Санкт-Петербургский государственный институт точной механики и оптики (технический университет)

Кафедра “Компьютерные технологии”

К. А. Бондаренко, А. А. Шалыто

Разработка XML - формата для описания внешнего вида видеопроигрывателя с использованием конечных автоматов

Гипертекстовое программирование с явным выделением состояний

Проектная документация

Проект создан в рамках
“Движения за открытую проектную документацию”
http://is.ifmo.ru

Санкт-Петербург
2003

Содержание

Введение

Для алгоритмизации и программирования задач логического управления была предложена SWITCH-технология, которая в дальнейшем была развита для событийных и объектно-ориентированных программ. Подробно ознакомиться с этой технологией и с конкретными примерами ее использования можно на сайтах http://is.ifmo.ru и http://www.softcraft.ru.

Эта технология удобна не только для решения задач управления техническими объектами, но, и как будет показано в настоящей работе, ее целесообразно применять при описании внешнего вида динамических приложений и интернет-страниц с помощью гипертекстового программирования.

Стандартный подход при разработке формата для описания внешнего вида состоит в введении в гипертекстовый формат для обеспечении его гибкости дополнительных вставок, называемых скриптами. Скрипты представляют собой выполняемые программы, и поэтому разработчик должен быть весьма квалифицированным программистом.

В данной работе предлагается **отказаться от стандартного подхода** за счет применения при проектировании программ *конечных автоматов*. При этом скипты не используются, а формат остается гибким.

Обратим внимание, что описание динамического образа можно представить в некотором абстрактном виде за счет выделения состояний, соответствующих основным стадиям развития динамики. Поясним сказанное.

Человек представляет себе образ того или иного объекта в неком абстрактном виде. Если объект изменяется, то можно запомнить несколько его состояний и причины, приводящие к его переходам из состояния в состояние, а не алгоритм, по которому можно построить вид объекта в конкретный момент времени.

Далее этот подход излагается на примере описания внешнего вида Windows-приложения – видеопроигрывателя [4]. Для краткости внешний вид приложения называется **скином** (от английского “skin” – оболочка). Это название и будет применяться в дальнейшем. Отметим, что скин является аналогом программы в традиционном программировании, а предлагаемый формат – аналогом языка программирования. Формат за счет применения автоматов обеспечивает не только гибкость, но и простоту разработки скинов.

Схема взаимодействия разработчика, пользователя, плейера, скинов, формата и автоматов приведена на **рис. 1**.

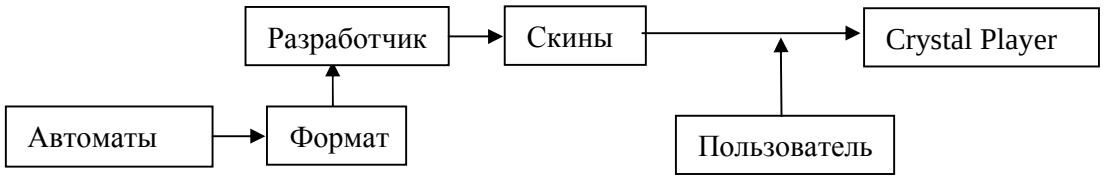

Рис. 1. Схема взаимодействия, используемая в предлагаемом подходе.

В соответствии с этой схемой разработчик на базе созданного на основе автоматов формата проектирует скин. Применение конечных автоматов значительно упрощает ему задачу – благодаря автоматам разработчик может создать динамический скин, не прибегая к скриптам.

Затем пользователь загружает скин в Crystal Player, кардинально меняя внешний вид проигрывателя.

Если в версии 0.31 проигрыватель выглядел только как стандартное Windows – окно (**рис. 2**), то начиная с версии 0.32, пользователь может достаточно просто изменить внешний вид проигрывателя. На **рис. 3** приведены примеры внешних видов проигрывателя, разработаные студентами СПбГИТМО (ТУ) Скаковым П. и Саитовым А.

Рис. 2. Традиционный внешний вид проигрывателя Crystal Player

Рис. 3. Внешние виды видеопроигрывателя, разработанные на основе нового формата

1. Постановка задачи

Требуется написать скин для видеопроигрывателя – приложения, позволяющего просматривать на компьютере анимационные файлы (фильмы, презентации, телепередачи, клипы, концерты, словом все, что можно записать на обычную видео кассету для VHS-видеомагнитофонов).

Разработанный гипертекстовый формат позволит любому человеку с помощью текстового и графического редакторов легко написать **динамический** скин, не имея никаких навыков в программировании.

Внешний вид приложения составляется из графических элементов – *контролов*. В скине будет указано, где находится конкретный *контрол* и какое изображение (файл в формате bmp) с ним связано. *Конролы* могут быть кнопками, изображениями и т.д. Если *контрол* – кнопка, то с ним может быть связано несколько изображений (нажатое и отпущенное состояние) и какое-нибудь действие, например, запустить проигрывание файла. При этом форма всего окна задается наложением простых геометрических фигур.

Таким образом, достаточно просто описывается статический скин. Динамика обычно задается скриптами, но в нашем случае это делается иначе и проще.

Автоматный подход позволяет разбить скин на логически отдельные составные части, которые могут принимать несколько состояний. Например, панель управления с кнопками “**Пуск”, “Пауза”** и **“Стоп”** может “прятаться”, когда в ней нет необходимости, и может расширяться, показывая дополнительные кнопки “**Вперед”** и **“Назад”**. Таким образом, панель управления может принимать три состояния **“Спрятана”, “Выдвинута”, “Расширена”**. Разработчик описывает эти состояния и задает правила перехода между ними. Каждая логически отдельная часть скина (например, вышеупомянутая панель) становится для разработчика конечным автоматом, а каждое ее состояние – состоянием этого автомата.

