Лабораторная работа №4

Основы работы с графикой в C# с использованием Windows Forms. Класс Graphics.

1. Цель

ознакомиться с компонентами и методами работы с графикой в C# с использованием Windows Form в простых Windows приложениях. Ознакомится с свойствами и методами класса Graphics.

2. Методические указания

1. При изучении языка программирования C# будет использоваться интегрированная среда разработки программного обеспечения Microsoft Visual Studio Express 2013 для Windows Desktop. Будут использованы основные элементы .NET Framework и связь C# с элементами платформы .NET.
2. По окончанию работы сохраните все созданные файлы на своих носителях.
3. Защита лабораторной работы производится только при наличии электронного варианта работающего скрипта.

3. Теоретические сведения

Пространство имен **using System.Drawing** обеспечивает доступ к функциональным возможностям графического интерфейса GDI+. Пространства имен System.Drawing.Drawing2D, System.Drawing.Imaging, и System.Drawing.Text обеспечивают дополнительные функциональные возможности.

Класс Graphics предоставляет методы рисования на устройстве отображения. Такие классы, как Rectangle и Point, инкапсулируют элементы GDI+. Класс Pen используется для рисования линий и кривых, а классы, производные от абстрактного класса Brush, используются для заливки фигур.

Синтаксис

C# | C++ | F# | VB

```
public sealed class Graphics : MarshalByRefObject,
        IDeviceContext, IDisposable
```

Свойства

Имя	Описание
Clip	Возвращает или задает объект Region, ограничивающий область рисования данного объекта Graphics.
ClipBounds	Получает структуру RectangleF, которая заключает в себе вырезанную область данного объекта Graphics.
CompositingMode	Получает значение, задающее порядок рисования сложных изображений в данном объекте Graphics.
CompositingQuality	Возвращает или задает качество отрисовки составных изображений, которые выводятся в данном объекте Graphics.
DpiX	Получает горизонтальное разрешение данного объекта Graphics.
DpiY	Получает вертикальное разрешение данного объекта Graphics.

InterpolationMode	Возвращает или задает режим интерполяции, связанный с данным объектом Graphics.
IsClipEmpty	Получает значение, которое указывает, является ли вырезанная область данного объекта Graphics пустой.
IsVisibleClipEmpty	Получает значение, которое указывает, является ли видимая вырезанная область данного объекта Graphics пустой.
PageScale	Возвращает или задает масштабирование между универсальными единицами и единицами страницы для данного объекта Graphics.
PageUnit	Возвращает или задает единицу измерения для координат страницы данного объекта Graphics.
PixelOffsetMode	Возвращает или задает значение, которое задает порядок смещения пикселей во время отрисовки данного объекта Graphics.
RenderingOrigin	Возвращает или задает начало координат при отрисовке данного объекта Graphics для кистей сглаживания цветовых переходов и штриховки.
SmoothingMode	Возвращает или задает качество отрисовки данного объекта Graphics.
TextContrast	Возвращает или задает значение коррекции показателя гаммы для отрисовки текста.
TextRenderingHint	Возвращает или задает режим отрисовки текста, связанного с данным объектом Graphics.
Transform	Возвращает или задает копию геометрического универсального преобразования объекта Graphics.
VisibleClipBounds	Получает ограничивающий прямоугольник видимой вырезанной области данного объекта Graphics.

Основные методы

Имя	Описание
AddMetafileComment	Добавляет комментарий к текущему объекту Metafile.
Clear	Очищает всю поверхность рисования и выполняет заливку поверхности указанным цветом фона.
Dispose	Освобождает все ресурсы, используемые данным объектом Graphics.
DrawArc(Pen, Rectangle, Single, Single)	Рисует дугу, которая является частью эллипса, заданного структурой Rectangle.
DrawArc(Pen, RectangleF, Single, Single)	Рисует дугу, которая является частью эллипса, заданного структурой RectangleF.
DrawArc(Pen, Int32, Int32, Int32, Int32, Int32, Int32)	Рисует дугу, которая является частью эллипса, заданного парой координат, шириной и высотой.
DrawArc(Pen, Single, Single, Single, Single, Single, Single)	Рисует дугу, которая является частью эллипса, заданного парой координат, шириной и высотой.
DrawBezier(Pen, Point, Point, Point, Point)	Рисует кривую Безье, определяемую четырьмя структурами Point.
DrawBezier(Pen, PointF, PointF,	Рисует сплайн Безье, определяемый четырьмя

PointF, PointF)	структурами PointF.
DrawBezier(Pen, Single, Single, Single, Single, Single, Single, Single)	Строит кривую Безье, определяемую четырьмя упорядоченными парами координат, которые представляют собой точки.
DrawBeziers(Pen, Point[])	Рисует последовательность кривых Безье из массива структур Point.
DrawBeziers(Pen, PointF[])	Рисует последовательность кривых Безье из массива структур PointF.
DrawClosedCurve(Pen, Point[])	Строит замкнутую фундаментальную кривую, определяемую массивом структур Point.
DrawClosedCurve(Pen, PointF[])	Строит замкнутую фундаментальную кривую, определяемую массивом структур PointF.
DrawClosedCurve(Pen, Point[], Single, FillMode)	Строит замкнутую фундаментальную кривую, определяемую массивом структур Point с указанным натяжением.
DrawClosedCurve(Pen, PointF[], Single, FillMode)	Строит замкнутую фундаментальную кривую, определяемую массивом структур PointF с указанным натяжением.
DrawCurve(Pen, Point[])	Строит фундаментальную кривую через точки указанного массива структур Point.
DrawCurve(Pen, PointF[])	Строит фундаментальную кривую через точки указанного массива структур PointF.
DrawCurve(Pen, Point[], Single)	Строит фундаментальную кривую через точки указанного массива структур Point с указанным натяжением.
DrawCurve(Pen, PointF[], Single)	Строит фундаментальную кривую через точки указанного массива структур PointF с указанным натяжением.
DrawCurve(Pen, PointF[], Int32, Int32)	Строит фундаментальную кривую через точки указанного массива структур PointF. Смещение при рисовании начинается от начала массива.
DrawCurve(Pen, Point[], Int32, Int32, Single)	Строит фундаментальную кривую через точки указанного массива структур Point с указанным натяжением.
DrawCurve(Pen, PointF[], Int32, Int32, Single)	Строит фундаментальную кривую через точки указанного массива структур PointF с указанным натяжением. Смещение при рисовании начинается от начала массива.
DrawEllipse(Pen, Rectangle)	Рисует эллипс, определяемый ограничивающей структурой Rectangle.
DrawEllipse(Pen, RectangleF)	Рисует эллипс, определяемый ограничивающей структурой RectangleF.
DrawEllipse(Pen, Int32, Int32, Int32, Int32)	Рисует эллипс, определяемый ограничивающим прямоугольником, заданным с помощью координат для верхнего левого угла прямоугольника, высоты и ширины.
DrawEllipse(Pen, Single, Single,	Рисует эллипс, определенный ограничивающим

Single, Single)	прямоугольником, заданным с помощью пары координат, ширины и высоты.
DrawIcon(Icon, Rectangle)	Формирует изображение, представленное указанным объектом Icon в пределах области, заданной структурой Rectangle.
DrawIcon(Icon, Int32, Int32)	Формирует изображение, представленное указанным объектом Icon, расположенным по указанным координатам.
DrawIconUnstretched	Формирует изображение, представленное указанным объектом Icon без его масштабирования.
DrawLine(Pen, Point, Point)	Проводит линию, соединяющую две структуры Point.
DrawLine(Pen, PointF, PointF)	Проводит линию, соединяющую две структуры PointF.
DrawLine(Pen, Int32, Int32, Int32, Int32)	Проводит линию, соединяющую две точки, задаваемые парами координат.
DrawLine(Pen, Single, Single, Single, Single)	Проводит линию, соединяющую две точки, задаваемые парами координат.
DrawLines(Pen, Point[])	Рисует набор сегментов линий, которые соединяют массив структур Point.
DrawLines(Pen, PointF[])	Рисует набор сегментов линий, которые соединяют массив структур PointF.
DrawPath	Рисует объект GraphicsPath.
DrawPie(Pen, Rectangle, Single, Single)	Рисует сектор, который определяется эллипсом, заданным структурой Rectangle и двумя радиальными линиями.
DrawPie(Pen, RectangleF, Single, Single)	Рисует сектор, определяемый эллипсом, заданным структурой RectangleF и двумя радиальными линиями.
DrawPie(Pen, Int32, Int32, Int32, Int32, Int32, Int32)	Рисует сектор, определяемый эллипсом, который задан парой координат, шириной, высотой и двумя радиальными линиями.
DrawPie(Pen, Single, Single, Single, Single, Single, Single)	Рисует сектор, определяемый эллипсом, который задан парой координат, шириной, высотой и двумя радиальными линиями.
DrawPolygon(Pen, Point[])	Рисует многоугольник, определяемый массивом структур Point.
DrawPolygon(Pen, PointF[])	Рисует многоугольник, определяемый массивом структур PointF.
DrawRectangle(Pen, Rectangle)	Рисует прямоугольник, определяемый структурой Rectangle.
DrawRectangle(Pen, Int32, Int32, Int32, Int32)	Рисует прямоугольник, который определен парой координат, шириной и высотой.
DrawRectangle(Pen, Single, Single, Single, Single)	Рисует прямоугольник, который определен парой координат, шириной и высотой.
DrawRectangles(Pen, Rectangle[])	Рисует набор прямоугольников, определяемых

	структурами Rectangle.
DrawRectangles(Pen, RectangleF[])	Рисует набор прямоугольников, определяемых структурами RectangleF.
DrawString(String, Font, Brush, PointF)	Создает указываемую текстовую строку в заданном месте с помощью определяемых объектов Brush и Font.
DrawString(String, Font, Brush, RectangleF)	Рисует заданную текстовую строку в указанном прямоугольнике с помощью определяемых объектов Brush и Font.
DrawString(String, Font, Brush, PointF, StringFormat)	Рисует заданную текстовую строку в заданном месте с помощью определяемых объектов Brush и Font, используя атрибуты форматирования заданного формата StringFormat.
DrawString(String, Font, Brush, RectangleF, StringFormat)	Рисует заданную текстовую строку в заданном прямоугольнике с помощью определяемых объектов Brush и Font, используя атрибуты форматирования заданного формата StringFormat.
DrawString(String, Font, Brush, Single, Single)	Создает указываемую текстовую строку в заданном месте с помощью определяемых объектов Brush и Font.
EndContainer	Закрывает текущий графический контейнер и восстанавливает состояние данного объекта Graphics, которое было сохранено при вызове метода BeginContainer.

Класс Graphics предоставляет методы для вывода объектов в устройстве отображения. Объект Graphics связан с конкретным контекстом устройства. Объект Graphics можно получить путем вызова метода Control.CreateGraphics для объекта, который наследует из объекта System.Windows.Forms.Control, или путем обработки события Control.Paint элемента управления и обращения к свойству Graphics класса System.Windows.Forms.PaintEventArgs. Можно также создать объект Graphics из изображения, используя метод FromImage. Дополнительные сведения о создании объекта Graphics см. в разделе Практическое руководство. Создание объектов Graphics для рисования.

