитрам требуется один парный тег `<body>` (строки 18 и 25). Обязательным является также парный тег `<div>` (строки 19 и 24). Описание этого требования в документации несколько туманно. Удалив тег `<div>`, мы получили ошибку, в которой сообщалось, что тег `<body>` не поддерживает теги `<p>`. Таким образом, выяснилось, что тег `<div>` является обязательным. Аналогичным образом дело обстоит с тегами `<p>` (строки с 20 по 23): в разделе «Timed Text Tags» справки Flash говорится, что поддерживаются ноль или более тегов абзаца, но мы не нашли логичного способа применить временные и стилевые атрибутов к отдельным субтитрам без этих тегов. Например, теги `<span>` (строка 21) поддерживаются, но не в теге `<body>` непосредственно. Поэтому мы предлагаем считать теги `<p>` обязательными для каждого субтитра.

Атрибут `begin`, определяющий время показа субтитра, является обязательным для каждого субтитра (в нашем случае – в каждом теге `<p>`). Атрибуты `dur` (от «duration» – продолжительность) и `end` (время окончания отображения субтитра) необязательные, если они опущены, субтитр будет оставаться на экране до момента показа следующего. Время может быть задано с использованием полного формата времени (ЧЧ:ММ:СС.m, где m – миллисекунды), неполного формата времени (ММ:СС.m или СС.m) или в виде смещения времени (с помощью единиц времени, например, «1s» для смещения в одну секунду). Тики и кадры не поддерживаются.

Примечание

В нашем основном примере Timed Text мы для ясности и единообразия использовали полный формат задания времени (с указанием длительности) даже тогда, когда завершение отображения одного субтитра совпадает по времени с моментом начала отображения следующего. Однако неполный формат задания времени позволяет упростить задачу, опуская атрибуты продолжительности или окончания отображения, если субтитр должен оставаться на экране до момента замены его другим субтитром. Испаноязычный пример, представленный ниже в иллюстративных целях, отформатирован именно таким образом.

Использование файла в формате Timed Text

Стандартный вариант реализации субтитров в формате Timed Text – их отображение в рамках компонента FLVPlayback. Для этого необходимо лишь добавить компонент на сцену и задать его атрибуты. В при-

веденном ниже фрагменте новый код, дополняющий пример с компонентом FLVPlayback, выделен жирным шрифтом:

```
1  import fl.video.*;
2
3  var vid:FLVPlayback;
4  var cap:FLVPlaybackCaptioning;
5
6  vid = new FLVPlayback();
7  vid.source = "nero_720x480_tt.flv";
8  vid.skin = "SkinUnderAll.swf";
9  vid.skinBackgroundColor = 0xAEBEFB;
10 vid.skinBackgroundAlpha = 0.5;
11 addChild(vid);
12
13 cap = new FLVPlaybackCaptioning();
14 cap.source = "nero_timed_text.xml";
15 addChild(cap);
```

В строке 4 объявляется переменная экземпляра, в строке 13 создается экземпляр компонента, в строке 14 задается его источник, а в строке 15 экземпляр компонента добавляется в список отображения. Обратите внимание, что в строке 8 заменяется графическая тема, чтобы обеспечить наличие кнопки включения/выключения отображения субтитров (и, в данном случае, также кнопки для перехода в полноэкранный режим). Эти изменения позволят пользователю при воспроизведении видео отображать и скрывать субтитры по своему желанию.

Примечание

Хорошим тоном при использовании кнопки субтитров является изначальная установка свойства showCaptions в значение false, чтобы начать воспроизведение без них, предоставив пользователю возможность выбирать, будет он или нет видеть субтитры. Однако в наших примерах мы опустили этот шаг, чтобы быстрее перейти к тестированию.

Создание субтитров с использованием контрольных точек

Другой способ добавления субтитров – внедрение данных во временные метки, называемые *контрольными точками* (*cue points*), в процессе кодирования видео. Преимущество такого подхода состоит в постоянном наличии информации субтитров, но в этом же постоянстве заключается и недостаток: для изменения встроенных контрольных точек приходится повторно кодировать видео.

