

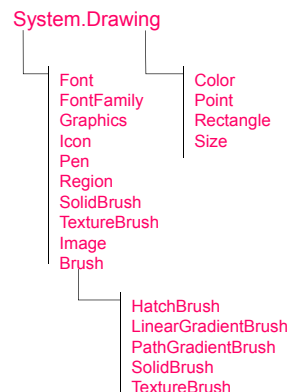
Графика

Графичен проектантски интерфейс GDI+ (Graphics Design Interface) за обработка на:

- графики;
- шрифтове;
- графични изображения.

Недостатък

- не винаги разполагаме с GDI+ в някои операционни системи (Linux, Mac OSX)



Графичен контекст и графични обекти

1. **Графичен контекст** – представлява графичната повърхност за изчертаване върху екрана.
2. **Графичен обект** – клас **Graphics**
 - управлява графичния контекст като управлява изчертаването на информацията;
 - съдържа методи за изчертаване, обработка на шрифтове, обработка на цветове и др.

Създаване на графика в Windows приложение

Windows приложение (наследник на **Form**) наследява манипулатора **virtual OnPaint** на събитието **Paint** за изпълнение на графични операции.

Всеки контрол при изчертаване извиква манипулатора **OnPaint** на събитието **Paint**.

Метод **Control.Invalidate** възстановява клиентската област на контрола и пречертава графичните компоненти.

1. Дефиниране на манипулатор

- а) **автоматично** – **предефиниране на OnPaint**
`protected override void OnPaint (PaintEventArgs e)`
- б) **програмно**
 - **добавяне на манипулатор в конструктора**
`this.Paint += new PaintEventHandler(<манипулатор>);`
 - **дефиниране на манипулатор**
`public void <манипулатор>(object source, PaintEventArgs e)`

2. Реализиране на манипулатора

- **извличане на графичния обект Graphics от PaintEventArgs аргумента**
`Graphics g = e.Graphics;`
- **изчертаване на графиката**
`<оператори за изчертаване>`

Структура Color

Дефинира константни цветове: Orange, Pink, Cyan, Magenta, Yellow, Black, White, Gray, DarkGray, Red, Green, Blue.

Клас Font

Дефинира вида, размера и стила на шрифта.

`Font myFont = new Font ("Arial", 24, FontStyle.Bold);`

Клас Pen

Дефинира писалка.

`Pen myPen = new Pen(Color.Red);`

Клас Brush

Дефинира четка за запълване на фигури.

`SolidBrush myBrush = new SolidBrush(Color.Yellow);`

Методи на класа Graphics за изчертаване и запълване на фигури и текст

```
public void DrawString (string s, Font font, Brush brush,
float x, float y);
```

Изчертава текста **s** в точка **(x, y)** с шрифт **font** и четка **brush**.

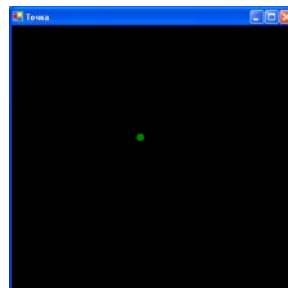
```
public void DrawLine (Pen pen, int x1, int y1, int x2, int y2);
```

Изчертава линия между две точки **(x1, y1)** и **(x2, y2)** с писалка **pen**.

```
public void FillEllipse (Brush brush, int x, int y,
int width, int height );
```

Запълва с четка **brush** елипсата, дефинирана от правоъгълника около нея, определен чрез координати на горен ляв ъгъл **(x, y)**, дължина **width** и височина **height**.

Пример: Изчертава зелена точка при натискане бутона на мишката (събитие **MouseDown** за формата).



```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
namespace Spot
{
public class Spot : System.Windows.Forms.Form
{
private Point clickPoint = new Point(0,0);
private const int RADIUS = 6;
private System.ComponentModel.Container components = null;
public Spot()
{
InitializeComponent();
this.Width=450;
this.Height=450;
}
protected override void Dispose( bool disposing ) { ...
```

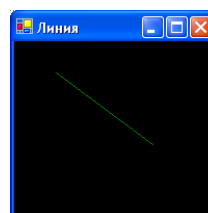
```
#region Windows Form Designer generated code
private void InitializeComponent()
{
// Spot
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.BackColor = System.Drawing.Color.Black;
this.ClientSize = new System.Drawing.Size(292, 266);
this.Name = "Spot";
this.Text = "Точка";

this.MouseDown += new System.Windows.Forms.MouseEventHandler
(this.MousePressed);
}
#endregion

[STAThread]
static void Main()
{
Application.Run(new Spot());
}
```

```
protected override void OnPaint (PaintEventArgs e)
{
// Извличане на графичния обект
Graphics g=e.Graphics;
// Изчертаване на зелена точка
SolidBrush brush=new SolidBrush(Color.Green);
if (clickPoint.X!=0 && clickPoint.Y!=0)
g.FillEllipse(brush, clickPoint.X - RADIUS,
clickPoint.Y - RADIUS, RADIUS*2, RADIUS*2);
}
private void MousePressed(object sender,
System.Windows.Forms.MouseEventArgs e)
{
// Създаване на нова точка в мястото на възникване на
// събитието MouseDown
clickPoint=new Point(e.X,e.Y);
// Опресняване на клиентската област и изпращане на съобщение
// за изчертаване (извикване на метод OnPaint)
this.Invalidate();
}
}
```

