Geometria

IFRN – Natal - RN

Geometria

Geometrias – A classe abstrata ***Geometry*** serve de base a vários tipos que permitem a representação de várias figuras geométricas 2D (ex.: retângulos, elipses, etc.)

- **Geometrias simples** – as geometrias podem ser agrupadas em três tipos: simples, baseadas em trajetos (paths) e compostas.
- As geometrias básicas englobam três classes (*rectangleGeometry*, *LienarGeometry* e *EllipdeGeometry*) e permitem a construção de gráficos básicos como linhas e retângulos.

Geometria

```
<Grid>
      <Path Stroke="Red" StrokeThickness="2">
         <Path.Data>
            <RectangleGeometry RadiusX="6" RadiusY="6" Rect="10,10,100,100" />
         </Path.Data>
      </Path>
   </Grid>
```

```
<Path Stroke="Green" StrokeThickness="5">
         <Path.Data>
            <LineGeometry StartPoint="150,150" EndPoint="300,300"/>
         </Path.Data>
</Path>
```

Geometria

Geometrias baseadas em trajetos (*PATHS*) – as geometrias baseadas em trajetos são construídas através da utilização da classe *PathGeometry*. Essa classe contém uma coleção de objetos do tipo *PathFigure* (propriedades Figures) que permitem a fácil construção de vários objetos do tipo *PathSegment*.

```
<Path Stroke="Gray" StrokeThickness="3">
        <Path.Data>
          <PathGeometry>
            <PathFigure StartPoint="10,10">
              <LineSegment Point="110,10"/>
              <LineSegment Point="110,110"/>
              <LineSegment Point="10,110"/>
            </PathFigure>
          </PathGeometry>
        </Path.Data>
      </Path>
```

```
<PathFigure StartPoint=“10,10” IsClosed=True”>
```

Geometria

Preenchimento de geometrias – a classe *GeometryPath* introduz uma propriedade (*FillRule*) que define a forma como a sobreposição de pontos é interpretada numa geometria deste tipo.

O valor da propriedade *FillRule* provém de uma enumeração e pode ser um dos dois valores:

- **EvenOdd**: a região é preenchida apenas quando temos de atravessar um número ímpar de segmentos para passar do interior da figura para o exterior.
- **NonZero**: o algoritmo usado neste caso é bem mais complexo do que o anterior. Este algoritmo leva em consideração a direção do segmento “atravessado” para passar do interior para o exterior da figura. Na maior parte dos casos, este algoritmo acaba simplesmente por preencher toda a área delimitada pela geometria.

Geometria

```
<Path Stroke="Gray" StrokeThickness="2" Fill="Red">
        <Path.Data>
          <PathGeometry FillRule="EvenOdd">
            <PathFigure IsClosed="True" >
              <LineSegment Point="0,100"/>
              <LineSegment Point="100,100"/>
            </PathFigure>
            <PathFigure StartPoint="70,0" IsClosed="True" >
              <LineSegment Point="0,100"/>
              <LineSegment Point="100,100"/>
            </PathFigure>
          </PathGeometry>
        </Path.Data>
</Path>
```

Teste as propriedades ***NonZero*** e ***IsFilled***

Geometria

Composição de geometrias – a classe GeometryGroup permite o agrupamento de vários objetos do tipo Geometry, resultando assim na criação de uma figura final composta à custa da interseção das várias geometrias usadas.

```
<Path Stroke="Gray" StrokeThickness="2" Fill="Red">
        <Path.Data>
          <GeometryGroup FillRule="EvenOdd">
            <RectangleGeometry Rect="10,10,100,100"/>
            <RectangleGeometry Rect="50,50,100,100"/>
          </GeometryGroup>
        </Path.Data>
</Path>
```

Geometria

MINI-LINGUAGEM DE REPRESENTAÇÂO – a plataforma Silverlight/WPF suporta uma mini-linguagem que permte definir os segmentos usados numa Figura através de uma string. Veja o exemplo a seguir:

```
<Path Stroke="Black" StrokeThickness="2" Fill="Red"
        Data="M 10,10 H 110 V 110 H 10 V 10 M 50,50 H 150 V 150 H 50 V 50"/>
```

Onde

M – indica o ponto de início de cada Figura (10,10) para o 1º e (50,50) para o 2º retângulo.

H – gera uma linha horizontal do ponto atual para a coordenada x associada ao comando.

V – gera uma linha vertical do ponto atual para a coordenada y associada ao comando.

Z – finaliza a figura ligando o último ponto ao ponto inicial.

