Цикл презентаций «ООП на Delphi» посвящен объектно – ориентированному программированию с использованием одной из самых распространенных систем быстрой разработки приложений – Delphi

Используя данный учебный курс, можно самостоятельно овладеть основами объектно – ориентированного программирования на Delphi. Для расширения Ваших знаний к курсу приложен ряд учебных пособий и справочников по Delphi

Цикл содержит 13 презентаций:

ООП на Delphi – 1: Знакомство с системой программирования Borland Delphi. Объекты (компоненты) и их свойства и методы

ООП на Delphi – 2: Первая программа на Delphi, сохранение и компиляция

ООП на Delphi – 3: Программное изменение свойств объектов

ООП на Delphi – 4: Условия в Delphi. Создание простого теста

ООП на Delphi – 5: Элементы ввода и вывода информации. Обработка исключений

ООП на Delphi – 6: Заставка программы и элемент таймер

ООП на Delphi – 7: Программируем свою игрушку

ООП на Delphi – 8: Меню программы, панель статуса, диалоги

ООП на Delphi – 9: Создаем свой текстовый редактор

ООП на Delphi – 10: Базы данных на Delphi

ООП на Delphi – 11: Калькулятор на Delphi. Обработка исключительных ситуаций

ООП на Delphi – 12: Создаем тестирующую систему

ООП на Delphi – 13: Графика на Delphi

Delphi использует язык программирования Объект Паскаль, поэтому лучше сначала изучить обычный Паскаль и поработать в ТурбоПаскале, а затем и переходить к Delphi – перейти будет очень просто, т.к синтаксис языка остается неизменным.

Изучение ООП на Delphi желательно проводить в старших профильных классах – количество часов, отводимое на информатику там вполне достаточно для освоения основ ООП на Delphi

Объектно – ориентированное программирование на

Borland®

DELPHI - 13

DELPHI - 13

На этом уроке:

Мы познакомимся с компонентами для работы с графикой и создадим свой графический редактор

Вопросы:

1. Введение в графику
2. Создаем свой графический редактор

Borland®

Введение в графику

Для работы с графикой в Delphi есть много средств. Рассмотрим некоторые из них.

У многих объектов Delphi есть свойство **Canvas**, что в переводе означает **холст**. Если у объекта имеется такое свойство, то это значит, что на нем можно рисовать. (Заметим, что работая в любой программе, да и в самой операционной системе Windows, мы видим только графику – все кнопочки, окна, инструменты … это всего лишь нарисованные, например на форме объекты (вернее их картинки) и при нажатии на кнопку она перерисовывается, показываясь нажатой)

Для рисования в Delphi есть два основных инструмента:

- **Pen** (карандаш) – для рисования линий
- **Brush** (кисть) – для раскраски объектов

У каждого из этих инструментов есть дополнительные свойства (например цвет карандаша, толщина линии, тип линии)

Рассмотрим рисование линий, прямоугольников, эллипсов:

1. Линия

Сначала откроем событие для формы – OnPaint и запишем там код:

```
procedure TForm1.FormPaint(Sender: TObject);
begin
canvas.Pen.Color:=rgb(255,0,0);
canvas.Pen.Width:=5;
canvas.MoveTo(10,20);
canvas.LineTo(200,20);
```

Устанавливаем цвет карандаша - красный

Устанавливаем толщину карандаша в 5 пикселей

Устанавливаем карандаш в точку с координатами 10,20

Проводим линию до точки с координатами 200,20

Рассмотрим рисование линий, прямоугольников, эллипсов:

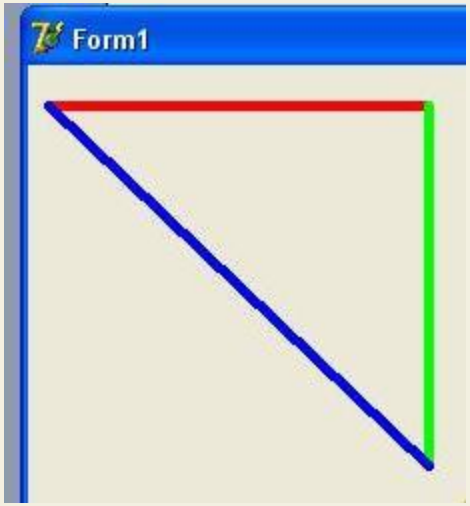
1. Линия

```
procedure TForm1.FormPaint(Sender: TObject);
begin
canvas.Pen.Color:=rgb(255,0,0);
canvas.Pen.Width:=5;
canvas.MoveTo(10,20);
canvas.LineTo(200,20);
canvas.Pen.Color:=rgb(0,255,0);
canvas.MoveTo(200,20);
canvas.LineTo(200,200);
canvas.Pen.Color:=rgb(0,0,255);
canvas.MoveTo(200,200);
canvas.LineTo(10,20);
```

