МГУПИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПРИБОРОСТРОЕНИЯ И ИНФОРМАТИКИ

В.М.Баканов

Программирование мультимедиа-систем

Учебно-методическое пособие по

выполнению лабораторных работ

Москва 2008

УДК 681.3

Рекомендовано к изданию в качестве учебно-методического пособия редакционно-издательским советом МГУПИ

Рецензенты:

проф. А.В.Рощин, доц. И.В.Степанова

Баканов В.М.

Программирование мультимедиа-систем: учебно-методическое пособие по выполнению лабораторных работ. —М.: МГУПИ, 2008. –47 с.

В данном учебно-методическом пособии представлены лабораторные работы по учебной дисциплине “Программирование мультимедиа-систем”.

Пособие предназначено для студентов, обучающихся по специальности “Вычислительные машины, комплексы и сети”, а также может быть использовано для студентов иных специальностей, изучающих учебную дисциплину “Программирование мультимедиа-систем”.

В данной работе рассматриваются в основном мультимедиа-системы на основе IBM PC-совместимых ПЭВМ и операционной системы (ОС) Windows, однако основные положения верны и для ЭВМ общего класса.

В лабораторных работах рассматривается практика работы со штатным программным обеспечением (ПО) и основы разработки оригинального ПО для создания и обработки мультимедиа-данных. При разработке оригинального ПО предполагается использование систем быстрой разработки приложений (RAD – *Rapid Application Design*) Delphi и C++Builder фирмы Borland Int. в среде ОС Windows, при этом знание основ программирования на языках Pascal и C/C++ априори предполагается. Для каждой из работ описываются соответствующие возможности компонент систем Delphi / C++Builder.

© В.М.Баканов, 2008
© МГУПИ, 2008

Содержание

Стр.

Введение .....……

1. *Лабораторная работа 1.* Создание простых программ для визуализации файлов изображений и видео…….

2. *Лабораторная работа 2.* Принципы создания ПО для обработки изображений……..……...

3. *Лабораторная работа 3.* Цифровая запись, редактирование и воспроизведения звука……

4. *Лабораторная работа 4.* Получение и цифровая обработка фотоизображений ……

5. *Лабораторная работа 5.* Получение и цифровая обработка видеоданных…….

6. *Лабораторная работа 6.* Создание видеофильмов…….

Список литературы.....……...

Введение

Мультимедиа-системы (ММС) суть системы обработки и представления мультимедиа-данных (ММД) и имеют два аспекта - аппаратный и программный. В данной работе рассматривается последний, причем как на пользовательском (приемы создания и обработки ММД с помощью ПО сторонних разработчиков) уровне, так и на уровне программиста (технологии программной реализации стандартных функций обработки ММД). Программирование в основном ориентировано на ОС Windows, переход на UNIX-подобные ОС облегчен использованием сред программирования Borland Kulix.

Усвоенные приемы разработки ПО реально полезны при создании программ обработки данных картографии, аэро- и космической съемки, системах автоматизированного проектирования, подготовке к распознавания образов (поиск и выделение границ) и др. При реальном программировании используется система C++Builder фирмы Borland Int., причем полученные навыки ценны при переходе на Delphi (в обоих системах разработки приложений совпадают имена объектов и компонентных функций, разница заключается лишь в синтаксисе языков программирования); при этом переход на Borland Kulix (для ОС Linux) максимально облегчен. Особое внимание обращено на особенности программирования (включая типовые ошибки) графических приложений и пути повышения производительности (что необходимо для создания профессиональных приложений).

При работе со звуковыми и видеоданными использованы известные (распространенные) пакеты сторонних разработчиков, хорошо зарекомендовавшие себя в практической работе.

1 Лабораторная работа 1. **Создание простых программ для визуализации файлов изображений и видео**

1.1 *Цель работы* – приобретение практических навыков в создании ПО (*‘вьюверов’*) для просмотра (визуализации) растровых изображений и аудио-видеофайлов с использованием систем быстрой разработки приложений Delphi и C++Builder фирмы Borland Int. в среде ОС Windows [1,3].

1.2 *Теоретические основы*. Любые видеоданные в конечном счете представляют собой последовательность блоков информации о каждой точке изображения (пикселов, *pixels - picture element*); сжатие информации (без потери качества или с некоторой потерей **о**ного) необходимо лишь для уменьшения объема дискового пространства, необходимого для сохранения соответствующего файла. При этом каждый раз при считывании файла происходит декомпрессия файла, при сохранении – повторная компрессия (согласно заданной технологии сжатия).

В системах Delphi/C++Builder определен компонент TImage (страница Additional палитры компонентов), предназначенный для визуализации изображений в виде битовой карты (TBitmap), метафайла (TMetafile) или ‘иконки’ (TIcon); JPEG-изображения могут быть обработаны с помощью специального (дополнительного) компонента Delphi/C++Builder.

Загрузка файла изображения в Image1 осуществляется соответствующими компонентными функциями (дополнительно используются компоненты OpenPictureDialog и SavePictureDialog):

```
if (OpenPictureDialog1->Execute())
 Image1->Picture->LoadFromFile(“имя_файла”);
```

сохранение

```
if (SavePictureDialog1->Execute())
 Image1->Picture->SaveToFile(“имя_файла”);
```

Размер (в пикселах) области визуализации (полотна) определяется как

```
int nX=Image1->Width;
int nY=Image1->Height;
```

Размер загруженного в Image изображения:

```
int pX=Image1->Picture->Width;
int pY=Image1->Picture->Height;
```

В целях визуализации изображения при его размерах, значительно превышающих размеры окна, компонент Image помещен в контейнер ScrollBox (причем свойство Align последнего установлено в alClient); этим достигается скроллинг окна показа изображения относительно окна программы.

Режим масштабирования (с показом загруженного изображения во весь размер полотна) включается переводом булева свойства полотна в true (соответственно и обратное действие):

```
Image1->Stretch = ! (Image1->Stretch);
```

Однако при этом (если отношение горизонтального и вертикального размеров изображения не совпадает с таковым полотна) изображение будет искажено (исказятся пропорции). Для сохранения пропорций (заметим, что большинство современных дисплеев используют ‘квадратные’ пикселы, т.е. AspectRatio=1) необходимо соответствующим образом изменять размер полотна; в модуле Main.cpp (проект View.bpr) описаны функции MakeScale (масштабировать с сохранением пропорций, выполняется при загрузке файла изображения) и ClearScale (вернуть режим визуализации без масштабирования). Соответственно функция MakeScale должны быть вызвана всякий раз при изменении размеров окна программы визуализации (реакция на событие OnResize), рис.1.1.

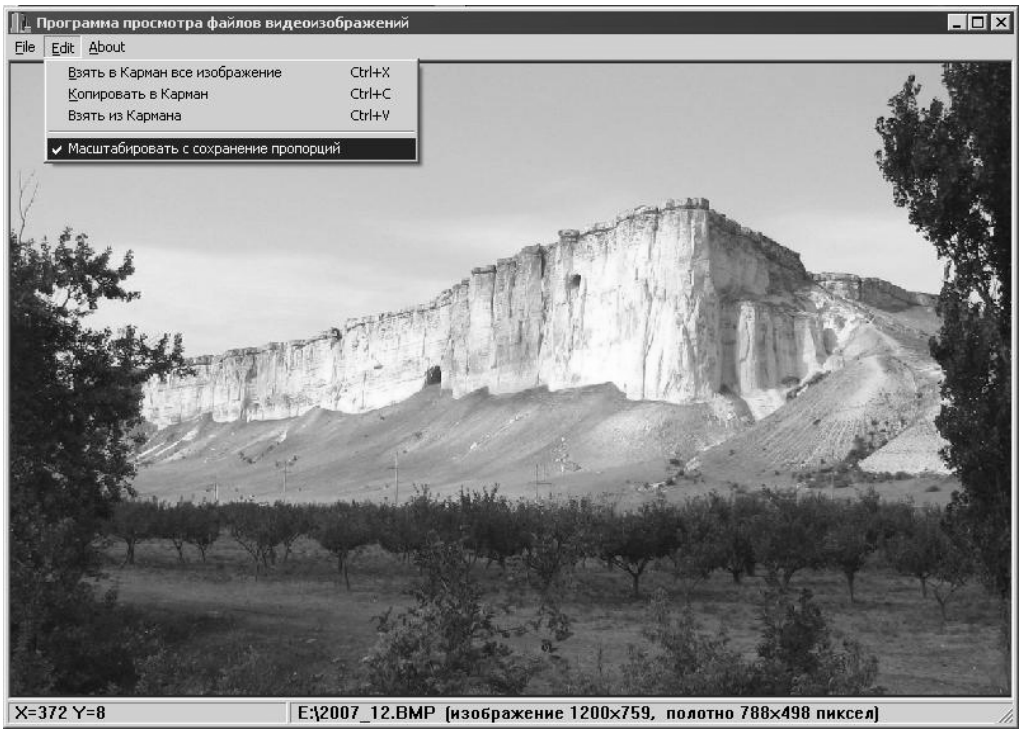


Рисунок 1.1 — Вид окна программы View.exe визуализации файлов изображений

Для работы с аудиовидеофайлами в системе Delphi/C++Builder используется компонент MediaPlayer (*'проигрыватель'*), представляющий интерфейс к мультимедийным возможностям Windows посредством MCI (*Media Control Interface*) - совместимых драйверов; причем большинство методов компонента MediaPlayer реализованы как вызовы стандартной функции mciSendCommand с соответствующими параметрами [3]. MediaPlayer может управлять приводами CD-ROM, MIDI-секвенсорами и рядом других устройств. Типы обрабатываемых (проигрываемых) файлов – MID, RMI, WAV, MP3, AVI, MPG (MPEG-1 поддерживаются в штатной инсталляции Windows, MPEG-2 – нет).

Проигрыватель визуализируется в виде линейки управления (набор кнопок, часть или все из которых могут быть сделаны невидимыми путем настройки свойства VisibleButtons в ObjectInspector'е или во время RunTime), см. рис.1.2.

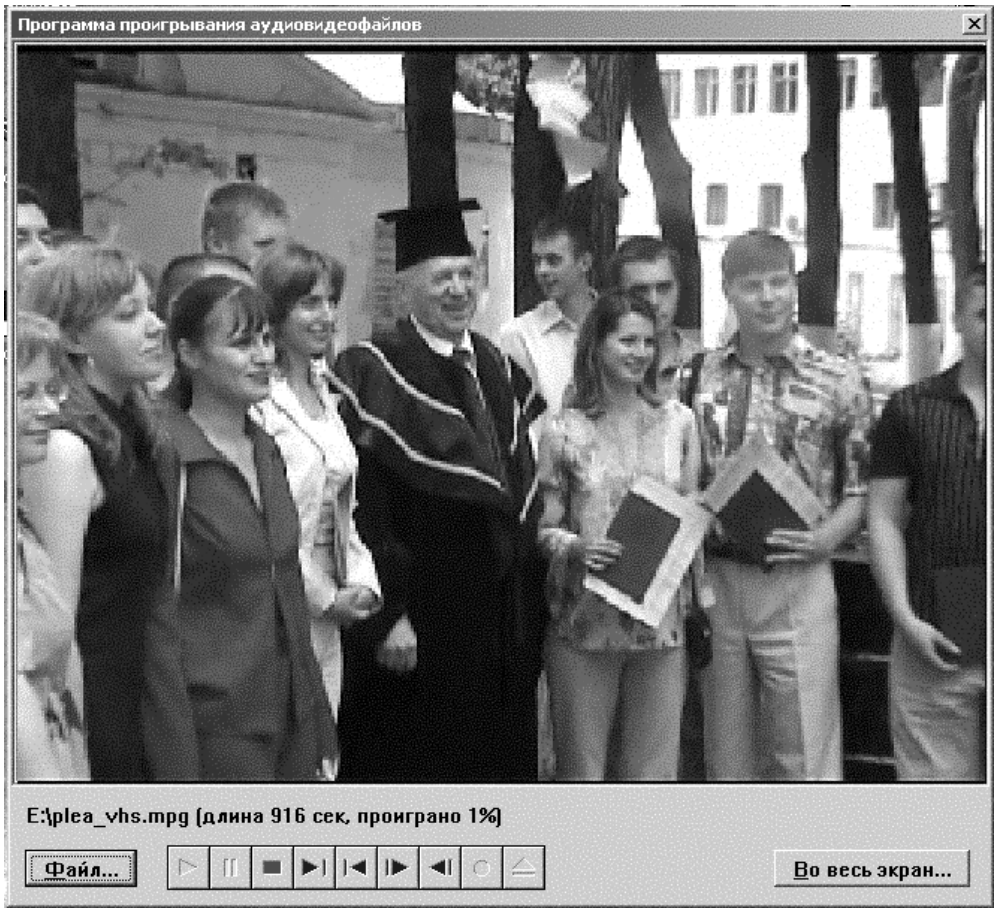


Рисунок 2.1 — Вид окна программы Play.exe проигрывания аудиовидеофайлов

Стандартная загрузка аудиовидеофайла для обработки компонентом MediaPlayer осуществляется следующимм образом:

```
…
MediaPlayer1->Close(); // сначала закрыть проигрыватель
```

```
if (OpenDialog1->Execute())
  {
   MediaPlayer1->FileName = OpenDialog1->FileName;
   MediaPlayer1->Open(); // открыть проигрыватель
  }
```

Начать проигрывание можно нажатием левой кнопки линейки управления (или вызовом MediaPlayer1→Play() при RunTime); при установке свойства AutoOpen в true проигрывание начинается сразу (нажатия кнопки или вызова метода Play не требуется). В случае DeviceType = dtAutoSelect проигрыватель самостоятельно определит тип устройства по расширению загруженного для проигрывания файла. При RunTime - управлении проигрывателем используются компонентные методы Play, Pause, Stop, Next, Prev, Step, StartRecording и Eject, соответствующие (слева направо) кнопкам линейки управления.

