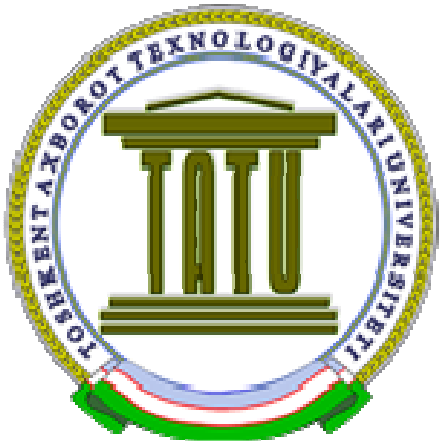
МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

НУКУССКИЙ ФИЛИАЛ ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫЙ ИНЖИНИРИНГ



Курсовая работа

По предмету: Программирование на языке C++
На тему: Воспроизведение аудио-файлов в Borland C++ Builder

Выполнил: студент 2-в курса
направления Компьютерный инжиниринг
Нукусбаев Н
Принял: _____

НУКУС 2016г.

План:

1) Введение
2) Функция PlaySound
3) Кнопки компонента MediaPlayer
4) Своиства компонента MediaPlayer
5) Методы компонента MediaPlayer
6) Заключение
7) Список использованных литература

Большинство современных программ являются мультимедийными, что подразумевает использование возможности компьютера отображать графику, воспроизводить видео, анимацию, музыку. Типичными примерами мультимедийных программ являются игры и обучающие программы.
Функция PlaySound
Для реализации звуковых эффектов (например, в играх) весьма удобна функция PlaySound. Она позволяет проиграть звуковой фрагмент, находящийся в wav-файле.
Инструкция вызова функции PlaySound в общем виде выглядит так:
PlaySound(wav-файл, 0, Режим);
Параметр wav-файл задает звуковой файл, параметр Режим — режим воспроизведения (синхронный или асинхронный). Если задан синхронный режим воспроизведения, то функция PlaySound возвращает управление программе сразу после того, как будет инициирован процесс воспроизведения звука. Если задан асинхронный режим, то программа, вызвавшая функцию PlaySound, продолжит работу только после того, как завершится воспроизведение звукового файла. В качестве значения параметра Режим можно указать именованную константу SND_SYNC (синхронный режим) или SND_ASYNC (асинхронный режим).
Например, инструкция
PlaySound('ringin.wav',0,SND_ASYNC);
активизирует процесс воспроизведения файла ringin.wav.

Для того чтобы функция PlaySound стала доступной, в программу надо добавить директиву #include <mmsystem.hpp>.
В качестве примера использования функции PlaySound в листинге 1.1 приведен фрагмент программы Будильник — функция обработки сигнала от таймера. Когда наступает время, на которое установлен будильник, на экране появляется окно с сообщением. Появление окна сопровождается звуком notify.wav. Так как задан асинхронный режим воспроизведения, то окно появляется сразу после начала воспроизведения звукового файла.

Использование функции PlaySound

```
сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
if ( CompareTime(Now(),AlarmTime) >= 0)
{
Timer1->Enabled = false;
if ( CheckBox1->Checked)
```

```
PlaySound("notify.wav",0,SND_ASYNC);
ShowMessage(FormatDateTime(" hh:nn — ", Now() ) + Edit3->Text);
Form1->Show(); // отобразить (развернуть) окно
TrayIcon1->Visible = false;
}
}
```

Следует обратить внимание на то, что если в имени wav-файла путь не указан, то функция PlaySound сначала будет искать звуковой файл в текущем каталоге (в каталоге, из которого запущена программа), затем в каталоге C:\Windows\Media. Если ни в одном из этих каталогов нужного файла нет, то будет воспроизведен так называемый "стандартный звук" (задается в настройках Windows). Программист может запретить воспроизведение стандартного звука. Для этого в качестве параметра Режим надо указать константу
SND_NODEFAULT.

Компонент MediaPlayer

Компонент MediaPlayer обеспечивает воспроизведение звуковых файлов различных форматов (wav, mid, mp3), компакт-дисков, сопровождаемых звуком анимации и видеороликов (avi).

