Графика

GDI+

Важные пространства имен

System.Drawing	Основные типы для вывода графики, а также тип Graphics
System.Drawing.Drawing2D	Выполнение сложных операций с двумерной графикой
System.Drawing.Imaging	Типы, позволяющие напрямую работать с графическими изображениями
System.Drawing.Printing	Работы с принтером
System.Drawing.Text	Работа с системными шрифтами

Типы System.Drawing (I)

Bitmap	Инкапсулирует файл изображения и определяет набор методов различных операций
Brush, Brushes, SolidBrush, SystemBrushes, TextureBrushes	Объекты-кисти. Brush – базовый абстрактный класс
Color, SystemColors, ColorTranslator	Набор статических полей для настройки цвета
Font, FontFamily	Характеристики шрифта. FontFamily – набор шрифтов, относящихся к одному семейству
Graphics	Важный класс определяет область рисования и набор методов для вывода текста и геометрических фигур
Icon, SystemIcons	Классы для работы с пользовательскими и системными значками

Типы System.Drawing (II)

Image, ImageAnimator	Image – абстрактный класс, производит Bitmap, Icon, Cursor. ImageAnimator – показ серии изображений через интервал времени
Pen, Pens, SystemPens	Классы, при помощи которых рисуются все линии. Определен ряд статических свойств.
Point, PointF	Работа с координатами точки (int, float)
Rectangle, RectangleF	Работа с прямоугольными областями (int, float)
Size, SizeF	Работы с размерами (ширина, высота) (int, float)
StringFormat	Используется для форматирования текста
Region	Определяет область, занятую геометрической фигурой

Перечисленения System.Drawing

ContnetntAlignment	Определяет расположение содержимого в области ввода (слева, справа, по центру)
FontStyle	Свойства шрифта (полужирный, курсив)
GraphicsUnit	Единицы измерения графического элемента
KnownColor	Дружественные имена системных цветов
StringAlignment	Выравнивание текстовых строк
StringFormatFlags	Форматирование текстовых строк
StringTrimming	Отсечение текстовых строк
StringUnit	Единицы измерения для строки текста

Тип Point (PointF)

```
namespace DrawingUtilTypes
{
using System;
using System.Drawing;  // Это пространство имен нужно для работы с типами GDI+

public class UtilTypes
{
  public static int Main(string[] args)
  {
    Point pt = new Point(100, 72); // Создаем точку, а затем смещаем ее

    System.Console.WriteLine(pt);
    pt.Offset(20, 20);
    System.Console.WriteLine(pt);

    Point pt2 = pt; // Применяем перегруженные операторы Point

    if (pt = = pt2)Console.WriteLine("Points are the same");
    else       Console.WriteLine("Different points");

    pt2.X = 4000; // Меняем значение координаты X для pt2

    // А теперь выводим информацию о координатах каждой точки:
    Console.WriteLine("First point: {0}", pt.ToString());
    Console.WriteLine("Second point: {0}", pt2.ToString());
    return 0;
  }
 }
}
```

Члены класса Point (PointF)

+, –	Перегруженные операторы
==, !=	
X, Y	Свойства устанавливают и получают координаты
IsEmpty	true для точки (0, 0)
Offset()	Производит смещение точки относительно текущей позиции

Члены класса Rectangle (RectangleF)

==, !=	Перегруженные операторы
X, Y	Определяют координаты верхнего левого угла
Inflate(), Intersect(), Union	Статические методы позволяют увеличивать размеры прямоугольника и создавать новые путем пересечения о объединения
Top, Left, Bottom, Right	Свойства устанавливают измерения прямоугольника
Height, Width	Определяют высоту и ширину
Contains()	Позволяет определить, попадает ли точка с заданными координатами внутрь прямоугольника

Работа метода Contains()

```
public static int Main(string[] args)
{
   ...
   Rectangle r1 = new Rectangle(0, 0, 100, 100);
   Point pt3 = new Point(101, 101);

   if (r1.Contains(pt3))
    Console.WriteLine("Point is within the rect!");
   else
    Console.WriteLine("Point is not within the rect!");

   // Теперь помещаем точку внутрь области прямоугольника
   pt3.X = 50;
   pt3.Y = 30;

   if (r1.Contains(pt3))
    Console.WriteLine("Point is within the rect!");
   else
    Console.WriteLine("Point is not within the rect!");

   return 0;
}
```

Члены класса Size (SizeF)

+	Перегруженные операторы
–	
==	
!=	
Height, Width	Определяют высоту и ширину

Методы класса Region

Complement	Дополняет объект Region другими графическими объектами, которые не пересекаются с исходным Region
Exclude	Исключает область, занимаемую другим графическим объектом из объекта Region
GetBounds	Возвращает объект класса RectangleF, представляющий треугольник в который помещается объект Region
Intersect	Пересекает два Region
IsEmpty MakeEmpty	Определить/установить нулевой размер области
IsInfinite MakeInfinite	Определить/установить бесконечный размер области
Transform	Преобразует область объекта Region на основе передаваемого объекта Matrix
Translate	Сдвигает координаты объекта Region на указанную величину
Union	Объединяет указанный объект с другим графическим объектом
Xor	Объединяет указанный объект с другим, исключая общие области

Сеансы вывода графики (1)

```
public class MainForm : Form
{
  public MainForm()
  {
   CenterToScreen();
   this.Text = "Basic Paint Form";
  }

  public static void Main(string[] args)
  {
   Application.Run(new MainForm());
  }

  protected override void OnPaint(PaintEventArgs e)
  {
   Graphics g = e.Graphics;
   g.DrawString("HelloGDI+", new Font("Times New Roman", 20),
    new SolidBrush(Color.Black), 0, 0);
  }
}
```

Сеансы вывода графики (вариант 2)

```
public class MainForm : Form
{
  public MainForm()
  {
   ...
   // Добавляем обработчик события
        this.Paint +=
           new System.Windows.Forms.PaintEventHandler(MainForm_Paint);
  }

  // Обратите внимание на сигнатуру обработчика событий
  public void MainForm_Paint( object sender, PaintEventArgs e )
  {
   Graphics g = e.Graphics;
    ...
  }

  public static void Main(string[] args)
  {
   Application.Run(new MainForm());
  }
}
```

