Графика под Windows

Лекция 10

Съдържание

- Компютърна графика – определение, видове
- Създаване на Windows приложение с графика
- Възможности на **Graphical devices interface (GDI+)** Пакетът **`System.Drawing`**
- Създаване на обекти за рисуване (pen, brush ….)
- Изчертаване на базови геометрични фигури
- Примери

Компютърна графика

- **Определение**

Компютърната графика разглежда методите и средствата, свързани със създаването, преобразуването и възпроизвеждането на графични изображения

- **Компютърната графика** обхваща няколко подобласти:
  - **2D/3D компютърна графика**
  - **компютърна анимация**
  - **обработка на изображения**
  - **геометрично моделиране** (често използвано за инженерни и медицински цели)
- Първите успехи на компютърната графика датират от 1963, когато **Иван Съдърланд** разработва първата компютърна програма с графичен потребителски интерфейс

Компютърната графика

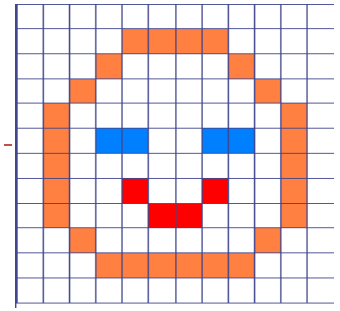
- **Компютърната графика** (Computer graphics (CG)
- **Двуизмерната (2D) компютърна графика** се среща в **издателската продукция,** която използва компютри (DTP – Desk Top Publishing), дизайна във вестниците, списанията, билбордовете и т.н.
- Инструменти на 2D се използват при **web дизайна**, **презентациите, дигиталното видео** и създаването на специални ефекти, в дигиталната фотография
- Двуизмерната (2D) компютърна графика е решаваща за ГПИ (graphical user interfaces)

Компютърен монитор

- Компютърният монитор е правоъгълна таблица от **квадратчета** – пиксели, наречена **растер**
- Разделителната способност - основна характеристика на монитора, от която зависи детайлността
- Разделителна способност (резолюция) е броят точки (пиксели), които могат да се изобразят в хоризонтална и вертикална посока
- Всеки пиксел може да “свети” в един от няколко милиона цвята
- **Компютърното изображение** се получава, като за **всеки пиксел** се избере подходящ **цвят**

Photo Shop; Paint

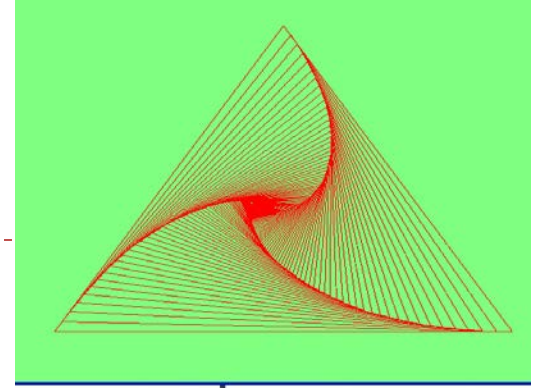
Видове графика



Според **начина на построяване на изображението** компютърната графика се разделя на:

- **Растерна графика** – разлагане на изображението **(bitmap)** на отделни точки (пиксели), всяка със собствен цвят и яркост
  - Растерното изображение е обемисто и след като веднъж е изработено, не може лесно да се променя
  - При опит да се увеличи или намали, качеството му силно се влошава
- **Векторна графика** – векторното изображение се получава чрез изчертаване на геометрични линии **(вектори)**
- **Векторната графика** или **геометричното моделиране** представлява използване на геометрични базисни елементи, като точки, линии, криви, правоъгълници и др. за представяне на изображения в компютърната графика

Векторна графика



- Отделните графични елементи са представени с математическо описание
- Цветният компонент се задава като запълване или щриховка
- Размерът на файла с векторна графика винаги е по-малък от съответния файл със същото изображение като **bit-map**

- Основното **предимство на векторната графика** е нейната **мащабируемост**
  - **Промяна в размерите** на изображението **не довежда до промяна** в неговото **качество** при визуализация
- Същото **не важи** за едно **bit-map** изображение
  - При разширяване на неговите стандартни размери се проявява **ефект на пикселизация** на изображението с помътняване на контурите и мозаечен ефект при визуализация