1. Значок компонента MediaPlayer

Значок компонента MediaPlayer находится на вкладке System (рис. 1.1). Внешне компонент MediaPlayer представляет собой группу кнопок (рис. 1.2), подобных тем, которые можно видеть на аудио- или видеоплеере. Назначение
этих кнопок пояснено в табл. 1.1. Свойства компонента MediaPlayer, доступные во время разработки формы, приведены в табл. 1.2

2. Компонент MediaPlayer

Таблица 1.1 Кнопки компонента MediaPlayer

Кнопка		Обозначение	Действие
	Воспроизведение	btPlay	Воспроизведение звука или видео
	Пауза	btPause	Приостановка воспроизведения
	Стоп	btStop	Остановка воспроизведения
	Следующий	btNext	Переход к следующему кадру
	Предыдущий	btPrev	Переход к предыдущему кадру
	Шаг	btStep	Переход к следующему звуковому фрагменту, например, к следующему треку (композиции) на CD
	Назад	btBack	Переход к предыдущему звуковому фрагменту, например, к предыдущей песне на CD
	Запись	btRecord	Активизирует процесс записи
	Открыть	btEject	Открывает CD-дисковод компьютера

Таблица 1.2 Своиства компонента MediaPlayer

Свойство	Описание
Name	Имя компонента. Используется для доступа к свойствам компонента и управления работой плеера
DeviceType	Тип устройства. Определяет конкретное устройство, которое представляет собой компонент MediaPlayer. Тип устройства задается именованной константой: dtAutoSelect — тип устройства определяется автоматически по расширению файла; dtVaweAudio — проигрыватель звука; dtAVIVideo — видеопроигрыватель; dtCDAudio — CD-проигрыватель
FileName	Имя файла, в котором находится воспроизводимый звуковой фрагмент или видеоролик
AutoOpen	Признак автоматической загрузки сразу после запуска программы, файла видеоролика или звукового фрагмента
Display	Определяет компонент, поверхность которого используется в качестве экрана для воспроизведения видеоролика (обычно в качестве экрана для отображения видео используют компонент Panel)
VisibleButtons	Составное свойство. Определяет видимые кнопки компонента

Помимо свойств, доступных в процессе разработки (эти свойства отображаются в окне Object Inspector), у компонента MediaPlayer есть и другие, доступные только во время работы программы свойства (табл. 1.3). Они позволяют получить информацию о состоянии медиаплеера, воспроизводимом файле или треке CD. Следует обратить внимание, что значения свойств, содержащих информацию о длительности, могут быть представлены в различных форматах. Наиболее универсальным форматом является формат tfMilliseconds, в котором длительность выражается в миллисекундах. Некоторые устройства поддерживают несколько форматов. Например, если MediaPlayer используется для воспроизведения CD, то информация о воспроизводимом треке может быть представлена в формате tfTMSF (Track, Minute, Second, Frame — трек, минута, секунда, кадр). Для преобразования миллисекунд в минуты и секунды надо воспользоваться известными соотношениями. Если значение свойства представлено в формате tfTMSF, то для преобразования можно использовать функции MCI_TMSF_TRACK, MCI_TMSF_SECOND и MCI_TMSF_MINUTE.

Таблица 1.3 Своиства компонента MediaPlayer, доступно во время работы

Свойство	Описание
Length	Длина (время, необходимое для воспроизведения) открытого файла (например, wav или avi) или всех треков Audio CD
Tracks	Количество треков на открытом устройстве (количество композиций на Audio CD)
TrackLength	Длина (длительность) треков CD. Свойство представляет собой массив, каждый элемент которого содержит информацию о длине трека (времени воспроизведения)
Position	Позиция (время от начала) в процессе воспроизведения трека
TimeFormat	Формат представления значений свойств Length, TrackLength и Position. Наиболее универсальным является формат tfMilliseconds. Если медиаплеер представляет собой CD-проигрыватель, то удобно использовать формат tfTMSF
Mode	Состояние устройства воспроизведения. Устройство может быть в состоянии воспроизведения (mpPlaying). Процесс воспроизведения может быть остановлен (mpStopped) или приостановлен (mpPaused). Устройство может быть не готово к работе (mpNotReady) или в устройстве (CD-дисководе) может отсутствовать носитель (mpOpen)
Display	Экран — поверхность, на которой отображается клип. Если значение свойства не задано, то клип отображается в отдельном, создаваемом во время работы программы, окне
DisplayRect	Размер и положение области отображения клипа на поверхности экрана

Компонент MediaPlayer предоставляет методы (табл. 1.4), используя которые можно управлять работой медиаплеера из программы так, как будто это делает пользователь.

