C#.6 Графика.

Платформа .**NET** обеспечивает целый набор пространства имен для поддержки визуализации двумерной графики.
Для рисования мы будем использовать пространство имен **Drawing**, нам этого достаточно, так как мы будем рисовать простейшие рисунки. Во время рисования следует обратить внимание на следующее, так как некоторые фигуры нам понадобится заливать цветом, то тут важен порядок, обязательно сначала выполнять заливку, а после обводку.
Для рисования мы будем использовать карандаш **Pen**, точнее статические свойства семейства **Pens**.

Например:
Pens.Blue;//карандаш синего цвета

Для заливки используем кисть **Brush**, аналогично будем использовать его статические свойства **brushes**.

Например:
Brushes.Blue;//кисть синего цвета.

Непосредственно для рисования используются следующие объекты:

Линия:
DrawLine();.// Для рисования необходимы 5 параметров:
1.цвет линии, используем семейство Pens.
2.Координаты верхнего левого угла
3.Координаты нижнего правого угла
Например:
DrawLine(Pens.Black, 100, 100, 500, 500);

Прямоугольник: (не забываем что сначала заливка а потом обаодка)
FillRectangle();Для рисования необходимы 5 параметров:
1.цвет заливки, используем семейство Brushes.
2.Координаты верхнего левого угла
3.Ширина и высота
Например:
FillRectangle(Brushes.Beige, 100, 100, 500, 500);//залили прямоугольную область
DrawRectangle(Pens.Red, 100, 100, 500, 500);// выполнили обводку

Эллипс:
FillEllipse();Для рисования необходимы 5 параметров:
1.цвет заливки, используем семейство Brushes.
2.Координаты верхнего левого угла
3.Ширина и высота
Например:
FillEllipse(Brushes.Beige, 100, 100, 500, 500);//залили прямоугольную область
DrawEllipse(Pens.Red, 100, 100, 500, 500);// выполнили обводку

Сектор эллипса:
FillPie();Для рисования необходимы 7 параметров:
1.Цвет заливки, используем семейство Brushes.
2.Координаты верхнего левого угла
3.Ширина и высота
4.Угол начала сектора
5.Угол конца сектора
Например:
FillPie(Brushes.Beige, 100, 100, 500, 500, 0, 90);//залили прямоугольную область
DrawPie(Pens.Red, 100, 100, 500, 500, 0, 90);// выполнили обводкуМногоугольник:

Многоугольник, строится по массиву тоыек, поэтому сначала нужно создать массив
Point [] tt={new Point(20, 10), new Point(10, 40),new Point(45,100),new Point(50б 80)};
А после рисуем многоугольник
g.**DrawPolygon**(Pens.Green,tt);
И при необходимости его заливаем
g.**FillPolygon**(Pens.Green,tt);

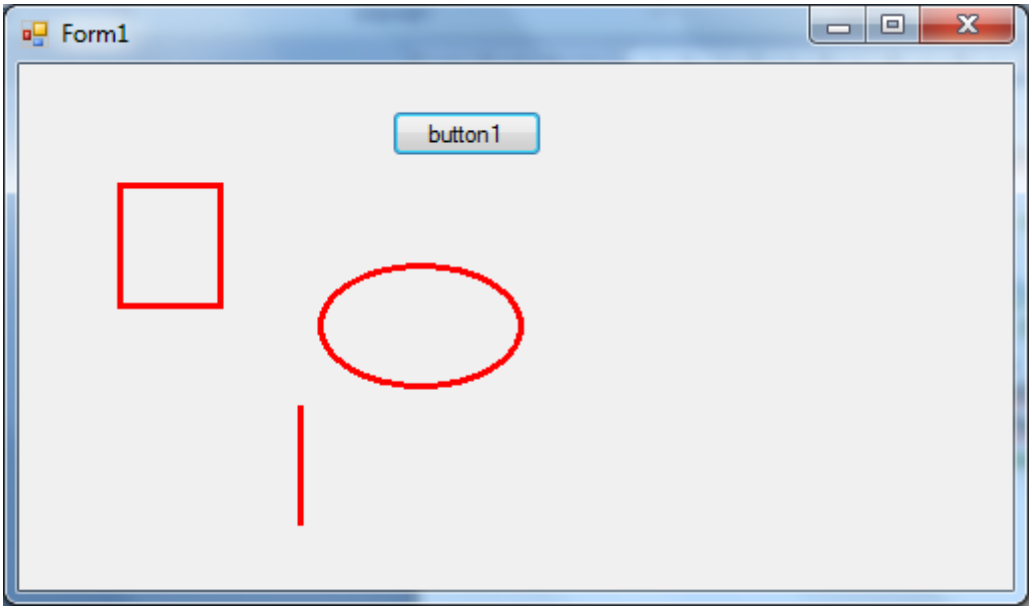
Масштабируемое изображение
Все что мы рассмотрели позволяет рисовать графические примитивы на C#. Но есть одно но, если после создания изображения мы попытаемся изменить размер формы, то мы увидим что размер изображения не меняется. А по мере уменьшения формы изображение срезается, то есть ее часть оказывается, как бы за пределами формы и мы его не видим.
Проблема в том что во время масштабирования формы уменьшается не единица измерения(это не возможно в принципе, так как размер формы измеряется в пикселях), а уменьшается количество пикселей.
Для того чтобы обойти эту проблему нам придется создать свою систему координат. Начало останется там же в верхнем левом углу. А вот единицу измерения мы изменим, мы привяжем ее непосредственно к размерам формы и, тем самым при изменении размеров формы будет меняться и наша единица измерения.
Первое что стоит понимать, мы рисуем только на клиентской части формы, то есть нельзя рисовать на заголовке, меню и т.п.

Пример 1. При нажатии кнопки, появляются фигуры (квадрат, эллипс, линия)

```
    private void button2_Click(object sender, EventArgs e)
      {

     Graphics g = this.CreateGraphics();

       // Подготавливаем перо, рисующее красную линию толщиной 3
пикселя
       Pen redPen = new Pen(Color.Red, 3);
       // С помощью пера рисуем прямую линию, прямоугольник и эллипс
       g.DrawLine(redPen, 140, 170, 140, 230);
       g.DrawRectangle(redPen, 50, 60, 50, 60);
       g.DrawEllipse(redPen, 150, 100, 100, 60);
       g.Dispose(); // Очистка
      }
```



Пример 2. Ветка параболы.

```
private void button2_Click(object sender, EventArgs e)
        {
            Graphics gr = this.CreateGraphics();
            Pen myPen = new Pen(Color.BlueViolet);
            int k = 100;
            float xmin = 0, xmax = 2, ymin = 0, ymax = 4;
            float xstep = (xmax - xmin) / k;
            float kx = this.Width / xmax, ky = this.Height / ymax;
            float x1 = xmin, y1 = x1 * x1, x2, y2;
            for (int i = 0; i < k; i++)
            {
                x2 = x1 + xstep;
                y2 = x2 * x2;
     gr.DrawLine(myPen, kx * x1, this.Height - ky * y1, kx * x2,
this.Height - ky * y2);
                x1 = x2;
                y1 = y2;
            }
        }
```

Пример 3. «Летящая окружность»

```
private void button1_Click(object sender, EventArgs e)
        {
 Graphics g = this.CreateGraphics();
 Pen redPen = new Pen(Color.Red, 3);  // красное перо

 for (int i = 0; i < 100; i+=1) // координаты центра изменяются
            {
          g.DrawEllipse(redPen, i+100, i+40, 10, 10);// рисуем эллипс

 for (int j = 0; j < 10000000; j++) //пустой цикл для задержки времени
                {
                }
                g.Clear(Color.White); //стираем нарисованный эллипс
            }
g.Dispose();
        }
```

Задание 1. Нарисовать снеговика (три закрашенных шара, нос морковкой, на голове колпак)

Задание 2. Доработать программу с графиком. Программа должна рисовать параболу y=a*x (с двумя ветками) и параметр

“a” считывать с текстового поля. (Желательно вывести оси координат)

Задание 3. Сделать эффект летящего объекта, который, долетая до края формы или любой другой фигуры, отлетает от нее.

Задание 4. Сделать эффект дождя или мерцающего звездного неба. (случайным образом появляются и исчезают фигуры)

Задание 5. При движении и нажатии на кнопку мыши, рисуется линия.