Векторна графика

- Когато векторното изображение се представя на екрана, математическото описание се трансформира в множество от цветни точки, които изменят съответните пиксели на екрана

- Класът **`Graphics`**, **дефиниран в пространството** **`System.Drawing`**
- Класът **`Graphics`** е имплементация на системата за съставяне на графични изображения GDI+

- C# предлага инструментариум за създаване на **векторна графика**, включваща и **текст**

Пакетът System.Drawing и GDI+ (Graphics Device Interface)

- Пакетът **`System.Drawing`** осигурява достъп до **GDI+** функциите на Windows:
  - **повърхности за чертане**
  - работа с **графика** и графични трансформации
  - изчертаване на **геометрични фигури**
  - работа с **изображения**
  - работа с **текст и шрифтове**
  - **печатане** на принтер

Пакетът System.Drawing

System.Drawing се състои от:

- **System.Drawing**
  - Основни класове
  - Повърхности, моливи, четки
  - Основни геометрични фигури
  - Изобразяване на текст
- **System.Drawing.Imaging**
  - Работа с изображения
  - Картинки и икони
  - Четене и записване в различни файлови формати
  - Оразмеряване на изображения

Пакетът `System.Drawing`

`System.Drawing` се състои от:

- `System.Drawing.Drawing2D`
  - Графични трансформации
  - Бленди, матрици и др.
- `System.Drawing.Text`
  - Достъп до шрифтовете
- `System.Drawing.Printing`
  - Печатане на принтер
  - Системни диалогови кутии за печатане

Класът `Graphics`

- Класът **`System.Drawing.Graphics`**
  - Предоставя абстрактна повърхност за чертане
    - Такава повърхност може да бъде, **както част от контрола** на **екрана**, така и част от **страница на принтер** или друго устройство.
  - Най-често **чертането се извършва в обработчика** на **събитието `Paint`**
    - **`Paint`** преизчертава контролата при необходимост
    - Параметърът **`PaintEventArgs`** съдържа **`Graphics`** обекта
  - **`Graphics` обект** може да се създава чрез **`Control.CreateGraphics()`**
  - Трябва да се освобождава чрез **`finally`** блок или с конструкцията **`using`**

Класът Graphics

- Класът **Graphics** капсулира **GDI + повърхност за чертане**
- Преди съставянето на някаква фигура (например кръг или правоъгълник), **трябва да се създаде повърхност**, като се използва Graphics class
- Основно се използва **събитие Paint** на WindowsForm, за да се укаже, че ще се създава графика

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

- Друг начин е чрез използване на предефиниран **OnPaint метод**

```
protected override void OnPaint(PaintEventArgs e)
        {
            Graphics g = e.Graphics;
        }
```

Основни методи на класа `Graphics`

`DrawArc`	Draws an arc from the specified ellipse.
`DrawBezier`	Draws a cubic Bezier curve.
`DrawBeziers`	Draws a series of cubic Bezier curves.
`DrawClosedCurve`	Draws a closed curve defined by an array of points.
`DrawCurve`	**Draws a curve defined by an array of points.**
`DrawEllipse`	**Draws an ellipse.**
`DrawImage`	**Draws an image.**
`DrawLine`	**Draws a line.**
`DrawPath`	Draws the lines and curves defined by a GraphicsPath.
`DrawPie`	**Draws the outline of a pie section.**
`DrawPolygon`	**Draws the outline of a polygon.**
`DrawRectangle`	**Draws the outline of a rectangle.**
`DrawString`	**Draws a string.**
`FillEllipse`	Fills the interior of an ellipse defined by a bounding rectangle.
`FillPath`	Fills the interior of a path.
`FillPie`	Fills the interior of a pie section.
`FillPolygon`	**Fills the interior of a polygon defined by an array of points.**
`FillRectangle`	**Fills the interior of a rectangle with a Brush.**
`FillRectangles`	**Fills the interiors of a series of rectangles with a Brush.**
`FillRegion`	Fills the interior of a Region.