Вставить контрольные точки просто. В процессе настройки параметров кодирования, который мы уже обсуждали, перейдите в раздел Cue Points (Контрольные точки) и перетащите ползунок под окном предварительного просмотра видео к тому моменту, в который необходимо вы-

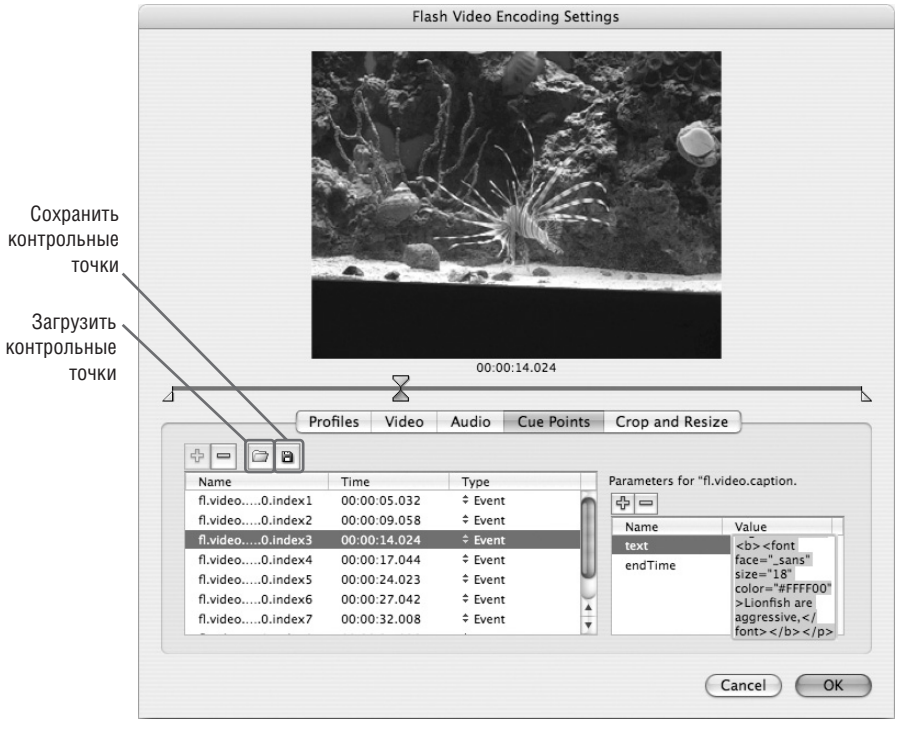

Рис. 12.8. *Субтитры, встроенные в файл FLV с использованием контрольных точек*

вести субтитр. Затем щелкните по кнопке + (плюс) в интерфейсе под видеорядом и введите в поле соответствующее значение. Рис. 12.8 иллюстрирует этот процесс.

Значения для субтитров должны подчиняться специальным рекомендациям, разработанным для компонента FLVPlaybackCaptioning. Вы можете получить более развернутые сведения на эту тему, выполнив поиск в справочной системе Flash CS3 по запросу «cue point standards» (стандарты для контрольных точек). Вот краткое описание атрибутов контрольных точек:

`Time` (время)

Заполняется автоматически исходя из позиции ползунка в Flash Video Encoder.

`Name` (имя)

Значение должно начинаться с «fl.video.caption.2.0.», за которым следует строка с положительным целым числом, увеличивающимся на единицу для каждой следующей контрольной точки (например, «fl.video.caption.2.0.index1», «fl.video.caption.2.0.index2» и т. д.).

`Type` (тип)

Должен имеет значение `Event`.

Сами субтитры затем задаются как параметры контрольных точек. Добавление параметров в контрольную точку аналогично добавлению контрольной точки в видео. Выбрав контрольную точку, нажмите кнопку + (плюс). Это обеспечит добавление пары «имя – значение». У параметра есть следующие опции:

`text` (текст)

Собственно текст субтитра, в нем могут использоваться HTML-теги, которые поддерживает Flash. Этот параметр является обязательным.

`endTime` (время окончания)

Продолжительность отображения субтитра в секундах. Этот параметр является необязательным, но если он не используется, субтитр будет оставаться на экране до завершения видео. Это приводит к эффекту наложения субтитров, что не имеет какой-либо практической пользы, поэтому, с нашей точки зрения, этот параметр должен присутствовать всегда.

`backgroundColorAlpha` (прозрачность фонового цвета)

Булево значение, задающее прозрачность фона. Значение `true` означает, что субтитр не имеет фонового цвета. Этот параметр является необязательным, значение по умолчанию – `true`.

`wrapOption` (опция переноса)

Также булево значение, которое определяет, будет ли выполняться перенос субтитров, и в случае необходимости обеспечивает добавление дополнительных строк для отображения всего текста. Этот параметр является необязательным, значение по умолчанию – `true`.

Внимание

Важные замечания, касающиеся данного процесса, можно найти во врезке «Проблемы создания субтитров с использованием контрольных точек».