Пример: Изчертава зелена гъвкава линия (събития **MouseDown**, **MouseMove** и **MouseUp** за формата). Началната точка се определя при натискане бутона на мишката (**MouseDown**). При преместване на мишката (**MouseMove**) крайната точка се пречертава всеки път и линията се „разтяга“. При освобождаване бутона на мишката (**MouseUp**) завършва изчертаването на линията.



```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace RubberLine
{
    public class Form1 : System.Windows.Forms.Form
    {
        private Point point1 = new Point(0,0); // начална точка
        private Point point2 = new Point (0,0); // крайна точка
        private Line line = null;
        private System.ComponentModel.Container components = null;

        public Form1()
        {
            InitializeComponent();
        }

        protected override void Dispose( bool disposing ) { ... }
    }
}
```

```
#region Windows Form Designer generated code
private void InitializeComponent()
{
    // Form1
    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
    this.BackColor = System.Drawing.SystemColors.ControlText;
    this.ClientSize = new System.Drawing.Size(192, 166);
    this.Name = "Form1";
    this.Text = "Линия";
    this.MouseDown += new System.Windows.Forms.MouseEventHandler
        (this.MousePressed);
    this.MouseMove += new System.Windows.Forms.MouseEventHandler
        (this.MouseDragged);
    this.MouseUp += new System.Windows.Forms.MouseEventHandler
        (this.MouseReleased);
    this.Paint += new System.Windows.Forms.PaintEventHandler(this.PaintLine);
}
#endregion

[STAThread]
static void Main()
{ Application.Run(new Form1()); }
```

```
private void MousePressed(object sender,
    System.Windows.Forms.MouseEventHandler e)
{
    point1=new Point(e.X,e.Y);
}
private void MouseDragged(object sender,
    System.Windows.Forms.MouseEventHandler e)
{
    point2=new Point(e.X,e.Y);
    this.Invalidate();
}
private void MouseReleased(object sender,
    System.Windows.Forms.MouseEventHandler e)
{
    line = new Line(point1, point2);
    point1.X=0;
    point1.Y=0;
    point2.X=0;
    point2.Y=0;
    this.Invalidate();
}
```

```
private void PaintLine(object sender,
    System.Windows.Forms.PaintEventArgs e)
{
    Graphics g=e.Graphics;
    Pen pen=new Pen(Color.Green);
    if (line != null)
        g.DrawLine(pen,line.Point1, line.Point2);
    if((point1.X!=0 && point1.Y!=0 && point2.X!=0 && point2.Y!=0)
        g.DrawLine(pen, point1, point2);
    }
}
```

```
// Project => Add Class
using System;
using System.Drawing;
namespace RubberLine
{
    public class Line
    {
        private Point p1;
        public Point Point1
        { get { return p1; } }
        private Point p2;
        public Point Point2
        { get { return p2; } }
        public Line(Point p1, Point p2)
        {
            this.p1=p1;
            this.p2=p2;
        }
    }
}
```

Запазване състоянието на чертеж

Клас GraphicsPath (System.Drawing.Drawing2D)
Представя последователност от свързани линии и криви.

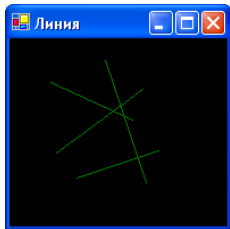
public void AddLine (Point pt1, Point pt2);
Добавя линеен сегмент към обект GraphicsPath.

public void StartFigure();
Стартира нова фигура, без да затваря текущата фигура.

public void CloseFigures();
Затваря текущата фигура и стартира нова фигура.

public void DrawPath (Pen pen, GraphicsPath path);
Изчертава GraphicsPath обект (метод на клас Graphics).

Пример: Запазване изчертаните линии.



```
...
using System.Drawing.Drawing2D;

public class Form1 : System.Windows.Forms.Form
{
    // ...
    GraphicsPath myPath = null;

    public Form1()
    {
        InitializeComponent();
        myPath = new GraphicsPath();
    }

    private void MousePressed(object sender,
        System.Windows.Forms.MouseEventArgs e)
    {
        myPath.StartFigure();
        point1=new Point(e.X,e.Y);
    }
}
```

```
private void MouseDragged(object sender,
    System.Windows.Forms.MouseEventArgs e)
{
    point2=new Point(e.X,e.Y);
    this.Invalidate();
}

private void PaintLine(object sender,
    System.Windows.Forms.PaintEventArgs e)
{
    Graphics g=e.Graphics;
    Pen pen=new Pen(Color.Green);
    e.Graphics.DrawPath(pen, myPath);
    if(point1.X!=0 && point1.Y!=0 && point2.X!=0 && point2.Y!=0)
        g.DrawLine(pen, point1, point2);
}

private void MouseReleased(object sender,
    System.Windows.Forms.MouseEventArgs e)
{
    line = new Line(point1, point2);
    myPath.AddLine(line.Point1, line.Point2);
    myPath.CloseFigure();
    point1.X=0; point1.Y=0;
    point2.X=0; point2.Y=0;
    this.Invalidate();
}
```