Область показа видео задается присвоением свойству Display проигрывателя имени компонента, в область которого осуществляется вывод видео:

```
MediaPlayer1->Display = Panel1; // показывать на Panel1
```

Имеется возможность явного определения размера области вывода видео (что важно, например, для реализации режима показа в полноэкранном режиме):

```
MediaPlayer1->DisplayRect = Rect(2,2,
        Panel1->Width-1,Panel1->Height-1); // задать размеры видеоэкрана
```

Большинство из описанных возможной реализовано в тексте файла Main.cpp проекта Play.bpr; о дополнительных возможностях проигрывателя MediaPlayer см. [2,5], систему контекстной помощи Delphi/C++Builder и др.

1.3 *Необходимое оборудование* – IBM PC-совместимая ЭВМ, предустановленная ОС Windows, пакет C++Builder версии выше 4; заготовки проектов View.bpr и Play.bpr (могут быть получены как http://pilger.mgapi.ru/metods/mm_src.zip).

1.4 *Порядок проведения работы*. Студент знакомится с общими сведениями по практике программирования визуализаторов файлов изображений и видео (уровень усвоения проверяется с использованием вопросов для самопроверки), рассматривает заготовки проектов, добивается корректного функционирования программы, по требованию преподавателя выполняет расширяющую возможности программы модификацию конкретные задания.

Возможные задания по модификации программ:

- Расширить функциональность программы просмотра файлов изображений возможностью задания произвольного масштаба (согласно линейке 500, 300, 200, 100, 75, 50, 25, 10% относительно размера изображения в файле).
- Создать проигрыватель двух мультимедиа-файлов одновременно, использующий один комплект кнопок управления.
- Снабдить проигрыватель шкалой управления, отражающей текущее положение в проигрываемом файле и имеющей возможность управлять позицией проигрывания.
- Реализовать возможность убыстренного и замедленного проигрывания мультимедиа-файла (линейка 2, 1.5, 1.0, 0.75, 0.5 относительно единичной скорости).

1.5 *Оформление отчета по работе.* Содержимое отчета должно отражать результаты самостоятельной работы над проектом (особенности предлагаемой программной реализации и предложения по расширению функциональности), по требованию преподавателя приводятся твердые копии (распечатки) внешнего вида интерфейса пользователя и в**ы**дачи программ. Фиксация имеющихся в заготовках проекта ошибок и (особенно) возможностей оптимизации кода весьма приветствуются.

1.6 *Вопросы для самопроверки.*

- Какие высокоуровневые компоненты Delphi/C++Builder применяются в программах мультимедиа и какова их функциональность?
- По каким причинам компонент типа TImage в стандартной поставке не может быть использован при работе с изображениями в формате JPEG?
- Чем отличается масштабирование с сохранением пропорций изображения от использования свойства Stretch? Каким приемом достигается скроллирование изображения большого размера?
- Каким путем возможно достичь неизменности режима вписывания изображения в окно при изменении (вплоть до ‘растягивания’ во весь экран дисплея) размеров окна программы?
- Какими программными средствами может быть реализован показ видеофайлов во весь экран дисплея?

2 Лабораторная работа 2. **Принципы создания ПО для обработки изображений**

2.1 *Цель работы* – приобретение практических навыков в создании ПО для обработки изображений в виде двумерной растровой однослойной графики с использованием систем быстрой разработки приложений Delphi и C++Builder фирмы Borland Int.

2.2 *Теоретические основы.*. При наличии в составе языка программирования средств д**о**ступа к информации о каждом пикселе (обычно интенсивность каждой из трех составляющих цвета) не представляет труда реализовать практически любую обработку видеоизображения. Несмотря на это, разработка профессиональных приложений (уровня PhotoShop, www.adobe.com, PaintShop, www.jasc.com и т.п.), включающих сотни самых разнообразных видеоэффектов, тем не менее требует серьезной работы коллектива программистов. Т.к. необходимо последовательно проводить преобразования многих тысяч пикселов, для достижения приемлемой производительности часть программного кода создается с использованием ассемблера. Заметим, что зачастую при пиксельных преобразованиях конечный результат получается независимо от других; это позволяет эффективно распараллеливать процесс вычислений.

Приведенными ниже примерами иллюстрированы простые возможности изменения изображений, при этом функции компрессии/декомпрессии, изменения палитры и др. выполняются с помощью штатных средств компонент сред быстрой разработки приложений. Профессиональные пакеты работы с графикой поддерживают сло**и** (совмещенные по координатам независимые изображения, причем обычно часть слоя в ф**о**новом цвете считается прозрачной), при этом допускается обработка каждого слоя как независимо, так и совместно с другими. Ниже рассматривается работа с растровой графикой; векторная графика (типичный представитель – пакет CorelDraw, www.adobe.com) в целом проще и полученные с ее помощью изображения легко конвертируются в растр (обратное утверждение неверно).

Д**о**ступ к контексту дисплея в GDI (*Graphic Device Interface*) Windows API реализуется посредством свойства Canvas (*полотно*) компонента TImage. Для копирования цвета (32-х битовое целое в Windows) из пиксела iX,iY (правая нижняя декартова система координат) в переменную iColor применяется следующий код (аналогично происходит обратное копирование):

```
long int iColor; // код цвета (32 бита)
…
iColor=Image1->Canvas->Pixels[iX][iY];
```

Размер (в пикселах) области визуализации определяется как:

```
int nX=Image1->Width;
int nY=Image1->Height;
```

Размер загруженного в Image изображения:

```
int pX=Image1->Picture->Width;
int pY=Image1->Picture->Height;
```

Типовой прием изменения цвета массива пикселов заключается в организации циклов по координатам:

```
for (int iX=100; iX<=300; iX++) // цикл по оси абсцисс
 for (int iY=200; iY<=400; iY++) // цикл по оси ординат
   if (Image1->Canvas->Pixels[iX][iY]==clRed) // если цвет пиксела красный...
    Image1->Canvas->Pixels[iX][iY]=clBlue; // сделать его синим!
```

Заметим, что приведенный выше код недостаточно эффективен по скорости выполнения (реально вызываются Windows API функции GetPixel и SetPixel, выполняющиеся достаточно медленно) и область его применения ограничена несложными демонстрационными примерами; профессиональный подход заключается в реализации соответствующего кода на ассемблере (с последующей ручной оптимизацией, использовании инструкций MMX и SSE [2]) и работе с расположенной в оперативной памяти (ОП) копией массива пикселов (изображение при этом визуализируется только после полной его обработки). Эффективные решения могут быть достигнуты при использовании возможностей разработанной Microsoft Corp. библиотеки DirectX (примеры С-программирования с использованием компонентов DirectDraw, DirectSound, DirectMusic, DirectInput и DirectPlay этой библиотеки приведены в [1,3]).

При работе на Delphi/C++Builder'е более профессиональным является создание внеэкранного б**и**тового **о**браза (часто с помощью компонента типа TBitmap, в дальнейших примерах используется TImage), обработка его в ОП и 'сброс' **о**ного на экран (режим копирования CopyMode=cmSrcCopy) с помощью метода Draw [3]. Ниже приведен один из (примитивных) примеров использования внеэкранного образа (объект tmpImage):

```
TImage *tmpImage; // описание объекта tmpImage
…
Image1->Picture->LoadFromFile("имя_файла"); // загрузить и показать
                                            // изображение из файла
tmpImage = new TImage(this); // создать экземпляра объекта TImage
tmpImage->Picture = Image1->Picture; // скопировать содержимое Image
                                     // во внеэкранный контейнер tmpImage
for (int iX=100; iX<=300;iX++) // цикл по оси абсцисс
```

```
 for (int iY=200; iY<=400; iY++) // цикл по оси ординат
   if (tmpImage->Canvas->Pixels[iX][iY] == clRed) // если цвет пиксела во
                                                  // внеэкранном контейнере красный...
    Image1->Canvas->Pixels[iX][iY]=clBlue; // то выдать его на экран синим!
…
delete tmpImage; // уничтожить tmpImage
```

Перо (Pen) предназначено для отрисовки одномерных графических объектов (линий) и определяется толщиной (Width), цветом (Color) и стилем (Style):

```
Canvas->Pen->Width = 1; // толщина в 1 пиксел
Canvas->Pen->Color = clRed; // цвет - красный
Canvas->Pen->Style = psDashDot; // штрих-пунктир
```

Кисть (Brush) предназначено для отрисовки двумерных графических объектов и определяется цветом (Color) и стилем (Style):

```
Canvas->Brush->Color = clGreen; // цвет - зеленый
Canvas->Brush->Style = bsDiagCross; // ‘заливка’ – диагональная решетка
```

Например процедура отрисовки эллипса (в полный размер Image1) черной кистью с ‘заливкой’ диагональной решеткой осуществляется так:

```
void __fastcall TForm1::ButtonClick(TObject *Sender)
{
 Image1->Canvas->Brush->Color=clBlack;
 Image1->Canvas->Brush->Style=bsDiagCross;
 Image1->Canvas->Ellipse(0, 0, Image1->Width, Image1->Height);
}
```

Методы MoveTo(int X, int Y) и LineTo(int X, int Y) устанавливают текущую позицию и отрисовывают прямую (текущим пером) в заданную точку соответственно:

```
void __fastcall TForm1::FormMouseDown(TObject *Sender,
                                      TMouseButton Button,
                                      TShiftState Shift, int X, int Y)
{ // при щелчке кнопкой мыши отрисовать линию из точки (0,0) в (X,Y)
 Image1->Canvas->MoveTo(0, 0);
 Image1->Canvas->LineTo(X, Y);
}
```

Текущая позиция пера определяется функцией PenPos (функция возвращает структуру типа TPoint, содержащую пиксельные координаты абсциссы и ординаты точки).

Метод Draw(int X, int Y, TGraphic* Graphic) отображает графический объект Graphic в точке с координатами X,Y (левый верхний угол). Метод FillRect(TRect &Rect) выполняет заливку прямоугольной области Rect цветом, который задан значением свойства Brush→Color. Метод Draw(int X, int Y, TGraphic* Graphic) отображает графический объект Graphic в точке с координатами X,Y (TRect представляет собой структуру, содержащую определяющие прямоугольную область целочисленные поля Left, Top, Right, Bottom в пикселах).

Метод CopyRect(TRect &Dest, TCanvas* Canvas, TRect &Source) копирует заданную полотном Canvas прямоугольную область Source в прямоугольную область Dest текущего полотна; режим копирования может быть предварительно задан свойством CopyMode (например, DrawForm→Canvas→CopyMode=cmPatCopy для копирования с логической операцией XOR при смешении цветов).

Копирование всего изображения из Image в *Карман* (Clipboard) осуществляется методом Assign (при этом *Карман* очищается от предыдущего содержимого):

```
…
#include <clipbrd.hpp> // хидер прототипов функций работы с Clipboard
…
Clipboard->Assign(Image1->Picture);
```

Обратное копирование требует проверки типа содержимого *Кармана* с помощью функции HasFormat (предыдущее содержимое Image затирается):

```
if (Clipboard()->HasFormat(CF_BITMAP)) // если в Кармане именно Bitmap…
  Image1->Picture->Assign(Clipboard());
```

Операция взятия изображения в *Карман* дополнительно требует очищения исходного изображения (при этом в качестве цвета заполнения обычно берется clWindow - фоновый цвет Windows):

```
void __fastcall TFormMain::CutToClipboard(TObject *Sender)
{ // взять все изображение в Clipboard
 Clipboard()->Assign(Image1->Picture);
 TRect r = Rect(0, 0, Image1->Width, Image1->Height);
 Image1->Canvas->Brush->Color = clWindow; // цвет заливки
 Image1->Canvas->FillRect(r); // залить весь Image1 выбранным цветом
}
```

Работа с *Карманом* с (прямоугольным) фрагментом изображения осуществляется с использованием функции CopyRect (режим копирования задается переменной CopyMode; возможные значения: cmSrcCopy – копирование с замещением исходного изображения, cmWhiteness - предварительное заполнение области копирования белым цветом, cmSrcAnd - копирование с

комбинацией имеющегося и наложенного изображения с логическим AND и др.).

Копировать часть изображения в *Карман* удобно с помощью внеэкранного контейнера (необходимое условие применения нижеприведенной функции суть xLeft<xRight && yTop<yBottom, в противном случае необходима перестановка координат – см. функцию ShiftRectCoord в модуле Main.cpp программы Paint; структура типа TRect представляет описывающие прямоугольник координаты {xLeft, yTop, xRight, yBottom}):

```
void __fastcall TForm1::CopyRectToClipboard(int xLeft, int yTop,
                                            int xRight, int yBottom)
{ // копировать прямоугольную часть исходного изображения
  // с левым верхним углом (xLeft,yTop) и правым нижним (xRight,tBottom)
 TRect rs, rd;
 TImage *tmpImage;
 tmpImage = new TImage(this); // создать экземпляр объекта TImage
 rs = Rect(xLeft,yTop, xRight,yBottom); // область копирования в Image1
 rd = Rect(0,0, xRight - xLeft,yBottom - yTop); // область копирования в tmpImage
 tmpImage->BoundsRect = rd; // настройка размеров tmpImage
 // tmpImage->Width  = xRight  - xLeft;  // та же самая настойка (ширина)
 // tmpImage->Height = yBottom - yTop; // … высота
 tmpImage->Canvas->CopyRect(rd,Image1->Canvas,rs); // копировать в tmpImage
 Clipboard()->Assign(tmpImage->Picture); // копировать из tmpImage в Clipboard
 delete tmpImage; // уничтожить tmpImage
}
```

Частой процедурой перед применением преобразований к части изображения (копированием в *Карман*, деформации, цветокоррекцией и др.) является выделение (прямоугольной) части изображения ‘резиновой рамкой’, перемещаемой по экрану при манипуляциях мышью. Проблемой при этом является необходимость восстановления (после каждого перемещения мыши) части изображения, ‘испорченного’ перемещающейся рамкой. Стандартным приемом является копирование прямоугольного участка изображения перед каждым перемещением правого нижнего угла рамки (левый верхний считается зафиксированным щелчком кнопки мыши) во внеэкранный образ и восстановление этой части изображения после смещения рамки.