Таблица 1.4. Методы компонента MediaPlayer

Метод	Действие
Play	Активизирует процесс воспроизведения. Действие метода аналогично щелчку на кнопке **Play**
Stop	Останавливает процесс воспроизведения
Pause	Приостанавливает процесс воспроизведения
Next	Переход к следующему треку (например, к следующей композиции на Audio CD)

Таблица 5.4 *(окончание)*

Метод	Действие
`Previous`	Переход к предыдущему треку (например, к следующей композиции на Audio CD)
`Step`	Переход к следующему кадру
`Back`	Переход к предыдущему кадру

Простой MP3-player

Как было сказано ранее, компонент MediaPlayer обеспечивает воспроизведение звуковых файлов, в том числе и mp3-формата. Следующий пример показывает, как на основе компонента MediaPlayer можно создать mp3 плеер. Форма программы приведена на рис. 1.3, значения свойств компонентов — в табл. 1.5. Следует обратить внимание, что кнопка Eject компонента MediaPlayer не используется, а ее функцию выполняет кнопка BitBtn. Это объясняется тем, что кнопками управляет сам компонент и кнопка Eject доступна только тогда, когда компонент MediaPlayer используется для воспроизведения CD (значение свойства DeviceType равно dtCDAudio). Во время работы программы в поле компонента Label2 отображается длительность — время, необходимое для воспроизведения выбранного файла, а в поле компо-
нента Label1 — время, прошедшее от момента начала воспроизведения. Щел-
чок на кнопке BitBtn1 открывает стандартное окно Выбор папки. Текст программы приведен в листинге 1.2. Объявление функций PlayList и TrackInfo, а также переменных SoundPath, min и sec надо поместить в h-файл.

Рис. 1.3.