Заполнение контрольных точек вручную может оказаться очень трудоемким процессом, в частности, из-за размера и ограниченных возможностей интерфейса редактирования. К счастью, благодаря новым возможностям Flash CS3 эта процедура значительно облегчается. Теперь мы можете импортировать и экспортировать списки контрольных точек в формате XML. Это означает, что вы можете добавить контрольные точки вручную, используя для точной синхронизации предварительный просмотр видеоряда и не беспокоясь при этом обо всех остальных параметрах. Затем эти данные могут быть экспортированы во внешний файл и дополнены остальными необходимыми деталями в обычном текстовом редакторе. Кнопками для загрузки и сохранения являются значки открытой папки и флоппи-диска соответственно (рис. 12.8).

В сопроводительных материалах есть XML-файл с контрольными точками для видеофрагмента, с которым мы работаем в этой главе, так что вы можете опробовать процесс кодирования на собственном видеоматериале. Ниже представлен фрагмент этого XML-файла. Он представляет собой простой XML-документ, включающий тег CDATA при использовании HTML в тексте субтитра. При этом стоит обратить внимание на некоторые моменты, касающиеся создания подобных файлов, которые обсуждаются во врезке «Проблемы создания субтитров с использованием контрольных точек».

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FLVCoreCuePoints>
  <CuePoint>
    <Time>5032</Time>
    <Type>event</Type>
    <Name>fl.video.caption.2.0.index1</Name>
    <Parameters>
      <Parameter>
        <Name>text</Name>
        <Value><![CDATA[<p align="center"><b><fontface="_sans"size="18"
        color="#FFFF00">Неро - рыба-зебра<br>(<i>Pterois volitans</i>),
        </font></b></p>]]></Value>
      </Parameter>
      <Parameter>
        <Name>endTime</Name>
        <Value>9.000</Value>
      </Parameter>
      <Parameter>
        <Name>backgroundColorAlpha</Name>
        <Value>true</Value>
      </Parameter>
    </Parameters>
  </CuePoint>
</FLVCoreCuePoints>
```

Использование субтитров, создаваемых с помощью контрольных точек, в FLV-файле

Вывод субтитров, созданных с использованием контрольных точек, аналогичен выводу субтитров в формате Timed Text. Однако, поскольку в этом случае субтитры встроены в видеоданные, требуется на одну строку ActionScript-кода меньше: отпадает необходимость в свойстве `source` компонента FLVPlaybackCaptioning, потому что субтитры автоматически извлекаются путем синтаксического разбора FLV-файла. Внесенные в код нашего примера изменения выделены ниже жирным шрифтом. В строке 7 в качестве ресурса задается встроенный источник контрольных точек, а строка 8 устраняет возможность отображения во весь экран путем задания соответствующей графической темы. Строки 13 и 14 выделены жирным шрифтом не потому, что они новые,

а потому, что ранее между ними находилась строка, задающая источник субтитров, которая теперь опущена.

```
import fl.video.*;

var vid:FLVPlayback;
var cap:FLVPlaybackCaptioning;

vid = new FLVPlayback();
vid.source = "nero_320x240_cp.flv";
vid.skin = "SkinUnderAllNoFullscreen.swf";
vid.skinBackgroundColor = 0xAEBEFB;
vid.skinBackgroundAlpha = 0.5;
addChild(vid);