Проводим **зеленую** линию

Проводим **синюю** линию

В результате мы получаем на форме следующую картинку:



Рассмотрим рисование линий, прямоугольников, эллипсов:

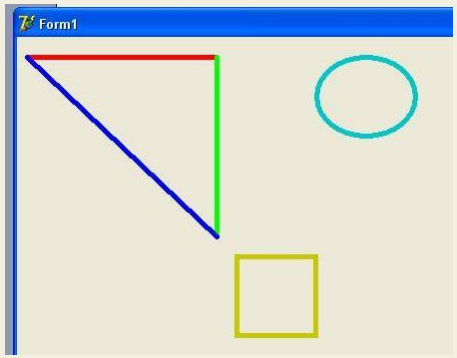
2. Прямоугольник, эллипс

```
canvas.LineTo(200,200);
canvas.Pen.Color:=rgb(0,0,255);
canvas.MoveTo(200,200);
canvas.LineTo(10,20);
canvas.Pen.Color:=rgb(200,200,0);
canvas.Rectangle(220,220,300,300);
canvas.Pen.Color:=rgb(0,200,200);
canvas.Ellipse(300,20,400,100);
```

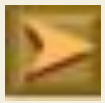
Выбираем произвольный цвет и рисуем **прямоугольник**

Выбираем произвольный цвет и рисуем **эллипс**

В результате мы получаем на форме следующую картинку:

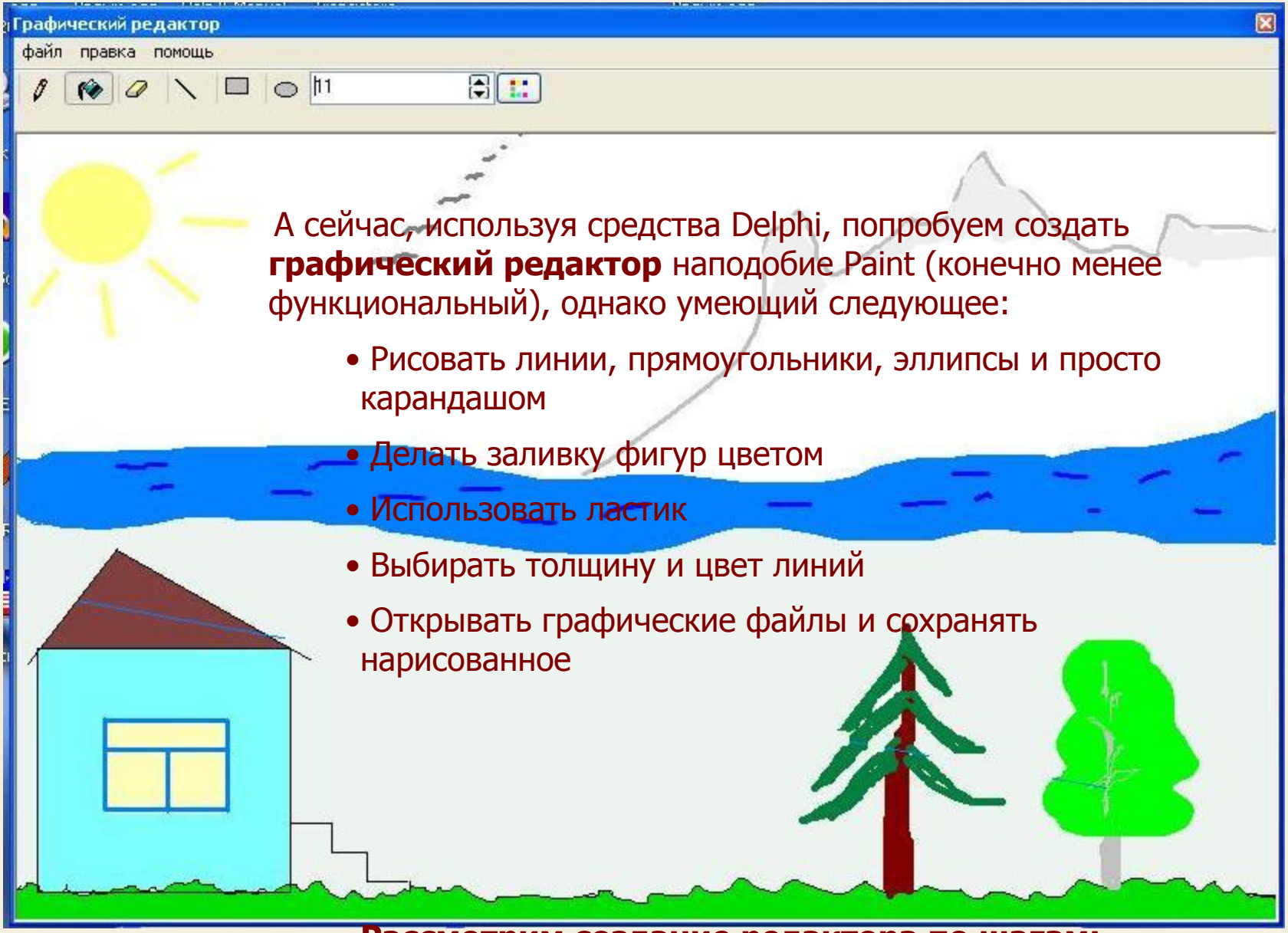


Посмотреть ->



Borland®

Создаем свой графический редактор



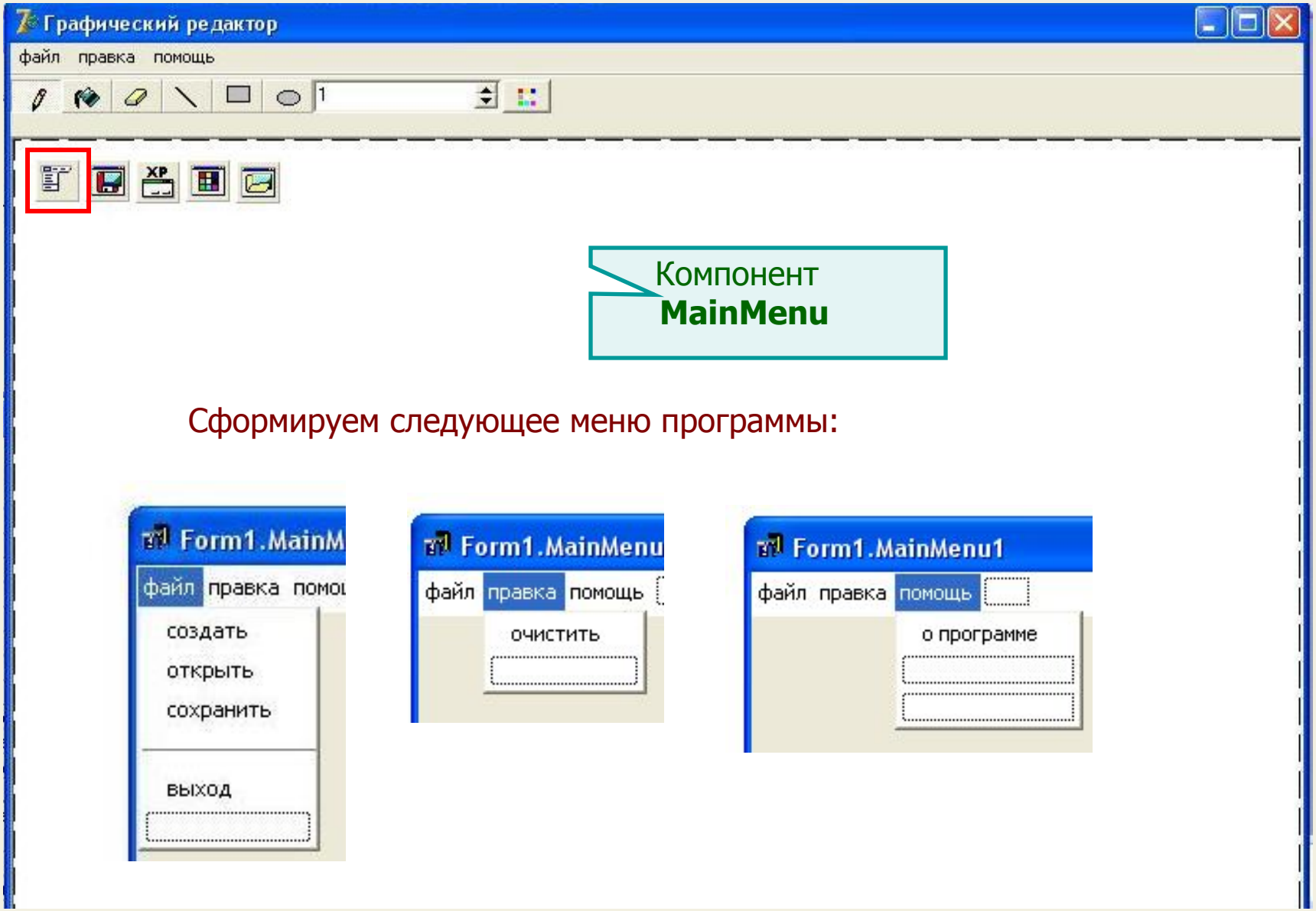
А сейчас, используя средства Delphi, попробуем создать **графический редактор** наподобие Paint (конечно менее функциональный), однако умеющий следующее:

- Рисовать линии, прямоугольники, эллипсы и просто карандашом
- Делать заливку фигур цветом
- Использовать ластик
- Выбирать толщину и цвет линий
- Открывать графические файлы и сохранять нарисованное

Рассмотрим создание редактора по шагам:

ШАГ 1

Откроем Delphi разместим на форме следующие элементы:

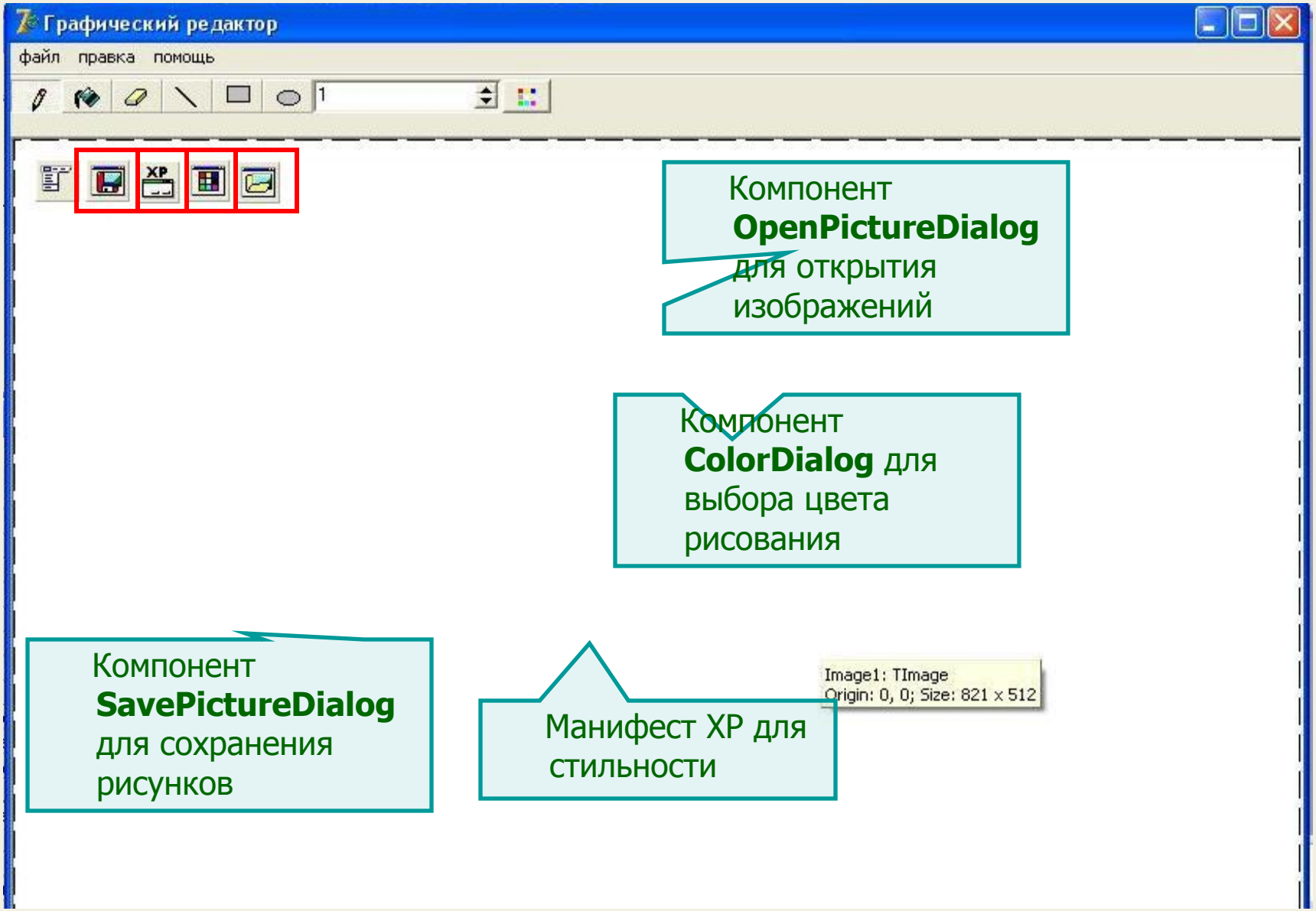


Компонент **MainMenu**

Сформируем следующее меню программы:

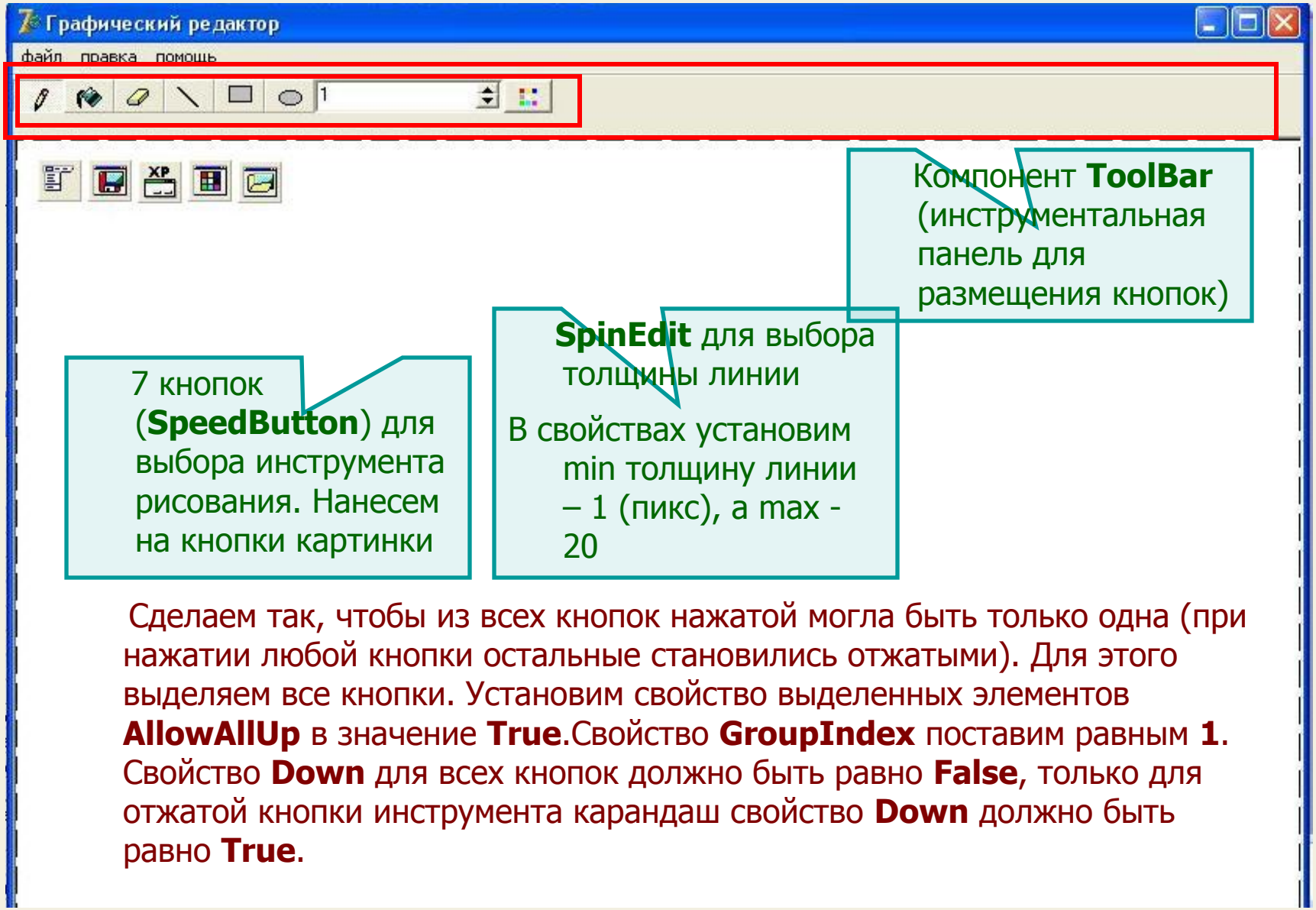
ШАГ 1

Откроем Delphi разместим на форме следующие элементы:



ШАГ 1

Откроем Delphi разместим на форме следующие элементы:



Компонент **ToolBar** (инструментальная панель для размещения кнопок)

7 кнопок (**SpeedButton**) для выбора инструмента рисования. Нанесем на кнопки картинки

