# WPF 2D图形编程入门

## 课程内容剪辑

## 课前准备

1. 学员基础：
   1. 熟悉C#
   2. XAML
   3. 已学完《Microsoft Blend 2019》更佳
   4. 已学完《WPF经典编程实例》更佳
2. 演示环境：
   1. Windows 10
   2. Visual Studio Community 2022 /.NET桌面开发

<https://visualstudio.microsoft.com/zh-hans/>

* 1. WPF应用程序（用于创建**.NET** WPF应用程序项目）
  2. XAML Styler 插件

1. 课程源码：[https://github.com/**zmrbak**/Wpf2D](https://github.com/zmrbak/Wpf2D)
2. 课程类型：演示、实战

## WPF中的统一编程模型

**DirectX：**游戏程序

WPF之前

控件：按钮、列表框之类。本质上一个个窗口。

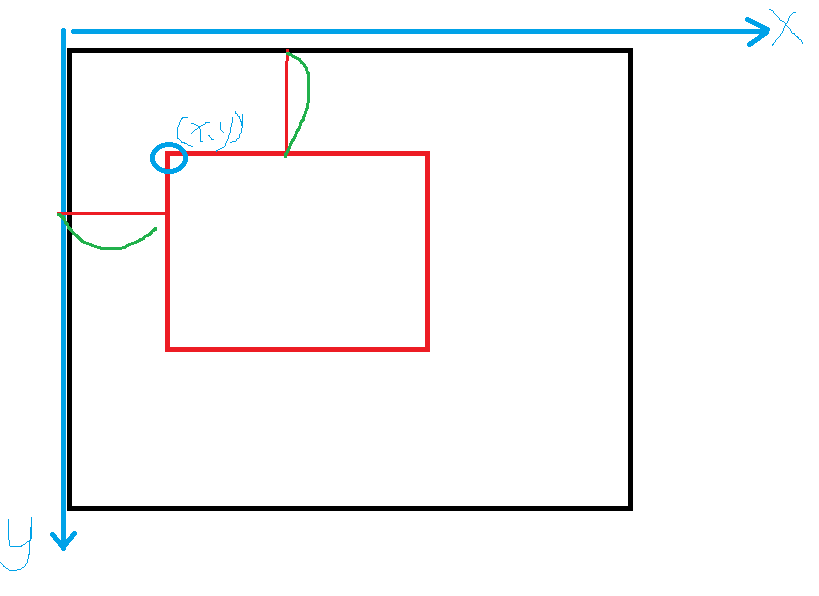
几何图形：直线、圆、矩形。

## WPF中Shape

坐标的单位与显示器的分辨率、显卡无关。

1/96英寸（1英寸=25.4毫米）

Width=96，在屏幕上1英寸宽



WPF中两个独立的图形系统：

以Shape为基类，Rectangle

以Goemetry为基类，RectangleGoemetry

System.Windows.Shapes

Ellipse 绘制椭圆形。

Line 在两个点之间绘制直线。

Path 绘制一系列相互连接的直线和曲线。

Polygon 绘制多边形，它是由一系列相互连接的线条构成的闭合形状。

Polyline 绘制一系列相互连接的直线。

Rectangle 绘制矩形。

Shape 为 Ellipse、Polygon 和 Rectangle 之类的形状元素提供基类。

shape可在界面中显示自己

shape可直接参与排版

shape含有画笔画刷

shape可以进行动画

System.Windows.Media

CombinedGeometry

EllipseGeometry

GeometryGroup

LineGeometry

PathGeometry

RectangleGeometry

StreamGeometry

Geometry通过shape显示自己

Geometry通过其他界面显示自己

Geometry可以组成非常复杂的形状。

## 使用代码设置控件属性

private void canvas\_SizeChanged(object sender, SizeChangedEventArgs e)

{

slider\_x.Maximum = canvas.ActualWidth;

slider\_y.Maximum = canvas.ActualHeight;