cap = new FLVPlaybackCaptioning();
addChild(cap);
```

Какой бы способ добавления субтитров в проект вы ни выбрали, в вашем распоряжении будут одни и те же функциональные возможности и способы оформления.

Проблемы создания субтитров с использованием контрольных точек

При создании субтитров в FLV-файле через встроенные контрольные точки с использованием ресурсов, поставляемых с версией Flash CS3, возникают некоторые ошибки. Однако оснований для беспокойства нет, поскольку мы предлагаем вам приемы для решения этих проблем и заменяющий компонент, который поможет исправить ситуацию.

Проявляющиеся проблемы можно разбить на две категории: мелкие программные ошибки, связанные с компонентом FLVPlaybackCaptioning, и некоторые моменты, касающиеся форматирования при кодировании контрольных точек путем импорта внешнего XML-файла. Начнем с обсуждения ошибок, потому что они имеют большее значение и в то же время легко устраняются.

Первая ошибка связана с заданием значения для свойства `backgroundColorAlpha`. Это свойство указывает компоненту FLVPlayback удалить цвет фона из внутреннего поля субтитров. Документация говорит, что это свойство требует булева значения, но, к сожалению, в действительности значение интерпретируется как строка. В итоге вы можете изначально получить непрозрачный фон как значение по умолчанию, но любое изменение этого свойства контрольной точки будет интерпретировано как `true`.

Это означает, что можно один раз перейти к прозрачному фону, но нельзя вернуться к непрозрачному фону.

Вторая ошибка касается значения свойства `track`. Это свойство подробно рассматривается в следующем разделе, «Предоставление субтитров на нескольких языках», а если говорить коротко, то оно предназначено для переключения наборов субтитров. Например, можно переключаться между субтитрами на разных языках или между субтитрами и описанием. Однако в поставляемом компоненте задать значение этому свойству невозможно, поэтому использоваться могут только субтитры, указанные как значение свойства `text` по умолчанию.

На сопроводительном веб-сайте этой книги объясняется, как изменить код. Там же предлагается компонент для замены, который можно просто поместить в каталог установки (инструкции прилагаются) для корректировки данной функциональности. (Спасибо Джеффу Камереру (Jeff Kamerer) за помощь в решении проблемы с `backgroundColorAlpha`.)

Покончив с известными ошибками, мы хотим сэкономить ваше время, указав на некоторые проблемы форматирования, которые могут возникнуть при написании собственного XML-файла субтитров для Flash Video Encoder. Кодировщик использует для этого типа ресурса собственную структуру файла, поэтому не пытайтесь перед процедурой импорта привести XML к строгому соответствию стандартам. Например, если унифицировать регистр тегов, операция импорта даст сбой. Просто четко следуйте образцу, который задан в представленном примере, – и все будет замечательно.

Отметим также, что обязательный атрибут `Time` задается в миллисекундах, тогда как необязательный параметр `endTime` – в секундах. Например, первый субтитр начинает отображаться в момент 5032 миллисекунды и прекращает отображаться в момент 9,000 секунд. Эти единицы не взаимозаменяемы.

Наконец, в отличие от формата Timed Text, отсутствие параметра `endTime` не приведет к замене субтитра следующим. Вместо этого новый субтитр будет добавлен в текстовое поле предыдущего субтитра. Это позволяет создавать многострочные субтитры в несколько шагов, но не является типичным для показа субтитров поведением. По этой причине мы предлагаем считать параметр `endTime` обязательным.

Предоставление субтитров на нескольких языках

DVD-фильмы с богатым набором функций часто предлагают субтитры на нескольких языках. Это дает возможность распространять фильм в разных странах и среди разных зрительских аудиторий, делая его доступным в том числе для людей с физическими ограничениями. Такого же результата можно достичь, используя компонент FLVPlaybackCaptioning в Flash CS3. В данной главе мы обсуждали два подхода работы с этим компонентом – с использованием внешнего XML-файла в формате Timed Text (DXFP) и с помощью встроенных контрольных точек. Оба варианта поддерживают многоязычность, но очень разными способами.

Timed Text

В случае с использованием формата Timed Text необходимо лишь подготовить набор DXFP-файлов – по одному для каждого языка – и переключаться между ними по мере необходимости. Однако поведение поставляемого компонента FLVPlaybackCaptioning приводит при этом к довольно странным эффектам.

Во-первых, этот компонент разработан так, что при изменении субтитров он перезаписывает содержимое поля субтитров, только если исходное содержимое состоит из одних пробелов. В противном случае, как это происходит при переключении субтитров с одного языка на другой, он добавляет новый текст к уже существующему. В результате мы получаем текст одновременно на двух языках (например, английском и испанском), который будет отображаться на экране до тех пор, пока контрольная точка не обеспечит его перезапись. Во-вторых, для определения того, загружен ли файл DXFP, этот компонент использует метод, не обеспечивающий мгновенного изменения. Поэтому, чтобы увидеть обновление языка, придется подождать следующего субтитра.

На сопроводительном веб-сайте представлены более развернутые сведения на эту тему, но, к счастью, существует простой прием для решения описанных проблем. Все, что требуется сделать, – отключить отображение субтитров перед переключением DXFP-источника, а затем включить субтитры снова. Файл примера *full_screen_tt.fla* демонстрирует применение компонента Button для переключения файлов субтитров. Этот компонент, расположенный в категории User Interface (Пользовательский интерфейс) панели Components (Компоненты), должен присутствовать в вашей библиотеке.

Примечание

При необходимости вернитесь к разделу «Работа с компонентом FLVPlayback», где приведено более подробное описание процедуры добавления компонентов в файл.