В рассматриваемой программе Pain (модуль Main.cpp проекта Paint.bpr):

- функция Image1MouseDown (вызываемая по событию Image1MouseDown – ‘нажатие кнопки мыши’) запоминает (передаваемые через формальные параметры) координаты левой верхней точки рамки в переменных InitialX, InitialY, создает внеэкранный контейнер tmpImage и копирует в него изображение,
- функция Image1MouseMove (событие Image1MouseMove – ‘перемещение указателя мыши’) путем вызова отрисовывает ‘резиновую рамку’ и осуще-

ствляет необходимые действия по восстановлению изображения при изменении размеров рамки (тело функции Image1MouseMove выполняется только при постоянно нажатой левой кнопки мыши),

- функция Image1MouseUp (событие Image1MouseUp – ‘отпуск**а**ние кнопки указателя мыши’) осуществляет финальные действия по отрисовке и уничтожению внеэкранного контейнера tmpImage.

Заметим, что эффективнее было бы каждый раз при перемещении указателя мыши копировать во внеэкранный контейнер не все изображение, а лишь закрываемую ‘резиновой рамкой’ часть его (манипуляции со значительными объемами информации при работе с большими изображениями зачастую вызывают неприятное ‘мерцание’ экрана).

Именно операции копирования и восстановления части экрана являются основой эффектов перемещения частей изображения по экрану в нужное место и мультипликации (из внеэкранных контейнеров последовательно копируются в нужное место экрана соответствующие изображения). Профессиональные графические редакторы поддерживают возможность выделения части изображения произвольной формы (режим ‘лассо’) и соответствующие операции с *Карманом*; реально, конечно, копируется прямоугольный участок изображения, а находящиеся вне выделяющей нужную область замкнутой кривой (описываемой последовательностями прямых или сплайном) объявляются ‘прозрачными’ (*transparent*) и не участвуют в операциях. Режим ‘волшебной палочки’ (*magic wand*) позволяет ‘автоматически’ выделить (для дальнейшей обработки – копирования, пере**о**са и др.) область изображения (при этом используется анализ на принадлежность к заданному диапазону цветов или интенсивностей прилежащих к выбранной точке пикселов изображения).

Важной операцией обработки изображений (понижение резкости или, наоборот, ок**о**нтуривание и др.) осуществляется с помощью фильтров. Фильтр определяется (обычно квадратной) матрицей коэффициентов преобразования k_{ij} ($i=1 \div iN$, $j=1 \div jN$, iN и jN - нечетные) и постоянными DivisionFactor и Bias. При преобразовании текущей точки (i,j) изображения интенсивности ее цвета умножаются на значения k_{ij}, полученные значения складываются, делятся на DivisionFactor, к полученной сумме прибавляется Bias (считается, что обрабатываемая точка изображения совпадает с центральным элементом матрицы):

$$color_ij = Bias + \frac{1}{DivisionFactor} \sum_{i=1}^{i=iN} \sum_{j=1}^{j=jN} k_{ij}\, c_{ij},$$

где color_ij – определяемое значение интенсивности цвета в точке ij,

k_{ij} – коэффициенты фильтра,

c_{ij} – интенсивность цвета в точках исходного изображения.

Например, при использовании фильтра с iN=jN=5 обрабатываемая (ij-тая) точка изображения совпадает с k_{33} (при этом точка изображения i-3, j-3 проецируется на k_{11}, точка изображения i+3, j+3 проецируется на k_{55} и т.д.). Таким образом преобразование для каждой точки исходного изображения арифметически сводится приблизительно к N^2 (N – размер матрицы фильтра) операций целочисленного умножения и сложения; при значительном разрешении изображения необходимо использовать ассемблер и функции MMX и/или SSE [3].

При реализации подобных операций с использованием языков высокого уровня значительную часть времени отнимает считывание значений интенсивности цвета исходного изображения в c_{ij} ($i=1 \div iN$, $j=1 \div jN$) и запись рассчитанного color_ij в образ модифицированного изображения; при этом целесообразно работать не с (визуализирующим изображение) объектом Image, а с внеэкранным контейнером (для визуального контроля изменения изображения полезно регулярно копировать порции обработанного изображения в Image). В предлагаемом ПО (файл Paint.exe) изображение загружается в Image и копируется во внеэкранный **о**браз fltImage, оттуда же копируются элементы матрицы c_{ij}, после описанного преобразования значение интенсивности цвета color_ij записывается непосредственно в Image (в дисплейное окно) - см. процедуру ApplyFilter в модуле Main.cpp (проект Paint.bpr).

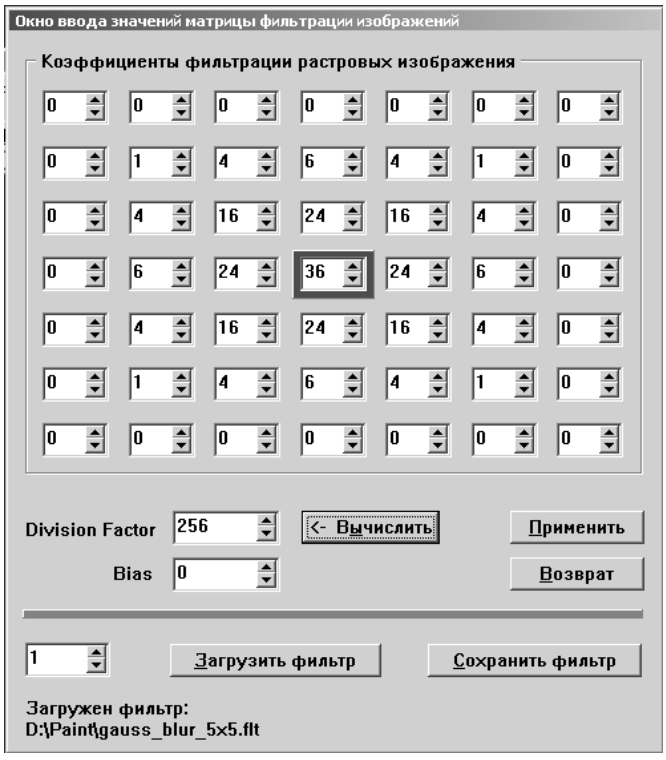


Рисунок 2.1 — Вид окна определения параметров фильтра в программе Paint.exe

Вид окна определения коэффициентов матрицы фильтрации программы Paint.exe приведен на рис.2.1 (д**о**ступ к окну осуществляется посредством выбора варианта Фильтр… главного меню).

Находящееся в левом нижнем углу окна поле ввода дает возможность установить р**а**вные значения всех коэффициентов матрицы (с целью минимизации дальнейшей настройки), кнопка ‘Вычислить’ позволяет определить значение DivisionFactor как сумму значений всех элементов матрицы фильтра. Выбранный фильтр *применяется ко всем тр**е**м составляющим цвета* (предварительно можно конвертировать изображение в серый тон посредством Edit → Преобразование цвета → В оттенки серого и, если необходимо, сохранить преобразованный таким образом файл).

Типовые значения коэффициентов матрицы фильтра приведены в табл.2.1 (для упрощения рассматривается матрица 5×5 или даже 3×3). Простейшим фильтром является Blur (снижение резкости, размывание изображения); в этом случае $k_{ij}\equiv1$ $(i,j=1\div n)$, $\mathrm{DivisionFactor}=\sum_{i=1,j=1}^{i=n,j=n}k_{ij}$ и Bias=0 (здесь n – размер матрицы, k_{ij} – значения коэффициентов матрицы). Из общих соображений ясно, что при $\mathrm{DivisionFactor}<\sum_{i=1,j=1}^{i=n,j=n}k_{ij}$ совместно с разм**ы**тием происходит потемнение изображения (в противном случае – осветл**е**ние); аналогично Bias>0 приводит к осветлению, а Bias<0 – к потемнению. В графическом редакторе PaintShop версии 8 создание нового фильтра возможно через Effects|User Defined…(для версии 4 - Image → User Defined Filters…). В версии 8 редактора PaintShop интересные эффекты можно получить (логической) комбинацией пикселов двух изображений (Image → Arithmetic… в версии 8); необычные (не все из них представ**и**мы матрицей) эффекты доступны посредством Effects → Artistic Effects.

С точки зрения программиста важно, что в описываемом случае параметры цвета каждого пиксела зависят от цветов окружающих и от цвета исходного; при этом применение внеэкранного контейнера изображения неизбежно (интенсивности пикселов считываются из помещенной во внеэкранный контейнер копии изображения, преобразуются и записываются на полотно экрана); образец исходного кода для работы с внеэкранным контейнером см. ниже – фрагмент функции ChangeColorAndIntensity.

Таблица 2.1. — Типовые матрицы фильтрации изображений.

Матрица фильтра	*Division Factor, Bias*	*Область применения*
$\begin{bmatrix} 1 & 1 & 2 & 1 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 2 & 2 & 8 & 2 & 2 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 1 & 2 & 1 & 1 \end{bmatrix}$	44,0	Снижение резкости (размывание, Blur-эффект).
$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	25, 0	Более сильное размывание (Blur More)
$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	256,0	‘Размывание’ по Гауссу (Gaussian Blur). Фильтр хорошо подходит для ‘размытия’ сюжетно незначимых фрагментов изображений (фотографий)
$\begin{bmatrix} -1 & -2 & -1 & -2 & -1 \\ 2 & -1 & 2 & -1 & 2 \\ -1 & -2 & 5 & -2 & -1 \\ 2 & -1 & 2 & -1 & 2 \\ -1 & -2 & -1 & -2 & -1 \end{bmatrix}$	-7,0	Эффект двойного изображения (Double Vision)
$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	1,0	Простейший фильтр поиска границ (Find Edges); значение центральной точки равно сумме остальных с обратным знаком.
$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	1,256	Выделение контуров (Trace Contour)

$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	1,0	Выделение горизонтальных границ (Find Horizontal Edges)
$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	1,0	Выделение горизонтальных границ (Find Vertical Edges)
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$	1,0	Вариант фильтра поиска границ – гравировка (Emboss 1)
$\begin{bmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$	1,0	Второй вариант фильтра поиска границ – гравировка (Emboss 2)
$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 10 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	6,0	Повышение резкости (Sharpen)
$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -2 & -2 & -2 & -1 \\ -1 & -2 & 64 & -2 & -1 \\ -1 & -2 & -2 & -2 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$	32,0	Дополнительное повышение резкости (Sharpen More)
$\begin{bmatrix} 9 & 0 & 3 & 0 & 9 \\ 0 & 0 & 0 & 0 & 9 \\ 3 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 3 & 0 & 9 \end{bmatrix}$	49,0	Эффект вибрации (Vibrate)

$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -7 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	1,0	‘Зас**а**ленность, зам**а**сливаемость’ (Grease)
$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 7 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	1,165	‘Сумерки’ (эффектно применение к однотонным изображениям или к выбранному цвету)
$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -2 & -2 & -2 & -1 \\ -1 & -2 & 34 & -2 & -1 \\ -1 & -2 & -2 & -2 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$	1,128	Имитация грубого рисунка цветными карандашами (модифицированный Sharpen)
$\begin{bmatrix} 1 & -1 & -3 \\ 1 & 5 & 2 \\ 1 & -1 & -3 \end{bmatrix}$	1,0	‘Психодел**и**ческий’ фильтр

Часто применяемым эффектом является изменение интенсивностей каждого из цветов текущего пиксела. Т.к. при этом цвет каждого пиксела не зависит от окружающих, внеэкранный контейнер, вообще говоря, не нужен (его применение оправдано лишь надеждой увеличения производительности и возможностью реализовать ‘откат’ изменений путем простого присваивания $\text{Image1} \to \text{Picture} = \text{colImage} \to \text{Picture};$).

На рис.2.2 приведен вид типовых зависимостей (копии графиков, полученных путем выбора Изображение → Настройка → Кривые… из главного меню PaintShop) интенсивности пикселов преобразованного цвета от интенсивности исходного; из общих соображений ясно, что ни величина аргумента, ни значение функции не должно выходить за границы $0 \div 255$. В табл.2.2 приведена дополнительная информация: в круглых скобках - определяющие положение кривых характерные точки (соответствуют рис.2.2), в квадратных – полученные методом наименьших квадратов (МНК) коэффициенты соответствующих уравнений вида $y = \sum_{i=0}^{i=n} a_i x^i$ (x,y – интенсивности исходной и преобразованной цветовых составляющих соответственно).

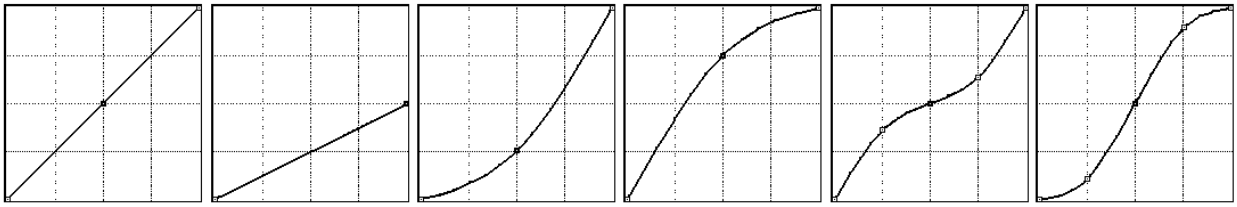


Рисунок 2.2 — Типовые зависимости интенсивности (яркости) преобразованной точки изображения (ось ординат) от интенсивности точки исходного изображений (ось абсцисс)

Таблица 2.2.— Характерные кривые преобразования яркости цветовых точек.