Как сделать клиентскую область вашего приложения недействительной

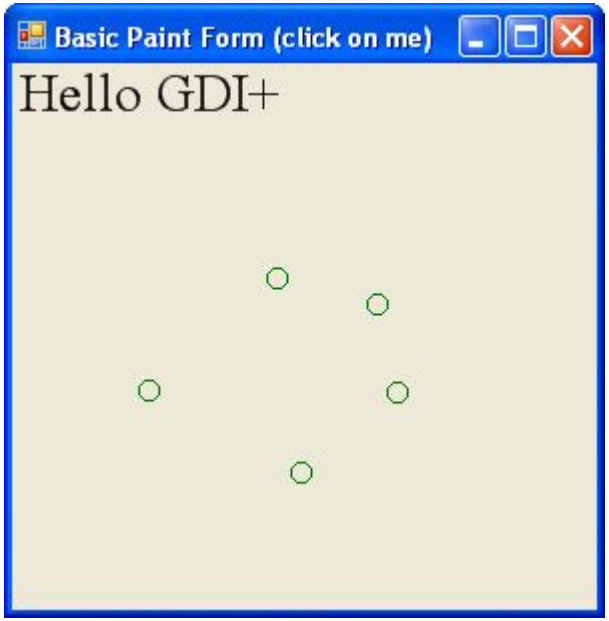
```
public class MainForm : Form
{
...
   private void MainForm_Paint(object sender, PaintEventArgs e)
   {
    Graphics g = g.Graphics;
    // Логика для вывода изображения
     ...
   }

   private void GetNewBitmap()
   {
    // Выводим окно диалога и получаем выбранное изображение
   ...
   // А теперь перерисовываем клиентскую часть формы
   Invalidate();
   }
}
```

Указание области перерисовки

```
 private void UpdateUpperArea()
{
     Rectangle myRect = new Rectangle(0,0,75,150);
Invalidate(myRect);
 }
```

Вывод графики вне события Paint

```
private void NainForm_MouseDown(object sender, MouseEventArgs e)
{
   // Получаем объект Graphics
   Graphics g = Graphics.FromHwnd(this.Handle);

         // По щелчку мыши рисуем кружок
   g.DrawEllipse(new Pen(Color.Green), e.X, e.Y, 10, 10);

}
```



«Правильный» вывод графики

```
public class MainForm : System.Windows.Forms.Form
{
  // Коллекция для хранения координат всех кружков
  private ArrayList myPts = new ArrayList();
  ...

  private void NainForm_MouseDown(object sender, MouseEventArgs e)
  {
                                    бавляем координаты указателя при щелчке
                                    м Invalidate()
                                     e.Y));
                                    int(object s
                                    ", new Font
                                    r,Black), 0
    g.DrawEllipse(new Pen(Color.Green), p.X, p.Y, 10, 10);
  }
}
```

Возможности класса Graphics

- **FromHdc(), FromHwnd(), FromImage()** – статические методы, обеспечивают возможность получения объекта Graphics из элемента управления или изображения
- **Clear()** – заполняет объект Graphics выбранным цветом, удаляя предыдущее содержимое
- **MeasureString()** – возвращает структуру Size, представляющую границы блока текста

Методы рисования класса Graphics

- **DrawArc(), DrawBezier(), DrawBeziers(), DrawCurve(), DrawEllipse(), DrawIcon(), DrawLine(), DrawLines(), DrawPie(), DrawPath(), DrawRectangle(), DrawRectangles(), DrawString()** – все это методы вывода изображений

- **FillEllipse(), FillPath(), FillPie(), FillPolygon(), FillRectangle()** – заполнение внутренних областей объектов

Свойства класса Graphics

- **Clip, Graphics, clipBounds, VisibleClipBounds, IsClipEmpty, IsVisibleClipEmpty** – позволяют настроить параметры отсечения для объекта
- **Transform** – для проведения преобразований координат
- **PageUnit, PageScale, DpiX, DpiY** – задают единицы измерения
- **SmoothingMode, PixelOffsetMode, TextRenderingHint** – задают плавность пере-ходов для геометрических объектов и текста
- **CompositingMode, CompositingQuality** – будет ли графический объект выводиться над фоном или будет происходить мешение с фоном
- **InterpolationMode** – определяет интерполяцию между соседними точками

Система координат по умолчанию

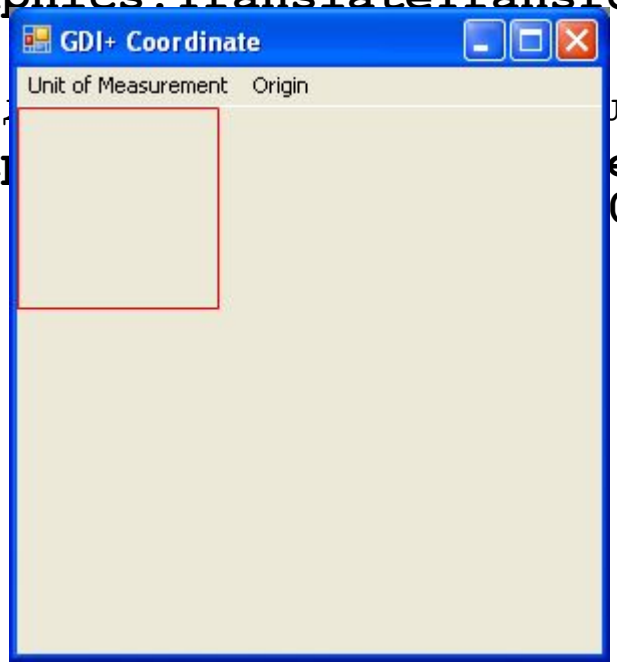
Применение других единиц измерения

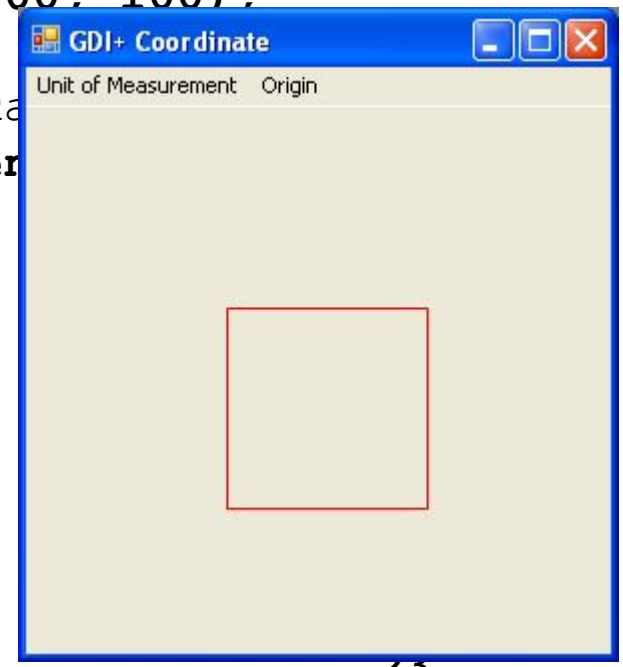
Свойству **Graphics.PageUnit** присваиваем значение из перечисления **GraphicsUnit**