}

private void slider\_x\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

//button.SetValue(Canvas.LeftProperty,slider\_x.Value);

Canvas.SetLeft(button, slider\_x.Value);

}

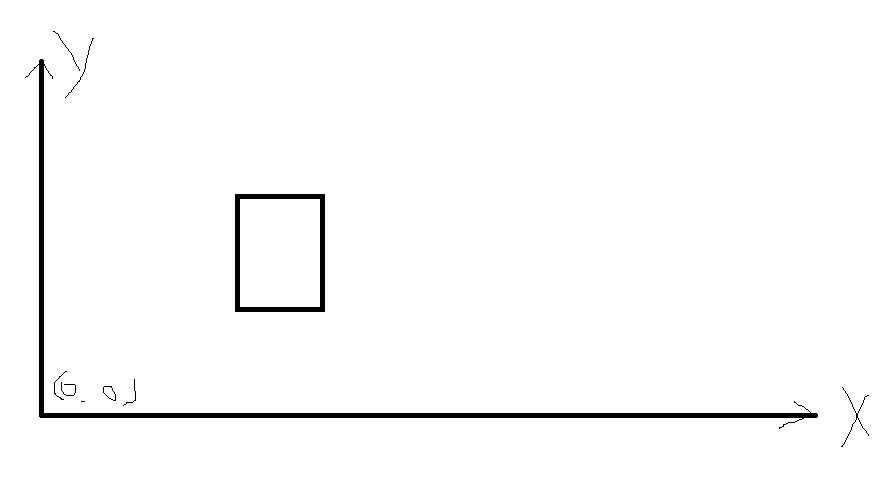
private void slider\_y\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

button.SetValue(Canvas.TopProperty, slider\_y.Value);

}

## 使用自定义坐标



RenderTransform

获取或设置影响此元素的呈现位置的转换信息。 这是依赖项属性。

描述所需呈现转换的细节。 默认值为 Identity。

呈现转换不会重新生成布局大小或呈现大小信息。 呈现转换通常用于对元素进行动画处理或应用临时效果。 例如，当焦点或鼠标悬停时，元素可能会缩放，或者在加载时抖动，以便将眼睛吸引到用户界面 (UI) 部分。

Transform

定义实现二维平面中的转换的功能。 转换包括旋转 (RotateTransform)、缩放 (ScaleTransform)、倾斜 (SkewTransform) 和平移 (TranslateTransform)。 此类层次结构不同于 Matrix 结构，因为它是一个类，而且支持动画和枚举语义。

使用 MatrixTransform 类创建自定义转换，这些转换不是由 RotateTransform、 ScaleTransform、 SkewTransform和 TranslateTransform 类提供的。

* **MatrixTransform**

在二维 x-y 坐标系中平移（移动）对象。

* RotateTransform

在二维 x-y 坐标系内围绕指定点按顺时针方向旋转对象。

* ScaleTransform

在 2-D x-y 坐标系统内缩放对象。

* SkewTransform

表示 2-D 倾斜。

* **TransformGroup**

表示由其他 Transform 对象组成的复合 Transform

* TranslateTransform

在二维 x-y 坐标系中平移（移动）对象。

## 路径标记语法

<Grid>

<Rectangle

Width="100"

Height="100"

Stroke="Black" />

</Grid>

<Grid>

<Path

Width="100"

Height="100"

**Data="M0.5,0.5 L99.5,0.5 L99.5,99.5 L0.5,99.5 z"**

Stretch="Fill"

Stroke="Black" />

</Grid>

在创建路径后**无需修改路径**时，请使用 StreamGeometry；

<Grid>

<Path

Width="100"

Height="100"

Stretch="Fill"

Stroke="Black">

<Path.Data>

<PathGeometry Figures="M0,0 L100,0 L100,100 L0,100 z" />

</Path.Data>

</Path>

</Grid>

如果需要**修改路径**，请使用 PathGeometry。

<Grid>

<Path

Width="100"

