**2D graphics in WPF**

2013-09-04 PV

# Drawings and Shapes

WPF provides both Drawing and Shape objects to represent graphical drawing content. However, Drawing objects are simpler constructs than Shape objects and provide better performance characteristics.

## Shapes

A Shape allows you to draw a graphical shape to the screen. Because they are derived from the FrameworkElement class, Shape objects can be used inside panels and most controls.

WPF offers several layers of access to graphics and rendering services. At the top layer, Shape objects are easy to use and provide many useful features, such as layout and event handling. WPF provides a number of ready-to-use shape objects. All shape objects inherit from the Shape class. Available shape objects include Ellipse, Line, Path, Polygon, Polyline, and Rectangle.

## Drawing

Drawing objects, on the other hand, do not derive from the FrameworkElement class and provide a lighter-weight implementation for rendering shapes, images, and text.

The Geometry class lacks some of the functionality of the Shape class, such as the ability to draw itself. To draw a geometry object, another class such as DrawingContext, Drawing, or a Path (it is worth noting that a Path is a Shape) must be used to perform the drawing operation. Rendering properties such as fill, stroke, and the stroke thickness are on the class which draws the geometry object, while a shape object contains these properties.

There are four types of Drawing objects:

* GeometryDrawing Draws a shape.
* ImageDrawing Draws an image.
* GlyphRunDrawing Draws text.
* DrawingGroup Draws other drawings. Use a drawing group to combine other drawings into a single composite drawing.

### GeometryDrawing

The GeometryDrawing object is used to render geometry content. The Geometry class and the concrete classes which derive from it, such as CombinedGeometry, EllipseGeometry, and PathGeometry, provide a means for rendering 2D graphics, as well as providing hit-testing and clipping support. Geometry objects can be used to define the region of a control, for example, or to define the clip region to apply to an image. Geometry objects can be simple regions, such as rectangles and circles, or composite regions created from two or more geometry objects. More complex geometric regions can be created by combining PathSegment-derived objects, such as ArcSegment, BezierSegment, and QuadraticBezierSegment.

# StreamGeometry Objects

The StreamGeometry object is a light-weight alternative to PathGeometry for creating geometric shapes. Use a StreamGeometry when you need to describe a complex geometry. StreamGeometry is optimized for handling many PathGeometry objects and performs better when compared to using many individual PathGeometry objects.

The following example uses attribute syntax to create a triangular StreamGeometry in XAML.

<StackPanel>

<Path Data="F0 M10,100 L100,100 100,50Z"

StrokeThickness="1" Stroke="Black"/>

</StackPanel>

# DrawingVisual Objects

The DrawingVisual object is a lightweight drawing class that is used to render shapes, images, or text. This class is considered lightweight because it does not provide layout or event handling, which improves its performance. For this reason, drawings are ideal for backgrounds and clip art.

# Images

There are several ways to display an image in a Windows Presentation Foundation (WPF) application. Images can be displayed using an Image control, painted on a visual using an ImageBrush, or drawn using an ImageDrawing.

## Image Control

Image is a framework element and the primary way to display images in applications.

<Image Width="200">

<Image.Source>

<!-- To save significant application memory, set the DecodePixelWidth or

DecodePixelHeight of the BitmapImage value of the image source to the desired

height and width of the rendered image. If you don't do this, the application will

cache the image as though it were rendered as its normal size rather then just

the size that is displayed. -->

<!-- Note: In order to preserve aspect ratio, only set either DecodePixelWidth

or DecodePixelHeight but not both. -->

<BitmapImage DecodePixelWidth="200"

UriSource="C:\Documents and Settings\All Users\Documents\My Pictures\Sample Pictures\Water Lilies.jpg" />

</Image.Source>

Many of the examples use a BitmapImage object to reference an image file. BitmapImage is a specialized BitmapSource that is optimized for Extensible Application Markup Language (XAML) loading and is an easy way to display images as the Source of an Image control.