Для реализации сложного поведения скина автоматы могут взаимодействовать между собой. Например, на расширенной выдвинутой панели, описываемой автоматом **“Панель управления”** в состоянии **“Расширена”**, может находиться кнопка помощи (со знаком “?”). При нажатии на эту кнопку “выдвигается“ другая панель – с текстом помощи. Эта панель, в свою очередь, также описывается автоматом **“Панель помощи”,** который имеет два состояния - **“Спрятана”** и **“Выдвинута”**. Таким образом, кнопка помощи, описываемая в состоянии **“Расширена”** автомата **“Панель управления”,** задает переход автомата **“Панель помощи”** в состояние **“Выдвинута”**.

Это очень гибкая возможность, которая реализуется использованием гипертекста вместо программирования. Достаточно разработать наглядный *граф переходов* (**рис. 4**) и записать в выбранном текстовом формате.

Цель работы состоит в разработке такого формата и внедрение его в реальное приложение. Код, поддерживающий этот формат, оформлен в виде интегрированного модуля на языке C++. Исходный текст этого модуля занимает 180 килобайт кода. Отметим, что этот модуль занимает более 20% кода видео проигрывателя в целом.

2. Пример графа переходов

Обычно при использовании конечных автоматов разрабатывают *графы переходов*. В данном случае граф переходов позволяет наглядно иллюстрировать взаимодействие отдельных частей скина. На **рис. 4** изображено два графа переходов, отвечающих за поведение панелей управления и помощи, каждая из которых представлена автоматом.

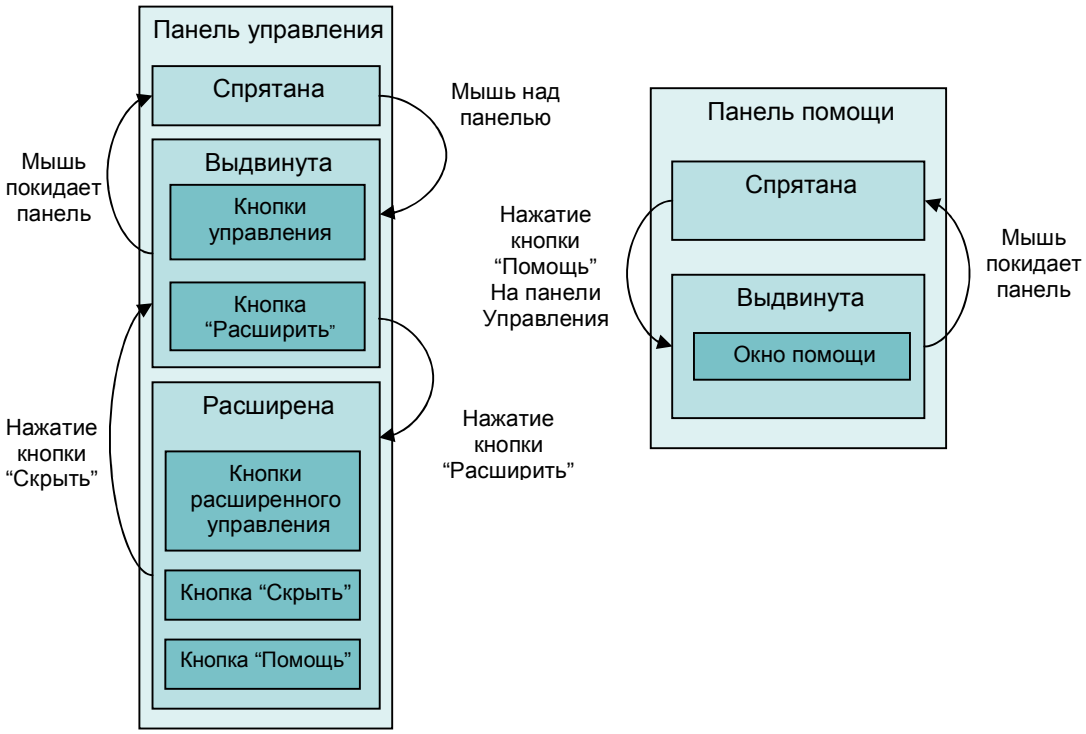

Рис. 4. Графы переходов, описывающие поведение двух панелей скина

Светлым цветом отмечены логические единицы – автоматы. Каждый автомат может находиться в одном из состояний. Они отмечены более темным цветом. В каждом состоянии описываются элементы управления (самый темный цвет). Это могут быть кнопки и текстуры. Также в поле состояния могут описываться некоторые свойства (настройки) приложения, например, цвет списка проигрываемых файлов. С элементами управления связаны определенные переходы. Например, нажатие кнопки “Скрыть” автомата “Панель управления” в состоянии “Расширена” переводит автомат “Панель управления” в состояние “Выдвинута”. В результате панель управления уменьшается, и с нее исчезают дополнительные кнопки.

Далее мы подробно разберем синтаксис языка описания автоматной структуры скина.

3. Описание синтаксиса

Для описания автоматной структуры скина выбран широко известный **XML** – формат. Этот формат очень похож на гипертекстовый формат **HTML**, и поэтому он понятен многим разработчикам. Кроме того, синтаксический анализатор XML включен в операционную систему **Windows** как **COM**-объект, и поэтому нет необходимости вручную разбирать грамматику формата.