Height="100"

Stretch="Fill"

Stroke="Black">

<Path.Data>

<PathGeometry Figures="M0,0 L100,0L100,100L0,100z" />

</Path.Data>

</Path>

</Grid>

**语法部分有一个空格，没有空格，或者多个空格，不影响结果**

<Grid>

<Path

Width="100"

Height="100"

Stretch="Fill"

Stroke="Black">

<Path.Data>

<PathGeometry Figures="M0 0 L100 0L100,100L0,100z" />

</Path.Data>

</Path>

</Grid>

两个数字之间不必用逗号或空格分隔，但仅适用于生成的字符串明确时。 例如，2..3 实际是两个数字：“2.”和“.3”。 同样，2-3 是“2”和“-3”。 命令前面或后面也无需加空格。

<Grid>

<Path

Width="100"

Height="100"

Stretch="Fill"

Stroke="Black">

<Path.Data>

<PathGeometry Figures="M0,0 100,0 100,100 0,100 z" />

</Path.Data>

</Path>

</Grid>

大写字母 M 表明 startPoint 是一个绝对值；小写字母 m 表明 startPoint 是对前一点的偏移，如果都不存在的话，则为 (0,0)。 如果在**移动命令之后列出多个点，可以绘制一条连接到这些点的直线**，尽管已指定了直线命令。

<Grid>

<Path

Width="100"

Height="100"

Stretch="Fill"

Stroke="Black">

<Path.Data>

<PathGeometry Figures="M0,0h100v100h-100z" />

</Path.Data>

</Path>

</Grid>

**Path 类**

绘制一系列相互连接的直线和曲线。

**Path.Data 属性**

获取或设置用于指定所要绘制的形状的 Geometry

**直线命令**

大写字母 L 表明 endPoint 是一个绝对值；小写字母 l 表明 endPoint 是对前一点的偏移，如果都不存在的话，则为 (0,0)。

**水平线命令**

在当前点和指定的 x 坐标之间创建一条水平线。 H 90 是有效水平线命令的示例。

大写字母 H 表明 x 是一个绝对值；小写字母 h 表明 x 是对前一点的偏移，如果都不存在的话，则为 (0,0)。

**竖线命令**

在当前点和指定的 y 坐标之间创建一条竖线。 v 90 是有效竖线命令的示例。

大写字母 V 表明 y 是一个绝对值；小写字母 v 表明 y 是对前一点的偏移，如果都不存在的话，则为 (0,0)。

**关闭命令**

结束当前图形，并创建一条将当前点连接到图形起点的直线。 此命令可以在图形最后一段与第一段之间创建一条连接线（角）。

<Application.Resources>

<DrawingBrush

x:Key="MyGridBrushKey"

TileMode="Tile"

Viewport="0 0 10 10"

ViewportUnits="Absolute">

<DrawingBrush.Drawing>

<DrawingGroup>

<DrawingGroup.Children>

<GeometryDrawing Brush="White">

<GeometryDrawing.Geometry>

<RectangleGeometry Rect="0 0 1 1" />

</GeometryDrawing.Geometry>

</GeometryDrawing>

<GeometryDrawing Brush="#EEEEEE">

<GeometryDrawing.Geometry>

M0,0 L1,0 L1,0.1 L0,0.1 z

</GeometryDrawing.Geometry>

</GeometryDrawing>

<GeometryDrawing Brush="#EEEEEE">

<GeometryDrawing.Geometry>

M0,0 L0,1 L0.1,1 L0.1,0 z

</GeometryDrawing.Geometry>

</GeometryDrawing>

</DrawingGroup.Children>

</DrawingGroup>

</DrawingBrush.Drawing>

</DrawingBrush>

</Application.Resources>

<Grid Background="{StaticResource MyGridBrushKey}">