Graphics Objects

- Трябва да се **добави** пространството:

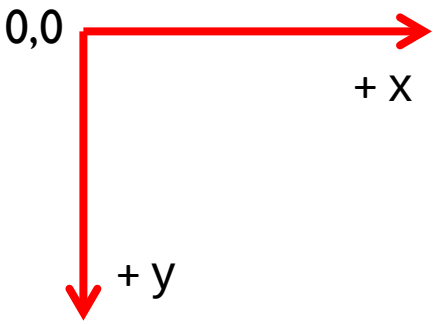
```
using System.Drawing.Drawing2D;
```

- След създаване на **обект** `Graphics g`, може да се използва за **чертане на линии и геометрични фигури**, **запълване на фигури, изчертаване на текст** и т.н.
- Основни **обекти** са:

`Brush`	Използва се за запълване на затворени повърхности с модели, цветове, или растерни изображения
`Pen`	Изчертава линии и многоъгълници, включително правоъгълници, дъги и части от кръг (pies)
`Font`	Използва се за описание на щрифта за изписване на текст
`Color`	Описва цвета, използван за изчертаване на конкретния обект

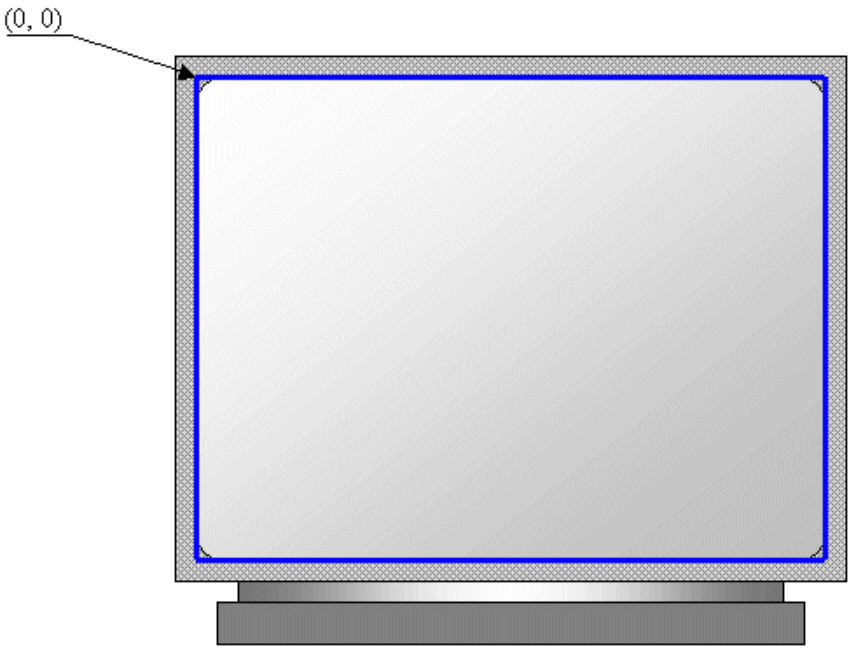
Чертожно поле

- В компютърната графика равнината е ограничена до т.н. чертожно поле – правоъгълна част от екранния растер с **r** реда и **c** стълба, която заема вътрешността на екранната форма
- **Пикселът в горния ляв ъгъл** на чертожното поле е **началото** на координатната система – **координати (0,0)**



  - x – координати са **целите числа** от 0 **до c-1** и се увеличават **отляво надясно**
  - y – координати са **целите числа** от 0 **до r-1** и се увеличават **отгоре надолу**

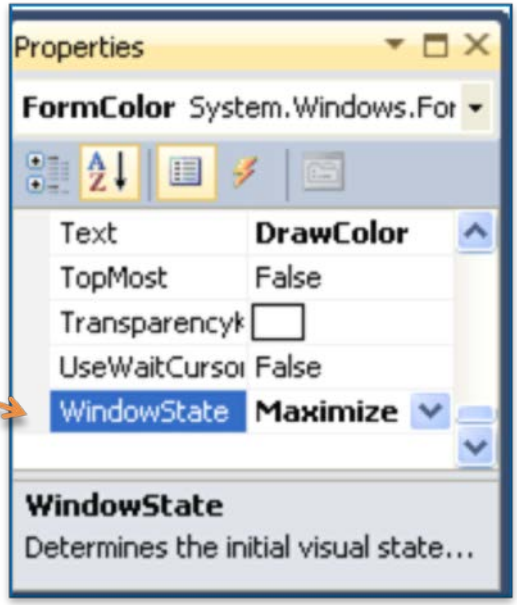
Екранна форма



- Екранната форма (за разлика от Декартова КС) е дискретна – екранните точки са пикселите и **координатите** им са **само** цели положителни числа