Схема уа рис.1	*Характерные точки кривой и коэффициенты a_i уравнения, аппроксимирующего кривую*	*Эффект от применения*
а)	(0,0), (128,128), (255,255) [0.0, 1.0]	Нет преобразования
б)	(0,0), (255,128) [0.0, 0.5]	Линейное снижение яркости
в)	(0,0), (128,64), (255,255) [0.0, -0.00394, 0.0039370079]	Нелинейное снижение яркости изображения в области низкой яркости совместно с повышением контраста в области высокой яркости
г)	(0,0), (128,192), (255,255) [0.0, 2.003937, -0.0039370079]	Нелинейное повышение яркости изображения в области низкой яркости с повышением контраста в области низкой яркости
д)	(0,0), (64,96), (192,159), (255,255) [0.0, 2.3491512, -0.015940975, 0.000041765435]	Снижение контраста в области средней яркости
е)	(0,0), (64,40), (192,215), (255,255) [0.0, -0.0065828087, 0.011595172, -0.000030193795]	Повышение контраста в области средней яркости

При работе с цветом согласно кривых преобразования изменяются интенсивности каждого из трех компонентов цвета (возможно применение собственной кривой для каждого цвета, что равносильно изменению цветов**о**го баланса).

В модуле main.cpp проекта Paint.bpr изменение цвета осуществляется функцией ChangeColorAndIntensity (приведен пример нескольких преобразований, определяемых значением управляющей переменной rule):

```
Int iX,iY;
short colR, colG, colB, intCol, // 0-255 интенсивность цвета (8 бит)
        selColor = clRed;
float A=0.0, B=-0.0039370079, C= 0.0039370079; // кривая в) на рис.2.2 и табл.2
…
colImage = new TImage(this); // внеэкранный битовый контейнер
colImage->Picture = Image1->Picture; // скопировали изображение в colImage
…

for (iX=0; iX<nX; iX++) // цикл по абсциссам пикселов ИЗОБРАЖЕНИЯ
 for (iY=0; iY<nY; iY++) // цикл по ординатам
  {
   // получить интенсивности цветов из внеэкранного контейнера
   colR=GetRValue(colImage->Canvas->Pixels[iX][iY]); //  интенсивность цвета R
   colG=GetGValue(colImage->Canvas->Pixels[iX][iY]); // …цвета G
   colB=GetBValue(colImage->Canvas->Pixels[iX][iY]);  // …цвета B

switch (rule) // переменная rule определяет режим преобразования цвета
 {
  case 1: // преобразовать в оттенки  серого ============================
            // среднее арифметическое из интенсивностей всех цветов
            intCol=(colR+colG+colB)/3;
            Image1->Canvas->Pixels[iX][iY] = RGB(intCol, intCol, intCol);
            break;
  case 2: // преобразовать в заданный цвет selColor ======================
            Image1->Canvas->Pixels[iX][iY] = RGB(GetRValue(selColor)*intCol/255,
                                                 GetGValue(selColor)*intCol/255,
                                                 GetBValue(selColor)*intCol/255);
            break;
  case 3: // уменьшить интенсивность всех цветов вдвое ==================
            Image1->Canvas->Pixels[iX][iY] = RGB(colR/2, colG/2, colB/2);
            break;
  case 3: // заданные полиномом преобразования цветов ================
           // используем схему Горнера для вычисление полинома y=A+B*x+C*x*x
           // где A,B,C – известные коэффициенты полинома, x,y – интенсивности
           // исходного и преобразованного цветов соответственно
           colR = A + colR * (B + C * colR); // интенсивность цвета R
           colG = A + colG * (B + C * colG); // …цвета G
           colB =  A + colB * (B + C * colB); // …цвета B
           Image1->Canvas->Pixels[iX][iY] = RGB(colR, colG, colB);
           break;
  default: // другие значения rule пока не рассматриваются… ==============
           break;
 } // конец CASE

} // конец цикла по пикселам изображения (по iX,iY)
```

```
…
delete colImage; // отдали память системе
```

Как видно из вышеприведенного исходного кода, наиболее трудоемко функциональное преобразование интенсивностей цветов; причем на показанном примере функция преобразования (заданная коэффициентами A,B,C полинома) ед**и**на для всех трех цветов (но, естественно, может быть и различной).

Заметим, что в используемой при проведении работ программе Paint показанные на рис.2.2 и табл.2.2 варианты б)÷е) реализуются выбором соответствующего варианта подменю Edit→Преобразование яркости→*. Повторим, что в известном пакете PhotoShop д**о**ступ к (визуальной) настройке кривых преобразования возможен через меню Изображение→Настройка→Кривые…

Фильтрация часто применяется в составе других эффектов обработки изображения – например, ретуширования (‘разм**ы**тие’ ограниченно области изображения, указываемой курсором мыши и многих других).

При создании текстовых надписей необходимо определить параметры фонта (напомним, что полотно Canvas является атрибутом некоторого объекта – обычно это объект типа TImage):

```
Canvas->Font->Name = “MS Serif”; // название фонта
Canvas->Font->Size = 14; // размер оного
Canvas->Font->Color = clRed; // цвет символов
Canvas->Font->Style<<fsItalic<<fsUnderline; // наклонный подч**е**ркнутый
```

В некоторых случаях используется свойство Pitch, задающее шаг отрисовки символов; при этом fpDefault задает шаг по умолчанию (соответствующий описанию в свойстве Name), fpVariable – возможность изменения шага, fpFixed - постоянную ширину символов (если символы выбранного свойством Name фонта имеют переменную ширину, этот фонт будет заменен ближайшим по начертанию моноширинным фонтом):

```
Canvas->Font->Pitch = fpDefault; // питч
```

Для б**о**льшей гибкости настройки фонтов применяется настройка свойства Charset (имеет смысл, если выбранный свойством Name фонт включает несколько наборов символов):

```
Image1->Canvas->Font->Charset = SYMBOL_CHARSET; // стандартный
```

В большинстве случае свойства Pitch и Charset не нужны.

Реально для определения описанных параметров используется компонент FontDialog, позволяющий в привычном диалоге выбрать параметры фонта:

```
if (FontDialog1->Execute())
 Image1->Canvas->Font = FontDialog1->Font;
```

Определить высоту и ширину (в пикселах, в целях оценки возможности вывода в нужную позицию полотна) текстовой строки можно так:

```
char str[] = “Нет лучше кафедры ИТ-4”;
…
int y = Image1->Canvas->TextHeight(str);
int x = Image1->Canvas->TextWidth(str);
```

Соотношение между параметрами Size и Height таков**о** (значение Size принято оценивать в *пунктах*, причем обычно *пункт=дюйм/72≈0,353 мм*):

```
Font->Size = -Font->Height * 72 / Font->PixelsPerInch;
Font->Height = -Font.Size * Font->PixelsPerInch / 72;
```

Отрисовывается (горизонтально слева направо) строка текста на полотне с помощью метода TextOut (x,y – левая верхняя точки прямоугольной рамки, мысленно охватывающей текст):

```
Image1->Canvas->TextOut(x,y, “Это отрисовываемая строка”);
```

Отрисовка строки с отсеч**е**нием вне заданной прямоугольной области r осуществляется методом TextRect (x,y – аналогично TextOut):

```
TRect r;
…
r.Top = 10;
r.Left = 10;
r.Bottom = 100;
r.Right = 100;
Image1->Canvas->TextRect(r, 20, 20, "Привет, МГУПИ!");
```

Для вывода текста на прозрачный фон обычно используется следующий прием:

```
Canvas->Brush->Style = bsClear;
```

Для профессиональных пакетов обработки изображения обязательным является наличие эффектов геометрического преобразования (скручивание, эффект линзы и т.д.); причем возможно применение этих эффектов как ко всему, так и к выделенной части изображения. В общем случае геометриче-

ские преобразования являются результатом применения заданных функций преобразования:

$$\begin{cases} x_2 = \varphi_x(x_1, y_1), \\ y_2 = \varphi_y(x_1, y_1), \end{cases}$$

где x_1, y_1 - исходные координаты пикселов изображения,

x_2, y_2 - преобразованные координаты,

φ_x, φ_y - заданные функции преобразования.

Простейшими преобразованиями являются аффинные, при этом:

$$\begin{cases} x_2 = a_{xx} x_1 + a_{xy} y_1 + a_x, \\ y_2 = a_{yx} x_1 + a_{yy} y_1 + a_y, \end{cases}$$

где a_{ij} - постоянные коэффициенты.

Приведенные зависимости удобны для использования в векторной графике, применение их для растровой графики затруднен**о** по чисто практическим соображениям (т.к. координаты пикселов преобразованного изображения не совпадают с таков**ы**ми изображения исходного, прямое определение цвета преобразованных пикселей невозможно).

Выходом из положения является определение интенсивности цвета в некоторой точке изображения путем усреднения интенсивностей цвет**о**в точек-соседей (на исходном изображении). Для этого используют интерполяцию (обычно с применением сплайнов) величины интенсивности каждого из б**а**зовых цветов RGB в окрестности рассматриваемой точки. На рис.2.2 показаны 9 точек $i \in [i-1,i,i+1]$, $j \in [j-1,j,j+1]$, по которым строится интерполяционная поверхность интенсивности цвета, и значение интенсивности A=f(x,y) в заданной точке x,y. На рис.2.2 матрица параметров имеет размерность 3×3, но может иметь размер и 4×4 и 5×5 и больше.

Особенно удобно применять эту технологию при *масштабировании изображений* (увеличении или уменьшении числа пикселов по горизонтали и вертикали, причем коэффициенты масштабирования по горизонтали и вертикали необязательно совпадают). При масштабировании изображения проводится преобразование параметров функции X и Y в соответствие с заданными коэффициентами масштабирования и по значению интерполирующей функции в нужной точке определяется интенсивность цвета пиксела (для каждого цвета RGB независимо). В Windows операции преобразований (и

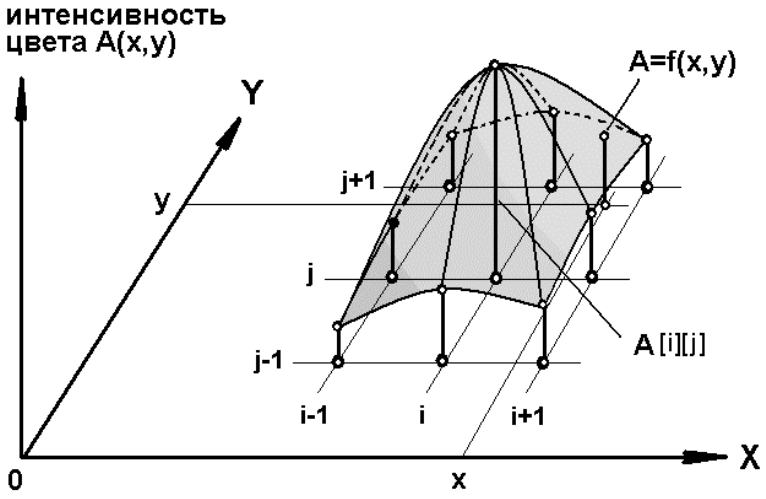


Рисунок 2.2 — Интерполяция функции интенсивности цвета A(x,y) по 9 точкам (матрица 3×3).

масштабирования в т.ч.) реализованы очень корректно (в состав Windows API включен**ы** эффективные функции сплайн-интерполяции).

Печать изображений (представляющая особую сложность для Windows-программирования) в системах Delphi/C++Builder легко осуществляется с использованием штатно входящего в системы программирования фирмы Borland Int. набора компонентов QuickReport (фирма QuSoft AS Fred Olsens gt. 1 N-0152 Oslo Norway, http://www.qusoft.no, макет будущей страницы проектируется в визуальном режиме аналогично внешнему виду форм, для вывода изображений используется компонент QRImage); выбор принтера осуществляется компонентом PrintDialog, настройка режимов печати - PrinterSetupDialog.

2.3 *Необходимое оборудование* – IBM PC-совместимая ЭВМ, предустановленная ОС Windows, пакет C++Builder версии выше 4; заготовка проекта Paint.bpr (может быть получена как http://pilger.mgapi.ru/metods/mm_src.zip).

2.4 *Порядок проведения работы*. Студент знакомится с общими сведениями по практике программирования приложений, направленных на обработку файлов изображений (при проверке уровня усвоения целесообразно использовать вопросы для самопроверки), анализирует исходные тексты заготовки проекта, добивается корректного функционирования программы, по требованию преподавателя выполняет расширяющую возможности программы модификацию конкретные задания.

Рекомендуется использовать изображения размерами стандарта не выше VGA (640×480 пиксел), причем для лучшего уяснения действия фильтров желательно иметь как крупные, так и мелкие детали на изображении.

Возможные задания по модификации программ:

- Расширить функциональность программы возможностью печати текста на экране под заданным углом к горизонтали и на заданном цвете фона.

- Реализовать дополнительную (заданную преподавателем) функцию преобразования интенсивности (яркости) цветов.
- Указать последовательность действий (с точки зрения программирования) для реализации перемещения части изображения по экранному полотну.

2.5 *Оформление отчета по работе.* Содержимое отчета должно отражать результаты самостоятельной работы над проектом (особенности предлагаемой программной реализации и предложения по расширению функциональности), по требованию преподавателя приводятся твердые копии (распечатки) внешнего вида интерфейса пользователя и в**ы**дачи программ. Фиксация имеющихся в заготовках проекта ошибок и возможностей оптимизации кода весьма приветствуются.

2.6 *Вопросы для самопроверки.*

- Что такое внеэкранный битовый **о**браз и почему его применение при операциях модификации цвета и иных преобразованиях повышает быстродействие программы?
- Какие пути повышения быстродействия операций модификации изображений известны?
- Каким образом реализуется перемещение ‘резиновой нити’ по экрану?
- В чем суть применения фильтров при модификации изображения? Какими свойствами обладают фильтры сглаживания? подчеркивания границ? эффекта гравировки?
- Какими свойствами должны обладать функции преобразования интенсивности цвета каждого пиксела?
- В чем заключается суть технологии сплайн-интерполяции при преобразованиях изображения?

