CHƯƠNG V:
XỬ LÝ ĐỒ HỌA TRÊN SILVERLIGHT

1 Giới thiệu

Silverlight cung cấp nhiều lựa chọn cho việc thêm các tính năng trực quan thú vị cho ứng dụng của bạn. Bạn có thể sử dụng vẽ, Shape, Path, và những hình học phức tạp. Những khu vực được xác định bởi dạng hình thì có thể tô hiệu ứng, như là ảnh, dải màu, hoặc là đoạn video, thông qua việc sử dụng Brush.
Silverlight kế thừa một thư viện đồ họa khá đầy đủ từ WPF bởi vậy sẽ không quá khó khăn để một lập trình viên đã quen với WPF chuyển qua làm việc với **Silverlight**. Dưới đây chúng ta sẽ lần lượt làm quen với các đối tượng đồ họa như Ellipse, Line, Path, Polygon, Geometries, Brushes…

2 Shapes and Drawing

Trong Silverlight, Shape là kiểu UIElement nên bạn có thể dễ dàng hiển thị một đối tượng dạng Shape lên màn hình. Bởi vì chúng là những thành phần đồ họa nên những đối tượng Shape này có thể đi kèm với những container như **Grid** và **Canvas. Silverlight** cung cấp những dạng hình(**Shape**) mà bạn có thể dùng ngay được như **Ellipse**, **Line**, **Path**, **Polygon**, **Polyline**, và **Rectangle**. Những **Shape** đều có chung những đặc tính dưới đây:

- **Stroke**: dùng để vẽ viền ngoài của **Shape.**
- **StrokeThickness**: độ dày của viền ngoài của **Shape.**
- **Fill**: Mô tả cách phía bên trong của Shape được vẽ.
- Đặc tính **Data** chỉ rõ tọa độ và các đỉnh được định nghĩa tùy theo thông tin đầu vào

Những đối tượng **Shape** có thể dùng bên trong **Canvas**. **Canvas** hỗ trợ chỉ ra vị trí tuyệt đối của đối tượng con bên trong thông qua đặc tính đính kèm là **Canvas.Left** và **Canvas.Top**

2.1 Ellipse

Bạn có thể tạo một Ellipse bằng cách xác định hai thuộc tính cơ bản là rộng(Width) và cao(Height) như ví dụ minh họa dưới đây.

```
<Canvas>
    <Ellipse
    Fill="Yellow"
    Height="100"
    Width="200"
    StrokeThickness="2"
    Stroke="Black"/>
</Canvas>
```

2.2 Line

Cho phép bạn vẽ một đoạn thẳng giữa hai điểm. Ví dụ dưới đây chỉ ra một số cách để bạn xác định tọa độ của đoạn thẳng và đặc tính Stroke của nó.

```
<Canvas Height="300" Width="300">
    <!—Vẽ một đường chéo từ tọa độ (0, 0) tới (100,100). -->
    <Line X1="0" Y1="0" X2="100" Y2="100" Stroke="Black" StrokeThickness="4" />
    <!—Vẽ một đường chéo từ tọa độ (0, 0) tới (100,100) và di chuyển nó tới gốc
tọa độ mới thêm 100 pixels về phía bên phải. -->
    <Line X1="0" Y1="0" X2="100" Y2="100" StrokeThickness="10" Canvas.Left="100">
        <Line.Stroke>
            <RadialGradientBrush GradientOrigin="0.5,0.5" Center="0.5,0.5"
        RadiusX="0.5" RadiusY="0.5">
                <RadialGradientBrush.GradientStops>
                    <GradientStop Color="Red" Offset="0" />
                    <GradientStop Color="Blue" Offset="0.5" />
                </RadialGradientBrush.GradientStops>
            </RadialGradientBrush>
        </Line.Stroke>
    </Line>
    <!—Vẽ một đoạn ngang từ tọa độ (10,80) tới (150,80). -->
    <Line X1="0" Y1="80" X2="200" Y2="80" Stroke="Black" StrokeThickness="4"/>
</Canvas>
```