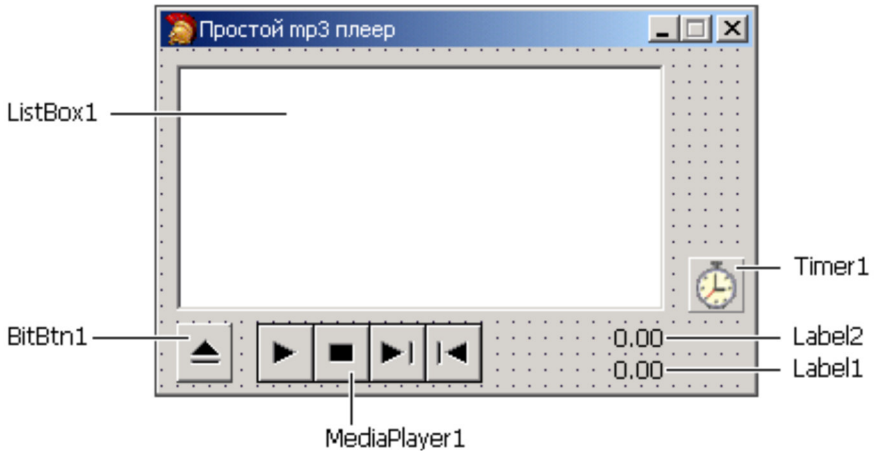


Рис 1.3 Форма программы простой mp3-player

Компонент	Свойство	Значение
MediaPlayer1	DeviceType	dtAutoSelect
	VisibleButtons.btPause	false
	VisibleButtons.btStep	false
	VisibleButtons.btBack	false
	VisibleButtons.btRecord	false
	VisibleButtons.btject	false
	ColoredButtons.btPlay	false
	ColoredButtons.btStop	false
	ColoredButtons.btNext	false
	ColoredButtons.btPrev	false
BitBtn1	Glyph	
	Hint	Выбор папки
	ShowHint	true

```
#include "FileCtrl.hpp" // для доступа к TSearchRec
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
PlayList("");
}
// формирует список mp3-файлов, находящихся в указанном каталоге
void __fastcall TForm1::PlayList(AnsiString path)
{
TSearchRec SearchRec; // структура SearchRec содержит информацию
// о файле, удовлетворяющем условию поиска
ListBox1->Clear();
// сформировать список mp3-файлов
if ( FindFirst(path + "*.mp3", faAnyFile, SearchRec) != 0 )
return;
// в каталоге есть файл с расширением mp3
// добавим имя этого файла в список
ListBox1->Items->Add(SearchRec.Name);
MediaPlayer1->FileName = SoundPath + ListBox1->Items->Strings[0];
MediaPlayer1->Open();
TrackInfo();
// пока в каталоге есть другие файлы с расширением wav
```

```
while (FindNext(SearchRec) == 0)
ListBox1->Items->Add(SearchRec.Name);
ListBox1->ItemIndex = 0;
}
// сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
AnsiString st;
if ( MediaPlayer1->Position == MediaPlayer1->Length )
// воспроизведение текущей композиции не завершено
{
Timer1->Enabled = false;
return;
}
// индикация времени воспроизведения
if ( sec < 59 )
sec++;
else {
sec = 0;
min++;
}
st = IntToStr(min) + ".";
if ( sec < 10)
st = st + "0" + IntToStr(sec);
else
st = st + IntToStr(sec);
Label1->Caption = st;
}
// щелчок на кнопке компонента MediaPlayer
void __fastcall TForm1::MediaPlayer1Click(TObject *Sender, TMPBtnType
Button,
bool &DoDefault)
{
switch ( Button ) {
case btPlay: // кнопка Play
min = 0;
sec = 0;
Timer1->Enabled = true;
break;
case btStop: // кнопка Stop
Timer1->Enabled = false;
break;
```

```
case btNext: // кнопка Next
if (ListBox1->ItemIndex < ListBox1->Items->Count-1)
{
ListBox1->ItemIndex += 1;
MediaPlayer1->FileName = SoundPath +
ListBox1->Items->Strings[ListBox1->ItemIndex];
MediaPlayer1->Open();
TrackInfo();
Timer1->Enabled = false;
}
break;
case btPrev: // кнопка Prev
if (ListBox1->ItemIndex > 0 )
{
ListBox1->ItemIndex -= 1;
MediaPlayer1->FileName = SoundPath +
ListBox1->Items->Strings [ListBox1->ItemIndex];
MediaPlayer1->Open();
TrackInfo();
Timer1->Enabled = false;
}
break;
}
}
// выводит информацию о текущем треке
void __fastcall TForm1::TrackInfo()
{
long int TrackLength; // длина трека в миллисекундах
int min,sec; // длина трека: минут, секунд
AnsiString st;
TrackLength = MediaPlayer1->TrackLength[1]/1000;
min = TrackLength / 60;
sec = TrackLength % 60;
st= IntToStr(min) + ".";
if (sec < 10)
st = st + "0" + IntToStr(sec);
else
st = st + IntToStr(sec);
Label2->Caption = st;
Label1->Caption = "0.00";
}
// щелчок на имени файла (композиции)
```

```
void __fastcall TForm1::ListBox1Click(TObject *Sender)
{
Timer1->Enabled = false;
MediaPlayer1->Stop(); // остановить воспроизведение текущей композиции
MediaPlayer1->FileName = SoundPath +
ListBox1->Items->Strings [ListBox1->ItemIndex];
MediaPlayer1->Open();
TrackInfo();
}
// щелчок на кнопке Выбор папки
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
if ( SelectDirectory("Укажите каталог, в котором находятся
mp3-файлы", "",SoundPath) )
{
SoundPath = SoundPath +"\\";
PlayList(SoundPath);
}
}
```

Компонент MediaPlayer позволяет просматривать видеоролики и сопровож-
даемую звуком анимацию. В качестве примера использования компонента
для решения этой задачи рассмотрим программу Video Player (рис. 1.9),
с помощью которой можно просмотреть небольшой ролик или анимацию.
Форма программы приведена на рис. 1.10, значения свойств компонентов —
в табл. 1.7.