3 Лабораторная работа 3**. Цифровая запись, редактирование и воспроизведения звука**

3.1 *Цель работы* – приобретение практических навыков в получении и цифровой обработке звука с использованием стандартного ПО.

3.2 *Теоретические основы*. Источником представляющих практический интерес звуковых колебаний являются микрофон, радио и теле- передачи, ресурсы сети InterNet, устройства хранения (CD, магнитная лента, flash) и синтеза звука.

Звук представляет собой механические колебания физической сред**ы** (воздуха, скорость распространения около 340 м/сек) частотой приблизительно $20 \div 20000$ Гц, современные системы обработки звука основаны на преобразовании этих колебаний в электрический сигнал, последующей его (ан**а**логовой или цифровой) обработки и вывода вновь в виде колебаний физической сред**ы**. Эффект стереофонии достигается временн**о**й разницей колебаний, легко улавливаемой благодаря наличию приблизительно 20-сантиметровой базы между приемниками аудиоинформации – ушами человека (разница по времени $0{,}2/340 \approx 6 \times 10^{-4}$ сек).

Преобразование аналогового сигнала в цифровую форму выполняет аналого-цифровой преобразователь (АЦП), служащий для дискретизации сигнала по времени (частота оцифровки) и квантования по уровню (собственно цифровое представление сигнала). Обычно в АЦП применяется технология преобразования с импульсно-кодовой модуляцией (PCM, *Pulse Code Modulation*). Временн**ы**е промежутки между моментами преобразования сигнала называют интервалами выборки (*Sampling Interval*); эта величина обратно пропорциональна частоте выборки, или сэмплингом (*Sampling Rate*). Амплитуда аналогового сигнала (*Sample Value*) при каждом преобразовании делится (*квантуется*) по уровню и кодируется в соответствующий параллельный цифровой код (*Digital Sample*), время преобразования аналогового сигнала в цифровой код именуется временем выборки (*Sampling Time*), рис. 3.1. Разреш**а**ющая способность АЦП - наименьшее значение ан**а**логового сигнала, которое приводит к изменению цифрового кода. Например, если АЦП выдает 8-разрядный код, разрешающая способность равна $1/(2^8)=1/256$ от максимальной амплитуды аналогового сигнала (около 0,4% в относительных единицах), 16-разрядный АЦП имеет точность представления сигнала не хуже $1/(2^{16})=1/65536$ (0,0015%). Увеличение разрядности АЦП приводит к росту его динамического диапазона (каждый дополнительный бит соответствует увеличению приблизительно на 6 дБ). 8-разрядное преобразование обеспечивает динамический диапазон 48 дБ (качество кассетного магнитофона), 12-

разрядное – 72 дБ (качественный катушечный магнитофон), 16-разрядное – 96 дБ (качество аудио компакт-диска).

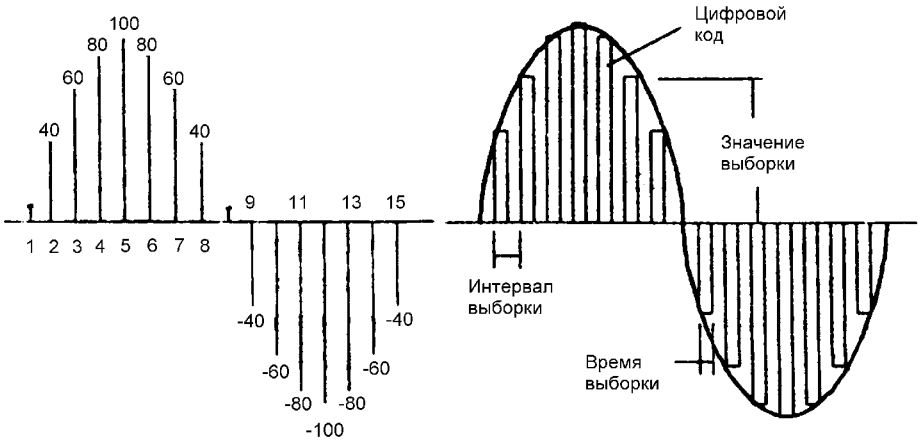


Рисунок 3.1 — Характеристики процесса преобразования между аналоговым и цифровым сигналами

С практической точки зрения качество аудиовоспроизведения определяется скорость передачи данных (*bitrate,* битрейт). Качество, неотличимое большинством рядовых слушателей от качества CD, достигается при скорости передачи $112 \div 128$ Кбайт в секунду; при этом коэффициент MPEG-сжатия достигает 14:1 относительно исходного объема. Специалисты обычно требуют скорости передачи $256 \div 320$ Кбайт/сек (это легко достижимо для CD-проигрывателя, но для многих отечественных InterNet -линий недоступно).

При цифровой обработке звуковой информации на ЭВМ преобразованию подвергается именно цифровое (а не аналоговое) представление сигнала. В данной работе используются формат аудиоинформации WAV (метаформат, при котором способ кодирования аудиосигнала может быть любым – напр., PCM без сжатия) и сильно компрессированный формат MP3 (MPEG-Audio Layer-3, алгоритм сжатия учитывает *психоакустическую модель* слуха человека).

При выполнении работы используется программа Nero Wave Editor из комплекта Ahead Nero (фирма Ahead Software AG, http://www.nero.com), обладающей значительными возможностями по редактированию и преобразованию аудиформатов при удобном пользовательском интерфейсе. В цифровом канале легко достигаются многие аудиоэффекты, труднодостижимые при ан**а**логовой обработке; напр., реверберация (эффект многократного эха), подавление акустической зав**я**зки выхода усилителя звуковой частоты на его вход (методом снижения звукового уровня в определенном частотном диапазоне) и др.

Б**о**льшую часть главного окна интерфейса программы занимает область графического представления амплитуды звукового сигнала от времени (ам-

плитудная характеристика, АХ), в центре нижней части представлены окна мгновенной амплитудно-частотной характеристики (АЧХ) сигнала (для каждого стереоканала отдельно), рис. 3.2. Управление программой стандартное – через главное меню и ‘горячие’ кнопки в верхней части главного окна. Через вариант главного меню Edit доступны функции копирования всего или части (выбор нужного фрагмента осуществляется проведением мыши с нажатой левой кнопкой вдоль графика АХ) аудиофайла и конвертации его формата, вариант View позволяет управлять формой изображения (напр., вместо графика АХ - Wave Display могут быть выбраны Spectrogram Display или Wavelet Display). Nero Wave Editor дает возможность также осуществлять запись аудиоклипов (Audio → Record).

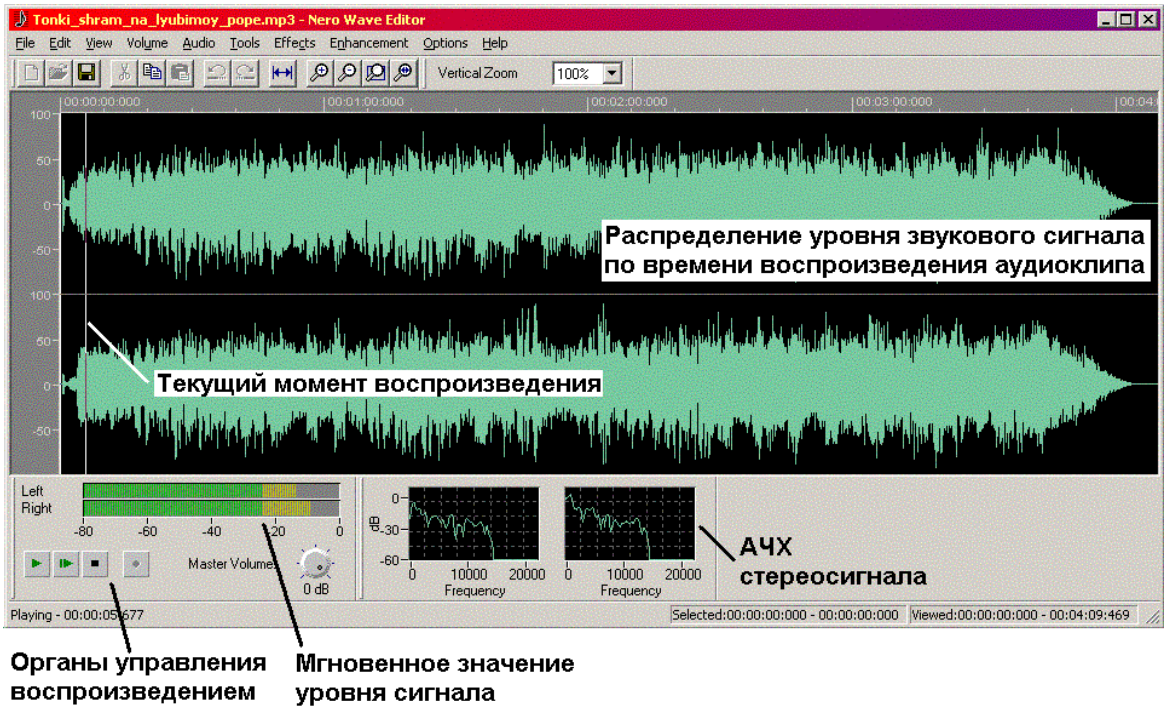


Рисунок 3.2 — Главное окно пользовательского интерфейса программы Nero Wave.

Указанный потенциал программы Nero Wave Editor дает возможность гибкого редактирования аудиофайлов:

- Изменения темпа (скорости) воспроизведения (Tools → Time Correction).
- Заданного изменения огибающей уровня звука (Volume → FadeIn, Volume → FadeOut).
- Редактирования АЧХ аудиоклипа (Tools → Equaliser, Transpose, Dynamic Processor).
- Внесения стандартных эффектов в звучание (Effects → Delay, Flanger, Chorus, Reverb).

- Тонкое регулирование качества звука и применение шумоподавления (Enhancement → High Frequence Correction, Noise Reduction).

3.3 *Необходимое оборудование* – IBM PC-совместимая ЭВМ с комплектом аудиоаппаратуры (микрофон, устройства аудио воспроизведения – желательно наличие высококачественных наушников), предустановленные ОС Windows и программа Nero Wave Editor.

3.4 *Порядок проведения работы*. Студент знакомится с практикой получения аудиофайлов различными методами (включая оцифровку записи с линии и/или микрофона), производит заданное преподавателем редактирование файлов (задаваемое обычно *субъективными* выражениями) и проверяет (путем индивидуального прослушивания через наушники) наличие и качество полученного аудиоэффекта.

Типовые задания на редактирование аудиоданных:

- Изменение воспроизведения темпа аудиоклипа (получение ‘голоса Буратино’ или ‘утрированного баса’). Провести подобные преобразования методом Tools → Transpose (использовать настройки ‘Helium Speech’ и ‘Monster Speech’).
- Установка АЧХ аудиоклипа методом использования эквалайзера (Tools → Equalizer). При работе учесть, что регулировка ширины полосы каждого из 6 каналов эквалайзера осуществляется регуляторами Bandwidths (в октавах) в окне Equalizer.
- Настройка динамических свойств воспроизведения (зависимость выходного уровня сигнала от уровня входного, Tools → Dynamic Processor).
- Добавление эффекта однократного эха (Tools → Delay), ‘дрожания’ (Tools → Flanger), хорового исполнения (Tools → Chorus), реверберации (Tools → Reverb).
- Понижение уровня шумов аудиоклипа (Enhancement → Noise Reduction).

3.5 *Оформление отчета по работе.* Отчет должно содержать технические параметры компонентов аудиосистемы и обработанных аудиофайлов (включая частоту семплинга, разрядность оцифровки, используемый битрейт), описание произведенных действий по редактированию файлов и полученного эффекта.

Полученные в результате обработки аудиофайлы предъявляются для прослушивания и оценки выполненной работы.

3.6 *Вопросы для самопроверки.*

- Назвать источники звуковой информации (в порядке личного мнения по снижению распространенности).
- Что представляет собой звук? Каков диапазон частот, воспринимаемых человеческим ухом? Какова природа стереоэффекта?
- Что такое амплитудная (АХ) и амплитудно-частотная характеристики (АЧХ), в каких координатах они строятся?
- Какие параметры оцифровки звука применяются? Что такое ‘битрейт’ и какому качеству звучания соответствуют стандартные величины битрейта?
- Что такое эффект реверберации и каким образом он может быть достигнут?
- Что такое ‘звуковая зав**я**зка’ и каковы возможные пути борьбы с ней?

4 Лабораторная работа 4. **Получение и цифровая обработка фотоизображений**

4.1 *Цель работы* – приобретение практических навыков в получении фотоизображений и их цифровой обработке с использованием стандартного ПО.

4.2 *Теоретические основы.* Источником цифровых изображений являются сканеры, цифровые фотоаппараты и графические редакторы. В настоящее время сканеры и цифровые фотоаппараты подключаются к ПЭВМ посредством интерфейса USB (http://www.usb.org). USB (*Universe Serial Bus*, 1996÷2000 г.) является последовательной шиной и обеспечивает пропускную способность 1,5 (*LS – Low Speed*) и 12 Мбит/сек (*FS – Full Speed*); в версии USB 2.0 определена скорость в 480 Мбит/сек (*HS – High Speed*), причем в одной системе могут одновременно работать устройства со всеми тремя скоростями.

Преимуществами USB является простое подключение до 127 периферийных устройств, возможность коммутации, подключения/отключения устройств при работающей системе, поддержка технологии PnP (*Plug and Play*, автоматическое конфигурирование в момент включения/выключения). Информация по шине USB передается посредством пакетов (специальным образом организованным цепочками бит).