Display	1/75 дюйма
Document	1/300 дюйма
Inch	Дюйм
Millimeter	Миллиметр
Pixel	Пиксел
Point	1/72 дюйма

Смена начала координат

```
private void MainForm_Paint(object sender, PaintEventsArgs e)
{
   // Настраиваем единицы измерения
   e.Graphics.PageUnit = GraphicsUnit.Point;

   // Настраиваем новую точку отсчета для системы координат
   e.Graphics.TranslateTransform(100, 100);

   // Ко                                    льника
   e.Gra                                 ew Per
                                         0);
}
```





Работа с цветом

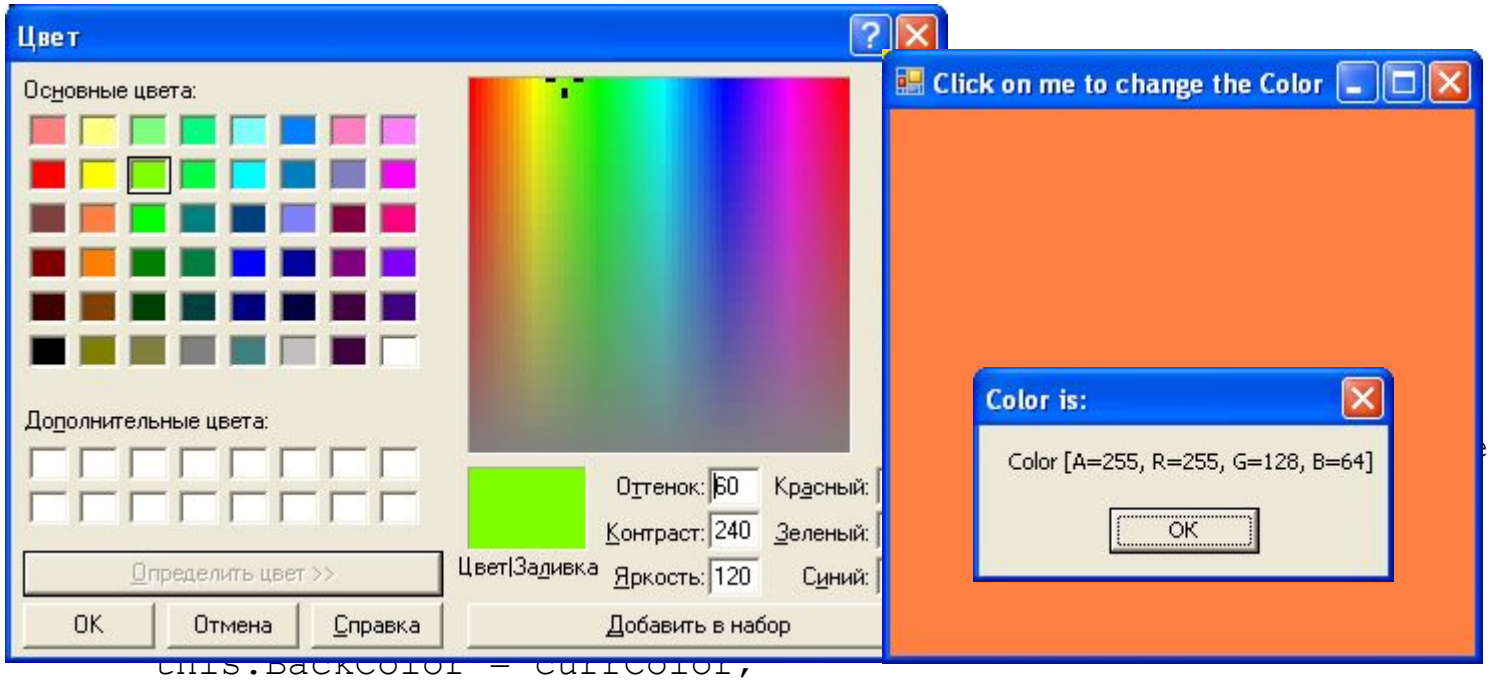
```
// Один из множества  предопределенных цветов
Color c = Color.PapayaWhip

// Названия цветов – статические поля
```

Члены класса Color

FromArgb()	Возвращает объект типа **Color** по прозрачности и RGB компонентам
FromKnownColor()	Возвращает объект типа **Color**. Используются значения из перечисления KnownColor
FromName()	Возвращает объект типа Color по имени. Используется текстовая строка
A, R, G, B	Возвращают значения, присвоенные α-каналу и RGB полям
IsNamedColor() Name	Позволяет узнать, принадлежит ли цвет какому-либо из известных цветов
GetBrightness() GetHue() GetSaturation()	Получает значения в системе HSB
ToArgb()	Возвращает числовые значения для объекта Color
ToKnownColor()	Возвращает значение из перечисления KnownColor

Возможности класса ColorDialog

```
public class ColorDlgForm : System.Windows.Forms.Form
{
..                                                    ;
..                                                    )
            this.BackColor = curColor;
            // Выводим информацию о выбранном цвете
            string strARGB = colorDlg.Color.ToString();
            MessageBox.Show(strARGB, "Color is:");
        }
    }
}
```



Работа со шрифтами

```
// Создаем объект Font, указывая имя шрифта и его размер
Font f = new Font("Times New Roman", 12)

// Создаем объект Font, указывая имя, размер и стиль
Font f2 = new Font("Arial", 20,
   FontStyle.Bold | FontStyle.Underline)

// public void DrawString(String, Font, Brush, Point)
g.DrawString("My string", new Font("Arial",24),
   new SolidBrush(Color.Black), new Point (0,0));

// public void DrawString(String, Font, Brush, float, float)
g.DrawString("Another string", f2, Brushes.Black,10,30);
```

Семейства шрифтов

```
// Создаем объект FontFamily
FontFamily myFamily = new FontFamily("Verdana");

// Передаем созданный нами объект как параметр
Font myFont = new Font(myFamily, 12);
e.Grapgics,DrawString("Hello",myFont,Brushes.Blue,10,10)
```

Члены типа FontFamily позволяют узнать различные типографские характеристики шрифта: GetCellAscent(), GetCellDescent(), GetEmHeigth(), GetLineSpacing(), GetName(), IsStyleAvailable()