Рис. 1.9. Окно программы Video Player

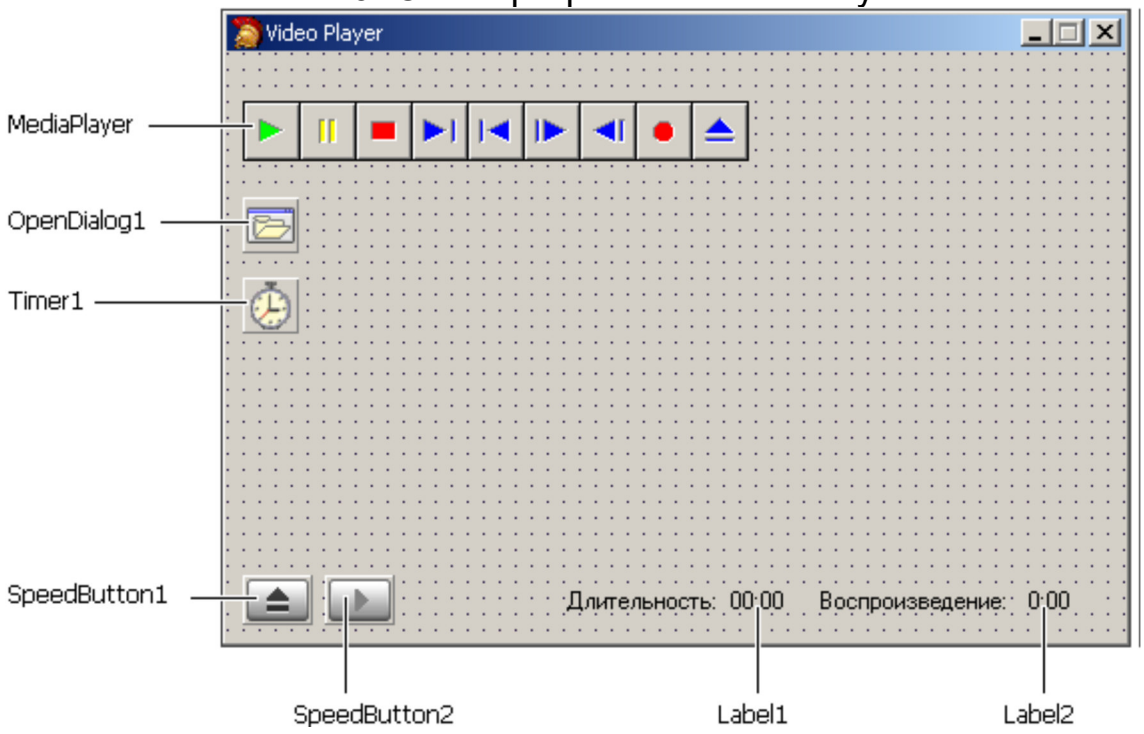


Таблица 1.7 Значение свойств компонентов

Компонент	Свойство	Значение
MediaPlayer	DeviceType	dtAutoSelect
	Visible	false
SpeedButton1	NumGlyphs	4
	Glyph	
	Flat	true
	Enabled	true
SpeedButton2	NumGlyphs	4
	Glyph	
	Flat	true
	Enabled	false
	GroupIndex	1
	AllowAllUp	true
Timer1	Interval	1000
	Enabled	false

Компонент OpenDialog1 обеспечивает отображение стандартного диалогового окна Открыть файл для выбора файла. Окно Открыть файл становится доступным во время работы программы в результате щелчка на кнопке Eject (SpeedButton1). Следует обратить внимание, что для управления процессом воспроизведения кнопки компонента MediaPlayer1 не используются, поэтому свойству Visible компонента MediaPlayer присвоено значение false. Также необходимо обратить внимание на свойство GroupIndex кнопки SpeedButon1. Его значение равно единице, поэтому после щелчка кнопка остается в зафиксированном (нажатом) состоянии и на ее поверхности появляется значок "Stop" (значение свойства NumGlyhts равно 4, это значит, что в битовом образе есть картинка для нажатого состояния). Компонент Timer обеспечивает обновление информации на индикаторе: функция обработки сигнала от таймера (события Timer) выводит в поле Label2 информацию о времени воспроизведения клипа.

Текст программы приведен в листинге

```
/*
Простой видеоплеер. Демонстрирует использование:
— компонента MediaPlayer для воспроизведения
видеороликов (формат avi, mpg);
```