USB осуществляет обмен данными между HOST-компьютером и множеством периферийных устройств. Устройства (*devices*) могут являться *хабами*, *функциональными устройствами* или *их комбинацией*. Устройство-хаб (*hub*) всего лишь обеспечивает дополнительные точки подключения устройств к шине, функциональной устройство (*function*) предоставляет системе дополнительные функциональные возможности (цифровые джойстики, цифровой интерфейс с акустическими колонками и др.), комбинированные устройства (*compound device*) представляют хаб с подключенными к нему несколькими устройствами. Работой всей системы управляет хост-контроллер (*host controller*), представляющий аппаратно-программную подсистему хост-компьютера.

Физическая топология USB представляется многоярусной звездой - вершина – хост-контроллер, к каждому порту которого подключаются периферийные устройства или промежуточный хаб, причем всего допустимо до 5 уровней каскадирования (при длине кабеля каждого уровня до 5 м имеем всего до 25 м). USB допускает питание устройств током до 0,5 А (в момент подключения – не более 0,1 А) при напряжении 5 В; поддерживается режим удаленного пробуждения (*remote wakeup*), при котором устройство может подать сигнал (находящемуся в приостановленном состоянии) хост-компьютеру с целью его активизации. Типы разъемов USB приведены на рис.4.1.

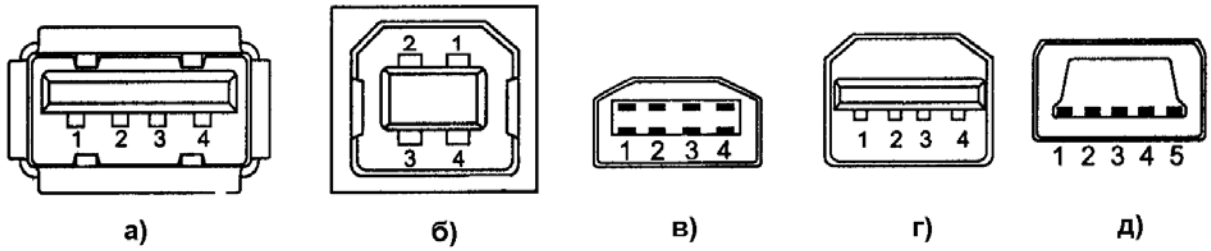
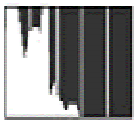


Рисунок 4.1 — Гнезда USB: а) – типа ‘A’, б) – типа ‘B’ стандартное, в,г,д) – миниатюрные типа ‘B’

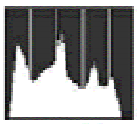
В качестве источника изображений используется *цифровая фотокамера* (ЦФК*)*. Важной характеристикой ЦФК является *разреше**н**ие* (измеряется в числе пиксел, составляющих изображение); современные бытовые фотокамеры имеют разрешение до $5 \div 7$ миллионов пиксел (*мегапиксел*, Мпкс), профессиональные – до 10 и более. Разрешение определяется параметрами используемой в данном ЦФК *полупроводниковой матрицы*, служащей для преобразования воспринимаемого изображения в пикселы (для матриц обычно применяется технология ПЗС – *прибор с зар**я**довой связью*). Информация с матрицы считывается и обрабатывается встроенным микропроцессором (МП), демонстрируется на расположенном с задней стороны ЦФК миниатюрном жидкокристаллическом дисплее и в виде файлов (при необходимости компрессированных стандартными методами – обычно JPG) сохраняется на сменном ПЗУ (обычно Flash-карта), для хранения может использоваться миниатюрный жесткий диск. Обычно видеофайлы в дальнейшем переносятся на ПЭВМ для окончательной обработки; в последнее время практикуется прямое подключение ЦФУ к малогабаритным цветным принтерам.

Вторая важная характеристика – величина *оптического* (достигающегося путем изменения *фокусного расстояния объектива*) увеличения изображения (явление трансфок**а**ции, оптический *zoom*); для хороших моделей величина трансфокации достигает $10 \div 12$. Важно отметить, что при трансфокации четкость изображения практически не снижается (разрешение остается прежним), цифровое же увеличение всегда снижает четкость изображения (т.к. снижается составляющее изображение число пикселов). Величина *экспозиции* (определяемая *диафрагмой* и *выдержкой*) и наводка на резкость во всех современных ЦФК осуществляются автоматически (режим AUTO) путем использования известных алгоритмов встроенного МП, хотя допускаются предпочтения (приоритет выдержки Tv или диафрагмы Av).

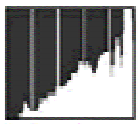
Встроенный МП во время съемки вычисляет и визуализирует гистограмму яркости изображения (см. рис. справа ниже); стремиться следует к наиболее равномерной гистограмме.



Излишне темное изображение

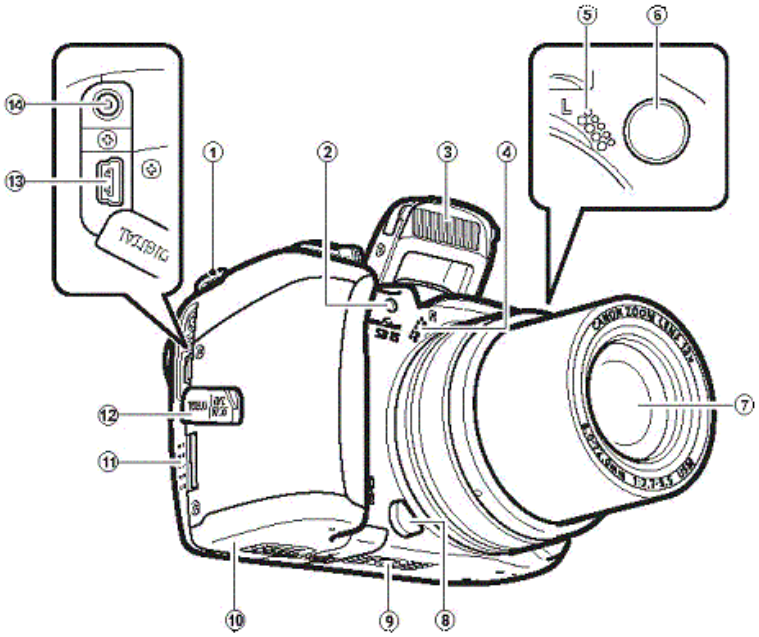


Сбалансированное изображение



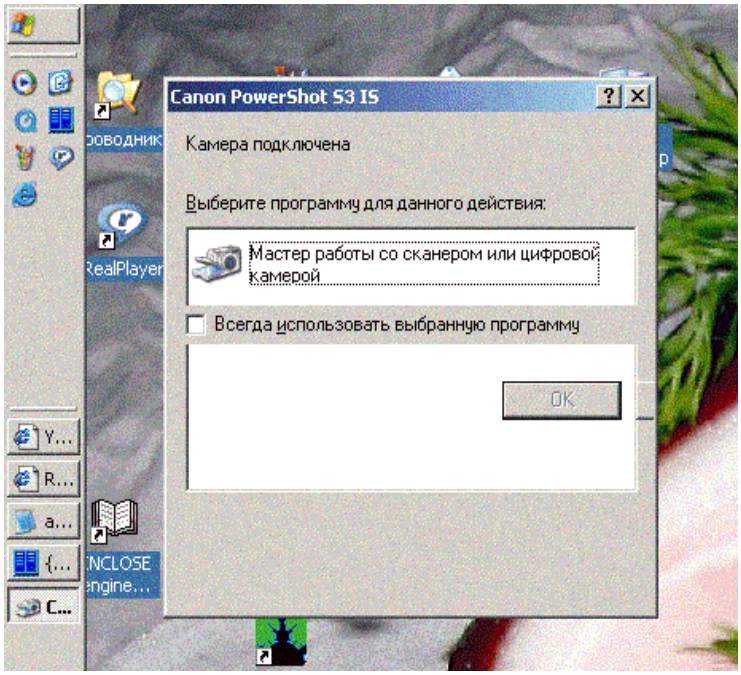
Чрезмерно яркое изображение

На рис. слева ниже показан внешний вид одной из лучших непрофессиональных ЦФК Power Shot S3 IS фирмы Canon (здесь 3-фотовспышка, 4, 5 – правый и левый микрофоны, 7 – объектив, 11 – контейнер для Flash-карты типа SD – *Secure Digital*, 10 – отсек батарей питания (4 шт. типа AA), 13 – розетка разъема USB 2.0, 14 – разъем подключения внешнего питания; имеется разъем коаксиального стерео/видео кабеля).



ЦФК S3 IS имеет разрешение 6 Мпкс, позволяет получать фотоизображения максимального размера 2816×2112 пикселов (объем видеофайла 2,7 Мбайт при наилучшем качестве сохранения S, т.е. около 370 кадров на каждый Гбайт Flash-карты) и видеоизображения VGA-формата (640×480 пиксел при 30 кадрах/сек, видеопоток 2 Мбайт/сек, максимальный размер видеоклипа – до 1 Гбайт или до 1 часа непрерывной записи).

В состав комплекта ЦФК входит специальное ПО для переноса полученных данных на ПЭВМ и манипуляций с ними. Однако Windows'XP позволяет использовать технологию 'Мастеров…' для работы с внешними устройствами, при этом часто можно обойтись без поставляемого совместно с ЦФК специализированного ПО. 'Мастер' автоматически отслеживает подключение внешнего устройства и предлагает возможности (ограниченные, обычно только перенос данных с ЦФК



на ПЭВМ; см. рис. справа). Окончательную обработку данных следует проводить с использованием специализированных растровых графических редакторов (от простейшего MS Paint до ухищренных уровня Adobe Photoshop).

В данной работе используется пакет Paint Shop Pro фирмы Jask Software (http://www.jasc.com), обладающий не меньшими по сравнению с Adobe Photoshop функциональными возможностями и, по мнению автора данной работы, более удобным пользовательским интерфейсом (рис. 4.2).

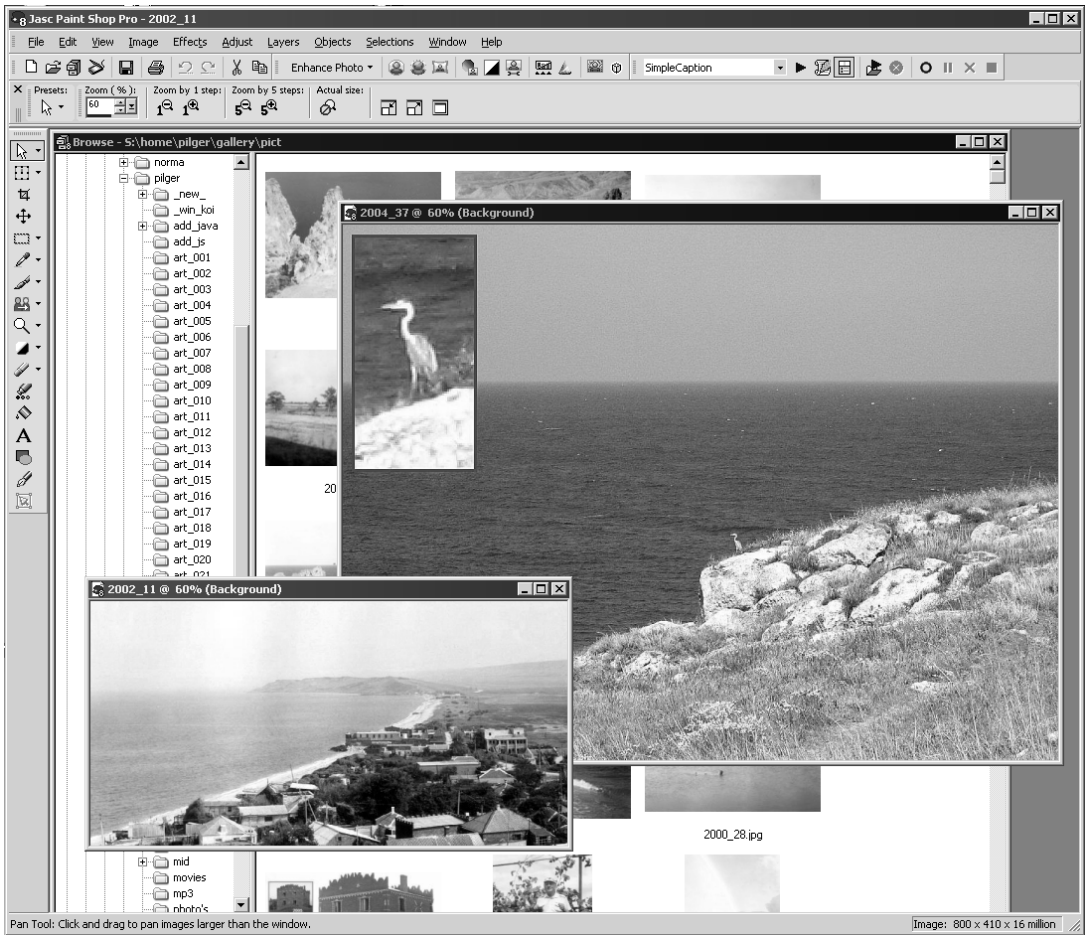


Рисунок 4.2 – Внешний вид главного окна графического пакета Paint Shop Pro 8.

Для лучшей ориентации в возможностях Paint Shop Pro полезно воспользоваться системой встроенной помощи (управление через главное меню посредством Help → Help Topics, Help → Contex Help, Help → Learning Center).

Стандартные действия коррекции изображений:

- Установка *цветового баланса* (Enchange Photo → Automatic Color Balance). Цветовой баланс управляется ползунком : вправо – ‘холоднее’ (голубой оттенок) и влево – ‘горячее’ (оранжевой оттенок) или прямой установкой цветовой температуры (нейтраль соответствует $6000^{O}C$ – эффективная

цветовая температура фотосферы Солнца, точка ‘Лунный свет’ - 6500^{O}C, ‘Флуоресцентная лампа’ - 4200 ^{O}C).