Используя объект Graphics, можно нарисовать много разных фигур и линий. Дополнительные сведения о рисовании линий и фигур см. в описании метода Draw графического элемента для линии или фигуры, которую требуется нарисовать. К этим методам относятся DrawLine, DrawArc, DrawClosedCurve, DrawPolygon и DrawRectangle. Дополнительные сведения о рисовании линий и фигур см. в разделах Рисование линий и фигур с помощью пера и Использование кисти для заливки фигур.

Рисунки и значки можно также рисовать с помощью методов DrawImage и DrawIcon, соответственно. Чтобы выполнить передачу данных о цвете блоками битов с экрана на поверхность рисования объекта Graphics, см. CopyFromScreen. Дополнительные сведения о рисовании рисунков с помощью объекта Graphics см. в разделе Работа с растровыми и векторными изображениями.

Кроме того, можно манипулировать системой координат, используемой объектом Graphics. Дополнительные сведения о системе координат и манипуляциях с ней см. в разделе Системы координат и преобразования.

Пример использования класса Graphics:

В формах Windows существует следующая система позиционирования комппонентов по координатам рис. 4.1.

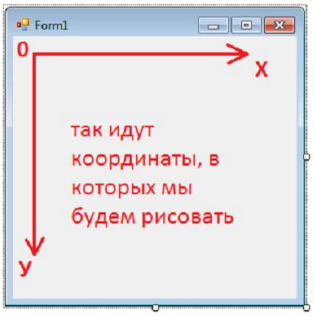


Рис. 4.1. Система координат в Windows формах

Создайте форму и добавьте на нее, с панели toolBox, перетаскиваем, элемент Button рис. 4.2.

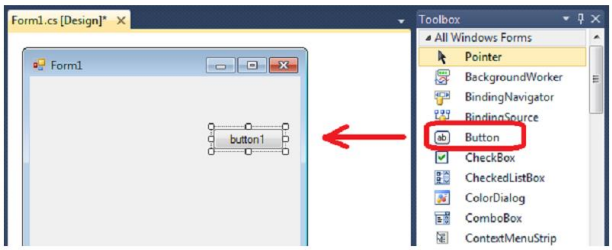
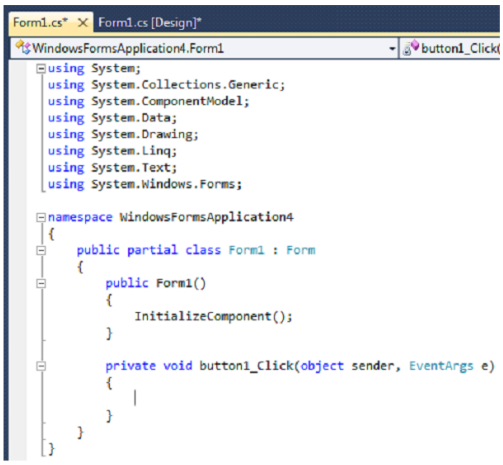


Рис. 4.2. Добавление кнопки на форму

Кликаем два раза по создавшейся кнопке button1 и переходим к файлу Form1.cs



Добавляем в функцию button1_Click, следующий код:

```
Graphics g = this.CreateGraphics(); //g - то, где мы будем рисовать
Pen a = new Pen(Color.Red, 6); // a - ручка которой будем рисовать, 6 - толщина
g.DrawRectangle(a, 100, 100, 50, 100); //рисуем прямоугольник, ручкой a, левый верхний
//угол которого находится в x=100, y=100, 50 - его ширина, 100 - высота
```

Запускаем программу, затем на появившемся окне, нажимаем кнопку button1 рис. 4.3:

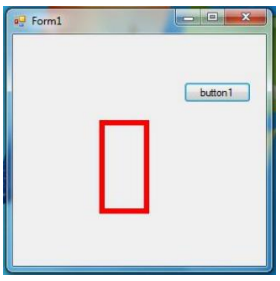


Рис. 4.3. Результат работы программы

добавляем еще код

```
Point[] polygon =                   //создаем массив точек, по которым
 {                                  //затем будем рисовать полигон
     new Point(200, 250),
     new Point(250,  220),
     new Point(270,  270),
     new Point(200, 300),
     new Point(200, 250),
 };

Pen b = new Pen(Color.Green, 3);
Pen c = new Pen(Color.Blue, 5); //добавили еще одну ручку
g.DrawEllipse(b, 200, 100, 200, 100); //рисуем эллипс, по аналогии с прямоугольником
g.DrawPolygon(c, polygon); //рисуем полигон по массиву точек
```

Результат работы программы, после добавления кода рис. 4.4.

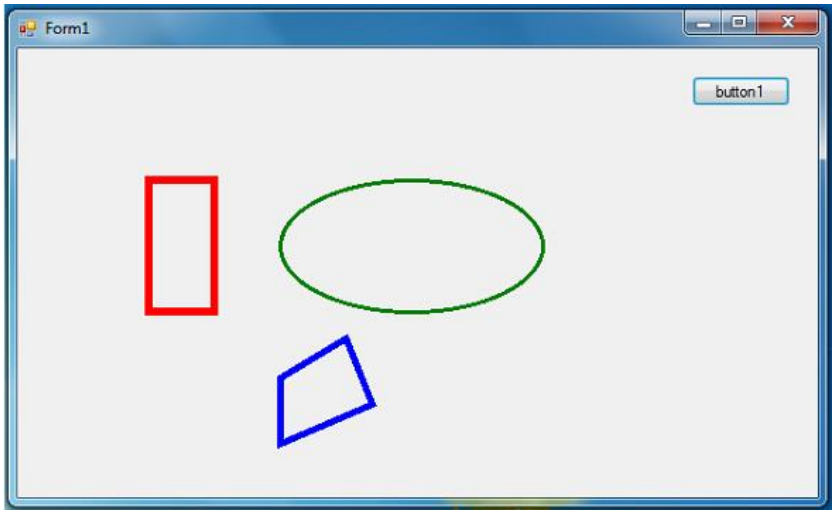


Рис. 4.4. Результат работы программы

Создание простейшей анимации

Подключаем библиотеку **System.Threading** содержащую классы для работы с многопоточными приложениями (в том числе понадобится для создания анимации). Необходимо щелкнуть по рамке окна, открыть вкладку Properties и поменять цвет фона на белый рис. 4.5.

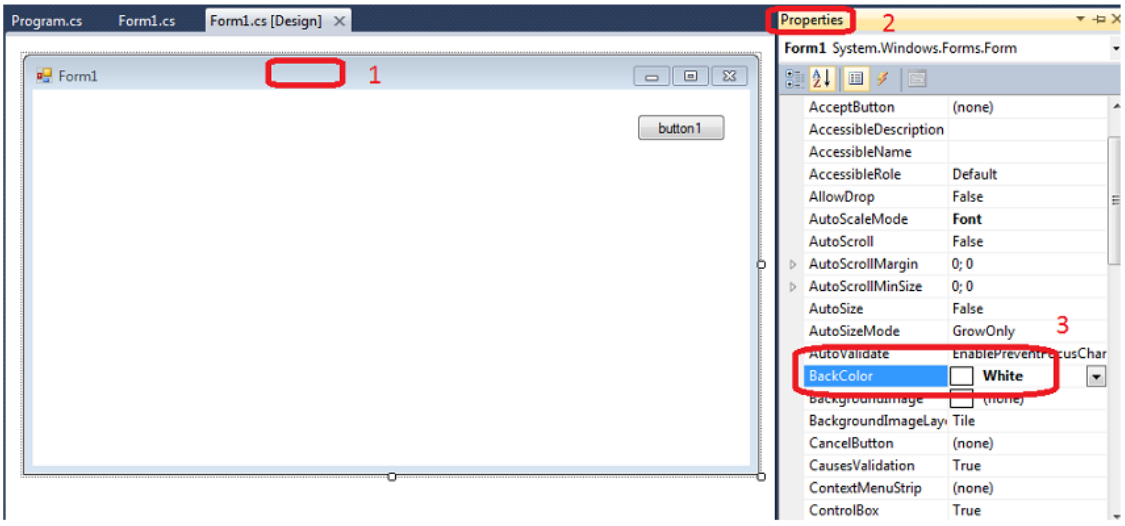


Рис. 4.5. Изменение цвета формы

Добавляем следующий код:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Threading;
namespace WindowsFormsApplication4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Graphics g = this.CreateGraphics();
            Pen a = new Pen(Color.Red, 6); //ручка которой мы будем рисовать
            Pen b = new Pen(Color.White, 6); //ручка которой мы будем "стирать",
            int step = 0; //шаг - на который перемещается наш объект

            for (int i = 0; i < 20; i++)
            {
                step += 10;
                g.DrawEllipse(a, 50 + step, 100, 100, 100); //рисуем эллипс,
                Thread.Sleep(100); //останавливаем выполнение программы на 100 милли секунд
                g.DrawEllipse(b, 50 + step, 100, 100, 100); //"стираем" эллипс,
            }
        }
    }
}
```

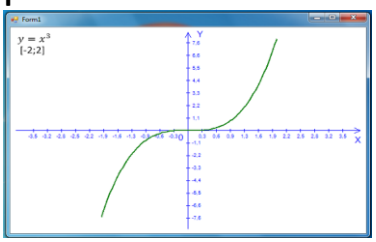
Результат работы программы:

Если мы во время анимации попробуем переместить окно по экрану или закрыть его, у нас не чего не получится. Причина этого в том, что программа выполняется в один поток, который занят рисованием. Переделаем код следующим образом.

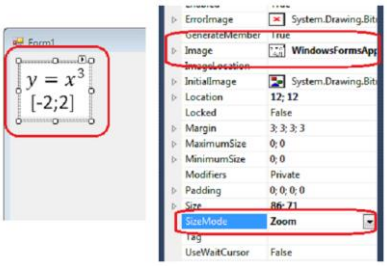
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Threading;
namespace WindowsFormsApplication4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Thread a = new Thread(one); //создали поток и отправили в него ф-ию
            a.Start(); //запустили поток
        }

        void one() //добавили функцию и поместили в нее код отрисовки
        {
            Graphics g = this.CreateGraphics();
            Pen a = new Pen(Color.Red, 6); //ручка которой мы будем рисовать
            Pen b = new Pen(Color.White, 6); //ручка которой мы будем "стирать",
            int step = 0; //шаг - на который перемещается наш объект

            for (int i = 0; i < 20; i++)
            {
                step += 10;
                g.DrawEllipse(a, 50 + step, 100, 100, 100); //рисуем эллипс,
                Thread.Sleep(100); //останавливаем выполнение программы на 100 милли секунд
                g.DrawEllipse(b, 50 + step, 100, 100, 100); //"стираем" эллипс,
            }
        }
    }
}
```

Теперь во время анимации окно можно перемещать по экрану или закрыть его.