— компонента SpeedButton.
Замечание. Если окно программы, когда воспроизведение клипа закончено, перекрыть другим окном, то кадр будет испорчен. Чтобы этого не было, надо написать функцию обработки события Paint для формы.
*/
```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
// эти макросы обеспечивают перевод интервала времени,
// выраженного в миллисекундах, в минуты и секунды
#define MINUTE(ms) ((ms/1000)/60)
#define SECOND(ms) ((ms/1000)%60)
__fastcall TForm1::TForm1(TComponent* Owner)
: TForm(Owner)
{
MediaPlayer1->Display = Form1;
}
// возвращает размер кадра
void __fastcall GetFrameSize(AnsiString f, int *w, int *h)
{
if ( f.Pos(".avi") == 0 )
{
// пользователь выбрал mpg-файл
*w = 352;
*h = 240;
return;
}
// *** Пользователь выбрал avi-файл ***
// В заголовке avi-файла есть информация о размере кадра
struct {
char RIFF[4]; // строка RIFF
long int nu_1[5]; // не используется
char AVIH[4]; // строка AVIH
long int nu_2[9]; // не используется
long int w; // ширина кадра
long int h; // высота кадра
} header;
TFileStream *fs; // поток для чтения заголовка файла
/* операторы объявления потока и его создания
```

```
можно объединить: TFileStream *fs = new TFileStream(f,fmOpenRead); */
fs = new TFileStream(f,fmOpenRead); // открыть поток для чтения
fs->Read(&header, sizeof(header)); // прочитать заголовок файла
*w = header.w;
*h = header.h;
delete fs;
}
// щелчок на кнопке Eject (выбор видеоклипа)
void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
int fw, fh; // размер кадра клипа
int top,left; // левый верхний угол экрана
int sw, sh; // размер экрана (ширина, высота)
int mw, mh; // максимально возможный размер экрана
// (определяется текущим размером формы)
float kw, kh; // коэф-ты масштабирования кадра по ширине и высоте
float k; // коэффициент масштабирования кадра
OpenDialog1->Title = "Выбор клипа";
OpenDialog1->InitialDir = "";
OpenDialog1->Filter =
"Все форматы|*.avi;*.mpg;*.mpeg|"
"AVI|*.avi|MPG|*.mpg|MGEG|*.mpeg";
if ( ! OpenDialog1->Execute() )
return; // пользователь нажал кнопку Отмена
/* При попытке открыть файл клипа, который уже открыт,
возникает ошибка. */
if ( MediaPlayer1->FileName == OpenDialog1->FileName )
return;
/* Пользователь выбрал клип. Зададим размер и положение "экрана",
на котором будет выведен клип. Для этого надо знать размер
кадров клипа. */
//Form1->Caption = "Video Player — " + OpenDialog1->FileName;
GetFrameSize(OpenDialog1->FileName,&fw, &fh);// получить размер кадра
// вычислим максимально возможный размер кадра
mw = Form1->ClientWidth;
mh = Form1->Panel1->Top-10;
if ( fw < mw)
kw = 1; // кадр по ширине меньше размера экрана
else kw = (float) mw / fw;
if ( fh < mh)
kh = 1; // кадр по высоте меньше размера экрана
else kh = (float) mh / fh;
```

```
// масштабирование должно быть пропорциональным
if ( kw < kh)
k = kw;
else k = kh;
// здесь масштаб определен
sw = fw * k; // ширина экрана
sh = fh * k; // высота экрана
left = (Form1->ClientWidth — sw) / 2;
top = (Panel1->Top — sh) / 2;
MediaPlayer1->FileName = OpenDialog1->FileName;
MediaPlayer1->Open();
MediaPlayer1->DisplayRect = Rect(left,top,sw,sh);
/* если размер кадра выбранного клипа меньше размера
кадра предыдущего клипа, то экран (область формы)
надо очистить */
Form1->Canvas->FillRect(Rect(0,0,ClientWidth,Panel1->Top));
SpeedButton2->Enabled = True; // кнопка Play теперь доступна
// вывести информацию о времени воспроизведения
MediaPlayer1->TimeFormat = tfMilliseconds;
int ms = MediaPlayer1->Length;
AnsiString st = IntToStr(SECOND(ms));
if ( st.Length() == 1)
st = "0" + st;
st = IntToStr(MINUTE(ms)) + ":" + st;
Label1->Caption = st;
Label2->Caption = "0:00";
// активизируем процесс воспроизведения
SpeedButton2->Down = true;
SpeedButton2->Hint = "Стоп";
SpeedButton2->Tag = 1;
SpeedButton1->Enabled = false; // кнопка Eject недоступна
MediaPlayer1->Play();
Timer1->Enabled = true;
}
// щелчок на кнопке Play/Stop (воспроизведение/стоп)
void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{
if (SpeedButton2->Tag == 0)
{
// нажата кнопка Play
SpeedButton2->Down = true;;
SpeedButton2->Hint = "Стоп";
```

```
SpeedButton2->Tag = 1;
SpeedButton1->Enabled = false; // кнопка Eject недоступна
MediaPlayer1->Play();
Timer1->Enabled = true;
}
else // нажата кнопка Stop
{
MediaPlayer1->Stop();
SpeedButton2->Down = false;
SpeedButton2->Hint = "Play";
SpeedButton2->Tag = 0;
SpeedButton1->Enabled = true; // кнопка Eject доступна
Timer1->Enabled = false;
}
}
// сигнал от плеера
void __fastcall TForm1::MediaPlayer1Notify(TObject *Sender)
{
if ( ( MediaPlayer1->Mode == mpStopped ) && ( SpeedButton2->Tag ==
1))
{
Timer1->Enabled = false;
SpeedButton2->Down = false;
SpeedButton2->Hint = "Play";
SpeedButton2->Tag = 0;
SpeedButton1->Enabled = true; // сделать доступной кнопку Eject
}
}
/* Процедура обработки события Pain обеспечивает
отображение (перерисовку) первого кадра
при появлении окна, например, после того,
как пользователь отодвинет другое окно, перекрывающее
окно Video Player. */
void __fastcall TForm1::FormPaint(TObject *Sender)
{
if ( MediaPlayer1->Mode == mpStopped )
{
MediaPlayer1->Position = 1;
MediaPlayer1->Position = 0;
}
}
// завершение работы программы
```

```
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
MediaPlayer1->Close();
}
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
// вывести информацию о времени воспроизведения
// MediaPlayer1->TimeFormat = tfMilliseconds;
int ms = MediaPlayer1->Position;
AnsiString st = IntToStr(SECOND(ms));
if ( st.Length() == 1)
st = "0" + st;
st = IntToStr(MINUTE(ms)) + ":" + st;
Label2->Caption = st;
}
```