- Установка *контраста* (Enchange Photo → Automatic Contrast Enhancement). Контраст задается дискретно – через Bias (см. работу 2 этого руководства), Strength (*интенсивность*) и Appearance (*внешнее проявление*).
- Установка *насыщенности цвета* (Enchange Photo → Automatic Saturation Enhancement). Насыщенность устанавливается дискретно - через Bias и Strength.
- Установка *прозрачности* (Enchange Photo → Clarify). Прозрачность задается дискретно (диапазон $1 \div 5$) через Strenght off effect (*интенсивность эффекта*).

Широкие возможности доступны через вариант Effects главного меню, вариант Effect Browser дает возможность доступа к применению разнообразных инструментов редактирования изображений; Effects → User Defined позволяет применить к изображению матрицу фильта, подобную описанной выше в работе 2.

Полезной особенностью пакета Paint Shop Pro является возможность записи проведенных действий по редактированию изображений (*сценария* действий) и в дальнейшем неоднократное применение этого сценария (*script*). Имеются также предустановленные сценарии (например, автоматическая всесторонняя коррекция изображения, Enchange Photo → One Step Photo Fix).

Типовые задания для выполнения студентами:

- Исправить некорректное фотоизображение – повысить контраст, сбалансировать цвета.
- Создать изображение в выбранном гипертрофированном цвете (цветах).
- Создать из сделанной при дневном свете фотографии художественный ‘вид в лунную ночь’.
- Получить художественный ‘эффект раскаленной пустыни’.
- Из заданной фотографии получить часто используемое в рекламе ‘чеканное’ изображение (*Emboss*) и ‘оконтуривания’ (*Trace Contour*).
- Применить цифровое масштабирование к выбранной части фотоизображения, проанализировать влияние масштабирования на резкость изображения.
- Применить эффекты ‘размытия’ (*Dilate, Blur, Soften*) и ‘повышения резкости’ (*Sharpen*).

4.3 *Необходимое оборудование* – IBM PC-совместимая ЭВМ, оснащенная USB-входами и драйверами USB (для Windows’XP установка специализированного для конкретного цифрового фотоаппарата ПО необязательна), циф-

ровая камера с кабелем USB. Для редактирования фотоизображений следует установить пакет Paint Shop Pro версии не ниже 8.0.

4.4 *Порядок проведения работы*. Студент знакомится с общими сведениями по практике получении фотоизображений и их цифровой обработке, под руководством преподавателя производит фотосъемку, подключает фотокамеру к ПЭВМ, переносит файлы фотоизображений на ПЭВМ и проводит их обработку в соответствие с требованиями преподавателя или же собственно сформулированными на основе желаемого эффекта или с целью компенсации полученных при съемке недостатков (не полностью корректная цветовая гамма, яркость, план изображения).

4.5 *Оформление отчета по работе.* Отчет должно содержать технические параметры фотокамеры, кабеля связи и используемой ЭВМ. В процессе переноса файлов изображений и их обработки фиксируются параметры полученного файла (формат изображения, размер изображения в пикселах, занимаемое файлом место на диске и др.).

Полученные в результате оцифровки и редактированные изображения предъявляются для просмотра и оценки выполненной работы.

4.6 *Вопросы для самопроверки.*

- Какими количественными характеристиками определяется цифровое фотоизображение?
- Какие интерфейсы обычно используются для передачи файлов изображений?
- Что такое цветовая палитра и с какой целью используется это понятие?
- С какой целью вычисляется гистограмма изображения? Для чего используется гистограмма яркости? Гистограмма отдельных цветовых составляющих?
- Что такое ‘цветовой баланс’ изображения? С какой целью введено понятие *цветовой температуры*?

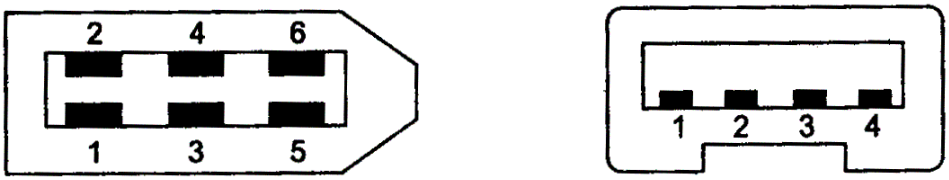
5 Лабораторная работа 5. **Получение и цифровая обработка видеоданных**

5.1 *Цель работы* – приобретение практических навыков в получении видеоизображений и их цифровой обработке с использованием стандартного ПО.

5.2 *Теоретические основы.* Источником видеоизображений изображений являются видеокамеры (включая цифровые фотоаппараты с функцией видео), телевизионный сигнал и специализированные программы синтезирования (не очень реалистичного в настоящее время) видео. Источник видео в большинстве случаев подключаются к ПЭВМ посредством интерфейса IEEE 1394 (*FireWire, iLink, Digital Link, MultiMedia Connection)*. IEEE 1394 (*High Perfomance Serial Bus*, 1995 г., стандарт основан на используемой в *Apple Computer* шине FireWire) является последовательной шиной и обеспечивает пропускную способность (округленно) 100, 200 и 400 Мбит/сек (планируется реализации скоростей 800, 1600 и 3200 Мбит/сек).

Преимуществами IEEE 1394 является подключение до 63 периферийных устройств без применения дополнительной аппаратуры (хабов), поддержка технологии PnP, свободная топология присоединения устройств (до 27 разъемов разветвления на одном устройстве), питание устройств током до 1,5 А при напряжении 8÷40 В. Длина сегмента кабеля до 4,5 м, суммарная длина – до 72 м, между любой парой узлов не может быть более 16 кабельных сегментов; в последующих реализациях предполагается автоматическое исключение петель физической топологии и использование оптоволокна.

Типы разъемов IEEE 1394 приведены на рис.5.1.



а) б)

Рисунок 5.1 — Разъемы IEEE 1394: а) – 6-ти контактное гнездо, б) – 4-х контактное гнездо (только для оконечных устройств)

Распространенным ПО для оцифровки данных с видеокамеры и последующего монтажа видеофильмов является Adobe Premiere (фирма Adobe Systems Inc., http://adobe.com), Ulead Media Studio (UleadSystems, http://ulead.com), Pinnacle Studio (Pinnacle Systems, Inc., http://pinnaclesys.com), WinDVD Creator (InterVideo, http://intervideo.com) и др.

Первым этапом является получение видеоданных в виде единого файла на жестком диске ПЭВМ. Заметим, что даже при использовании видеокамер с

цифровой записью этот этап необходим (ибо формат цифровой записи на видеокамере обычно не совпадает со стандартами видеофайлов в ПЭВМ).

В табл. 5.1 приведены основные форматы аудиовидеороликов, они используют 24-х битовой цвет и (в большинстве случаев) частоту смены кадров 25 кадров/сек [4]. Следует обратить внимание на необходимость иметь достаточно пространства на диске – для качества DVD это около 2,6 Гбайт на 1 час видео (440 Мбайт на 10 мин, 44 Мбайт на 1 мин). Конечно, бессмысленно оцифровывать сигнал с бытов**о**й ан**а**логовой видеокамеры в стандарте DVD – для этого лучше всего подойдет VHS или Video CD.

Таблица 5.1. – Основные стандарты видеоданных.

Общепринятое наименование	*Видеоданные*			*Аудиоданные*		
	Размер изображения, пиксел	*Формат видеофайла*	*Поток данных, кбит/сек*	*Формат аудиофайла*	*частота оцифровки, kHz*	*Поток данных, кбит/сек*
Video CD	352×288	MPEG-1	1 150	MPEG стерео	44,1	224
VHS	352×288	-.-.-	2 400	-.-.-	44,1	224
SVCD	480×576	MPEG-2	2 400	-.-.-	44,1	224
DVD	720×576	-.-.-	6 000	-.-.-	48	224
Micro MV	720×576	-.-.-	10 000	-.-.-	48	256

Главное окно удобной для создания видео программы Pinnacle Studio ver.9.0 приведено на рис.5.2 (стандарт DVD - захват в режиме 720×576 пиксел, выходной файл формата MPEG-2). Действие по преобразованию видеоданных из представления видеокамеры в формат дискового файла ПЭВМ инициируется выбором ‘Захват’ главного меню, параметры захваченного файла определяются в меню ‘Настройки’, во время оцифровки текущее изображение можно просматривать в окне видеоизображения.

Для последующего монтажа видеофильма необходимо разбить единый файл на сцены (части видеопотока, представляющие собой *законченные смыслов**о**вые блоки*). Практически всегда это действие является предварительным – при монтаже видеофильма сцены будут серьезным образом перекомпон**о**ваны в соответствие с художественным представлением режиссера.

В большинстве современных пакетов для создания видеофильмов реализован режим автоматического разбиения на сцены (новая сцена начинается в момент резкого изменения содержания видеопотока); часто этого вполне достаточно.

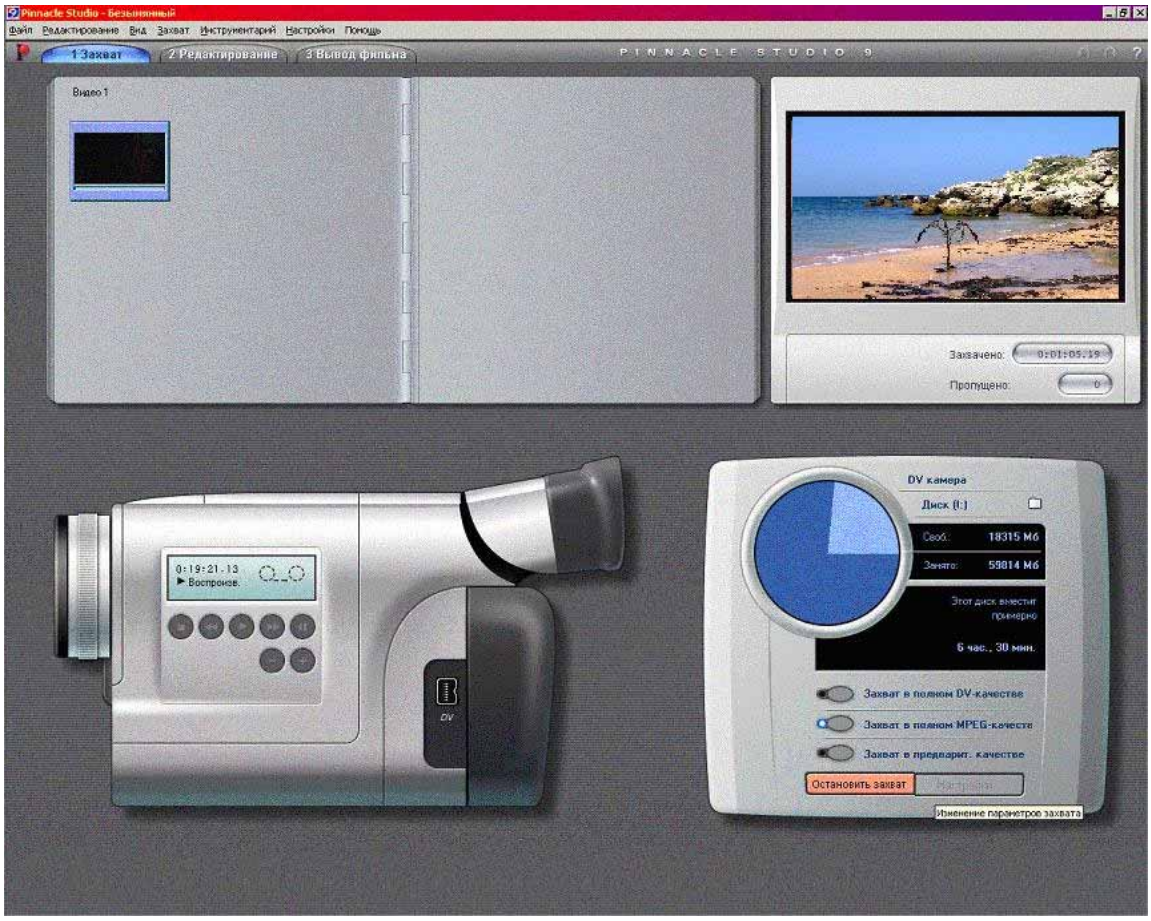
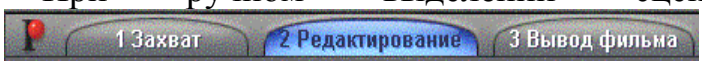


Рисунок 5.2 — Главное окно программы Pinnacle Studio 9.0 в режиме захвата информации с видеокамеры в DVD-совмест**и**мом режиме.

При ручном выдел**е**нии сцен (режим ‘Редактирование’ 

главного меню) появляется возможность предварительного редактирования снятого материала (в основном удаления явно неудачных кадров). На рис.5.3 приведена копия главного окна программы Pinnacle Studio ver.9.0 в режиме ‘линия времени’ (средняя кнопка три**а**ды в правой части главного окна). В этом режиме видеоданные представлены в виде непрерывной горизонтальной ленты (синхронная аудиодорожка находится ниже), поддерживаются функции разделения видеклипа в заданный момент времени и вырезания определенной таким образом части клипа (правая кнопка мыши).