Приложение с возможностью выбора шрифта

```
public class FontForm: System.Windows.Forms.Form
{
    private Timer timer;
    private int swellValue;
    // Этот шрифт будет использоваться по умолчанию:
    private string fontFace = "WingDings";

    public FontForm()
    {
    // Считаем, что система меню создана при помощи  Visual Studio.IDE
     InitializeComponents();

     timer = new Timer();
     Text = "Font App";
     Width = 425; Height = 150;
     BackColor = Color.Honeydew;
     CenterToScreen();

     // Настраиваем таймер
     timer.Enabled = true; timer.Interval = 100;
     timer.Tick += new EventHandler(FontForm_OnTimer);
    }
}
```

Приложение с возможностью выбора шрифта

```
private void FontForm_OnTimer(object sender, EventArgs e)
{
        // При каждом «тике» размер шрифта увеличивается на 5
    swellValue += 5;

         // При достижении максимального размера уменьшаем размер до 0
    if(swellValue >= 50)
            swellValue = 0;

        // Перерисовываем нужную нам прямоугольную область
    Invalidate(new Rectangle(0, 0, ClientRectangle.Width, 100));
   }
```

Приложение с возможностью выбора шрифта

```
private void FontForm_Paint(object sender, PaintEventsArgs e)
{
    Graphics g = e.Graphics;

// Размер шрифта будет меняться между 12 и 62 в зависимости от текущего значения  swellValue
    Font theFont = new Font(fontFace, 12 + swellValue);

    string message = "Hello GDI+";

    // Выводим сообщение по центру формы
    float windowCenter = this.DisplayRectangle.Width/2;
    SizeF stringSize = g.MeasureString(message, theFont);
    float startPos = windowCenter - (stringSize.Width/2);

    g.DrawString(message, theFont, new SolidBrush(Color.Blue), startPos, 10);
}
```

Обработка команд меню

```
private void FormatFont_Clicked(object sender, EventArgs e)
    {
     cmiFontChecked.Checked = false;

     MenuItem miClicked = (MenuItem)sender;
     fontFace = miClicked.Text.Remove(0,1);

     if (fontFace == "Arial")
                  cmiFontChecked = cmiArial;
     else if (fontFace == "Times New Roman")
                  cmiFontChecked = cmiTimesNewRoman;
     else if (fontFace == "WingDings")
                  cmiFontChecked = cmiWingDings;

     cmiFontChecked.Checked = true;
     Invalidate();
    }
```

Информация об установленных шрифтах

```
public class FontForm : System.Windows.Forms.Form
{
  // Для хранения списка шрифтов
  private string installedFonts;

  // Обработчик события меню для вывода списка установленных в системе
   шрифтов
  private void mnuConfigShowFonts_Clicked(object sender, EventArgs e)
  {
        InstalledFontCollection fonts = new InstalledFontCollection();
   for(int i=0; i < fonts.Families.Lenght; i++)
   {
    installedFonts += fonts.Families[i].Name + " ";
   }

   // На этот раз нам потребуется перерисовать всю клиентскую область
   // формы, поскольку вывод полученных значений будет производиться в
   // нижнюю часть  формы
   Invalidate();
   }
...
}
```

Вывод информации в нижнюю часть формы

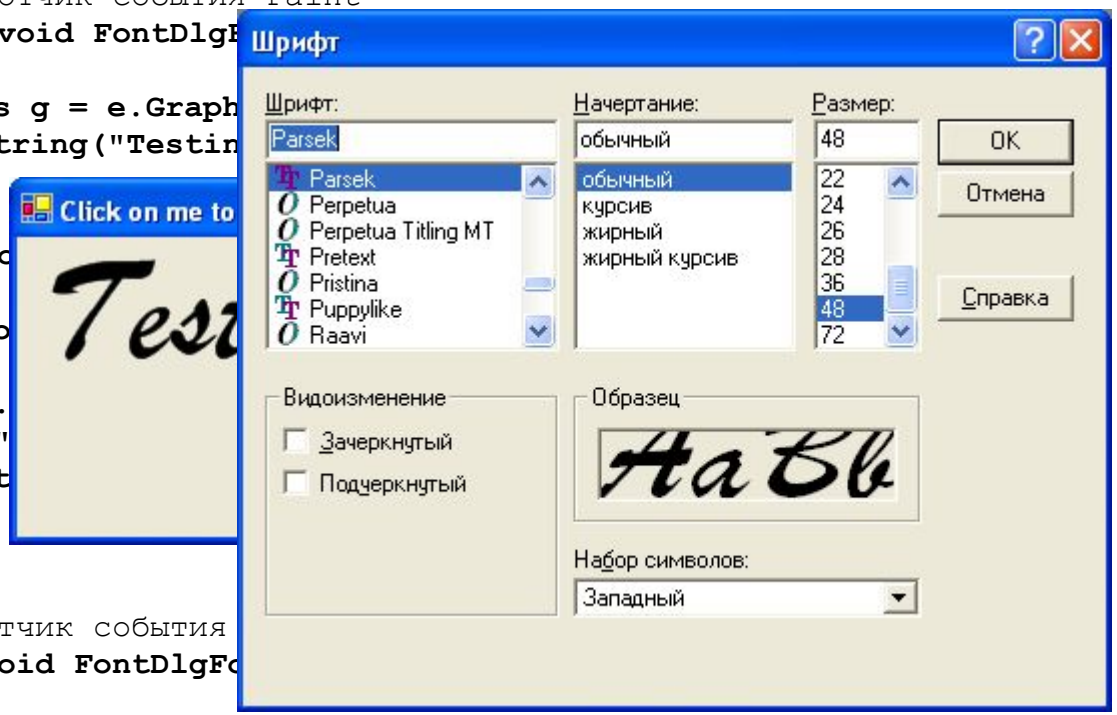
```
private void FontForm_Paint(object sender, PaintEventArgs e)
{
    Graphics g = g.Graphics;
    Font theFont = new Font(fontFace, 12 + swellValue);
    string message = "Hello GDI+";

    // Выводим сообщение "Hello GDI+" по центру формы
    float windowCenter = this.DisplayRectangle.Width/2;
    SizeF stringSize = g.MeasureString(message, theFont);
    float startPos = windowCenter - (stringSize.Width/2);
         g.DrawString(message, theFont, new SolidBrush(Color.Blue),
         startPos, 10);

    // Выводим информацию об установленных шрифтах под занятой областью
    Rectangle myRect = new Rectangle(0, 100, ClientRectangle.Width,
                       ClientRectangle.Height);

    // Будем рисовать в этой области белым по черному
    g.FillRectangle(newSolidBrush(Color.Black), myRect);
    g.DrawString(installedFonts, new Font("Arial", 12),
                  new SolidBrush(Color.White), myRect);
}
```

Обработка события Resize

```
private void FontForm_Resize(object sender, System.EventArgs e)
{
    Rectangle myRect = new Rectangle(0, 100,
    ClientRectangle.Width, ClientRectangle.Height);
    Invalidate(myRect);
}
```

Класс FontDialog

```
public class FontDlgForm : System.Windows.Forms.Form
{
   private System.Windows.Forms.FontDialog fontDlg;
   private Font currFont;

   // Обработчик события Paint
   private void FontDlgF
   {
    Graphics g = e.Graph
    g.DrawString("Testin                                    0, 0);
   }

   public Fo
   {
    CenterTo
    fontDlg
    fontDlg.
    Text = "
    currFont
    ...
   }

  // Обработчик события
  private void FontDlgFo
  {
    if (fontDlg.ShowDialog() != DialogResult.Cancel)
    {
      currFont = fontDlg.Font;
      Invalidate();
    }
  }
}
```



Классы System.Drawing.Drawing2D

AdjustableArrowCap CustomLineCap	Определяют наконечники перьев
Blend ColorBlend	Используются для смешивания цветов
GraphicPath GraphicPathIterator	Представляют набор связанных линий
PathData	Хранит графические данные для GraphicPath
HatchBrush LinearGradientBrush PathGradientBrush	Экзотические типы кистей

Перечисления System.Drawing.Drawing2D

DashStyle	Стиль штриховых линий пера
FillMode	Заполнение внутренней области фигуры
HatchStyle	Варианты штриховки
LinearGradientMode	Направление изменения градиентного цвета
LineCap	Стиль наконечника пера
PenAlignment	Ориентация пера относительно проводимой линии
PenType	Тип линии, создаваемый пером
QualityMode SmoothingMode RenderingHint	Качество вывода графического объекта

Определение качества вывода графического объекта

```
private void MainForm_Paint(object sender, PaintEventArgs e)
{
  Graphics g = e.Graphics;

  // Устанавливаем качество вывода графического объекта

  g.SmoothingMode = SmoothingMode.AntiAlias;

  //  Еще лучше   - HighQuality
  //  Еще быстрее - HighSpeed


}
```

Рисование пером

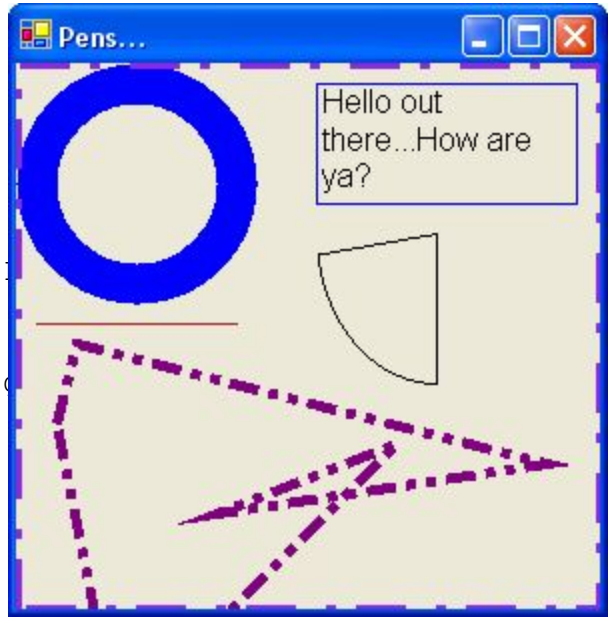
DrawPie()	Дуга (по эллипсу)
DrawPolygon()	Кривая Безье (кубическая кривая по четырем точкам)
DrawCurves()	Вывод кривой на основе массива точек
DrawEllipse()	Эллипс (по координатам описанного прямоугольника)
DrawLine() DrawLines()	Прямая или ломаная (по массиву точек)
DrawPath()	Выводи коллекцию прямых и кривых при помощи типа GraphicsPath
DrawPie()	Сектор эллипса
DrawPolygon()	Многоугольник (по массиву точек)
DrawRectangle() DrawRectangles()	Прямоугольник

Некоторые свойства класса Pen

Brush	Кисть
Color	Цвет
DashCap	Стиль наконечников для пунктирной линии
DashOffset	Смещение начала пунктира
DashStyle	Стиль пунктира
LineJoin	Способ пересечения линий
PenType	Стиль линий
StartCap, EndCap	Наконечники линий
Width	Толщина линии

Работа с перьями

```
private void MainForm_Paint(object sender, PaintEventArgs e)
{
  Graphics g = e.Graphics;

  // Создаем большое перо синего цвета
  Pen bluePen = new Pen(Color.Blue, 20);

  // Создаем еще одно перо при помощи заготовок из коллекции
  Pen pen2 = Pens.Firebrick;

  // Выводим при помощи созданных нами перьев геометрические
  g.DrawEllipse(bluePen, 10, 10, 100, 100);
  g.DrawLine(pen2, 10, 130, 110, 130);
  g.DrawPie(Pens.Black, 150, 10, 120, 150, 90, 80);

  // Выводим многоугольник пурпурного цвета
  Pen pen3 = new Pen(Color.Purple, 5);
  pen3.DashStyle = DashStyle.DashDotDot;
  g.DrawPolygon(pen3, new Point[] {     new Point (30, 140), new Point(265, 200),
              new Point(100, 225), new Point(190, 190),
              new Point(50, 330),  new Point(20, 180),} );

  // Добавляем прямоугольник со вписанным нами текстом
  Rectangle r = new Rectangle(150, 10, 130, 60);
  g.DrawRectangle(Pens.Blue, r);
  g.DrawString("Hello out there...How are ya?",   new Font("Arial", 12), Brushes.Black, r);
}
```



Работаем с наконечниками перьев (Перечисление LineCap)

ArrowAnchor	Стрелки
DiamondAnchor	Ромбы
Flat	Стандартное завершение
Round	Скругление
RoundAnchor	Шары
Square	Квадраты в толщину линии
SquareAnchor	Квадраты большего размера
Trinagle	Треугольное завершения

Наконечники перьев

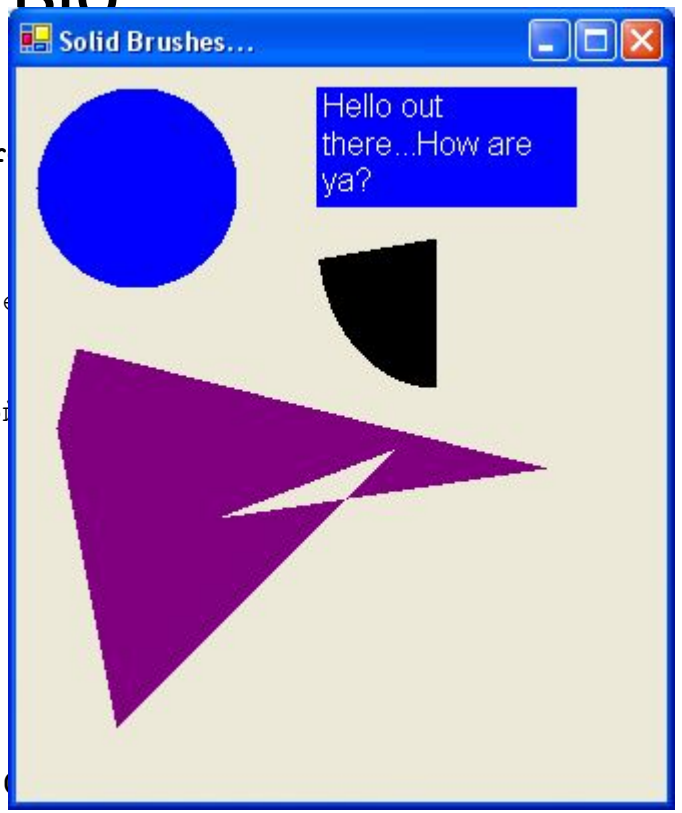
```
private void MainForm_Paint (object sender, PaintEventArgs e)
{
  Graphics g = e.Graphics;
  Pen thePen = new Pen(Color.Black, 10);
  int yOffSet = 10;

  // Получаем все значения перечисления LineCap
  Array obj = Enum.GetValues(typeof(LineCap));

  // Выводим линию с использованием значения из LineCap
  for(int x = 0; x < obj.Length; x++)
  {
    // Настраиваем "наконечник" пера
    LineCap temp = (LineCap)obj.GetValue(x);
    thePen.StartCap = temp;
    thePen.EndCap = temp;

    // Выводим имя значения перечисления
    g.DrawString(temp.ToString(), new Font("Times New Roman", 10),
                 new SolidBrush(Color.Black), 0, yOffSet);

    // Выводим линию с выбранным наконечником
    g.DrawLine(thePen, 100, yOffSet, Width - 50, yOffSet);
    yOffSet += 40;
  }
}
```

Работаем с кистью

FillClosedCurve()	Область внутри замкнутой кривой
FillEllipse()	Эллипс
FillPath()	Область внутри траектории
FillPie()	Сегмент эллипса
FillPolygon()	Многоугольник
FillRectangle() FillRectangles()	Прямоугольник, несколько прямоугольников
FillRegion	Область Region

Работа с кистью

```
private void MainForm_Paint(object sender, PaintEventAr
{
    Graphics g = e.Graphics;

    // Создаем "плотную кисть" – SolidBrush – синего цве
    SolidBrush blueBrush = new SolidBrush(Color.Blue);

    // Создаем еще одну кисть при помощи заранее готово
    SolidBrush pen2 = (SolidBrush)Brushes.Firebrick;

    // Закрашиваем этими кистями геометрические фигуры
    g.FillEllipse(blueBrush, 10, 10, 100, 100);
    g.FillPie(Brushes.Black, 150, 10, 120, 150, 90, 80)

    // Закрашиваем многоугольник пурпурным цветом
    SolidBrush brush3 = new SolidBrush(Color.Purple);

    g.FillPolygon(brush3, new Point[]{ new Point(30, 14
          new Point(100, 225), new Point(190, 190),
          new Point(50, 330), new Point(20, 180)} );

    // и прямоугольник с текстом – синим:
    Rectangle r = new Rectangle(150, 10, 130, 60);
    g.FillRectangle(Brushes.Blue, r);
    g.DrawString("Hello out there...Now are ya?",
          new Font("Arial", 12), Brushes.White, r);
}
```

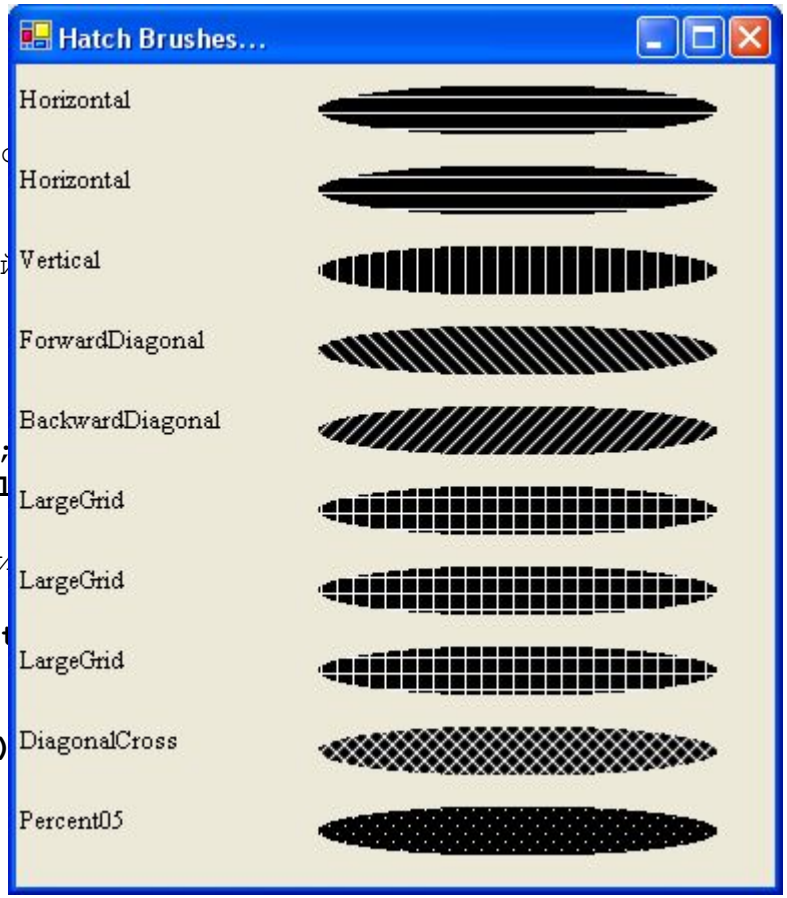


Работаем со штриховыми кистями (Перечисление HatchStyle)

BackwardDiagonal	Диагональная с наклоном вправо
Cross	Крестообразная
DiagonalCross	Диагональная крестообразная
ForwardDiagonal	Диагональная с наклоном влево
Hollow	Пустая
Horizontal	Горизонтальная
Pattern	По пользовательскому образцу
Solid	Сплошная
Vertical	Вертикальная

Работаем со штриховыми кистями

```
private void MainForm_Paint(object sender, PaintEventArgs e)
{
  Graphics g = e.Graphics;
  int yOffSet = 10;

  // Помещаем в массив все члены перечисления Hatc
  Array obj = Enum.GetValues(typeof(HatchStyle));

  // Выводим эллипс со штриховкой, соответствующей
  // с 1 по 10
  for(int x =0; x < 10; x++)
  {
    // Настраиваем кисть
    HatchStyle temp = (HatchStyle)obj.GetValue(x);
    HatchBrush theBrush = new HatchBrush(temp, Col

    // Выводим имя каждого из значений перечислени
    g.DrawString(temp.ToString(), new Font("Times
          new SolidBrush (Color.Black), 0, yOffSet

    // Закрашиваем эллипс штриховой кистью
    g.FillEllipse(theBrush, 150, yOffSet, 200, 25)
    yOffSet += 40;
   }
}
```



Работаем с текстурными кистями

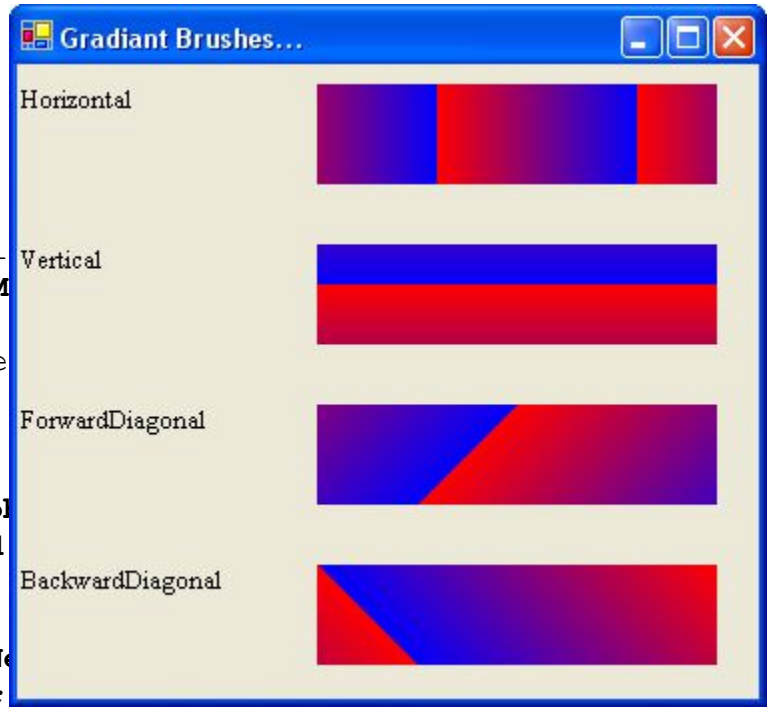
```
public class MainForm : System.Windows.Forms.Form
{
  // Потребуются переменные типа Brush для загрузки изображений
   private Brush texturedTextBrush;
   private Vrush texturedBGroundBrush;

  public MainForm()
  {
   ...
   // Загружаем изображение для текстуры формы
   Image bGroundBrushImage = new Bitmap(Coluds.bmp");
   texturedBGrounBrush = new TextureBrush(bBroundBrushImage);

   // Загружаем изображение для текстуры теста
   Image textBrushImage = new Bitmap("Soap Bubbles.bmp");
   texturedTextBrush = new TextureBrush(textBrushImage);
  }
...
}
```

Используем текстурные кисти

```
private void MainForm_Paint(object sender, PaintEventArgs e)
{
  Graphics g = e.Graphics;
  Rectangle r = ClientRectangle;

  // Выводим облака на форме
  g.FillRectangle(texturedBGroundBrush, r);

  // Текст должен быть большим – чтобы можно было разглядеть текстуру
  g.DrawString ("Bitmaps as brushes! Way cool...",
   new Font("Arial", 60, FontStyle.Bold | FontStyle.Italic),
        texturedTextBrush, r);
}
```

Работа с градиентными кистями

```
private void MainForm_Paint(object sender, PaintEventArgs e)
{
  Graphics g = e.Graphics;
  Rectangle r = new Rectangle(10, 10, 100, 100);

  LinearGradientBrush theBrush = null;
  int yOffSet = 10;

  // Получаем все значения перечисления LinearGradi
  Array obj = Enum.GetValues(typeof(LinearGradientM

  // Выводим прямоугольник, используя значения Line
  for(int x = 0; x < obj.Length; x++)
  {
    // Настраиваем кисть
    LinearGradientMode temp = (LinearGradientMode)ob
    theBrush = new LinearGradientBrush(r, Color.Red

    // Выводим имя значения перечисления
    g.DrawString(temp.ToString(), new Font("Times Ne
          new SolidBrush(Color.Black), 0, yOffSet);

    // Заполняем прямоугольник при помощи градиентной кисти
    g.FillRectangle(theBrush, 150, yOffSet, 200, 50);
    yOffSet += 80;
  }
}
```



Вывод изображений

Члены класса Image

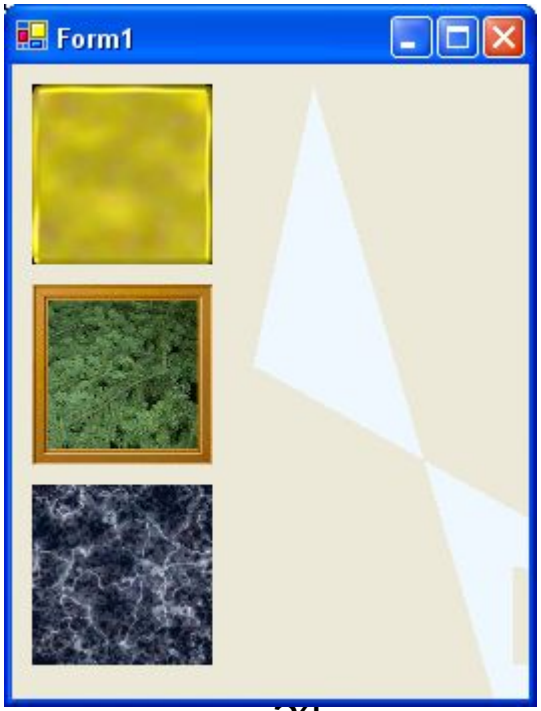
FromFile(), FromHBitmap(), FromStream()	Статические методы создания изображений
Height, Width, Size, PhysicalDimensions, HorizontalDimensions, VericalResolution	Свойства для работы с размерами
Palette	Свойство, возвращает объект Colorpaletter
GetBounds()	Прямоугольник, представляющий текущую область, занятую изображением
Save()	Сохраняет изображение в файл