```
1   import fl.video.*;
2   import fl.controls.Button;
3
4   var vid:FLVPlayback;
5   var cap:FLVPlaybackCaptioning;
6   var capsLangBtn:Button;
7   var vidSize:Rectangle;
8
9   vid = new FLVPlayback();
10  vid.source = "nero_720x480_tt.flv";
11  vid.skin = "SkinUnderAll.swf";
12  vid.skinBackgroundColor = 0x0066CC;
13  vid.skinBackgroundAlpha = 0.5;
14  addChild(vid);
15
16  cap = new FLVPlaybackCaptioning();
17  cap.source = "nero_timed_text.xml";
18  addChild(cap);
19
20  capsLangBtn = new Button();
21  capsLangBtn.label = "English/Spanish";
22  vidSize = vid.getBounds(this);
23  capsLangBtn.x = vidSize.right + 20;
24  capsLangBtn.y = vidSize.bottom;
25  addChild(capsLangBtn);
26  capsLangBtn.addEventListener(MouseEvent.CLICK, onSwitchTTCaps,
    false, 0, true);
27
28  function onSwitchTTCaps(evt:MouseEvent):void {
29    cap.showCaptions = false;
30    switch(cap.source) {
31      case "nero_timed_text.xml":
32        cap.source = "nero_timed_text_sp.xml";
33        break;
34      case "nero_timed_text_sp.xml":
35        cap.source = "nero_timed_text.xml";
36        break;
37
38    }
39    cap.showCaptions = true;
40  }
```

В строке 2 приведенного выше кода выполняется импорт класса `Button`, что обеспечивает возможность создания его экземпляра и работы с компонентом Button. В строках 6 и 7 описываются переменные экземпляров. Для позиционирования кнопки в нижнем правом углу компонента FLVPlayback будет использоваться прямоугольник. В строках с 20 по 24 задается ряд свойств кнопки, включая надпись и местоположение на сцене. Метод `getBounds()` в строке 22 использует систему координат главной временной диаграммы (ссылкой на которую является ключевое слова `this`) для получения значений `x`, `y`, `width` и `height`

компонента FLVPlayback. В строке 25 кнопка добавляется в список отображения, а в строке 26 описывается слушатель события для вызова функции switchTTCaps() по событию щелчка мыши. Наконец, функция switchTTCaps() (строки с 28 по 38) отключает отображение субтитров, проверяет, какой источник субтитров используется, переключается к другому файлу и вновь включает отображение субтитров.

Если вам необходимо поддерживать более двух языков, это решение придется доработать. Наша реализация ни в коей мере не является полной – это всего лишь базовый прототип, требующий минимума кода и специальных ресурсов. Если вы решите использовать эту технику, предлагаем вам расширить возможности приложения, например, добавив переключатель языка субтитров, показывающий текущий язык субтитров и отображаемый только тогда, когда свойство showCaptions имеет значение true, и дополнив список состояний вариантом отсутствия субтитров. Попробуйте использовать функцию *«Субтитры»* своего DVD-проигрывателя, чтобы увидеть еще один пример реализации этих функций.

Контрольные точки

Для обеспечения показа субтитров на нескольких языках необходимо встроить эти дополнительные языки в видеоматериал в процессе кодирования. Можно добавить любое количество языков. Для их различения служит положительное целое число, используемое свойством track компонента FLVPlaybackCaptioning. По умолчанию свойство track не используется и источником субтитров является значение параметра text. Однако, если свойство track имеет некое отличное от нуля положительное целое значение – скажем, *n*, – компонент будет использовать субтитры, являющиеся содержимым параметра text*n*. Так, при значении track, равном 1, будут использованы субтитры, обозначенные как text1, для значения 2 – субтитры, помещенные в параметр text2, и т. д.

Вот предыдущий пример XML-файла, в который добавлен второй язык субтитров:

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
2  <FLVCoreCuePoints>
3    <CuePoint>
4      <Time>5032</Time>
5      <Type>event</Type>
6      <Name>fl.video.caption.2.0.index1</Name>
7      <Parameters>
8        <Parameter>
9          <Name>text</Name>
10         <Value><![CDATA[<p align="center"><b><font face=" sans" size="18"
           color="#FFFF00">Иеро - рыба-зебра<br>(<i>Pterois volitans</i>),
           </font>< b></p>]]></Value>
11       </Parameter>
```

```
12        <Parameter>
13          <Name>text1</Name>
14          <Value><![CDATA[<p align="center"><b><font face="_ sans"
            size="18" color="#FFFF00">Nero es un lionfish<br>(<i>Pterois
            volitans</i>),</font></b></p>]]></Value>
15        </Parameter>
16        <Parameter>
17          <Name>endTime</Name>
18          <Value>9.000</Value>
19        </Parameter>
20        <Parameter>
21          <Name>backgroundColorAlpha </Name>
22          <Value>true</Value>
23        </Parameter>
24      </Parameters>
25    </CuePoint>
26  </FLVCoreCuePoints>
```