2.3 Path

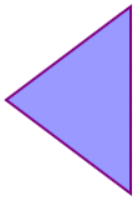
Lớp **Path** cho phép bạn vẽ những hình cong và những dạng hình phức tạp. Những hình cong và dạng hinh(shape) được diễn tả thông qua việc sử dụng đối tượng Geometry. Để sử dụng Path, bạn tạo một Geometry và dùng nó để xét cho đặc tính Data của đối tượng Path. Có nhiều loại đối tượng Geometry khác nhau để bạn chọn: LineGeometry, RectangleGeometry, và EllipseGeometry có liên quan tới những dạng hình(shape) đơn giản lần lượt tương ứng Line, Rectangle, Ellipse. Để tạo những dạng hình phức tạp hoặc hình tròn chúng ta sử dụng PathGeometry. Ví dụ dưới đây xử dụng cú pháp rút gọn để biểu diễn một dạng hình phức tạp.

```
<Canvas>
       <Path Stroke="DarkGoldenRod" StrokeThickness="3"
       Data="M 100,200 C 100,25 400,350 400,175 H 280" />
</Canvas>
```

2.4 Polygon

Để xác định một đa giác(Polygon) bạn cần cung cấp một danh sách các điểm (Point) và tô màu cho nó qua đặc tính Fill. Ví dụ dưới đây vẽ một đa giác với màu viền đỏ tía và bên trong được đổ màu xanh.

```
<Canvas>
    <Polygon
    Points="300,200 400,125 400,275 300,200"
    Stroke="Purple"
    StrokeThickness="2">
        <Polygon.Fill>
            <SolidColorBrush Color="Blue" Opacity="0.4"/>
        </Polygon.Fill>
    </Polygon>
</Canvas>
```



2.5 Polyline

Cũng tương tự như đối tượng Polygon ngoại trừ việc không nhất thiết Polyline phải khép kín. Bạn cũng có thể nhận được kết quả tương tự như Polyline khi vẻ chồng lên nhau nhiều Line. Chú ý Polyline với một điểm(Point) sẽ không được hiển thị lên. Ví dụ dưới đây vẽ một Polyline là một hình tam giác viền màu xanh khép kín.

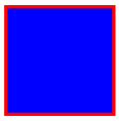
```
<Canvas>
    <Polyline Points="50,25 0,100 100,100 50,25"
    Stroke="Blue" StrokeThickness="10"
    Canvas.Left="75" Canvas.Top="50" />
</Canvas>
```

2.6 Rectangle

Một hình chữ nhật(Rectangle) được xác định bởi hai đặc tính rộng(Width) và cao(Height), để xác định vị trí của Rectangle trong container chúng ta sử dụng thuộc tính Margin hoặc Canvas.Left, Canvas.Top để tùy chỉnh. Rectangle không hỗ trợ đối tượng con bên trong bởi vậy nếu bạn muốn khu vực Rectangle chứa những đối tượng

khác bạn có thể sử dụng Canvas. Bạn cũng có thể dùng dạng hình hỗn hợp, nhưng trong trường hợp này có lẽ bạn nên sử dụng RectangleGeometry hơn là việc dùng Rectangle. Ví dụ dưới đây hiển thị một Rectangle viền màu đỏ và tô màu bên trong màu xanh.

```
<Canvas>
    <Rectangle Width="100" Height="100" Fill="Blue" Stroke="Red"
   Canvas.Top="20" Canvas.Left="20" StrokeThickness="3" />
</Canvas>
```



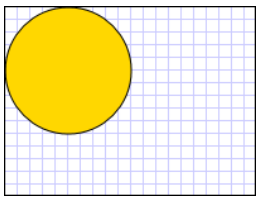
3 Geometries

Cung cấp lớp cơ sở cho những đối tượng để xác định dạng hình học cho đối tượng. Đối tượng Geometry có thể sử dụng để làm vùng giao(clipping regions) và cũng như định nghĩa dữ liệu cho đối tượng Path.

3.1 EllipseGeometry

Miêu tả hình dạng của đường tròn hoặc ellipse. **FillRule** là đặc tính kế thừa từ lớp cơ sở Geometry nhưng tùy chọn này không có tác dụng đối với **EllipseGeometry**. **EllipseGeometry** chỉ đơn giản định nghĩa hình dạng cho ellipse và không thể tự nó hiển thị được. Bởi vì nó rất đơn giản và phạm vi sử dụng rộng. Ví dụ dưới đây vẽ một path sử dụng dữ liệu dạng **EllipseGeometry**

```
<Canvas>
  <Path Fill="Gold" Stroke="Black" StrokeThickness="1">
    <Path.Data>
      <EllipseGeometry Center="50,50" RadiusX="50" RadiusY="50" />
    </Path.Data>
  </Path>
</Canvas>
```

