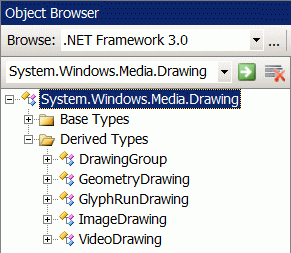
**Двумерная графика WPF**

Вся двумерная *графика* **WPF** реализуется классами, производными от абстрактных классов Drawing и Shape. Классы, наследующие Drawing, являются более простыми конструкциями, чем классы, произведенные от Shape, и поэтому требуют меньших системных ресурсов.

Абстрактный *класс* Drawing находится в пространстве имен System.Windows.Media.Drawing и содержит общее описание того, что должно быть нарисовано: фигура, точечный рисунок, строка текста или видео. *Производные* от него классы описывают конкретные типы содержимого:

1. GeometryDrawing - готовит внутри себя данные фигуры для рисования
2. ImageDrawing - готовит внутри себя данные изображения (точечный рисунок)
3. GlyphRunDrawing - готовит внутри себя текстовые данные
4. VideoDrawing - готовит внутри себя аудио- или видеофайл
5. DrawingGroup - послойно накапливает коллекцию данных предыдущих четырех объектов Drawing как один составной объект рисования для последующего вывода через класс отображения ( DrawingBrush, DrawingImage или Visual )

Все эти *производные* классы являются запечатанными ( sealed ) и не могут продолжать цепочку наследования.



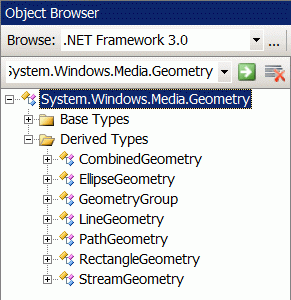
Нужно понимать, что перечисленные объекты содержат только описание того, что нужно нарисовать. Но чтобы воспроизвести содержимое любого из них, нужно передать этот *объект* в *конструктор* экземпляра класса-контейнера DrawingImage, а сам *контейнер* присоединить к свойству Source объекта-представления Image.

**Применение объекта GeometryDrawing для построения графиков**

*Объект* GeometryDrawing позволяет создать фигуру с заливкой и контуром путем совместного использования объектов Geometry, Pen и Brush, адресуемых его одноименными свойствами-ссылками, где

* объект Geometry описывает структуру самой геометрии фигуры
* объект Pen описывает контур фигуры
* объект Brush описывает заливку фигуры

Свойство-*ссылка* типа Geometry адресует экземпляр одного из классов, производных от абстрактного класса Geometry. Таковыми являются следующие *запечатанные* ( sealed ) классы:

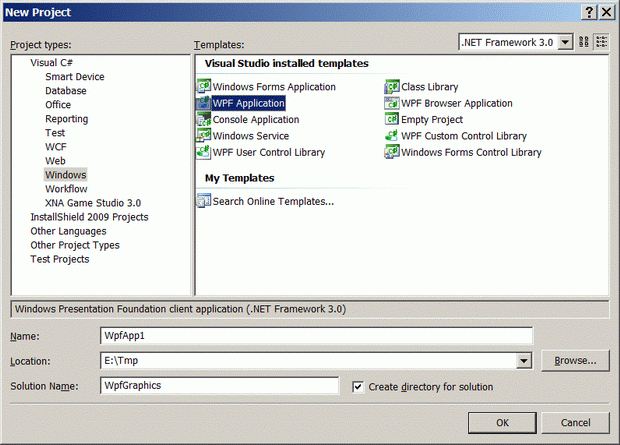


*Класс* GeometryGroup может накапливать описание геометрических примитивов своих родственников за счет свойства-коллекции Children типа GeometryCollection, а затем передавать их как единый составной *объект* свойству-ссылке Geometry объекта GeometryDrawing. В *дополнение* к такому описанию геометрии *объект* GeometryDrawing добавляет свое описание свойств Pen и Brush. Затем можно передать все это содержимое в коллекцию DrawingGroup как очередной готовый слой для последующего отображения. Такой механизм определяет векторную (в отличие от точечной) графику, которую можно произвольно масштабировать без потери качества рисунка на устройстве отображения с любым разрешением.

Продемонстрируем применение описанного на примере построения графиков (см. по ссылке <http://vitiy.info/?p=5>). Построим два *графика*: sin() и cos() для одного периода.

При выборе места размещения решения следите, чтобы путь к нему и проектам не содержал пробелов и кириллицы, иначе возможны сбои в работе!!!

Создайте новое решение Wpf с новым проектом WpfApp1 типа **WPF Application**

[](https://www.intuit.ru/EDI/05_07_17_1/1499206907-9549/tutorial/628/objects/11/files/68_01.gif)

Заполните декларативную часть Window1.xaml проекта следующей разметкой

<Window x:Class="WpfApp1.Window1"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Window1" Height="300" Width="300">

<Grid Margin="20">

<Image Name="image1" />

</Grid>

</Window>

Реализуйте бизнес-логику окна в застраничном файле Window1.xaml.cs с помощью следующего кода

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Globalization;

namespace WpfApp1

{

public partial class Window1 : Window

{

//

// Поля

//

const int countDot = 30;// Количество отрезков

// Список для хранения данных

List<double[]> dataList = new List<double[]>();

// Или можно DoubleCollection data = new DoubleCollection();