Формат XML представляет собой древовидную структуру. Каждый узел в XML описывается открывающим и закрывающим **тегом**:

<TAG [параметры]> тело тега </TAG>

В “теле тега” могут быть другие узлы – дочерние. Если узел является листом дерева (без дочерних узлов), то его можно описать следующим образом:

<TAG [параметры]/>

“[Параметры]” – это необязательное поле, в котором перечисляются некие параметры тега:

param1=”value1” [param2=”value2”]

Здесь param1 – параметр с именем “param1”, его значение – “value1”.
Приведем пример парного тега с двумя параметрами:

<TAG param1=”value1” param2=”value2”> …… </TAG>

Весь скин описывается между тегами **<SKIN RecuiredVersion=”Version”>** и **</SKIN>**. Обязательный параметр “RequiredVersion” – минимальный номер версии программы, необходимой для распознавания данного скина. Таким образом, может использоваться следующая запись:

<SKIN RequiredVersion=”101”> … </SKIN>

Она означает, что минимальный номер версии, которая поддерживает данный формат – 1.01. Это сделано для совместимости более поздних, расширенных форматов скинов. Данный формат уже трижды расширялся – добавлялись новые возможности, и скины, написанные для старых версий программы, корректно работают. Попытки загрузить новые скины в старую программу безуспешны – она откажется его интерпретировать.

В теле скина можно разместить парный тег:

<TITLE author="SkinAuthors"> комментарий </TITLE>

В заголовке обычно помещается название скина. *SkinAuthors* - авторы, разработчики и дизайнеры скина. “Комментарий” – некоторая информация о скине.

Автоматы в скине также описываются парным тегом:

<SWITCH name="SwitchName"> тело автомата </SWITCH>

SwitchName – имя автомата, его уникальный идентификатор в рамках скина. В скине должен присутствовать как минимум один автомат (хотя бы комбинационный, имеющий одно состояние). Автоматы могут взаимодействовать или существовать отдельно. Внешний вид программы собирается по всем автоматам ее скина по очереди. В “теле автомата” описываются состояния, которые он может принимать. Первое описанное состояние является исходным. В нем автомат начинает свое существование.

Состояние описывается парным тегом вида:

<STATE name=”StateName”> тело состояния </STATE>

StateName – имя состояния, его уникальный идентификатор в рамках автомата. В каждом автомате должно быть описано как минимум одно состояние. Автомат всегда находится в одном из своих состояний. Это состояние целиком определяет поведение и внешний вид автомата. Описание состояния размещается в “теле состояния”.

Стандартный элемент, описывающий внешний вид скина – *контрол*. *Контрол* представляет собой прямоугольник, который определенным образом выглядит и с которым связаны определенные действия. Он задается следующим образом:

```
<CONTROL topleft="TLCoord"
         bottomright="BRCoord"
         chit="ChitMode"
         gradient="ControlGradient"
         normal="NormalBitmap"
         mouse="MouseBitmap"
         leftdown="DownBitmap"
         leftaction="ControlAction"
         rightaction=”RightAction”
         mouseaction=”MouseAction”
         seal=”Seal”/>
```

Опишем параметры тега:

TLCoord – координата левого верхнего угла *контрола*. Координаты задаются относительно одного из четырех углов базового окна – окна вывода изображения, той прямоугольной части экрана, в которую выводится видео изображение. Координаты имеют вид: *TL(x, y), TR(x, y), BL(x, y), BR(x,y).* Эти примеры соответствуют координатам относительно левого верхнего, правого верхнего, левого нижнего, правого нижнего угла видео окна. Смещение (x, y) может состоять из отрицательных чисел, а отчет начинается с левого верхнего угла. Например, topleft=”BR(-10, 15)” означает, что левый верхний угол *контрола* находится на 10 пикселей левее и на 15 ниже правого нижнего угла вывода изображения.

BRCoord задает координаты правого нижнего угла *контрола*.

ChitMode задает способ управления котролом. Это не обязательный параметр. Его возможные значения:

title – за *контрол* можно “таскать” весь скин. Фактически, данный *контрол* ведет себя, как заголовок стандартного окна;

client – стандартное управление. Если параметр *ChitMode* не указан вообще, параметр автоматически принимает данное значение;

top – за *контрол* можно растягивать скин (точнее окно видеовывода) также, как за верхнюю часть стандартного окна;

bottom – за *контрол* можно растягивать скин также, как за нижнюю часть стандартного окна;

left – за *контрол* можно растягивать скин также, как за левую часть стандартного окна;

right – за *контрол* можно растягивать скин также, как за правую часть стандартного окна;

topleft – за *контрол* можно растягивать скин также, как за левую верхнюю часть стандартного окна;

topright – за *контрол* можно растягивать скин также, как за правую верхнюю часть стандартного окна;

bottomleft – за *контрол* можно растягивать скин также, как за левую нижнюю часть стандартного окна;

bottomright – за *контрол* можно растягивать скин также, как за правую нижнюю часть стандартного окна.

ControlGradient – четыре цвета, которыми первоначально заливается тело *контрола*. Цвета задаются следующим образом:

“#b0g0r0#b1g1r1#b2g2r2#b3g3r3”

b-синяя компонента, g-зеленая, r-красная в шестнадцатеричном виде. 0 – левый верхний угол, 1 – правый верхний, 2 – левый нижний, 3 – правый нижний. Например,

gradient=“#001122#a0b0c0#ffffff#8090a0”

Если все четыре цвета не совпадают, *контрол* будет “залит” градиентом – цвет буде плавно изменяться, принимая в углах заданные значения.

NormalBitmap – имя растрового файла (в формате *.bmp), которым следует заполнить тело *контрола*. Если параметр пропущен, *контрол* не заполняется текстурой. Если при этом не задан и *ControlGradient*, то *контрол* окажется невидимым. Это иногда очень удобно. Расположим, например, *контрол* поверх других и свяжем с ним какие-нибудь действия. В итоге действия выполняться будут, а *контрол* не будет “маячить” на переднем плане.

Если после имени файла поставить знак вопроса и указать цвет (например “control.bmp?#102030”), то растровое изображение будет выводиться “с ключом” - все точки файла control.bmp, имеющие цвет #102030, станут прозрачными. Если же после знака вопроса указать “z” (“control.bmp?z”), то изображение будет растянуто до размеров *контрола*. В противном случае оно будет многократно выводиться на его поверхности, как текстура.

MouseBitmap – задает растровое изображение *контрола*, когда над ним находится указатель мыши.

DownBitmap – задает растровое изображение *контрола*, над которым была нажата левая кнопка мыши.

LeftAction – действие, связанное с нажатием левой кнопки мыши. Действие может явно переводить произвольный автомат в произвольное состояние либо порождать сообщение программе-проигрывателю.

Список допустимых сообщений:

Menu_File - вызов меню “Файл”;
Menu_Subtitles - вызов меню “Субтитры”;
Menu_Playback - вызов меню “Проигрывание”;
Menu_Playlist - вызов меню “Плейлист”;
Menu_View - вызов меню “Просмотр”;
Menu_Options - вызов меню “Настройки”;
Menu_Help - вызов меню “Помощь”;
Maximize или Fullscreen – перейти в полноэкранный режим;
Minimize - минимизация окна приложения (свернуть окно);
Close - закрыть окно;
Menu_Playback_Play - начать (продолжить) проигрывание;
Menu_Playback_Pause - приостановить проигрывание;
Menu_Playback_Stop - завершить проигрывание;
Menu_File_Open – открыть видео файл;
Menu_Subtitles_Open – открыть файл субтитров.

Если действие должно переводить “Автомат” в определенное состояние, то в качестве параметра следует указать имя-идентификатор автомата и через точку имя-идентификатор состояния, в которое следует перейти автомату. Если действий несколько, то их следует перечислять через точку с запятой (;). Например, если требуется завершить проигрывание и перевести автомат “Main” в состояние “Initial” по нажатию левой кнопки мыши, то это записывается так: leftaction=“Stop; Main.Initial”

RightAction – действие, связанное с нажатием на *контроле* правой кнопки мыши.

MouseAction - действие, связанное с появлением над *контролом* указателя мыши.

Seal – если установить seal=”on”, то при нажатии на *контрол*, он будет “западать”, аналогично пунктам стандартного меню. В противном случае он будет вести себя, как обычная кнопка.

С помощью следующего тега описываются свойства плейлиста – списка проигрываемых файлов.

```
<PLAYLIST topleft="TLCoord"
          bottomright="BRCoord"
          back="BackGroundGradient"
          active="ActiveGradient"
          textcolor="TextColor"
          activecolor="ActiveColor"
          font="FontName"
          fontheight=”FontHeight”
          fontbold=”FontBold”
          height="Height"
          ground="GroundBitmap"
          grayground="GrayGroundBitmap"/>
```

Опишем параметры этого тега.

TLCoord и *BRCoord* задают положение плейлиста относительно окна видео вывода.

BackGroundGradient – градиент, заполняющий фон плейлиста.

ActiveGradient – градиент, использующийся для подсветки активного файла в плейлисте.

TextColor – цвет шрифта, которым печатаются элементы плейлиста.

ActiveColor – цвет шрифта, которым печатается активный, подсвеченный файл в плейлисте.

FontName – имя шрифта, которым печатаются элементы плейлиста. Допускаются любые зарегистрированные в системе шрифты, например “Arial” или “Times New Roman”

FontHeight – высота шрифта в пикселях.

FontBold – толщина шрифта в условных единицах (400 – обычный, 600 – толстый).

Height – ширина одной строки в плейлисте в пикселях. Отметим, что чем толще строка, тем крупнее шрифт, если *FontHeight* не указан.

GroundBitmap - растровое изображение заднего фона плейлиста – файл *.bmp. Картинка, задаваемая этим файлом, будет украшать плейлист, находясь в его центре на заднем фоне, в то время, как на переднем фоне будет расположен список файлов.

GrayGroundBitmap – приглушенное изображение заднего фона плейлиста. Эта картинка заменит предыдущую, если плейлист не пуст. Если этот параметр опущен, то картинка автоматически генерируется из *GroundBitmap* – картинки.

Следующий тег явно указывает те сообщения, при появлении которых автомату следует перейти в состояние, в котором тег описан:

<EVENT message="Msg" source="StateName"/>

При появлении сообщения *Msg*, если перед этим автомат имел состояние *StateName* или любое другое (когда параметр *source* опущен), активизируется состояние, в котором тег описывается. В этом теге используется параметр

Msg - сообщение, при котором происходит переход.

Этот параметр может принимать три значения:

Play – началось проигрывание файла.

Stop – проигрывание файла закончилось.

Pause – проигрывание приостановлено.

Следующий тег позволяет указать, чем будет управлять клавиатура.

<KEYBOARD playcontrol="PlayControl"/>

Параметр *PlayControl* может принимать следующие значения:

On – клавиатура будет использована для управления проигрыванием.

Off – клавиатура будет использована для управления плейлистом.

Этот тег позволяет корректно переключаться от управления проигрыванием к управлению плейлистом и обратно. При этом в соответствующем состоянии следует указать <KEYBOARD playcontrol=”On”/>, а в другом состоянии - <KEYBOARD playcontrol=”Off”/>. На **рис.5** изображен скин, реализующий данную возможность.