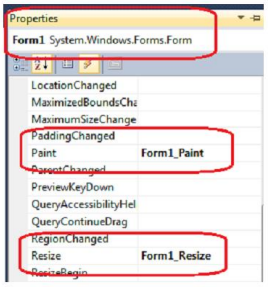
Построение графика функции



Построим график заданной функции, при изменении размера окна, график помещается в окне. В окне с функцией отобразим картинку с уравнением. Создадим новую форму и добавим на нее элемент pictureBox, в свойствах SizeMode -Zoom, Image – укажем картинку с формулой. (Картинку можно создать в Word'e (вставка->формула), а затем сделатьскриншот , сохранить и вставить сюда).



Далее заходим в события формы и два раза кликаем по событиям Paint и Resize, в итоге у нас создадутся две функции, третью – one(); мы создадим сами (в ней будем рисовать), и будем вызывать ее из двух других. Функция Paint выполняется при первоначальной отрисовке окна, Resize – при изменении размера окна.



Вид файла Form1.cs

```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            one();
        }

        private void Form1_Resize(object sender, EventArgs e)
        {
            one();
        }

        void one()
        {

        }
    }
}
```

Добавляем в функцию **one()** код для создания необходимых для рисования Инструментов

```
            Graphics g = this.CreateGraphics(); //задаем область рисования
            g.Clear(Color.White); //очищаем область (заливаем белым цветом)
            Pen a = new Pen(Color.Blue, 1); //ручка для рисования осей
            Pen b = new Pen(Color.Green, 2); //ручка для рисования графика
            Font drawFont = new Font("Arial", 12); //шрифт которым будем подписывать оси
            Font signatureFont = new Font("Arial", 7); //шрифт которым будем подписывать деления
оси
            SolidBrush drawBrush = new SolidBrush(Color.Blue); //цвет этого шрифта
            StringFormat drawFormat = new StringFormat(); //формат букв (для подписывания)
            drawFormat.FormatFlags = StringFormatFlags.DirectionRightToLeft; //направление текста
оси
            int sizeWidth = Form1.ActiveForm.Width; //ширина окна программы
            int sizeHeight = Form1.ActiveForm.Height; //высота окна программы
            //x, y - координаты центра (точки 0)
            Point center = new Point(((int)(sizeWidth / 2) - 8), (int)((sizeHeight / 2) - 19));
```

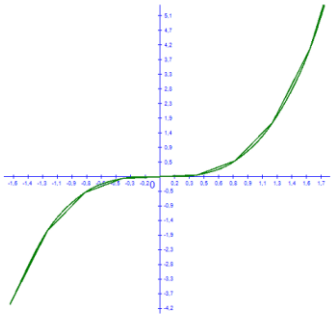
Добавляем код для рисования и подписи осей

```
            g.DrawLine(a, 10, center.Y, center.X, center.Y);                //ось X-
            g.DrawLine(a, center.X, center.Y, 2 * center.X - 10, center.Y); //ось X+
            g.DrawLine(a, center.X, 10, center.X, center.Y);                //ось Y+
            g.DrawLine(a, center.X, center.Y, center.X, 2 * center.Y - 10); //ось Y-
            g.DrawString("X", drawFont, drawBrush, new PointF(2 * center.X - 5, center.Y + 10),
drawFormat);  //подписали x
            g.DrawString("Y", drawFont, drawBrush, new PointF(center.X + 30, 5), drawFormat);
//подписали y
            // аналогично подписать 0

            g.DrawLine(a, center.X, 10, center.X + 5, 20);              //стрелки: y+
            g.DrawLine(a, center.X, 10, center.X - 5, 20);             // y-
            // аналогично нарисовать стрелки x+ x-


            int stepForAxes = 25; //расстояние между делениями на осях
            int lenghtShtrih = 3; //половина длины штришка деления
            int maxValueForAxesX = 4; //максимальное значение по оси X\
            int maxValueForAxesY = 9; //по оси Y

            float oneDelenieX = (float)maxValueForAxesX / ((float)center.X / (float)stepForAxes);
//то, чем подписывать деления X
            float oneDelenieY = (float)maxValueForAxesY / ((float)center.Y / (float)stepForAxes);
//то, чем подписывать деления Y

            for (int i = center.X, j = center.X, k = 1; i < 2 * center.X - 30; j -= stepForAxes, i
+= stepForAxes, k++)
            {
                g.DrawLine(a, i, center.Y - lenghtShtrih, i, center.Y + lenghtShtrih); //рисуем
штрихи по оси X вправо от центра
                g.DrawLine(a, j, center.Y - lenghtShtrih, j, center.Y + lenghtShtrih); //рисуем
штрихи по оси X влево от центра

                if (i < 2 * center.X - 55)
                {
                    g.DrawString((k * oneDelenieX).ToString("0.0"), signatureFont, drawBrush, new
PointF(i + stepForAxes + 9, center.Y + 6), drawFormat); //подписываем деления +
                    g.DrawString(((k * oneDelenieX).ToString("0.0").ToString() + "-"),
signatureFont, drawBrush, new PointF(j - stepForAxes + 9, center.Y + 6), drawFormat); //-
                }
            }

            //рисуем штрихи по оси Y
            //подписываем деления по оси Y
```

Далее рассчитаем значение функции на заданном интервале, и построим по этим значениям график.