Geometria

```
<Grid>
      <Path Stroke="Black" StrokeThickness="2" Fill="Red"
          Data="M 10,10 H 110 V 110 H 10 Z M 50,50 H 150 V 150 H 50 Z"/>
   </Grid>
```

Shapes

A classe Shape é abstrata e serve de base a vários tipos que permitem desenhar gráficos na tela. Veja algumas de suas propriedades:

- **Fill** : indica a cor (brush) do interior da figura.
- **GeometryTransform** : permite aplicar uma transformação a um shape antes dela ser renderizada.
- **Stretch** : ajusta a shape de acordo com as dimensões do elemento pai.
- **Stroke** : indica a cor (brush) usado para renderizar o contorno da shape.
- **StrokeDashArray** : define o padrão de traços usados na definição do contorno da shape.
- **StrokeDashCap** : indica a forma como o final de um traço deve ser representado.
- **StrokeDashOffSet** : permiti-nos definir a distância a que o padrão de traços definido deve começar.
- **StrokeEndLineCap** : define a forma como o final de uma linha deve ser representada.
- **StrokeMiterLimit** : define a forma como duas linhas pertencentes a uma sahpe se interceptam.
- **StrokeStartLineCap** : semelhante à propriedade StrokeEndLineCap, mas neste caso, influencia o início da linha.

shapes

Tipos de shapes: Ellipse, Line, Path, Polygon, Polyline e Rectangle.

```
<Line Stroke="Blue" X1="10" Y1="10" X2="100" Y2="110"/>
<Line Stroke="Blue" X1="5" Y1="100" X2="15" Y2="200"/>
```

```
<Polygon Stroke="Blue" StrokeThickness="5"
         Points="10,150,30,140,50,160,70,130,90,170,110,120,130,
         180,150,110,170,190,190,100,210,240"
         Fill="Yellow" />
```

```
<Polygon Stroke="Blue" StrokeThickness="1" Fill="Yellow"
         Points="15,200,68,70,110,200,0,125,135,125"
         FillRule="Nonzero"/>
```

Shapes

```
<StackPanel>
        <Ellipse Fill="Yellow" Stroke="Blue" Height="50" Width="100"
                Margin="5" HorizontalAlignment="Left" />
        <Rectangle Fill="Yellow" Stroke="Blue" Height="50" Width="100"
                 Margin="5" HorizontalAlignment="Left" />
</StackPanel>
```

Desafio, construa uma elipse a partir de um retângulo.

Polígono

Um desenha uma forma fechada, enquanto o polyline desenha uma forma aberta.

```
<Polygon Stroke="Black" Points="10,40 20,10 60, 10 70,40 10,40"/>
```

Brushes

Para colorir o interior de uma Shape ou outros vários elementos visuais, deve-se escolher uma das inúmeras opções de **Brushes**, incluindo SolidColorBrush, LinearGradientBrush, RadialGradientBrush, ImageBrush e VideoBrush.

SolidColorBrush

A mais rudimentar das opções de Brush. Uma SolidColorBrush usa uma única cor sólida para colorir uma área.

```
<Ellipse Stroke="Black" StrokeThickness="3" Width="64" Height="64">
        <Ellipse.Fill>
           <SolidColorBrush Color="Navy" />
        </Ellipse.Fill>
</Ellipse>
```

LinearGradientBrush

A LinearGradientBrush pinta uma área com uma alternância gradual e suave de cores ao longo de uma linha teórica.

```
<Ellipse Stroke="Black" StrokeThickness="3" Width="64" Height="64">
        <Ellipse.Fill>
          <LinearGradientBrush>
            <GradientStop Color="Navy" Offset="0" />
            <GradientStop Color="White" Offset="1" />
          </LinearGradientBrush>
        </Ellipse.Fill>
</Ellipse>
```

```
<Rectangle StrokeThickness="0" Width="200" Height="64">
        <Rectangle.Fill>
          <LinearGradientBrush StartPoint="0,0" EndPoint="1,0">
            <GradientStop Color="Yellow" Offset="0" />
            <GradientStop Color="Orange" Offset=".45" />
            <GradientStop Color="Blue" Offset=".55" />
            <GradientStop Color="Green" Offset="1" />
          </LinearGradientBrush>
        </Rectangle.Fill>
      </Rectangle>
```

RadialGradientBrush

A RadialGradientBrush é muito semelhante à LinearGradientBrush, exceto porque as transações de cor se iniciam em um Point de origem. Conforme a Brush se difunde a partir do centro, ela pinta transações elípticas gradualmente até que um GradientStop seja encontrado. Esse processo continua de uma GradientStop ao próximo até que cada um deles tenha sido renderizado.

```
<Ellipse Width="75" Height="75" Stroke="Black">
        <Ellipse.Fill>
           <RadialGradientBrush>
              <GradientStop Color="Black" Offset="0"/>
              <GradientStop Color="Gray" Offset="0.5"/>
              <GradientStop Color="Black" Offset="1"/>
           </RadialGradientBrush>
        </Ellipse.Fill>
</Ellipse>
```