Рис. 5. Скин с выдвигающимся плейлистом, реализованный на автоматах

Следующий тег позволяет задать положение физического окна Windows относительно окна видео-вывода. Логично требовать, чтобы все *контролы* и само окно видео-вывода лежали в пределах этого физического окна.

<WINDOW topleft="TLCoord" bottomright="BRCoord"
borderwidth="Width" bordercolor="Color"/>

Здесь:

TLCoord и *BRCoord* собственно координаты физического окна;
Width – толщина рамки вокруг физического окна в пикселях;
Color – цвет рамки.

Последние два параметра могут быть опущены.

Для описания свойств окна видео вывода используется следующий тег:

<VIDEO color="Color"
ground="Ground"
minwidth="MinWidth"
minheight="MinHeight"
minwidth="MaxWidth"
minheight="MaxHeight"/>

Здесь:

Color - цвет заливки пустой области вокруг выводимого изображения;
Ground – имя растрового файла, который будет изображен вместо видео пока проигрывание остановлено;
MinWidth, MinHeight, MaxWidth, MaxHeight – ограничения на изменение размера видео окна;

Форма физического окна скина может быть не прямоугольной и, более того, может меняться при переходе автоматов в различные состояния.

Форма окна – пересечение заданных геометрических фигур. Геометрическая фигура задается тегом <REGION/>.

Прямоугольник с диагональю *TLCoord* – BRCoord описывается тегом:

<REGION type=”Rect” topleft=”TLCoord” bottomright=”BRCoord”/>

Зададим формат для описания прямоугольника со скругленными краями. Дополнительные параметры *width* и *height* задают горизонтальный и вертикальный диаметры “скругляющего” эллипса следующим тегом:

<REGION type=”RoundRect” topleft=”TLCoord” bottomright=”BRCoord” width=”Width” height=”Height”/>

Следующий тег задает эллипс. Параметры *TLCoord* и *BRCoord* задают прямоугольник, в который следует вписать эллипс.

<REGION type=”Ellipse” topleft=”TLCoord” bottomright=”BRCoord”/>

Многоугольник задается следующим тегом. В этом теге параметр *Countour* задает вершины многоугольника, перечисленные через запятую. Например, “*TL(0, 0), TL(100, 50), TL(50, 100)*".

<REGION type=”Poly” countour="Countour”/>

Для описания внешнего вида контекстных меню также применена разработанная автоматная модель. По сути контекстное меню является отдельным скином, вложенным в основной. При этом в теле скина необходимо указать следующий тег:

<SKIN object="Menu"> тело вложенного скина </SKIN>

Здесь параметр *object*=”Menu” указывает, что данный вложенный скин будет использован для описания контекстных меню. Координаты в этом случае базируются не относительно видео окна, а относительно динамически определенной прямоугольной области, в которую помещается список пунктов меню при заданных настройках.

Вместо тега **<PLAYLIST/>** в описании меню допустим тег **<MENU/>**, описывающий внешний вид текста в меню.

<MENU
topleft="TLCoord"
bottomright="BRCoord"
back="BackGradient"
active="ActiveGradient"
textcolor="TextColor"
activecolor="ActiveColor"
graycolor="GrayColor"
keycolor="KeyColor"
keyactivecolor="KeyActiveColor"
font="FontName"
fontbold=”FontBold”
fontheight=”FontHeight”
height="Height"
separatorheight="SeparatorHeigth"
popupwidth="PopupWidth"
iconwidth="IconWidth"
iconx=”IconX”
borderwidth=”BorderWidth”

bordercolor=”BorderColor”
popup="Popup"
activepopup=”ActivePopup”
check="Check"
activecheck=”ActiveCheck”
separator="SeparatorGradiend">

Как и в теге **<PLAYLIST/>** параметры *TLCoord, BRCoord, BackGradient, ActiveGradient, TextColor, ActiveColor, FontName, FontHeight, FontBold, Height* отвечают соответственно за положение строк меню, за градиент фона, за градиент подсветки, за цвет текста, за цвет активной строки, за используемый шрифт и за высоту строки меню в пикселах. Кроме перечисленных выше, в теге имеются дополнительные параметры:

GrayColor – цвет недоступного пункта меню;

KeyColor – цвет “горячих клавиш”, по нажатию на которые можно быстро вызывать те или иные пункты меню;

KeyActiveColor - цвет “горячих клавиш” у активного пункта меню;

BorderWidth – ширина рамки вокруг активного пункта;

BorderColor – цвет рамки (градиент) вокруг активного пункта;

SeparatorHeight – высота разделителя в пикселях;

PopupWidth – ширина полоски справа для иконок вложенных меню;

IconWidth – ширина полоски слева для иконок-галочек меню;

IconX – горизонтальное смешение картинок иконок;

Popup – имя растрового файла – иконки, символизирующей всплывающее вложенное меню.

ActivePopup – имя растрового файла – иконки, символизирующей всплывающее вложенное меню для активной строки;

Check – имя растрового файла – иконки, символизирующей галочку слева от отмеченного пункта меню.

ActiveCheck – имя растрового файла – иконки, символизирующей галочку слева от отмеченного пункта меню для активной строки;

SeparatorGradient – четыре цвета, задающие две горизонтальные линии – разделитель.

На **рис. 6** изображены примеры меню, описанных с помощью данного формата.

Рис. 6. Внешний вид меню, описанных с помощью разработанного формата

Таким образом, простой гипертекстовый формат позволяет описать скин, обладающий широкими возможностями. Приведем пример листинга скина, внешний вид которого изображен на **рис. 5.**

4. Пример использования разработанного формата

Рассмотрим пример, иллюстрирующий простоту описания и одновременно широкие возможности разработанного формата.