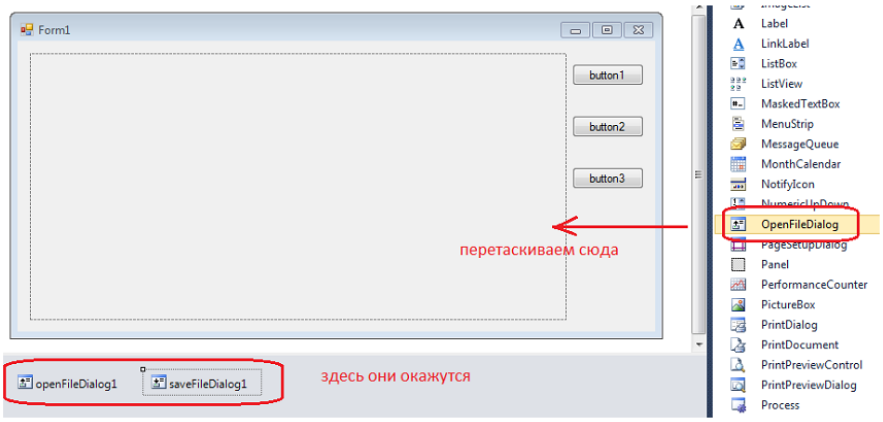
```
            int numOfPoint = 100; //количество точек для расчета значения функции

            float[] first = new float[numOfPoint];//точки для расчета

            for (int i = 0; i < numOfPoint; i++)
            {
                first[i] = (float)maxValueForAxesX / (float)numOfPoint * (i + 1) -
(float)(maxValueForAxesX / 2); //интервал от -2 до 2
            }

            float[] second = new float[numOfPoint];//значения в точках для расчета

            for (int i = 0; i < numOfPoint; i++)
            {
                //расчитывам значение ф-ии
            }

            Point[] pointOne = new Point[numOfPoint];

            float tempX = 1 / oneDelenieX * stepForAxes; //расчитываем новые координаты
            float tempY = 1 / oneDelenieY * stepForAxes;

            for (int i = 0; i < numOfPoint; i++)
            {
                pointOne[i].X = center.X + (int)(first[i] * tempX); //переход к новым координатам
                pointOne[i].Y = center.Y - (int)(second[i] * tempY);
            }

            g.DrawLines(b, pointOne); //рисуем график (ломанная линия)
            g.DrawCurve(b, pointOne); //рисуем сплайном
```

Для построения графика можно использовать две различных функции, рассмотрим разницу между ними:

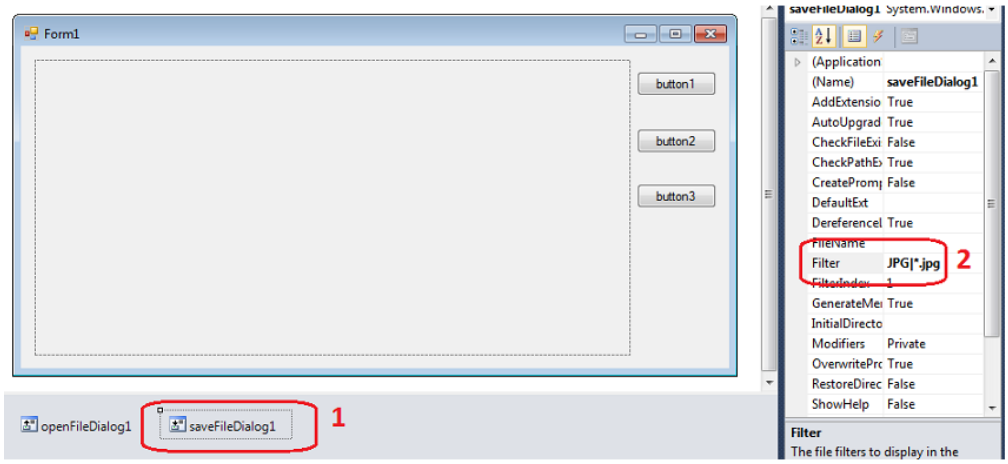


Создание диалогов сохранения и загрузки картинок.

Создадим форму с элементами button1, button2, button3, button4, pictureBox1, подключим System.Drawing.Imaging и System.Drawing.Drawing2D, а так же перетащим на форму элемент saveFileDialog1 и элемент openFileDialog1



в свойствах элемента saveFileDialog1 сделать как на картинке ниже это позволит не дописывать вручную расширение файла при сохранении поместить такой код:




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
using System.Threading;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            pictureBox1.Hide(); //скрываем элемент
            Graphics g = this.CreateGraphics(); // создаем объект касса Graphics
            Pen a = new Pen(Color.Red, 6); // создаем перо
            SolidBrush myBrush = new SolidBrush(Color.Yellow); //создам кисть
            g.DrawRectangle(a, 100, 100, 50, 100); // рисуем прямоугольник
            Pen b = new Pen(Color.Green, 3); // создаем другое перо
            g.DrawEllipse(b, 200, 100, 200, 100); // рисуем эллипс пером b
            g.FillEllipse(myBrush, 200, 100, 200, 100); // заливаем эллипс кистью myBrush
        }
```

Дважды кликнем по кнопке button2 и button3 и добавим в функции button2_Click и button3_Click соответственно следующий код:

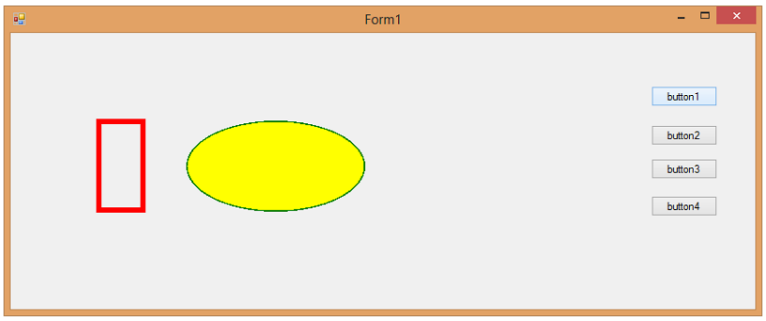
```
private void button2_Click(object sender, EventArgs e)
{
    ImageFormat img = ImageFormat.Jpeg; //задаем формат, имя и путь сохранения картинки
    saveFileDialog1.ShowDialog(); //вызываем диалог сохранения
    Rectangle bounds = this.Bounds; //берем размеры активного окна
    Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height); //создаем картинку с размерами текущего окна
    Graphics g = Graphics.FromImage(bitmap); //создали поверхность для рисования GDI+
    g.CopyFromScreen(new Point(bounds.Left, bounds.Top), Point.Empty, bounds.Size); //делаем скриншот
    bitmap.Save(saveFileDialog1.FileName, img); //сохраняем картинку по заданным: пути, имени, формату
    pictureBox1.Show(); //отображаем элемент
}