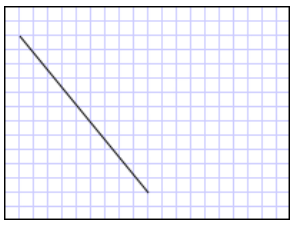


3.2 PathGeometry

Miêu tả dạng hình phức tạp có thể gồm có hình cung, đường cong, ellipse, đoạn thẳng, và hình chữ nhật. Mỗi PathGeometry xác định bởi một danh sách đối tượng PathFigure. Mỗi đối tượng PathFigure bao gồm một hoặc nhiều PathSegment, tương tự cũng như với ArcSegment và LineSegment. Để phủ kín diện tích của PathGeometry thì mọi đối

tượng PathFigure bạn phải xét thuộc tính IsFilled là "true" và FillRule để xác định khu vực được fill.

```
<Canvas>
  <Path Stroke="Black" StrokeThickness="1">
    <Path.Data>
      <PathGeometry>
        <PathGeometry.Figures>
          <PathFigure StartPoint="10,20">
            <PathFigure.Segments>
              <LineSegment Point="100,130"/>
            </PathFigure.Segments>
          </PathFigure>
        </PathGeometry.Figures>
      </PathGeometry>
    </Path.Data>
  </Path>
</Canvas>
```



3.3 GeometryGroup

Miêu tả một hỗn hợp hình dạng, được tạo thành từ nhiều đối tượng Geometry khác. Sử dụng FillRule cho GeometryGroup là rất hợp lý bởi sự kết hợp các hình dạng có thể có khả năng kết hợp các đoạn nơi mà sự áp dụng của quy luật fill có tác dụng.

```
<Canvas>
  <Path Stroke="Black" StrokeThickness="1" Fill="#CCCCFF">
    <Path.Data>

      <!-- Creates a composite shape from three geometries. -->
      <GeometryGroup FillRule="EvenOdd">
        <LineGeometry StartPoint="10,10" EndPoint="50,30" />
        <EllipseGeometry Center="40,70" RadiusX="30" RadiusY="30" />
        <RectangleGeometry Rect="30,55 100 30" />
      </GeometryGroup>
    </Path.Data>
  </Path>
</Canvas>
```

4 Brushes

Bạn có thể tô màu các đối tượng trong **Silverlight** với màu đơn, dải nghiêng, bán kính nghiêng và ảnh.

4.1 Solid Color

Một trong những thao tác chung trong bất cứ nền tảng nào là tô màu một khu vực với một màu đơn sắc. Để thực hiện việc này, Silverlight cung cấp lớp SolidColorBrush. Dưới đây chúng tôi sẽ chỉ ra những cách khác nhau để tô màu với SolidColorBrush.

Để tô màu một khu vực với màu đơn trong XAML, sử dụng theo những tùy chọn sau:

- SolidColorBrush được định nghĩa trước theo tên. Ví dụ dưới đây dùng SolidColorBrush để xét thuộc tính Fill của hình chữ nhật(Rectangle).

```
<StackPanel>
  <Rectangle Width="100" Height="100" Fill="Red" />
</StackPanel>
```

- Chọn từ bảng màu 32-bit với định dạng #rrggbb với rr, gg, bb lần lượt là 2 số mã 16 cho các màu đỏ, xanh lá, xanh lơ. Mở rộng thêm bạn có thể xét với định dạng như sau: #aarrggbb với r, g, b ý nghĩa như trên và aa ở đây chỉ giá trị alpha hoặc là độ trong suốt.

```
<StackPanel>
  <Rectangle Width="100" Height="100" Fill="#FFFF0000" />
</StackPanel>
```

- Sử dụng thuộc tính kiểu phần tử để mô tả SolidColorBrush(ở đây là thuộc tính Fill)

```
<StackPanel>
  <Rectangle Width="100" Height="100">
    <Rectangle.Fill>
      <SolidColorBrush Color="Red" />
    </Rectangle.Fill>
  </Rectangle>

  <Rectangle Width="100" Height="100">
    <Rectangle.Fill>
      <SolidColorBrush Color="#FFFF0000" />
    </Rectangle.Fill>
  </Rectangle>
</StackPanel>
```