Пусть требуется создать скин, внешний вид которого приведен на **рис. 5.** Помимо статической части (левая часть рисунка) скин имеет динамическую составляющую – панель с плейлистом. Эта панель может находиться в трех состояниях: “Спрятана”, “Выдвинута, но не активна”, “Выдвинута и активна”. Граф переходов для динамической части разрабатываемого скина представлен на **рис. 7**

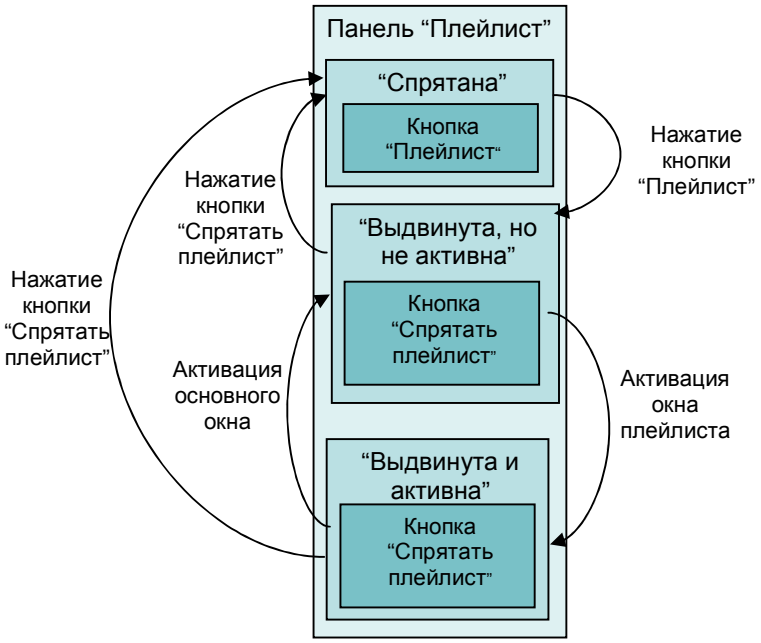

Рис. 7. Граф переходов для динамической части разрабатываемого скина

Описание скина

<SKIN RequiredVersion="33" object=”Main”> Необходимая версия проигрывателя, поддерживающего данный скин – 0.33

Информация об авторах скинов – стандартное поле

<TITLE author="A.Saitov, K.Bondarenko">
TV Skin
</TITLE>

Описание статической части

<SWITCH name="Main"> Этот автомат может находиться лишь в одном состоянии. Он описывает те элементы скина, которые остаются неизменными.

Единственное состояние статической части скина

<STATE name="Normal">

Описание плейлиста в этом состоянии

Три статичных прямоугольника, из которых состоит статичная часть скина

```
<REGION type="Rect" topleft="TL(-13, -15)" bottomright="BR(13, 18)"/>
<REGION type="Rect" topleft="BL(10, 17)" bottomright="BR(-10, 48)"/>
<REGION type="Rect" topleft="BL(-12, 47)" bottomright="BR(13, 63)"/>

Ограничения на изменение размера видео окна
<VIDEO color="#858585" minwidth="374" minheight="257"/>
<CONTROL topleft="TL(-13, -15)" bottomright="TL(0, 0)" chit="topleft" normal="tl.bmp"/>
<CONTROL topleft="TR(0, -15)"  bottomright="TR(14, 0)" chit="topright" normal="tr.bmp"/>
<CONTROL topleft="BL(-13, 0)" bottomright="BL(0, 17)" chit="bottomleft" normal="bl.bmp"/>
<CONTROL topleft="BR(0, -1)"  bottomright="BR(13, 18)" chit="bottomright"
      normal="br.bmp"/>

Рамка вокруг видео окна. За нее можно "растягивать" окно
<CONTROL topleft="TL(-13, 0)"  bottomright="BL(0, -51)" chit="left" normal="l.bmp"/>
<CONTROL topleft="TR(0, 0)"  bottomright="BR(13, -51)" chit="right" normal="r.bmp"/>
<CONTROL topleft="TL(0, -15)"  bottomright="TR(0, -13)" chit="top"/>

Так можно описать содержимое окна – градиент и картинка
<CONTROL topleft="TL(0, 0)"  bottomright="BR(0, 0)" chit="client"
      gradient="#858585#858585#858585#858585"/>
<CONTROL topleft="BL(0, 0)"  bottomright="BR(0, 17)" chit="bottom" normal="b.bmp"/>

Далее описаны статичные элементы управления и просто текстуры
 <CONTROL topleft="BL(-13, -51)"  bottomright="BL(0, 0)" normal="larr.bmp"
          mouse="larr_mouse.bmp" leftdown="larr_down.bmp" leftaction="Menu_Subtitles"/>
 <CONTROL topleft="BR(0, -51)"  bottomright="BR(13, 0)" normal="rarr.bmp"
          mouse="rarr_mouse.bmp" leftdown="rarr_down.bmp" leftaction="Menu_Subtitles"/>
 <CONTROL topleft="BL(10, 17)"  bottomright="BR(-10, 48)" chit="title"
          normal="panel.bmp"/>
 <CONTROL topleft="BL(10, 17)"  bottomright="BL(45, 48)" normal="switch.bmp"
          mouse="switch_mouse.bmp" leftdown="switch_down.bmp" leftaction="Close"/>
 <CONTROL topleft="BL(46, 17)"  bottomright="BL(76, 48)" normal="min.bmp"
          mouse="min_mouse.bmp" leftdown="min_down.bmp" leftaction="Minimize"/>
 <CONTROL topleft="BL(80, 17)"  bottomright="BL(175, 48)" normal="file.bmp"
          mouse="file_mouse.bmp" leftdown="file_down.bmp" leftaction="Menu_File"/>
 <CONTROL topleft="BL(176, 17)"  bottomright="BL(227, 48)" normal="play.bmp"
          mouse="play_mouse.bmp" leftdown="play_down.bmp"
          leftaction="Menu_Playback"/>
 <CONTROL topleft="BR(-146, 17)"  bottomright="BR(-96, 48)" normal="view.bmp"
          mouse="view_mouse.bmp" leftdown="view_down.bmp" leftaction="Menu_View"/>
 <CONTROL topleft="BR(-96, 17)"  bottomright="BR(-56, 48)" normal="options.bmp"
          mouse="options_mouse.bmp" leftdown="options_down.bmp"
          leftaction="Menu_Options"/>
 <CONTROL topleft="BL(-13, 48)"  bottomright="BR(13, 63)" chit="title"
          normal="stand.bmp"/>
 <CONTROL topleft="TR(-100, 3)" bottomright="TR(-3, 75)"  normal="about_dark.bmp"
          mouse="about_light.bmp" leftdown="about_light.bmp" leftaction="Menu_Help"/>
  </STATE>
</SWITCH>
```