private void button3_Click(object sender, EventArgs e)
{
    openFileDialog1.ShowDialog(); //вызываем диалог открытия файла
    pictureBox1.Image = Image.FromFile(openFileDialog1.FileName); //помещаем в pictureBox1.Image
    //выбранную нами картинку
}
```

Описание работы программы:

1. при нажатия на первую кнопку на форме рисуются два примитива;

2. при нажатии на вторую кнопку мы сохраняем скриншот окна программы и стираем нарисованные элементы;
3. при нажатии на третью кнопку открываем картинку и помещаем ее в элемент pictureBox.

Итоговый вид приложения:



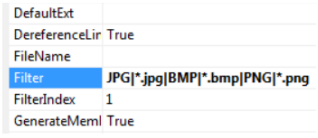
Теперь доработаем приложение: добавим еще одну кнопку, которая выводит текст.
В функцию button4_Click добавим следующий код.

```
        private void button4_Click(object sender, EventArgs e)
        {
            pictureBox1.Hide(); //скрываем элемент
            Graphics g = this.CreateGraphics(); // создаем объект Graphics
            LinearGradientBrush myBrush = new LinearGradientBrush(ClientRectangle, Color.Red,
Color.Yellow,
                System.Drawing.Drawing2D.LinearGradientMode.Horizontal); // создаем кисть
            Font myFont = new Font("Tahoma", 24, FontStyle.Regular); // выбираем шрифт
            g.DrawString("Text1", myFont, myBrush, new RectangleF(250,130,100,200)); // выводим
текст
        }
```

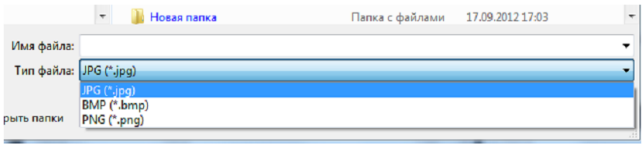
Так же модифицируем функцию button2_Click следующим образом