Чертожно поле

- **Реалните размери** r и c на чертожното поле зависят от това, колко голям е прозорецът на програмата
- Най-голямо чертожно поле ще получим, ако максимизираме прозореца
- Свойство **WindowsState = Maximized**
- Текущите размери на чертожното поле на формата се съдържат в свойствата **ClientSize.Width –** ширина **и ClientSize.Hight –** височина на чертожното поле



Създаване на писалка - The Pen Class

- Инструментът, който избираме, за да чертаем върху чертожното поле се характеризира с цвета и дебелината на следата, която оставя
- Тези характеристики са обобщени в класа Pen (писалка)
- Винаги трябва да се използва **конструктор Pen** за инициализиране на **нова инстанция** от класа Pen
- **Инициализиране** на нова инстанция **Pen** class със **специфичен цвят**
  - `public Pen(Color);`
- **Инициализиране** на нова инстанция **Pen** class със **специфична четка**
  - `public Pen(Brush);`

Създаване на писалка - The Pen Class

```
Pen p = new Pen(<Color>, <width>);
```

- Инициализира нова инстанция **Pen class** със специфичен **цвят** и **ширина на линията**
- Цветът е **обект от клас** `Color` със 140 свойства, всяко от които е някакъв цвят и се избира от списъчна кутия
- **Широчината** на линията е **число от тип** float, и може да считаме, че закръглено до цяло число, означава брой пиксели
  - `Pen pn = new Pen(Color.Blue,20);`
- `Pen pb = new Pen(<Brush>, <width>);`
- `public Pen(Brush, float);`

Създаване на писалка

- Всяка стойност на широчина **width < 1**, се **приема за 1**

`Pen p = new Pen(Color.Aqua, -1);` - създава писалка със син цвят и дебелина 1 пиксел

- Ако искаме да създадем писалка с цвят, който не е между включените като свойство на класа `Color`, трябва да използваме статичния метод:

`Color.FromArgb(<червен>, <зелен>, <син>);`

- който **задава цвят**, базиран на **модела RGB** (red-green-blue) т.е. чрез интензитетите на трите образуващи цвята

```
Pen p = new Pen(Color.FromArgb(45, 226, 29),3);
```

- Трите интензитета на избрания цвят могат да се вземат от цветовата палитра

The Font Class

- Класът Font определя конкретен формат за текст, като например шрифт, размер, стил и характеристики
- Инициализиране на нова инстанция **Font class** със специфични характеристики
- `public Font(string, float);`
  - `Font font = new Font("Times New Roman", 26);`
- `public Font(Font, FontStyle);`
- Членове на `FontStyle`

Member Name	Description
Bold	Bold text.
Italic	Italic text.
Regular	Normal text.
Strikeout	Text with a line through the middle.
Underline	Underlined text.

The Brush Class

- **Класът Brush** е **абстрактен базов клас** и **не може да бъде инстанциран**
- **Винаги** се използват негови **производни класове** за **създаване на инстанция** като:
  - **SolidBrush**
  - **TextureBrush**
  - **RectangleGradientBrush**
  - **LinearGradientBrush**

Фонов цвят на чертожното поле

- Фоновият цвят се задава като стойност на свойството **BackColor**
- Друг начин е с **метода `Clear`** на класа `Graphics`
- `g.Clear(Color.Red);`
- `g.Clear(Color.FromArgb(20, 140, 60));`

Създаване на Windows приложение с графика – пример 1

1) Стартирайте **VS.NET** и създайте нов **Windows Forms** проект
2) След това задължително трябва **да напишем** в класа `FormColor` метод за обработка на събитието `Paint`

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

Този **метод** се изпълнява при **отварянето на формата** и съобщава на програмата, че вътрешността на екранната форма няма да се използва за разполагане на компоненти, а за **създаване на графика** чрез програмния код, **написан за обект g**

Изчертаването на графичните обекти трябва да стане също в този метод.