Динамическая часть

Автомат, инкапсулирующий панель плейлиста. Панель может находиться в трех состояниях: “Спрятана”, “Выдвинута, но не активна”, “Выдвинута и активна”.

```
<SWITCH name="PlayList">
```

Состояние “Спрятана”

```
<STATE name="Hide">

Описание плейлиста в этом состоянии
Физическое окно уменьшено
 <WINDOW topleft="TL(-20, -20)" bottomright="BR(13, 64)" borderwidth="1"
          bordercolor="#000000"/>
<CONTROL topleft="TL(-1, -15)"  bottomright="TR(1, 0)" chit="title" normal="t_light.bmp"/>

Клавиатура управляет проигрыванием видео документов
<KEYBOARD playcontrol="on"/>
```

Кнопка, отвечающая за переход в другое состояние (“Плейлист”)

```
<CONTROL topleft="BR(-56, 17)"  bottomright="BR(-20, 48)" normal="list.bmp"
        mouse="list_mouse.bmp" leftdown="list_down_mouse.bmp"
        leftaction="PlayList.ActiveShow"/>
```

```
</STATE>
```

Состояние “Выдвинута и активна”

```
<STATE name="ActiveShow">
```

Описание плейлиста в этом состоянии

Клавиатура управляет плейлистом

```
<KEYBOARD playcontrol="off"/>
```

Далее перечислены визуальные *контролы* "Выдвинутого и активного" окна

```
<WINDOW topleft="TL(-20, -20)" bottomright="BR(400, 64)" borderwidth="1"
        bordercolor="#000000"/>
<REGION type="Rect" topleft="TR(20, -15)" bottomright="BR(400, 63)"/>
<REGION type="Rect" topleft="BR(-10, 28)" bottomright="BR(20, 35)"/>
<CONTROL topleft="BR(-10, 28)" bottomright="BR(20, 35)" chit="title" normal="bridge.bmp"/>
<PLAYLIST topleft="TR(30, 15)" bottomright="BR(390, 53)"
        back="#858585#858585#858585#858585"
        active="#000000#000000#000000#000000" textcolor="#000000" activecolor="#ffffff"
        playcolor="#000000" font="Arial Bold" height="18" />
<CONTROL topleft="TR(20, -15)"  bottomright="TR(400, 15)" chit="title"
        normal="Playlist_light.bmp"/>
<CONTROL topleft="BR(20, 53)"  bottomright="BR(400, 63)" chit="title" normal="pb.bmp"/>
<CONTROL topleft="TR(20, 15)"  bottomright="BR(30, 53)" chit="title" normal="pl.bmp"/>
<CONTROL topleft="TR(390, 15)"  bottomright="BR(400, 53)" chit="title" normal="pr.bmp"/>
<CONTROL topleft="BR(-56, 17)"  bottomright="BR(-20, 48)" normal="list_down.bmp"
        mouse="list_down_mouse.bmp" leftdown="list_mouse.bmp"
        leftaction="PlayList.Hide"/>
<CONTROL topleft="TR(319, -15)"  bottomright="TR(354, 15)" mouse="Arrow_light.bmp"
        leftdown="Arrow_down.bmp" leftaction="Menu_Playlist"/>
<CONTROL topleft="TL(-1, -15)"  bottomright="TR(1, 0)" chit="title" normal="t_dark.bmp"/>
```

Кнопки, отвечающие за переходы в другие состояния

Кнопка “Спрятать Плейлист”

```
<CONTROL topleft="BR(-56, 17)"  bottomright="BR(-20, 48)" normal="test2.bmp"
        mouse="test2.bmp" leftdown="list.bmp" leftaction="PlayList.Hide"/>
```

Невидимая кнопка “Деактивировать”.

```
<CONTROL topleft="TL(0, 0)" bottomright="BR(0, 0)" leftaction="PlayList.InactiveShow"/>
</STATE>
```