Вывод изображений

```
public class mainForm : System.Windows.Forms.Form
{
   // Объявляем переменные типа Image
   private Image bMapImageA;
   private Image bMapImageB;
   private Image bMapImageC;

   public MainForm()
   {
    ...
   // Используем для этих переменных объекты класса Bitmap
   bMapImageA = new Bitmap("ImageA.bmp");
   bMapImageB = new Bitmap("ImageB.bmp");
   bMapImageC = new Bitmap("ImageC.bmp");
   }
...
}
```

Вывод изображений

```
private void Form1_Paint(object sender,
    System.Windows.Forms.PaintEventArgs e)
{
    Graphics g = e.Graphics;

    // Выводим изображения
    g.DrawImage(bMapImageA, 10, 10, 90, 90);
    g.DrawImage(bMapImageB, rectB);
    g.DrawImage(bMapImageC, rectC);

}
```



Форматы изображений

```
// Тип Bitmap поддерживает все распространенные форматы!
Bitmap myBMP = new Bitmap("CoffeeCup.bmp");
Bitmap myGIF = new Bitmap("Candy.gif");
Bitmap myJPEG = new Bitmap("Clock.jpg");
Bitmap myPNG = new Bitmap("Speakers.png");
Bitmap myTIFF = new Bitmap("FooFighters.bmp");

// Выводим изображения при помощи Graphics.DrawImage()
g.DrawImage(myBmp, 10, 10);
g.DrawImage(myGIF, 220, 10);
g.DrawImage(myJPEG, 280, 10);
g.DrawImage(myPNG, 150, 200);
g.DrawImage(myTIFF, 300, 200);
```

Методы Add класса GraphicsPath

- AddArc
- AddBezier
- AddBeziers
- AddclosedCurves
- AddCurve
- AddEllipse
- AddLine
- AddLines
- AddPAth
- AddPie
- AddPolygon
- AddRectangle
- AddRectangles
- AddString

Построение пути

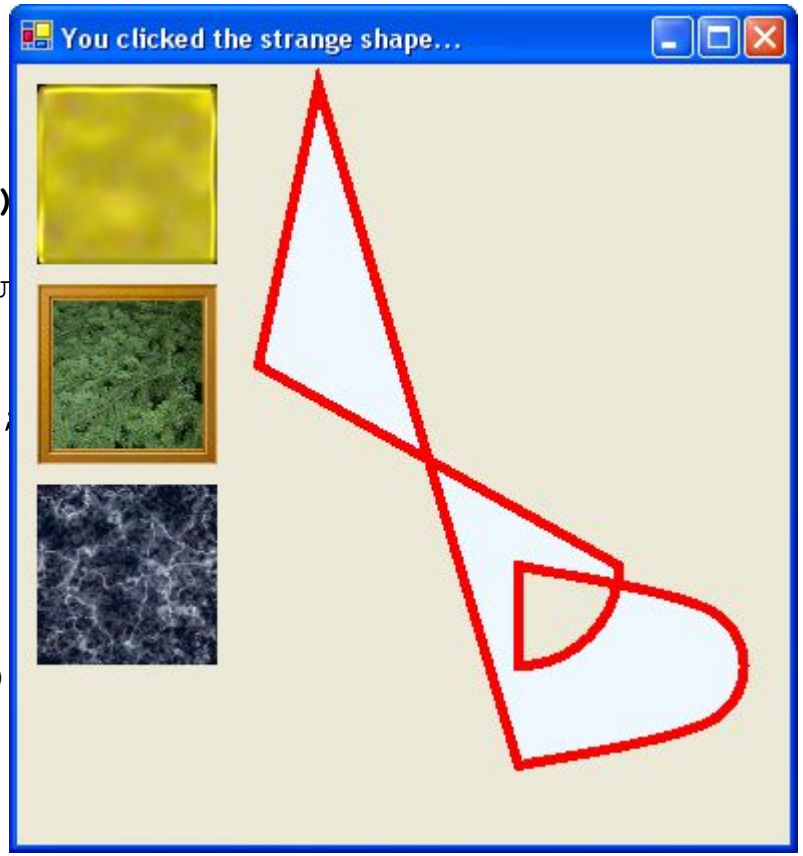
```
public MainForm : System.Windows.Forms.Form
{
  // Создаем объект GraphicsPath
  GraphicsPath myPath = new GraphicsPath();

  public MainForm()
  {
  // Добавляем в объект GraphicsPath элементы, из которых он будет
   состоять
   myPath.StartFigure();
    myPath.AddLine(new Point(150, 10), new Point(120, 150));
    myPath.AddArc(200, 200, 100, 100, 0, 90);
    Point point1 = new Point(250, 250);
    Point point2 = new Point(350, 275);
    Point point3 = new Point(350, 325);
    Point point4 = new Point(250, 350);
    Point[] points = {point1, point2, point3, point4};
        myPath.AddCurve(points);
   myPath.CloseFigure();
    ...
    }
}
```

Проверка попадания в фигуру

```
private void Form1_MouseUp(object sender, System.Windows.Forms.MouseEventArgs e)
{
  // Get (x, y) of mouse click.
  Point mousePt = new Point(e.X, e.Y);

  // See if the mouse is anywhere in the 3 regions...
  if (rectA.Contains(mousePt))
  {
    isImageClicked = true;
    imageClicked = 0;
    this.Text = "You clicked image A";
  }
  …
  else if (myPath.IsVisible(mousePt))
    {
     isImageClicked = true;
     imageClicked = 3;
     this.Text = "You clicked the strange shape...";
    }
        else   // Not in any shape, set defaults.
    {
     isImageClicked = false;
     this.Text = "Images";
    }

    // Redraw the client area.
    Invalidate();
}
```

Выводим объект GraphicsPath

```
private void MainForm_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    ...

    // Выводим объект GraphicsPath
    g.FillPath(Brushes.AliceBlue, myPath)

    // Обводим этот объект рамкой (по щел
    if(isImageClicked = = true)
    {
     Pen outline = new Pen(Color.red, 5);

     switch(imageClicked)
     {
          ...
          case 3:
               g.DrawPath(outline, myPath)
               break;
          default:
               break;
     }
    }
}
```



Ресурсы

- *.resx
- *.resources

- ResourceManager
- ResourceWriter
- ResourceReader

Что еще?

- Элементы управления
- Ввод/вывод и сериализация объектов
- Взаимодействие с DLL, созданными на C и VB 6.0
- Доступ к данным ADO.NET
- Разработка Web-приложений
- Web-службы