Для переключения языков необходимо задать значение свойства `track`. Обратите внимание: значение 1 свойства `track` используется для отображения содержимого параметров `text1` контрольных точек (испанских субтитров), тогда как значение 0 возвращает компонент к использованию для каждой контрольной точки параметра `text` (русских субтитров). Переключение между языками не приводит к каким-либо побочным эффектам вроде комбинации строк на разных языках, поэтому нет необходимости в отключении и последующем включении субтитров. В следующем фрагменте кода жирным шрифтом выделены только строки, отличающиеся от примера с использованием Timed Text.

```
1   import fl.video.*;
2   import fl.controls.Button;
3
4   var vid:FLVPlayback;
5   var cap:FLVPlaybackCaptioning;
6   var capsLangBtn:Button;
7   var vidSize:Rectangle;
8
9   vid = new FLVPlayback();
10  vid.source = "nero_320x240_cp.flv";
11  vid.skin = "SkinUnderAllNoFullscreen.swf";
12  vid.skinBackgroundColor = 0x0066CC;
13  vid.skinBackgroundAlpha = 0.5;
14  addChild(vid);
15
16  cap = new FLVPlaybackCaptioning();
17  addChild(cap);
18
19  capsLangBtn = new Button();
20  capsLangBtn.label = "English/Spanish";
21  vidSize = vid.getBounds(this);
22  capsLangBtn.x = vidSize.right + 20;
```

```
capsLangBtn.y = vidSize.bottom;
addChild(capsLangBtn);
capsLangBtn.addEventListener(MouseEvent.CLICK, onSwitchFLVCaps,
false, 0, true);

function onSwitchFLVCaps(evt:MouseEvent):void {
  if (cap.track == 0) {
    cap.track = 1;
  } else {
    cap.track = 0;
  }
}
```

Написание собственного кода для воспроизведения видео

До сих пор при воспроизведении FLV мы полагались исключительно на компоненты. Однако важно помнить, что написание собственного кода позволяет сократить размеры файла, дает возможность настраивать функциональность в широких пределах и устраняет зависимость от компонента FLVPlayback при проектировании пользовательского интерфейса. По этим причинам решения, использующие исключительно код, являются предпочтительными. Они, однако, ограничены с точки зрения дизайна, поскольку не позволяют применить элементы оформления, подготовленные дизайнером. Следовательно, выбор подхода на основе чистого кода должен быть результатом вдумчивого анализа всех плюсов и минусов, а не самоцелью в любой ситуации.

В данном разделе мы представляем готовый класс `BasicVideo`, позволяющий создать очень простой видеопроигрыватель с использованием исключительно программного кода. Даже кнопки рисуются динамически с помощью методик, которые мы обсуждали в главе 8. В результате получается SWF-файл размером всего лишь 4 Кб. Как обычно, этот класс может быть написан в любом текстовом редакторе и должен быть сохранен в текстовом файле с именем *BasicVideo.as*.

```
package {

  import flash.display.*;
  import flash.net.*;
  import flash.media.Video;
  import flash.events.*;
  import CreateRoundRectButton;

  public class BasicVideo extends Sprite {

    private var _vidConnection:NetConnection;
    private var _vidStream:NetStream;
    private var _vid:Video;
```

```
14      private var _vidURL:String;
15      private var _vidPlaying:Boolean;
16      private var _infoClient:Object;
17      private var _playBtn:Sprite;
18      private var _pauseBtn:Sprite;
19      private var _stopBtn:Sprite;
```

Первые 19 строк являются стандартным описанием структуры пакета. Сюда входит объявление пакета (строка 1), директивы импорта внешних классов (строки с 3 по 7), объявление класса (строка 9) и объявления переменных (строки с 11 по 19). Заметьте, что как для кнопок простого блока управления, так и для самого класса используется объект отображения `Sprite`. (Не забывайте также о закрывающих фигурных скобках для объявления класса и пакета в строках 99 и 100.)

```
20      public function BasicVideo () {
21
22        _vidConnection = new NetConnection();
23        _vidConnection.connect(null);
24        _vidStream = new NetStream(_vidConnection);
```

Конструктор класса, начинающийся в строке 20, первым делом создает экземпляр класса `NetConnection` для организации сетевого соединения. Это первый шаг в создании видеопроигрывателя, благодаря которому устанавливается соединение с удаленным сервером потоковой передачи – например, с Flash Media Server или одним из альтернативных сервисов, которых сейчас появляется все больше и больше. Для организации прогрессивной загрузки FLV-файлов (локально либо по сети) просто необходимо сообщить классу о том, что потоковой передачи не будет, передав в метод `connect()` экземпляра класса вместо URL-адреса приложения потоковой передачи значение `null`, как видно в строке 23.