Состояние “Выдвинута, но не активна”

```
<STATE name="InactiveShow">
```

Описание плейлиста в этом состоянии

Клавиатура управляет проигрыванием

```
<KEYBOARD playcontrol="on"/>
```

Далее описаны визуальные *контролы* “Выдвинутого, но не активного“ окна

```
<WINDOW topleft="TL(-20, -20)" bottomright="BR(400, 64)" borderwidth="1"
        bordercolor="#000000"/>
<REGION type="Rect" topleft="TR(20, -15)" bottomright="BR(400, 63)"/>
<REGION type="Rect" topleft="BR(-10, 28)" bottomright="BR(20, 35)"/>
<CONTROL topleft="BR(-10, 28)" bottomright="BR(20, 35)" chit="title" normal="bridge.bmp"/>
<PLAYLIST topleft="TR(30, 15)" bottomright="BR(390, 53)"
        back="#858585#858585#858585#858585" active="#505050#505050#505050#505050"
        textcolor="#000000" activecolor="#AAAAAA"
        playcolor="#000000" font="Arial Bold" height="18" />
<CONTROL topleft="TR(20, -15)"  bottomright="TR(400, 15)" chit="title"
        normal="Playlist_dark.bmp"/>
<CONTROL topleft="BR(20, 53)"  bottomright="BR(400, 63)" chit="title" normal="pb.bmp"/>
<CONTROL topleft="TR(20, 15)"  bottomright="BR(30, 53)" chit="title" normal="pl.bmp"/>
<CONTROL topleft="TR(390, 15)"  bottomright="BR(400, 53)" chit="title" normal="pr.bmp"/>
```

```
            <CONTROL topleft="TR(319, -15)"  bottomright="TR(354, 15)" mouse="Arrow_light.bmp"
                    leftdown="Arrow_down.bmp" leftaction="Menu_Playlist"/>
            <CONTROL topleft="TL(-1, -15)"  bottomright="TR(1, 0)" chit="title" normal="t_light.bmp"/>

            Кнопки, отвечающие за переходы в другие состояния
            Кнопка “Спрятать плейлист”
            <CONTROL topleft="BR(-56, 17)"  bottomright="BR(-20, 48)" normal="list_down.bmp"
                    mouse="list_down_mouse.bmp" leftdown="list_mouse.bmp"
                    leftaction="PlayList.Hide"/>

            Невидимая кнопка “Активировать плейлист”
            <CONTROL topleft="TR(30, 15)" bottomright="BR(390, 53)" leftaction="PlayList.ActiveShow"/>

             При запуске проигрывания из состояния "Выдвинута и активна" необходимо
             деактивировать плейлист.
             <EVENT message="Play" source="ActiveShow"/>
        </STATE>

    </SWITCH>
```

Главный скин описывает внешний вид окон проигрывателя. Внешний вид всплывающих меню также описывается скином. Для этого в теле главного тега <SKIN object=”Main”/> размещается вложенный тег <SKIN object=”Menu”/>

```
<SKIN object="Menu">
        <SWITCH name="Main">
            <STATE name="Main">
                <REGION type="Rect" topleft="TL(0, 0)" bottomright="BR(0, 0)"/>
                <WINDOW topleft="TL(0, 0)" bottomright="BR(0, 0)" borderwidth="1"
                        bordercolor="#000000"/>
                <MENU topleft="TL(0, 0)" bottomright="BR(0, 0)"
                        back="#858585#858585#858585#858585"
                        active="#505050#505050#505050#505050"
                        textcolor="#000000" activecolor="#AAAAAA" graycolor="#404040"
                        keycolor="#505050" keyactivecolor="#F0F0F0" font="Arial" height="18"
                        separatorheight="8" popupwidth="16" iconwidth="16"
                        popup="popup.bmp?#858585" check="check.bmp?#858585"
                        separator="#000000#000000#000000#000000">
                </MENU>
            </STATE>
        </SWITCH>
    </SKIN>
</SKIN>
```

5. Заключение

Если в настоящее время для сотовых телефонов модно иметь сменные панели управления, то для современных виртуальных устройств модно иметь сменные скины. Ярким примером этого является поддержка скинов (тем - themes) в системе *Windows XP* корпорации *Microsoft*. Скины сейчас являются незаменимым атрибутом любого профессионального приложения, имеющего дружелюбный интерфейс.

Естественное желание при разработке любого формата – дать максимальную гибкость авторам, которые в дальнейшем будут использовать этот формат. Обычно гибкость достигается за счет усложнения формата - введения в него исполняемого кода, для того, чтобы авторы могли писать в скинах целые программы. При этом, однако, появляется большая проблема: усложнение формата сильно уменьшает круг авторов, способных формат понять.

Однако, вносить в формат скинов исполняемый код нет необходимости. Для описания внешнего вида использовать код даже странно – гораздо проще описать несколько статических состояний скина и логику переходов между состояниями. Благодаря использованию явного выделения состояний логика описывается

гипертекстом. В результате круг людей, способных писать скины, значительно расширяется.

Примером этого являются пять скинов, написанные **Mario Sernicola** [4], который понял формат и применил его даже без документации, только посмотрев несколько примеров.

В заключение отметим, что среди уже разработанных скинов имеется один особенный. Он на основе предложенного подхода моделирует поведение скина, использующего стандартную визуализацию окон *Windows* (рис. 2).

Авторы надеются, что документация позволит расширить круг разработчиков скинов для программы ***Crystal Player*** [4] и станет примером для авторов других программ, нуждающихся в поддержке скинов. Фрагмент разработанной документации на английском языке приведен в [4].

Источники

1. *Шалыто А.А.* SWITCH – технология http://is.ifmo.ru, http://www.softcraft.ru
2. *Microsoft* “SDK Media Player Skin Documentation” http://microsoft.com
3. *Microsoft* “SDK MS XML3.0 Documentation”
4. *Сайт* программы ***Crystal Player*** www.crystalplayer.com