В качестве экрана, на котором осуществляется воспроизведение видеороликов, используется поверхность формы. Поэтому установить значение свойства Display компонента MediaPlayer1 во время разработки формы нельзя. Кроме того, размер экрана должен быть равен или пропорционален размеру кадров ролика. Значение свойства Display устанавливает функция обработки события Create для формы, а размер и положение экрана на форме — функция обработки события Click на кнопке Eject (SpeedButton1). Размер экрана устанавливается максимально возможным и таким, чтобы ролик воспроизводился без искажения (высота и ширина экрана пропорциональны высоте и ширине кадров). Размер кадров ролика возвращает функция GetFrameSize, которая извлекает нужную информацию из заголовка файла.

Список использованной литературы

1. Джарод Холингвэрт, Дэн Баттерфилд, Боб Сворт, Джэйми Оллсоп C++Builder 5. Руководство разработчика.

2. Borland C++ Builder 5. Энциклопедия программиста. Калверт Ч., Рейсдорф К., "ДиаСофт" - 2001, 944 стр.

3. http://www.codenet.ru/

4. Никита Культин, Самоучитель C++ Builder 2008г

Введение

Большинство современных программ являются мультимедийными, что подразумевает использование возможности компьютера отображать графику, воспроизводить видео, анимацию, музыку. Типичными примерами мультимедийных программ являются игры и обучающие программы. Таким образом мы сегодня в среде C++ Builder создаем аудио и видео плейер. Для создание этого приложение нам понадобится среда C++ Builder версия 6 или выше и еще базовые знаний языка C++.

Заключение

В ходе проведенного нами проекта по созданию аудио и видео плейера «», мы обнаружили, что скорость обработки увеличилась в два раза. Это позволяет значительно ускорить ход рабочего процесса, сокращая число необработанных вовремя данных, и снизить временные затраты. В долгосрочной перспективе мы видим возможность использования разработанного нами продукта в производственном процессе всех предприятиях, занимающихся подобного рода деятельностью.
В заключение отметим, что поставка приложений, созданных с помощью C++ Builder, осуществляется практически точно так же, как и поставка приложений, созданных с помощью Delphi. Не смотря на много строчных кодов, все коды программы понятно и очень грамотно написано. Над сложными коды специально написано комментарий что бы не были не понятном .