4.2 Gradient

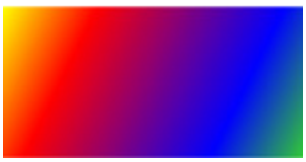
Gradient brush vẽ một khu vực với nhiều màu sắc pha trộn vào nhau dọc theo một trục. Bạn có thể sử dụng chúng để tạo ra hiệu ứng ánh sáng và bóng, mang lại cho thành phần đồ họa của bạn cảm giác 3 chiều. Bạn cũng có thể sử dụng chúng để mô phỏng kính, chrome, nước, và các bề mặt mịn. Silverlight cung cấp 2 loại gradient brush là : **LinearGradientBrush** và **RadialGradientBrush**.

a. Linear Gradients

LinearGradientBrush dùng để vẽ một khu vực với một dải nghiêng. Bạn chỉ ra màu sắc cho dải nghiêng và vị trí dọc theo trục nghiêng thông qua việc sử dụng đối tượng GradientStop. Bạn cũng có thể chỉnh sửa trục nghiêng, nó cho phép bạn tạo dải nghiêng theo chiều dọc hoặc ngang và đảo hướng của

dài nghiêng. Ví dụ dưới đây sử dụng dải nghiêng với 4 màu và sử sụng nó để vẽ lên hình chữ nhật.

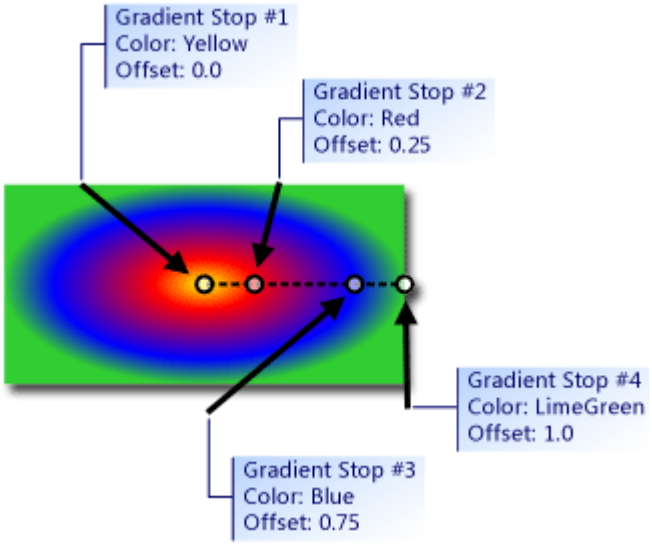
```
<StackPanel>
  <Rectangle Width="200" Height="100">
    <Rectangle.Fill>
      <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
        <GradientStop Color="Yellow" Offset="0.0" />
        <GradientStop Color="Red" Offset="0.25" />
        <GradientStop Color="Blue" Offset="0.75" />
        <GradientStop Color="LimeGreen" Offset="1.0" />
      </LinearGradientBrush>
    </Rectangle.Fill>
  </Rectangle>
</StackPanel>
```