Далее создается экземпляр класса `NetStream` со ссылкой на только что созданный экземпляр `NetConnection`, как показано в строке 24. Это поток, через который будет контролироваться видео даже в случае прогрессивной загрузки файлов.

```
25        _infoClient = new Object();
26        _infoClient.onMetaData = onMetaData;
27        _infoClient.onCuePoint = onCuePoint;
28        _vidStream.client = _infoClient;
29        _vidConnection.addEventListener(NetStatusEvent.NETSTATUS,
          onNetStatus, false, 0, true);
30        _vidConnection.addEventListener(AsyncErrorEvent.ASYNCERROR,
          onAsyncError, false, 0, true);
31        _vidStream.addEventListener(NetStatusEvent.NET_STATUS, onNetStatus,
          false, 0, true);
32        _vidStream.addEventListener(AsyncErrorEvent.ASYNC_ERROR,
          onAsyncError, false, 0, true);
```

Строки с 25 по 32 отвечают за обработку событий, состояний и ошибок. Это может быть реализовано множеством способов; в данном при-

мере используется самый базовый вариант. Программная обратная связь осуществляется двумя способами: с помощью предопределенных обработчиков, таких как `onMetaData` и `onCuePoint`, и путем создания слушателей событий и перехвата связанных с событиями данных.

В строках с 25 по 28 создается пользовательский объект, который будет использоваться для обработки данных, получаемых от экземпляра `NetStream`. Задавая этот объект как значение свойства `client` потока, мы обеспечиваем передачу всех метаданных или данных контрольных точек в этот объект. При наличии методов `onMetaData()` и `onCuePoint()` (задаются в строках 26 и 27 и описываются несколько ниже) эту информацию можно использовать по мере ее поступления.

В строках с 29 по 32 аналогичные действия выполняется с использованием слушателей событий. С их помощью этот класс перехватывает события, связанные с отчетами о состоянии и асинхронными ошибками как при подключении к видеопотоку, так и при его обработке. Эту технику можно применять для обработки поступающих данных или просто для предотвращения отображения ошибок.

```
33        _vid = new Video();
34        _vid.attachNetStream(_vidStream);
35        _vidURL = "nero_320x240_cp.flv";
36        _vidStream.play(_vidURL);
37        addChild(_vid);
38
39        createControlButtons();
40      }
```

Строки с 33 по 37 отвечают собственно за отображение видео. Первый шаг – динамическое создание объекта отображения `Video`. Затем ранее созданный экземпляр `NetStream` прикрепляется к этому объекту отображения для организации управления, далее указывается видеоресурс, выполняется его воспроизведение, и объект видео добавляется в список отображения, что обеспечит его появление на сцене.

В последней строке конструктора (строка 39) вызывается функция, создающая три кнопки для воспроизведения, приостановки и остановки воспроизведения видео. Эта функция описывается в самом конце сценария, в строках с 82 по 98, на которые мы далее обратим отдельное внимание.

```
41      private function onMetaData(info:Object):void {
42        trace(info.duration);
43      }
44
45      private function onCuePoint(info:Object):void {
46        trace(info.parameters.text);
47      }
48
49      private function onAsyncError(evt:AsyncErrorEvent):void {
```

```
50        trace(evt.text);
51      }
52
53      private function onNetStatus (evt:NetStatusEvent):void {
54        trace(evt.info.level + ": " + evt.info.code);
55        if (evt.info.code == "NetStream.Play.Start") {
56          _vidPlaying = true;
57        } else if (evt.info.code == "NetStream.Play.Stop") {
58          _vidPlaying = false;
59        }
60      }
```

Далее представлены функции, к которым обращаются обработчики обратных вызовов и слушатели событий `__vidConnection`. В качестве примера обработки метаданных функция `onMetaData()` (строка 42) выводит продолжительность FLV-файла. Аналогичным образом функция `onCuePoint()` (строка 46) отображает текст каждой контрольной точки. Все возникающие асинхронные ошибки отслеживаются функцией `onAsyncError()` (строка 50), а вывод сообщений о состоянии обеспечивается функцией `onNetStatus()` (строка 54). Мы воспользовались также функцией `onNetStatus()` для создания типового уведомления о воспроизведении FLV-файла. Код события `NetStatus.Play.Play` устанавливается при воспроизведении, а код события `NetStatus.Play.Stop` – при остановке воспроизведения видео.

```
61      private function onPlayVid(evt:MouseEvent):void {
62        if (_vidPlaying) {
63          _vidStream.resume();
64        } else {
65          _vidStream.play(_vidURL);
66        }
67
68        _vidPlaying = true;
69      }
70
71      private function onPauseVid(evt:MouseEvent):void {
72        _vidStream.togglePause();
73      }
74
75      private function onStopVid(evt:MouseEvent):void {
76        _vidPlaying = false;
77        _vidStream.close();
78        _vid.clear();
79      }
```

Следующий фрагмент описывает функции кнопок. При вызове метода потока `play()` воспроизведение запускается с начала файла. Следовательно, функции `onPlayVid` (строка 42) необходимо знать, было ли воспроизведение приостановлено или остановлено, чтобы не воспроизводить видео сначала при каждом нажатии кнопки воспроизведения. Для этого используется логическая переменная `_vidPlaying` и простое

условное выражение в функции onNetStatus(). Функция onNetStatus() в строке 71 использует метод togglePause() для поочередной приостановки и возобновления воспроизведения видеопотока.

С помощью метода close() функция onStopVid() (строка 75) полностью останавливает воспроизведение видео (в отличие от просто приостановки). Эта функция вызывает также метод clear() объекта видео, чтобы кадр, отображаемый в момент остановки воспроизведения, не оставался на экране, поскольку в зависимости от кадра это может привести к некорректному отображению приостановленного видео.

```
80      private function createControlButtons():void {
81        _playBtn = new CreateRoundRectButton(80,20,10,2,0x0066CC, "Play");
82        _playBtn.x = 20;
83        _playBtn.y = 260;
84        _playBtn.addEvenвtListener(MouseEvent.MOUSE_UP, onPlayVid,
          false, 0, true);
85        addChild(_playBtn);
86        _pauseBtn = new CreateRoundRectButton(80,20,10,2,0x0066CC,
          "Pause");
87        _pauseBtn.x = 120;
88        _pauseBtn.y = 260;
89        _pauseBtn.addEventListener(MouseEvent.MOUSE_UP, onPauseVid,
          false, 0, true);
90        addChild(_pauseBtn);
91        _stopBtn = new CreateRoundRectButton(80,20,10,2,0x0066CC, "Stop");
92        _stopBtn.x = 220;
93        _stopBtn.y = 260;
94        _stopBtn.addEventListener(MouseEvent.MOUSE_UP, onStopVid, false,
         0, true);
95        addChild(_stopBtn);
96      }
97    }
98  }
```

Наконец, мы видим функцию createControlButtons(), которая вызывается в последней строке конструктора класса. Она создает экземпляр внешнего класса, используемого для создания всех кнопок и их размещения под видеорядом. Выражение импорта пользовательского класса находится в строке 7. Класс CreateRoundRectButton можно найти в исходном коде; он является простой реализацией идей, рассмотренных в главах 8 и 10, поэтому здесь мы не будем возвращаться к его обсуждению. Важно помнить лишь то, что этот класс добавляет для каждой кнопки по слушателю события отпускания кнопки мыши, который вызывает функцию с соответствующим именем для управления воспроизведением видео.

Этот класс не является функционально полной реализацией, однако он демонстрирует, как можно реализовать воспроизведение видео с помощью ActionScript. В него можно было бы добавить функции перемотки, управления звуком и даже отображения субтитров, созданных

с использованием контрольных точек, если таковые имеются. Однако к этому стоит перейти после того, как вы полностью разберетесь в основах. На сопроводительном веб-сайте более подробно обсуждаются некоторые из этих возможностей, а также предлагается пример более сложного клиента NetStream и альтернативный подход с использованием класса `VideoPlayer`. Проработав примеры, предложенные в этой главе, вы можете рассмотреть дополнительные упражнения и темы, приведенные на веб-сайте.

Пакет проекта

Пакет проекта этой главы основан на упражнении из последнего раздела – «Написание собственного кода для воспроизведения видео». С помощью этого пакета можно передать путь к внешнему видеофайлу в класс инициализации, после чего этот класс автоматически подготовит необходимые элементы для воспроизведения видеофайла и сообщения об ошибках. Это позволит вам управлять воспроизведением с помощью своих любимых элементов управления. Более подробную информацию о сквозном проекте вы найдете в главе 6.

Что дальше?

В этой главе представлены на выбор два способа воспроизведения видео. Мы рассмотрели простой подход с использованием готовых компонентов, включая компонент FLVPlayback для отображения видео и компонент FLVPlaybackCaptioning, обеспечивающий работу с субтитрами на разных языках и доступность видео для людей с физическими ограничениями. Мы также продемонстрировали создание элементарного проигрывателя на ActionScript. Не забывайте про сопроводительный веб-сайт, на котором есть несколько дополнительных упражнений, реализующих представленные здесь идеи на более высоком уровне.

В следующей главе мы перейдем к части V нашей книги, посвященной вводу и выводу данных. В главе 13 рассматриваются основы загрузки внешних ресурсов, включая:

- Использование универсального класса `URLRequest`
- Загрузку визуальных ресурсов, включая изображения и другие SWF-файлы
- Загрузку внешних файлов в формате MP3
- Загрузку текста и переменных