Пример 1 - продължение

- Да се напише програма **DrawColors,** която отваря екранна форма, трансформира я в чертожно поле и с оператор за цикъл променя многократно цвета на фона с генериран по случаен начин цвят, при което се получава смяна на цветовете

- **Фоновият цвят** на чертожното поле се задава като стойност на свойството й **`BackColor`**.
- Друг начин за смяна на фоновия цвят на полето е с метода **Clear** на класа **Graphics**. Той има един параметър, който е обект от класа **Color.**
  - `g.Clear(Color.Red);` **или**
  - `g.Clear(Color.FromArgb(20, 140, 60));`

Добавяне на програмен код

```
private void FormColor_Paint(object sender, PaintEventArgs e)
{
        Graphics g = e.Graphics;
        int a = 255, freq = 20, rep = 20;
        Random c = new Random();
        for (int i = 1; i <=  rep; i++)
        {
          g.Clear(Color.FromArgb(c.Next()%a, c.Next() % a, c.Next() % a));
          System.Threading.Thread.Sleep(freq);
         }
}
```

- Този подход се изполва при създаване на анимационни ефекти
- Методът `System.Threading.Thread.Sleep(<време>)` - забавя изпълнението на следващия оператор за времето в милисекунди, зададено като параметър
- Всяко извикване Next() на обекта с от класа Random **връща случайно число,** което се превръща в **интензитет (0 до 255**), като се вземе остатъка му по модул 255

Геометрични фигури
Изчертаване на права линия

- Използване на метод **DrawLine** от класа **Graphics**

Име на метод	**Описание**
DrawLine(Pen, Point, Point)	Чертае линия, свързваща две точки
DrawLine(Pen, PointF, PointF)	Чертае линия, свързваща две точки
DrawLine(Pen, Int32, Int32, Int32, Int32)	Чертае линия, свързваща двете точки, определени от координатни двойки
DrawLine(Pen, Single, Single, Single, Single)	Чертае линия, свързваща двете точки, определени от координатни двойки, pen задава цвета и дебелината на линията

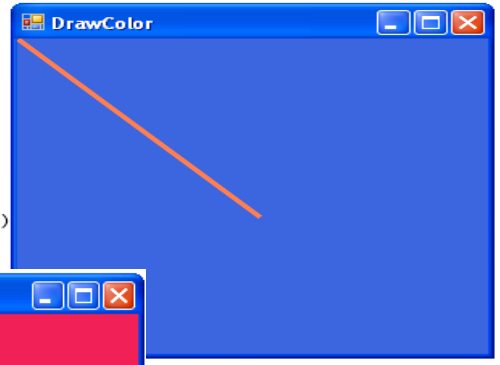
```
void DrawLine(Pen p, int x1, int y1, int x2, int y2);
```

- Писалката p задава цвета и дебелината на линията. Първата двойка са координати на първата точка, а втората двойка числа на втората

Примери

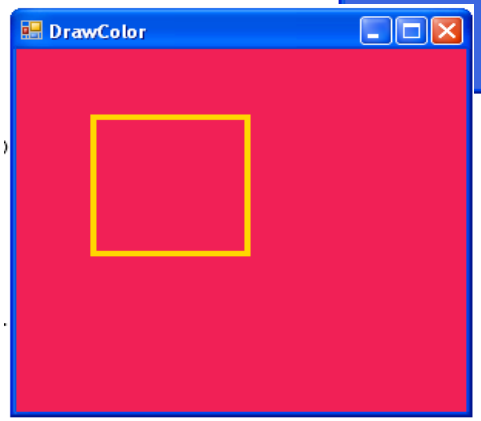
Добавете: using System.Drawing.Drawing2D

```
Pen p = new Pen(Color.Coral, 4);
g.DrawLine(p, 0, 0, 150, 150);
```



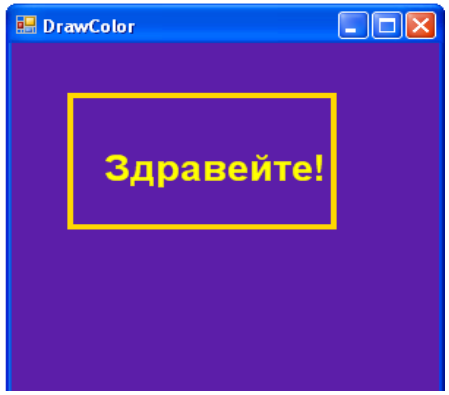
- Оцветява линията в цвят Корал с координати (0,0) до (150,150)

```
Pen p = new Pen(Color.Gold, 4);
g.DrawRectangle(p, 50, 50, 100, 100);
```