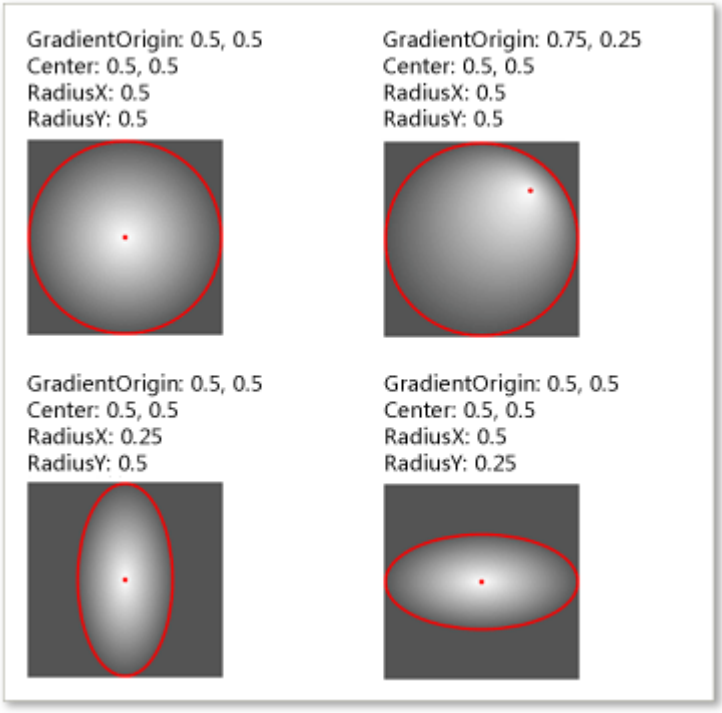
b. Radial Gradients

Giống như LinearGradientBrush, một RadialGradientBrush vẽ một khu vực với màu sắc mà pha trộn với nhau dọc theo một trục. Cách vẽ của RadialGradientBrush dựa theo một trục được xác định bởi đường tròn; màu sắc của nó tia ra ngoài từ phía gốc của nó. Ví dụ dưới đây vẽ một hình chữ nhật sử dụng RadialGradientBrush để tô màu bên trong.

```
<StackPanel>
  <Rectangle Width="200" Height="100">
    <Rectangle.Fill>
      <RadialGradientBrush GradientOrigin="0.5,0.5" Center="0.5,0.5"
        RadiusX="0.5" RadiusY="0.5">
        <GradientStop Color="Yellow" Offset="0" />
        <GradientStop Color="Red" Offset="0.25" />
        <GradientStop Color="Blue" Offset="0.75" />
        <GradientStop Color="LimeGreen" Offset="1" />
      </RadialGradientBrush>
    </Rectangle.Fill>
  </Rectangle>
</StackPanel>
```



Dưới đây là một vài tùy chọn khác nhau giúp bạn tạo một số kiểu hiệu ứng khác nhau với RadialGradientBrush:

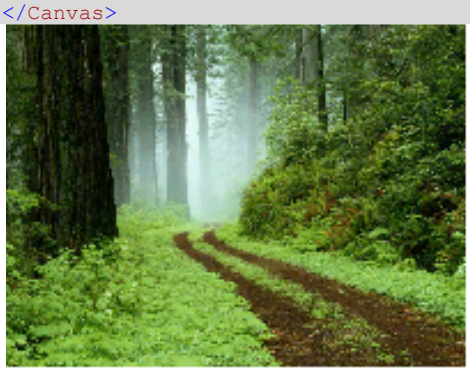


4.3 Images

Lớp ImageBrush cho phép bạn sử dụng ảnh để fill, để xét làm bacrkground, và viền ngoài. ImageBrush sử dụng được với những định dạng ảnh JPEG hoặc PNG và được dùng thông qua thuộc tính ImageSource. Bạn có thể cung cấp thông tin ImageSource với đường dẫn của ảnh để tải chúng.

Mặc định ImageBrush sẽ giãn ảnh ra hết toàn bộ khu vực được vẽ, có thẻ ảnh sẽ bị làm méo nếu khu vực được vẽ có tỉ lệ bề mặt khác với ảnh. Bạn có thể thay đổi trạng thái này bằng việc thay đổi thuộc tính Stretch mặc định của nó tới một trong những giá trị sau: None, Uniform, UniformToFill. Ví dụ dưới đây sử dụng ImageBrush để vẽ nền của một Canvas

```
<Canvas>
  <Grid x:Name="LayoutRoot">
    <Grid.Background>
      <ImageBrush ImageSource="Forest.jpg" />
    </Grid.Background>
  </Grid>
</Canvas>
```



4.4 Video

VideoBrush cho phép bạn vẽ một khu vực bằng video. Ví dụ dưới đây sử dụng VideoBrush để vẽ màu chữ(Foreground) của TextBlock

```
<Grid x:Name="LayoutRoot" Background="White">
    <MediaElement
    x:Name="butterflyMediaElement"
    Source="Butterfly.wmv" IsMuted="True"
    Opacity="0.0" IsHitTestVisible="False" />
    <TextBlock Canvas.Left="5" Canvas.Top="30"
    FontFamily="Verdana" FontSize="120"
    FontWeight="Bold" TextWrapping="Wrap"
    Text="Video">

    <!-- Paint the text with video. -->
    <TextBlock.Foreground>
      <VideoBrush SourceName="butterflyMediaElement" Stretch="UniformToFill" />
    </TextBlock.Foreground>
    </TextBlock>
</Grid>
```

4.5 Deep Zoom

Deep Zoom cung cấp khả năng phóng to thu nhỏ một cách tùy ý những ảnh lớn trong Siliverlight một cách hiệu quả nhất. Ảnh có thể hiển thị ở mức rất nhỏ và rất lớn mà không ảnh hưởng tới hiệu quả hoạt động của ứng dụng đang hiển thị ảnh. Chỉ có thuộc tính làm ảnh hưởng tới hiệu quả hoạt động đó là độ phân giải của màn hình. Để biết thêm chi tiết mời các bạn tham khảo địa chỉ sau: http://msdn.microsoft.com/en-us/library/cc645050(VS.95).aspx