```
private void button2_Click(object sender, EventArgs e)
{
    ImageFormat img = ImageFormat.Jpeg;
    saveFileDialog1.ShowDialog();
    switch (saveFileDialog1.FilterIndex)
    {
        case 0: img = ImageFormat.Bmp; break;
        case 1: img = ImageFormat.Jpeg; break;
        case 2: img = ImageFormat.Png; break;
    }
    Rectangle bounds = this.Bounds;
    Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height);
    Graphics g = Graphics.FromImage(bitmap);
    g.CopyFromScreen(new Point(bounds.Left, bounds.Top), Point.Empty, bounds.Size);
    bitmap.Save(saveFileDialog1.FileName, img);
    pictureBox1.Show();
}
```

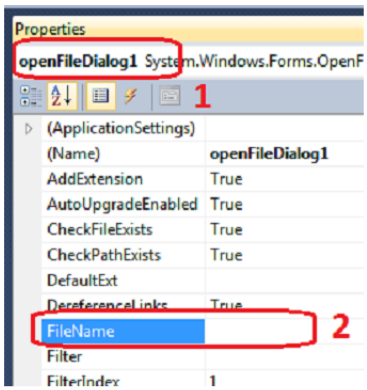
изменим свойства saveFileDialog1



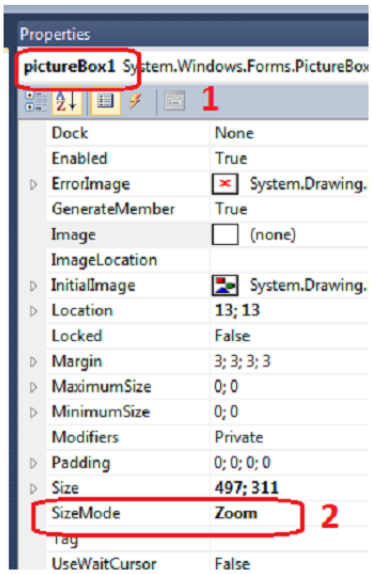
Теперь при сохранении картинки можно выбрать формат файла



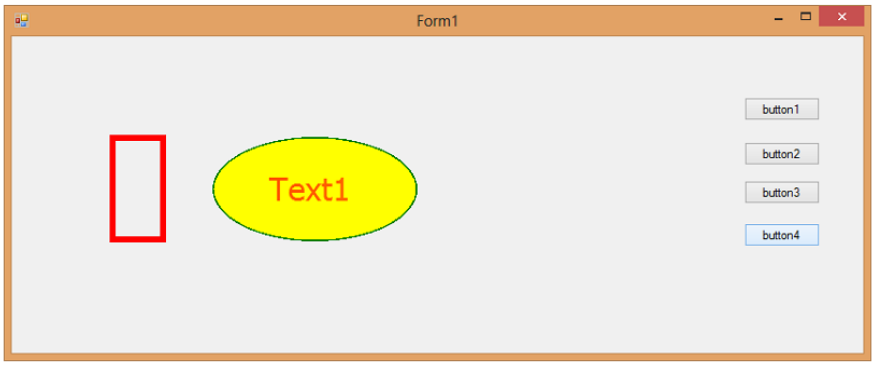
Установим свойства *saveFileDialog1* и open*FileDialog1* так:



Свойства *pictureBox1* так: (картинка справа)



Итоговый вид:



4. Содержание отчёта

Отчёт должен содержать название и цель. Ход работы с комментариями и выполненное задания из ПРИЛОЖЕНИЕ А. Выводы.

5. Контрольные вопросы

1. Опишите класс Graphics.
2. Свойства класса Graphics.
3. Методы класса Graphics.
4. Создание простой анимации.
5. Создание графика функций.
6. Типы отображения линий.
7. Для чего используется библиотека System.Threading
8. Опишите принципы работы pictureBox в Windows Form
9. Опишите принципы работы saveFileDialog1 в Windows Form
10. Опишите принципы работы openFileDialog1в Windows Form

1. На главной форме построить график функции соответствующий варианту в трех вариациях, каждый график должен отображаться другим цветом и изображаться на одной координатной сетке.
2. Создать меню на главной форме для вызова дополнительной формы, на которой создать анимацию для рисования графика функции по варианту.
3. В созданном меню создать обработчик вызова еще одной формы, где нарисовать на форме минимум 5 примитивов, разного размера, цвета и заливки. Создать кнопки загрузки и сохранения изображения.

Вариант	Задание
1	Парабола - график функции квадратного трёхчлена $y = ax^2 + bx + c$. Имеет вертикальную ось симметрии. Если a > 0, имеет минимум, если a < 0 - максимум. Точки пересечения (если они есть) с осью абсцисс - корни соответствующего квадратного уравнения $ax^2 + bx + c = 0$
2	Гипербола - график функции $y = \frac{a}{x}$. При a > O расположена в I и III четвертях, при a < 0 - во II и IV. Асимптоты - оси координат. Ось симметрии - прямая y = x (a > 0) или y - - x(a < 0).
3	Экспонента (показательная функция по основанию e) y = ex. (Другое написание y = exp(x)). Асимптота - ось абсцисс.
4	Логарифмическая функция $y = \log_a x$ (a > 0)
5	y = sinx. Синусоида - периодическая функция с периодом T = 2π
6	y = a•sin(ωx+ф) - функция гармонических колебаний. Обозначения: a - амплитуда, ω - частота (ω = 2π/T), ф - фаза (сдвиг).
7	Косинусоида y = cosx (графики y = sinx и y = cosx сдвинуты по оси x на $\frac{\pi}{2}$)
8	Тангенсоида y = tgx. Точки разрыва при x = $\frac{\pi}{2}$(2k -1), где k = 0, ±1, ±2,.. Вертикальные асимптоты в этих точках.
9	Гауссиана $y = Ae^{-(a*x*x)}$. Кривая "нормального" закона распределения ошибок, у которого $A = \frac{1}{\sigma\sqrt{2\pi}}$, $a = \frac{1}{\sigma\sqrt{2}}$, σ 2 - дисперсия ошибки. Симметрия относительно оси y.
10	y = secx - кривая "цепной линии", эту форму принимает абсолютно гибкая нить, подвешенная в параллельном поле тяжести. А полная функция периодична, и её асимптоты x = $\frac{\pi}{2}$(2k -1), как у функции y = tgx.
11	Круг с центром в точке (xo, yo) радиуса r. $(x-xo)^2 + (y-yo)^2 = r^2$
12	Эллипсс центром в точке (xo, yo). Большая полуось a, малая b, эксцинтриситет $e = \frac{\sqrt{a^2 + b^2}}{a}$, $\frac{(x - x_o)^2}{a^2} + \frac{(y - y_o)^2}{b^2} = 1$
13	Затухающее колебание $y = Ae^{-ax}$•sin(ωx+ф)
14	Если при заданной начальной скорости снаряда V менять угол α, то получится бесконечное множество парабол. Все параболы, для которых 45° ≤ α ≤ 90°, касаются одной и той же линии, имеющей уравнение $Y=½(V^2/g – gx^2/ V^2)$

15	Чтобы построить график функции $y=-f(x)$, можно построить изображение, симметричное графику функции $y=f(x)$ относительно оси $0x$. (На примере функции $y=\ln x$ и $y=-\ln x$).
16	Чтобы построить график функции $y=a\cdot f(x)$ при $a>0$, можно график функции $y=f(x)$ растянуть (сжать) вдоль оси $0y$, если $a>1$ ($0<a<1$). (На примере функции $y=\ln x$, $y=\ln\frac{1}{2}x$ и $y=\ln 2x$).
17	Функция $y=f\|x\|$ четная. Чтобы построить ее график, достаточно построить для $x\geq 0$ график функции $y=f(x)$, а затем достроить его левую часть, симметричную правой относительно оси $0y$. (На примере функции $y=\frac{1}{4}x^2-\|x\|-3$).
18	Можно данную функцию рассматривать как совокупность двух функций: $y=\begin{cases} f(x), \text{где } f(x)\geq 0 \\ -f(x), \text{где } f(x)<0 \end{cases}$. Чтобы построить график функции $y=\|f(x)\|$, достаточно построить график функции $y=f(x)$ и ту часть графика, которая расположена в нижней полуплоскости, симметрично отразить относительно оси $0x$. (На примере функции $y=\left\|\frac{1}{4}x^2-x-3\right\|$)
19	$y=\frac{1}{1+x^2}$
20	$y=\sqrt{x^2-1}$