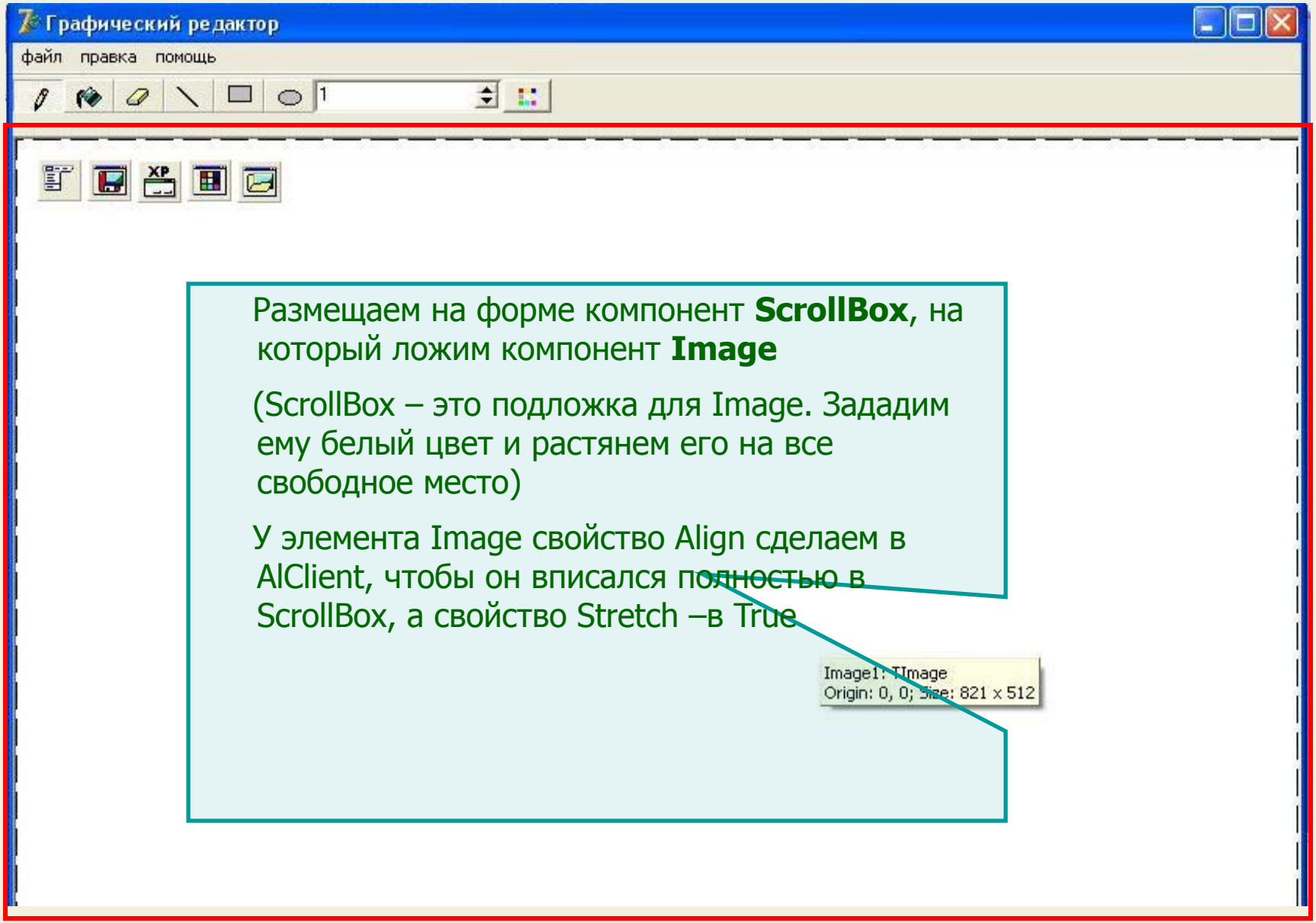
SpinEdit для выбора толщины линии

В свойствах установим min толщину линии – 1 (пикс), а max - 20

Сделаем так, чтобы из всех кнопок нажатой могла быть только одна (при нажатии любой кнопки остальные становились отжатыми). Для этого выделяем все кнопки. Установим свойство выделенных элементов **AllowAllUp** в значение **True**.Свойство **GroupIndex** поставим равным **1**. Свойство **Down** для всех кнопок должно быть равно **False**, только для отжатой кнопки инструмента карандаш свойство **Down** должно быть равно **True**.

ШАГ 1

Откроем Delphi разместим на форме следующие элементы:



ШАГ 2

Займемся описанием событий:

Сначала давайте подумаем, какие события возникают, когда мы рисуем. Ясно, что почти все мы делаем мышкой. Например для рисования линии мы ставим курсор в какое-то место, нажимаем левую кнопку мыши, перемещаем мышь до нужного места и отпускаем ее.

Таким образом в процессе рисования мы используем **3 основных события:**

1. **OnMouseDown** (нажатие кнопки мыши)
2. **On MouseMove** (перемещение мыши)
3. **OnMouseUp** (отпускание кнопки мыши)

Рисование карандашом

При нажатии на кнопку (**OnMouseDown)** карандаш должен встать на точку с координатами курсора , при перемещении мыши с нажатой клавишей (**On MouseMove**) должна рисоваться линия, при отпускании кнопки мыши (**OnMouseUp)** должно прекратиться рисование.

Поэтому для рисования карандашом необходим код в каждом из событий для мыши **OnMouseDown, On MouseMove**, **OnMouseUp.**

ШАГ 2

Рассмотрим код рисования карандашом:

(Нажатие кнопки мыши - MouseDown)

```
procedure TForm1.Image1MouseDown(Sender: TO
  Shift: TShiftState; X, Y: Integer);
begin
  r:=true;
if  speedbutton1.Down then
Image1.Canvas.MoveTo(x, y);
```

Введем переменную логического типа **r** – это своеобразное разрешение/запрет на рисование. Если **r** – true, то рисовать можно, иначе нельзя.

Зачем это?

Мы можем перемещать мышь (MouseMove) с нажатой или отпущенной клавишей: в первом случае должно рисоваться (r=true), а во втором не должно (r=false)

При нажатии кнопки мыши даем добро на рисование

Если нажата кнопка рисования карандашом, то на Image переходим к точке с координатами курсора

ШАГ 2

Рассмотрим код рисования карандашом:

(Перемещение мыши - MouseMove)

```
procedure TForm1.Image1MouseMove(Sender: TObject;
  Y: Integer);
begin
if r then
begin
  if  speedbutton1.Down then
  begin
  image1.Canvas.Pen.Color:=colordialog1.Color;
  Image1.Canvas.LineTo(x, y);
   end;
```

Проверяем, разрешено ли рисовать

Если рисовать можно и нажата кнопка рисования карандашом, то устанавливаем цвет карандаша, соответствующий выбранному в ColorDialoge и проводим линию

ШАГ 2

Рассмотрим код рисования карандашом:

(Отпускание мыши - MouseUp)

```
procedure TForm1.Image1MouseUp(Sender: TObje
  Shift: TShiftState; X, Y: Integer);
begin
    if  speedbutton1.Down then
    Image1.Canvas.LineTo(x, y);
    r:=false;
```

Проводим линию до координаты отпускания мыши и устанавливаем запрет на рисование

ШАГ 3

Стирание с помощью ластика

Что мы делаем при стирании ластиком? На самом-то деле мы ничего не стираем, а рисуем точно так же, как и карандашом, только белым цветом. Получается эффект стирания.

Это значит, что при нажатии кнопки ластика ему присваивается цвет белый, толщина линии берется из значения SpinEdita, а код рисования точно такой же.

ШАГ 4

Рисование прямоугольника, эллипса, заливка фигур

Не будем вдаваться в подробности рисования прямоугольника, эллипса и заливки фигур – здесь используются соответствующие методы Canvas: Rectangle, Ellipse, FloodFill – посмотрите внимательно код, соответствующий этим операциям в примере, приложенном к презентации

ШАГ 5

Выбор цвета

```
procedure TForm1.SpeedButton7Click(Sender: TObject);
begin
  colordialog1.Execute;
  image1.Canvas.Pen.Color:= colordialog1.Color;
  image1.Canvas.Brush.Color:= colordialog1.Color;

end;
```

При нажатию на кнопку цвета запускается ColorDialog, где мы выбираем нужный цвет и присваиваем его карандашу и кисти

(см. предыдущие уроки о диалогах)

ШАГ 6

Очистка холста

Конечно, можно очистить рисунок с помощью ластика, но это долго, поэтому при выборе меню Правка -> Очистить происходит очистка всего холста.

И это опять обман, как и в случае с ластиком. На самом деле мы ничего не очищаем, а рисуем белый прямоугольник размерами чуть больше размера поля рисования – создается эффект очистки

```
procedure TForm1.N8Click(Sender: TObject);
begin
  image1.Canvas.Pen.Width:=1;
  image1.Canvas.Brush.Color:=clwhite;
  image1.Canvas.Rectangle(-5,-5,image1.Width+5,image1.Height+5);
end;
```

Устанавливаем толщину линии = 1, цвет линии = белый и рисуем прямоугольник с размерами на 5 пикселей больше размеров поля для рисования (Image)

ШАГ 7

Меню Файл -> Создать

```
procedure TForm1.N2Click(Sender: TObject);
begin
  image1.Canvas.Brush.Color:=clwhite;
  image1.Canvas.Rectangle(-5,-5,image1.Width+5,image1.Height+5);
  Form1.Caption:=(' Новый файл '+' – Графический редактор');
  image1.Canvas.Pen.Color:=clblack;
end;
```

Цвет кисти делаем белым, рисуем прямоугольник чуть больше Image, меняем заголовок формы и устанавливаем начальный цвет карандаша - черный

ШАГ 8

Меню Файл -> Открыть

```
procedure TForm1.N3Click(Sender: TObject);
begin
   if openpicturedialog1.Execute then
      If OpenpictureDialog1.Filename<>'' Then
   begin
       Image1.Picture.Bitmap.LoadFromFile(OpenPictureDialog1.FileName);
       Form1.Caption:=(OpenpictureDialog1.Filename+' – Графический редактор');
       end;
```

Диалоги рассмотрены нами ранее довольно подробно, поэтому комментарии излишни

ШАГ 9

Меню Файл -> Сохранить

```
procedure TForm1.N4Click(Sender: TObject);
begin
 if savepicturedialog1.Execute then
   If savepictureDialog1.Filename<>'' Then
   begin
   Image1.Picture.Bitmap.SaveToFile(savePictureDialog1.FileName);
   Form1.Caption:=(savepictureDialog1.Filename+' – Графический редактор');
   end;
 end;
```

Тоже без комментариев

ШАГ 10

Меню Файл -> Выход – метод Close

ШАГ 11

Меню Помощь -> О программе

Здесь мы должны создать форму «О программе», познакомить с ней форму редактора и открыть методом ShowModal

На этом и остановимся. Давайте запустим наш редактор и порисуем.

Порисовать -> 

Итак, мы создали свой графический редактор, где использовали графические возможности Delphi, и не только графические – нам пригодились знания из всех прошлых уроков Конечно это очень простой редактор с элементарными функциями. Однако его разработка позволила нам понять суть работы с графикой в Delphi. Ну а дальше можно наращивать его возможности, добавлять новые функции – для этого надо читать литературу по Delphi, разбираться и пробовать.

Мы же на этом закончим наш урок

ИТОГИ УРОКА:

На этом уроке мы научились работать с графикой в Delphi и создали свой графический редактор

На этом цикл презентаций о программировании в среде Delphi закончен

Домнин Константин Михайлович
E – mail: kdomnin@list.ru
2006 год.