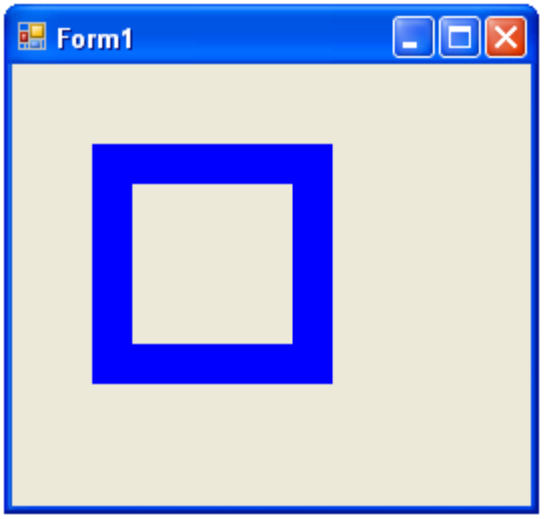
- // изписва на екрана Здравейте

```
g.DrawRectangle(p, 40, 40, 180, 100);
Graphics g2 = e.Graphics;
g2.SmoothingMode =SmoothingMode.AntiAlias;
Brush brush = new SolidBrush(Color.Yellow);
Font font = new Font("Arial", 20, FontStyle.Bold)
g2.DrawString("Здравейте!", font, brush, 60, 80);
```



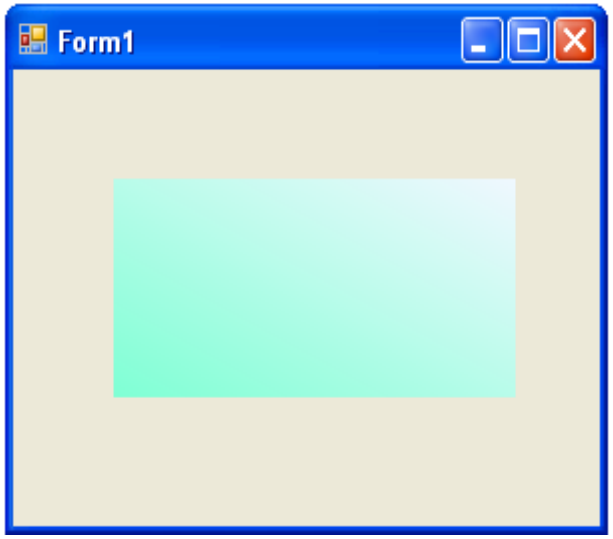
Изчертаване на правоъгълник

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
      Graphics g = e.Graphics;
      //създаване на писалка
      Pen pn2 = new Pen( Color.Blue, 20 );
      //изчертаване на правоъгълник
      g.DrawRectangle(pn2, 50, 50, 100, 100);
}
```



Изчертаване и запълване на правоъгълник

```
protected override void OnPaint(PaintEventArgs pe)
{
     Graphics g = pe.Graphics;
     Rectangle rect = new Rectangle(50, 50, 200, 100);
     LinearGradientBrush lBrush = new LinearGradientBrush(rect,
                         Color.AliceBlue,  Color.Aquamarine,
                         LinearGradientMode.BackwardDiagonal);
     g.FillRectangle(lBrush, rect);
}
```




```
Rectangle rect = new Rectangle(50, 50, 200, 200);
LinearGradientBrush lBrush = new
          LinearGradientBrush(rect, Color.Purple,
          Color.Aquamarine,
          LinearGradientMode.BackwardDiagonal);
 g.FillRectangle(lBrush, rect);
```

Изчертаване на елипса

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
      Graphics g = e.Graphics;
      //създаване на писалка
      Pen pn2 = new Pen( Color.Red, 100 );
      //създаване на инстанция обект от клас Rectangle
      Rectangle rect = new Rectangle(100, 50, 200, 100);
      //изчертаване на елипса
      g.DrawEllipse(pn2, rect);
}
```

Изписване на текст

- Създава се обект от тип Graphics

```
Graphics g = e.Graphics;
```

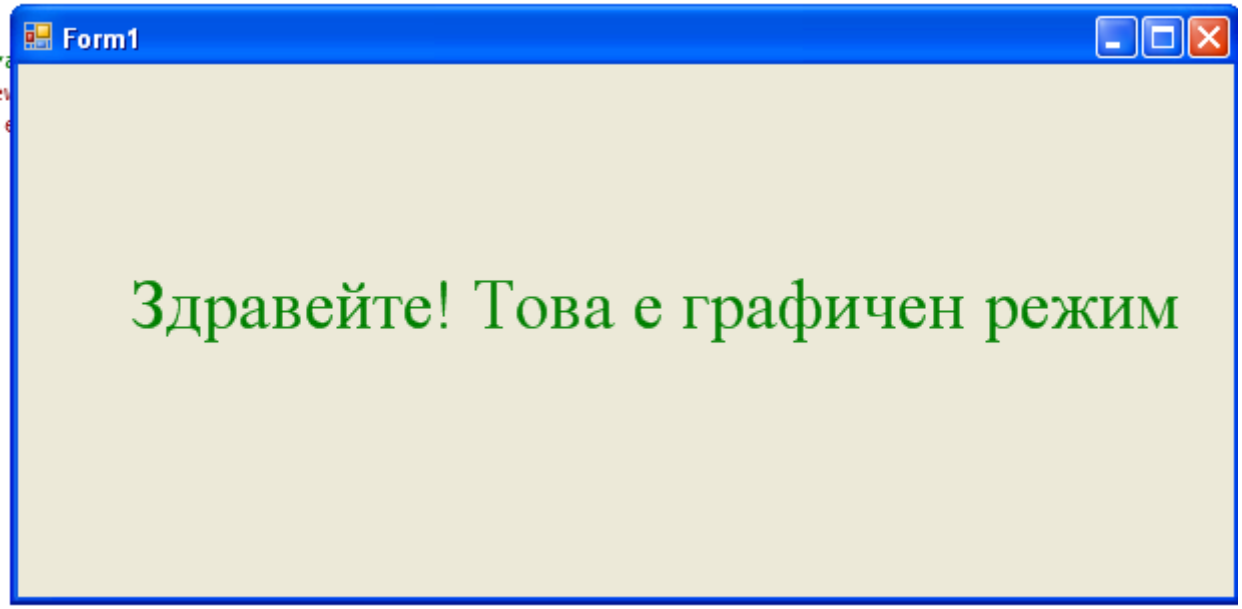
- Използва се метода **DrawString ()**
- Синтаксисът за този метод е следния:

```
g.DrawString(string, font, brush, x, y);
```

- Аргумента **string** определя **текста,** който ще се рисува
- **font** дефинира **стила на шрифта**. Трябва да се създаде обект от клас Font
- Обекта **brush** е подобен на обект Pen, използван за рисуване на различни фигури, но той **определя шаблон на запълване**
- Сройностите **x и y** задават **горният ляв ъгъл на текста**

Изписване на текст

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
      Graphics g = e.Graphics;
      Font font = new Font("Times New Roman", 26);
      g.DrawString("Здравейте! Това е графичен режим", font,
                        new SolidBrush(Color.Green), 50,100);
}
```



Пример – Движещ се обект 1/3

- Да се изчертае и запълни геометрична фигура на екрана

```
public partial class Form1 : Form
    {
        private int x;
        private int y;
        enum Position
        {
            Left, Right, Up, Down
        }
        private Position objPosition;

public Form1()
    {
         x = 50;
         y = 50;
         //objPosition = Position.Right;
         objPosition = Position.Down;
         InitializeComponent();
    }
```

Пример – Движещ се обект 2/3

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
        Graphics g = e.Graphics;
         g.FillRectangle(Brushes.Violet, x, y, 100, 100);
         // g.DrawImage(new Bitmap("джип.jpg"), x, y, 100, 100);
}
```

Пример – Движещ се обект 3/3

- Да се създаде компонент Timer

```
private void timer1_Tick(object sender, EventArgs e)
    {
        if (objPosition == Position.Right)
        {    x += 10;         }
        else if (objPosition == Position.Left)
        {    x -= 10;         }
        else if (objPosition == Position.Up)
        {     y -= 10;        }
        else if (objPosition == Position.Down)
        {     y += 10;        }
        Invalidate();
    }
```

Печатане на принтер

- Използват се 3 ключови класа:
  - **`PrintDialog`**
    - Стандартен диалог за печатане на принтер
  - **`PrintController`**
    - Управлява процеса на печатане и активира събития, свързани с него
    - Предоставя **`Graphics`** повърхността
  - **`PrintDocument`**
    - Описва характеристиките на отпечатвания документ
    - Съдържа **`PrinterSettings`** върнати от **`PrintDialog`**

Благодаря за вниманието!