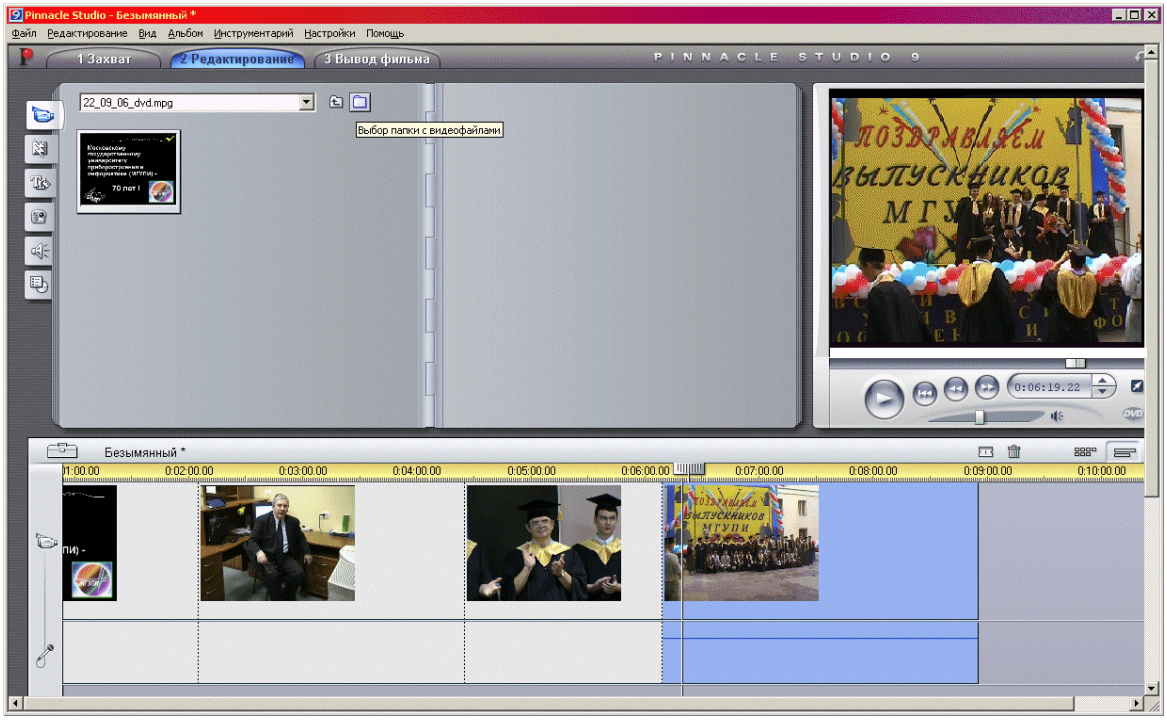


Рисунок 5.3 — Редактирование видеоданных в режиме ‘Линия времени’.

На рис.5.4 приведена копия главного окна в режиме ‘Сценарий’ (левая кнопка триады). Этот режим удобен, напр., для задания переходов между сценами (выбираются из имеющегося списка, см. левую верхнюю часть рис.5.4). Режим ‘Монтажный лист’ удобен при создании и редактирования списка видео и аудиоклипов.

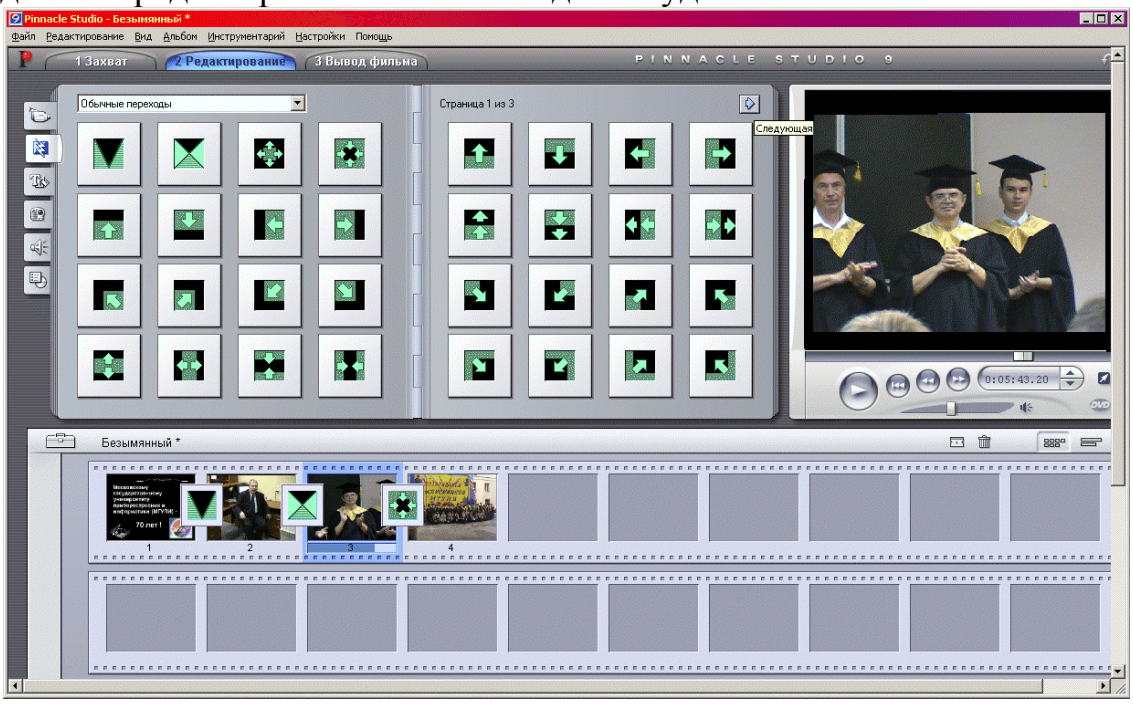


Рисунок 5.4 — Редактирование видеоданных в режиме ‘Сценарий’.

Цифровая обработка видеоданных включает:

- Изменение цветового фона, контраста, добавление визуальных эффектов (при необходимости, с целью коррекции исходно снятого видеоматериала или достижения определенных художественных эффектов).

  В Pinnacle Studio определены эффекты времени – изменение скорости, стробоскоп, цветовые видеоэффекты – цветокоррекция (изменение яркости, контраста, оттенка, насыщенности), стилевые эффекты – размытие, рельеф, мозаика, 'старое кино', витраж, занимательные эффекты – рассеяние в объективе, шум, капля воды и др.
- Добавление (заранее определенных) плавных переходов между сценами и/или титров.

  В Pinnacle Studio определено несколько десятков переходов (таких как 'растворение', 'выталкивание', 'слайд', 'спираль' и др.). Также существуют как заранее определенные титры (с заголовками, пригодными для многих случаев жизни), так и титры с функциональностью меню (предусмотрен прямой переход к заданной части видеофильма).
- Озвучивание фильма; при этом могут использоваться как записанные микрофоном камеры звуки, так звуки из специально подготовленных аудиофайлов, возможно их наложение (микширование).

  В Pinnacle Studio предусмотрено использование в качестве синхронного звукового сопровождения использование как внешних аудиофайлов, так и запись с микрофона и богатая библиотека стандартных звуков; аудиоэффекты позволяют выполнить как подавление шумов, так и имитацию искажений, эффект караоке, эхо и др.

Конечным этапом является создание готового видеоролика (режим 'Вывод файла'); возможны режимы записи на магнитную ленту, видеодиск, на сайт http://StudioOnline.com производителя в формате MPEG или вывод фильма в AVI-формате на жесткий диск и др.

5.3 *Необходимое оборудование* – IBM PC-совместимая ЭВМ с процессором P5 или Amdahl с частотой не менее 1,5÷2 GHz, размером ОП не менее 512 Мбайт и жестким диском объемом не менее 80÷120 Gб, предустановленные ОС, набор драйверов MPEG-2, пакет Pinnacle Studio версии не ниже 9.0.

5.4 *Порядок проведения работы*. Студент знакомится с общими сведениями по практике получении видеоизображений и их цифровой обработке, под руководством преподавателя подключает видеокамеру к ПЭВМ (видеоматериал обычно снимается на камеру заранее), выбирает подходящий видеоформат, оцифровывает данные и проводит их начальную обработку ('вырезает' явно неудачные фрагменты, разбивает видеоклип на сцены, сохраняет

сцены в отдельных файлах для последующего их использования для создания полноценного видеофильма).

5.5 *Оформление отчета по работе.* Отчет должно содержать технические параметры видеокамеры, кабеля связи и используемой ЭВМ. В процессе оцифровки фиксируются параметры полученного видеоролика (формат видео и аудиопотока, занимаемое клипом место на диске и др.).

Полученные в результате оцифровки и предварительно редактированные сцены предъявляются для просмотра и оценки выполненной работы.

5.6 *Вопросы для самопроверки.*

- На каком основном принципе основано восприятие человеком видеоинформации?
- Какими количественными характеристиками определяется качество цифрового видео?
- Назвать методы снижения потока данных при работе с видео. В чем их преимущества и недостатки?
- Какие интерфейсы обычно используются для передачи файлов изображений?
- Что такое качество VHS и DVD?

6 Лабораторная работа 6. **Создание видеофильмов**

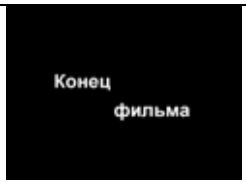
6.1 *Цель работы* – приобретение практических навыков в создании цифровых видеофильмов любительского уровня с помощью цифровых компьютерных технологий.

6.2 *Теоретические основы.* Создание художественного произведения (в том числе видеофильма) является творческим процессом, трудно поддающемся алгоритмизации [4]; поэтому здесь рассматривается в основном техническая сторона создания видеофильма уровня ‘домашнего видео’.

Исходными являются видеоклипы, снятые и оцифрованные согласно имеющегося (в виде твердой копии или в сознании режиссера) *сценария*. Непосредственно при создании видеофильма первым этапом является разработка *монтажного листа* (списка сцен и титров с указанием желаемой длительности, особенностей монтажа и др.). В таблице 6.1 приведен монтажный лист, реально использовавшийся при создании видеофильма о вручении дипломов выпускникам кафедры ИТ-4 МГУПИ 03 июля 2003 г. (титры подготавливались заранее в виде BMP-файлов, звуковые файлы брал**и**сь из различных источников).

Таблица 6.1.— Таблица-монтажный лист видеофильма_____

№ п/п	*Вводной титр (включая визуальные эффекты), видеоклип*	*Содержимое титра или клипа*
1	• 001.bmp • Вступление, о чем фильм | Информация об авторе • aa_001.wav (звук щелканья клавиш и писк динамика ЭВМ) • Переход - наплыв	
	• class.mpg (41 сек)	
2	• Переход наплывом (2 сек)	
	• Перед выпускниками выступает ректор МГУПИ • Естественная запись с микрофона камеры с переходом к a11_1.wav (классика) • mih_01.mpg (9,2 сек), mih_02 (2,05 мин), mih_03.mpg (11,2 сек)	

3	• 004.bmp • Переход – всплывание из глубины	
	• Приступаем к вручению дипломов. Акцентировать внимание - *несколько человек не явились за своими дипломами*! • Естественная запись с микрофона камеры с переходом к a12_3.wav • prize_01.mpg (1,33 мин), prize_02 (1,03 сек), prize_03 (1,02 сек)	
4	• 009.bmp • '…Довьерьяй, но провьерьяй !' • Переход – всплывание из глубины	
	• Естественная запись с микрофона камеры с переходом к a13_1.wav (очень медленный блюз) • test_01.mpg (9 сек), test_02.mpg (21 сек), test_03.mpg (8,2 сек)	
5	• ta_001.bmp • Изображение для вставки перед концом фильма	
	• Естественная запись с микрофона камеры • end_01.mpg (9,24 сек)	
…	…	…
22	• 000.bmp • Конец фильма… • aa_001.wav (щелканье клавиш и писк динамика ЭВМ) • Переход – плавное проявление	
23	• 000_.bmp • …Еще не совсем конец фильма • Звук – тот же (без прерывания) • Переход – такой же	
24	• 000__.bmp • Настоящий конец фильма • Звук – плавное микширование к 'Оде к радости' • Переход – 'всплывающее солнце'	

6.3 *Необходимое оборудование* – IBM PC-совместимая ЭВМ с процессором P5 или Amdahl с частотой не менее $1{,}5 \div 2$ GHz, размером ОП не менее 512 Мбайт и жестким диском объемом не менее $80 \div 120$ Gб, предустановленные ОС Windows, набор драйверов MPEG-2, пакет Pinnacle Studio версии не менее 9.0.

6.4 *Порядок проведения работы*. Студент обдумывает сценарий будущего фильма, проводит видеосьемку, переносит полученный материал на ПЭВМ, монтирует видеоряд, добавляет звук, записывает готовый материал на DVD или CD (последнее возможно при длине видеофильма формата MPEG-2 до ≈ 10 мин при разрешении 720×576 пиксел).

Возможные тематические варианты создания видеофильмов:

- Семейные праздники.
- Встреча старых друзей.
- Отдых на даче.
- Летний отдых.

6.5 *Оформление отчета по работе.* Отчет должен включать разработанный сценарий видеофильма и содержать технические параметры оборудования, с помощью которого создавался видеофильм. Смонтированный фильм обсуждается совместно со студенческой группой, причем обязательно фиксируются субъективные и объективные достоинства и недостатки работы.

СПИСОК ЛИТЕРАТУРЫ

1. Попов В.Б. Основы информационных и телекоммуникационных технологий. Мультимедиа. –М.: Финансы и статистика, 2007. –336 с.
2. Леонтьев Б.Н. Мультимедия Microsoft Windows XP. -М.: Новый издательский дом, 2005. –190 с.
3. Порев В.Н. Компьютерная графика (учебное пособие). -С.Птб.: BHV, 2004. -234 с.
4. Чепмен Н. Цифровые технологии мультимедиа. –М.: Вильямс, 2006. –624 с.
5. Баканов В.М. Программирование мультимедиа-систем (конспект лекций). –М.: МГАПИ, 2004. -120 с.

Учебное издание.

Баканов Валерий Михайлович

ПРОГРАММИРОВАНИЕ МУЛЬТИМЕДИА-СИСТЕМ

Учебно-методическое пособие по
выполнению лабораторных работ

Подписано в печать ___._____ 2008 г.
Формат 60 × 84 1/16.
Объем 3 п.л. Тираж 100 экз. Заказ _____

Отпечатано в типографии Московского государственного
университета приборостроения и информатики.
107996, Москва, ул. Стромынка, 20.