// Контейнер слоев рисунков

DrawingGroup drawingGroup = new DrawingGroup();

public Window1()

{

InitializeComponent();

DataFill();// Заполнение списка данными

Execute(); // Заполнение слоев

// Отображение на экране

image1.Source = new DrawingImage(drawingGroup); }

// Генерация точек графиков

void DataFill()

{

double[] sin = new double[countDot + 1];

double[] cos = new double[countDot + 1];

for (int i = 0; i < sin.Length; i++)

{

double angle = Math.PI \* 2 / countDot \* i;

sin[i] = Math.Sin(angle);

cos[i] = Math.Cos(angle);

}

dataList.Add(sin);

dataList.Add(cos);

}

// Послойное формирование рисунка в Z-последовательности

void Execute()

{

BackgroundFun(); // Фон

GridFun(); // Мелкая сетка

SinFun(); // Строим синус линией

CosFun(); // Строим косинус точками

MarkerFun(); // Надписи

}

// Фон

private void BackgroundFun()

{

// Создаем объект для описания геометрической фигуры

GeometryDrawing geometryDrawing = new GeometryDrawing();

// Описываем и сохраняем геометрию квадрата

RectangleGeometry rectGeometry = new RectangleGeometry();

rectGeometry.Rect = new Rect(0, 0, 1, 1);

geometryDrawing.Geometry = rectGeometry;

// Настраиваем перо и кисть

geometryDrawing.Pen = new Pen(Brushes.Red, 0.005);// Перо рамки

geometryDrawing.Brush = Brushes.Beige;// Кисть закраски

// Добавляем готовый слой в контейнер отображения

drawingGroup.Children.Add(geometryDrawing);

}

// Горизонтальная сетка

private void GridFun()

{

// Создаем коллекцию для описания геометрических фигур

GeometryGroup geometryGroup = new GeometryGroup();

// Создаем и добавляем в коллекцию десять параллельных линий

for (int i = 1; i < 10; i++)

{

LineGeometry line = new LineGeometry(new Point(1.0, i \* 0.1),

new Point(-0.1, i \* 0.1));

geometryGroup.Children.Add(line);

}

// Сохраняем описание геометрии

GeometryDrawing geometryDrawing = new GeometryDrawing();

geometryDrawing.Geometry = geometryGroup;

// Настраиваем перо

geometryDrawing.Pen = new Pen(Brushes.Gray, 0.003);

double[] dashes = { 1, 1, 1, 1, 1 };// Образец штриха

geometryDrawing.Pen.DashStyle = new DashStyle(dashes, -.1);

// Настраиваем кисть

geometryDrawing.Brush = Brushes.Beige;

// Добавляем готовый слой в контейнер отображения

drawingGroup.Children.Add(geometryDrawing);

}

// Строим синус линией

private void SinFun()

{

// Строим описание синусоиды

GeometryGroup geometryGroup = new GeometryGroup();

for (int i = 0; i < dataList[0].Length - 1; i++)

{

LineGeometry line = new LineGeometry(

new Point((double)i / (double)countDot,

.5 - (dataList[0][i] / 2.0)),

new Point((double)(i + 1) / (double)countDot,

.5 - (dataList[0][i + 1] / 2.0)));

geometryGroup.Children.Add(line);

}

// Сохраняем описание геометрии

GeometryDrawing geometryDrawing = new GeometryDrawing();

geometryDrawing.Geometry = geometryGroup;

// Настраиваем перо

geometryDrawing.Pen = new Pen(Brushes.Blue, 0.005);

// Добавляем готовый слой в контейнер отображения

drawingGroup.Children.Add(geometryDrawing);

}

// Строим косинус точками

private void CosFun()

{

// Строим описание косинусоиды

GeometryGroup geometryGroup = new GeometryGroup();

for (int i = 0; i < dataList[1].Length; i++)

{

EllipseGeometry ellips = new EllipseGeometry(

new Point((double)i / (double)countDot,

.5 - (dataList[1][i] / 2.0)), 0.01, 0.01);

geometryGroup.Children.Add(ellips);

}

// Сохраняем описание геометрии

GeometryDrawing geometryDrawing = new GeometryDrawing();

geometryDrawing.Geometry = geometryGroup;

// Настраиваем перо

geometryDrawing.Pen = new Pen(Brushes.Green, 0.005);

// Добавляем готовый слой в контейнер отображения

drawingGroup.Children.Add(geometryDrawing);

}

// Надписи

private void MarkerFun()

{

GeometryGroup geometryGroup = new GeometryGroup();

for (int i = 0; i <= 10; i++)

{

FormattedText formattedText = new FormattedText(

String.Format("{0,7:F}", 1 - i \* 0.2),

CultureInfo.InvariantCulture,

FlowDirection.LeftToRight,

new Typeface("Verdana"),

0.05,

Brushes.Black);

formattedText.SetFontWeight(FontWeights.Bold);

Geometry geometry = formattedText.BuildGeometry(new Point(-0.2, i \* 0.1 - 0.03));

geometryGroup.Children.Add(geometry);

}

GeometryDrawing geometryDrawing = new GeometryDrawing();

geometryDrawing.Geometry = geometryGroup;

geometryDrawing.Brush = Brushes.LightGray;

geometryDrawing.Pen = new Pen(Brushes.Gray, 0.003);

drawingGroup.Children.Add(geometryDrawing);

}

}

}

Результат:

