Course 2D_SL: 2D-Computer Graphics with Silverlight
Chapter C5: The Complete Code of PathAnimation

Copyright © by V. Miszalok, last update: 30-01-2009

Install 1) Visual Web Developer 2008 Express Edition with Service Pack 1 English
and 2) Silverlight Tools for Visual Studio 2008 SP1.

Preliminaries

Guidance for **Visual Web Developer 2008 Express**:

1) Main Menu after start of VWD Express: `Tools` → `Options` →
check lower left checkbox: `Show all Settings` →
→ `Projects and Solutions` → `Projects location:` → `C:\temp`.
→ `Text Editor` (double click) → `All Languages` (double click) → `Tabs` →
`Indenting: None` → `Tab size: 2` → `Insert spaces`.
→ `Text Editor` (double click) → `C#` (double click) → `Formatting` → uncheck all three check boxes → `OK`.
→ `Text Editor` (double click) → `XAML` (double click) → `Tabs` →
`Indenting: None` → `Tab size: 1, Indent size: 1` → `Insert spaces`.
→ `Text Editor` (double click) → `XAML` (double click) → `Formatting` →
uncheck all `Auto-Formatting Events` → `OK`.

2) Main Menu after start of VWD Express: `File` → `New Project...` →
`Project types: Visual C#` (double click) → `Silverlight` →
`Templates: Silverlight Application`
`Name: SL_path1` → `Location: C:\temp\SLProjects` →
`Create directory for solution:` switch off → `OK`.
An `Add Silverlight Application`-Window appears.
Choose "`Automatically generate a test page to host Silverlight at build time`" → `OK`.

In the `Solution Explorer - SL_path1` click the branch `Page.xaml`.
A split window will appear containing the default xaml code.
Delete this code completely and replace it by the code of **Page.xaml** which follows.

Page.xaml

```
<UserControl
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        x:Class="path1.Page"
        Width="540" Height="225">
  <UserControl.Resources>
    <Storyboard x:Name="Storyboard1"/>
  </UserControl.Resources>
```

```
<Border BorderBrush="Black" BorderThickness="2">
  <StackPanel Orientation="Vertical">
    <Canvas x:Name="LayoutRoot" Background="Cornsilk" Height="200">
      <Path Data ="M 10,50 h 10 v 100 h 25 v -100                          h 10
                   m 10, 0 h 10 l 12.5,100 l 12.5,-100                     h 10
                   m 10, 0 h 10 a 12.5,100 0 0 0  25,0                     h 10
                   m 10, 0 h 10 q 12.5,100,       25,0                     h 10
                   m 10, 0 h 10 c  0  ,100,25,100,25,0                     h 10
                   m 10, 0 h 10 c  0  ,100,25,100,25,0 c 0, 100,25, 100,25,0 h 10
                   m 10,50 h 10 c  0  ,100,25,100,25,0 c 0,-100,25,-100,25,0 h 10
                   m 10, 0 h 10 c  0  ,100,25,100,25,0 s        25,-100,25,0 h 10"
            x:Name="PathThickBlack" Stroke="Black" StrokeThickness="5"/>
      <Path x:Name="PathThinGreen"  Stroke="Green" StrokeThickness="2"
                    Loaded="PathThinGreenLoaded">
        <Path.Data>
          <PathGeometry>
            <PathGeometry.Figures>
              <PathFigure StartPoint=" 10  , 50">
                <LineSegment   Point=" 20  , 50"/>
                <LineSegment   Point=" 20  ,150"/>
                <LineSegment   Point=" 45  ,150"/>
                <LineSegment   Point=" 45  , 50"/>
                <LineSegment   Point=" 55  , 50"/>
              </PathFigure>
              <PathFigure StartPoint=" 65  , 50">
                <LineSegment   Point=" 75  , 50"/>
                <LineSegment   Point=" 87.5,150"/>
                <LineSegment   Point="100  , 50"/>
                <LineSegment   Point="110  , 50"/>
              </PathFigure>
              <PathFigure StartPoint="120  , 50">
                <LineSegment   Point="130  , 50"/>
                <ArcSegment    Point="155  , 50" Size="12.5,100"/>
                <LineSegment   Point="165  , 50"/>
              </PathFigure>
              <PathFigure StartPoint="175  , 50">
                <LineSegment   Point="185  , 50"/>
                <QuadraticBezierSegment Point1="197.5,150" Point2="210,50"/>
                <LineSegment   Point="220  , 50"/>
              </PathFigure>
              <PathFigure StartPoint ="230, 50">
                <LineSegment   Point ="240, 50"/>
                <BezierSegment Point1="240,150" Point2="265,150" Point3="265,50"/>
                <LineSegment   Point ="275, 50"/>
              </PathFigure>
              <PathFigure StartPoint ="285, 50">
                <LineSegment   Point ="295, 50"/>
                <BezierSegment Point1="295,150" Point2="320,150" Point3="320,50"/>
                <BezierSegment Point1="320,150" Point2="345,150" Point3="345,50"/>
                <LineSegment   Point ="355, 50"/>
              </PathFigure>
              <PathFigure StartPoint ="365,100">
                <LineSegment   Point ="375,100"/>
                <BezierSegment Point1="375,200" Point2="400,200" Point3="400,100"/>
                <BezierSegment Point1="400,  0" Point2="425,  0" Point3="425,100"/>
                <LineSegment   Point ="435,100"/>
              </PathFigure>
              <PathFigure     StartPoint ="445,100">
                <LineSegment       Point ="455,100"/>
                <PolyBezierSegment Points="455,200 480,200 480,100
                                           480,  0 505,  0 505,100"/>
                <LineSegment       Point ="515,100"/>
              </PathFigure>
            </PathGeometry.Figures>
          </PathGeometry>
        </Path.Data>
      </Path>
```

```
      <Path x:Name="PathDottedRed" Stroke="Red" StrokeThickness="5"
            Loaded="PathDottedRedLoaded">
        <Path.Data><PathGeometry/></Path.Data>
      </Path>
    </Canvas>

    <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">
      <Button Content="Start" Click="start_Button_Click"/>
      <Button Content="Stop"  Click="stop_Button_Click" />
      <TextBlock Text="Velocity: "   VerticalAlignment="Center" Margin="20,0,0,0"/>
      <Slider   x:Name="velocity_slider" Width="80"
                Minimum="100" Maximum="4000" IsDirectionReversed="True" Value="1000"
                ValueChanged="VelocitySliderValueChanged"/>
      <CheckBox x:Name="Red_Dots"    VerticalAlignment="Center" Margin="20,0,0,0"
                Content  ="Red Dots"
                Checked  ="ShowDottedRedPath"
                Unchecked="HideDottedRedPath"/>
      <CheckBox x:Name="BlackCurves" VerticalAlignment="Center" Margin="20,0,0,0"
                Content  ="Black Curves"
                Checked  ="ShowThickBlackPath"
                Unchecked="HideThickBlackPath"/>
    </StackPanel>
  </StackPanel>

  </Border>
</UserControl>
```

Page.xaml.cs

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace path1
{       public partial class Page : UserControl
        { PointCollection pp = new PointCollection();
    public Page()
                { InitializeComponent();
      PathThickBlack.Visibility = Visibility.Collapsed;
      PathDottedRed .Visibility = Visibility.Collapsed;
                }

    //This initial event occurs just once.
    //It copies all 53 vertices to the intermediary dynamic array pp and
    //animates 7 of the 29 segments of PathThinGreen
    //by calling the subroutine add_points_to_pp_and_to_Storyboard1(..)
    private void PathThinGreenLoaded( object sender, EventArgs e )
    { PathGeometry PG = (PathGeometry)PathThinGreen.Data;
      //read curves from XAML and buffer all vertices to pp
      for ( int i=0; i < PG.Figures.Count; i++ )
      { PathFigure PF = PG.Figures[i];
        pp.Add( PF.StartPoint );
        for ( int j=0; j < PF.Segments.Count; j++ )
          add_points_to_pp_and_to_Storyboard1( PF.Segments[j], j, i );
      }
    }
```

```
//This initial event occurs just once.
//It creates and animates the tiny red points of PathDottedRed
//It copies any vertex into PathDottedRed and animates the lower vertices by
//calling the subroutine add_a_PointAnimation_to_Storyboard1(...)
private void PathDottedRedLoaded( object sender, EventArgs e )
{ PathGeometry PG = (PathGeometry)PathDottedRed.Data;
  //All red "points" are packed inside a Path named PathDottedRed
  //of PathFigures each carrying one tiny horizontal LineSegment
  foreach ( Point p in pp )
  { PathFigure  PF = new PathFigure();
    LineSegment LS = new LineSegment();
    PF.StartPoint = new Point( p.X-2, p.Y );
    LS.Point      = new Point( p.X+2, p.Y );
    PF.Segments.Add( LS );
    PG.Figures .Add( PF  );
  }
  //This loop animates the lower "red points"
  //Storyboard1 will contain such a lower point triply:
  //1. as animated vertex of PathThinGreen (done by the first initial event handler)
  //2. as animated left  border of a 5-pixel horizontal line of PathDottedRed
  //3. as animated right border of a 5-pixel horizontal line of PathDottedRed
  for ( int i=0; i < pp.Count; i++ )
  {  if ( pp[i].Y > 100 )
     { //The LineSegments have a start and an end point with a horizontal distance = 5
       Point p_left  = new Point( pp[i].X-2, pp[i].Y );
       Point p_right = new Point( pp[i].X+2, pp[i].Y );
       String s1 = "(Path.Data).(PathGeometry.Figures)[" +
                    i.ToString() + "].(PathFigure.StartPoint)";
       String s2 = "(Path.Data).(PathGeometry.Figures)[" +
                    i.ToString() + "].(PathFigure.Segments)[0].(LineSegment.Point)";
       add_a_PointAnimation_to_Storyboard1( PathDottedRed, p_left , s1 );
       add_a_PointAnimation_to_Storyboard1( PathDottedRed, p_right, s2 );
     }
  }
}

//This subroutine is called from the initial event handler PathThinGreen
//It adds any vertex to the intermediary dynamic array pp and
//animates 7 of the 29 segments of PathThinGreen
private void add_points_to_pp_and_to_Storyboard1( PathSegment PS, int NoOfPS, int NoOfPF )
{ String s1 = "(Path.Data).(PathGeometry.Figures)[" + NoOfPF.ToString() +
                        "].(PathFigure.Segments)[" + NoOfPS.ToString() + "].";
  String s; //complete PropertyPath-string aimed to feed Storyboard.SetTargetProperty(...)
  Point  p;
  if ( PS.GetType() == typeof(LineSegment) )
  { p = ((LineSegment)PS).Point; pp.Add( p );
    if ( p.Y > 100 )
    { s = s1 + "(LineSegment.Point)";
      add_a_PointAnimation_to_Storyboard1( PathThinGreen, p, s );
    }
  }
  else if ( PS.GetType() == typeof(ArcSegment) )
  { ArcSegment AS = ((ArcSegment)PS);
    p = new Point( AS.Point.X-AS.Size.Width, AS.Point.Y+AS.Size.Height );
    pp.Add( p );
    //The Size.Height property of an ArcSegment cannot be animated
  }
  else if ( PS.GetType() == typeof(QuadraticBezierSegment) )
  { p = ((QuadraticBezierSegment)PS).Point1; pp.Add( p );
    if ( p.Y > 100 )
    { s = s1 + "(QuadraticBezierSegment.Point1)";
      add_a_PointAnimation_to_Storyboard1( PathThinGreen, p, s );
    }
    p = ((QuadraticBezierSegment)PS).Point2; pp.Add( p );
  }
```

```
      else if ( PS.GetType() == typeof(BezierSegment) )
      { p = ((BezierSegment)PS).Point1; pp.Add( p );
        if ( p.Y > 100 )
        { s = s1 + "(BezierSegment.Point1)";
          add_a_PointAnimation_to_Storyboard1( PathThinGreen, p, s );
        }
        p = ((BezierSegment)PS).Point2; pp.Add( p );
        if ( p.Y > 100 )
        { s = s1 + "(BezierSegment.Point2)";
          add_a_PointAnimation_to_Storyboard1( PathThinGreen, p, s );
        }
        p = ((BezierSegment)PS).Point3; pp.Add( p );
      }
      else if ( PS.GetType() == typeof(PolyBezierSegment) )
        foreach ( Point point in ((PolyBezierSegment)PS).Points ) pp.Add( point );
        //Points inside a PolyBezierSegment cannot be animated
    }

    //This subroutine is called from add_points_to_pp_and_to_Storyboard1(...) and
    //directly from the second initial event handler PathDottedRedLoaded(...)
    //It sets the common properties that all 42 animations share.
    private void add_a_PointAnimation_to_Storyboard1( Path path, Point p, string s )
    { PointAnimation PA = new PointAnimation();
      PA.To = new Point ( p.X, p.Y-150 );
      PA.Duration = TimeSpan.FromMilliseconds( 1000 );
      PA.AutoReverse = true;
      PA.RepeatBehavior = RepeatBehavior.Forever;
      Storyboard.SetTarget        ( PA, path );
      Storyboard.SetTargetProperty( PA, new PropertyPath( s ) );
      Storyboard1.Children.Add    ( PA );
    }

    private void start_Button_Click( object sender, EventArgs e )
    { Storyboard1.Begin(); }

    private void stop_Button_Click( object sender, EventArgs e )
    { Storyboard1.Stop(); }

    private void ShowDottedRedPath( object sender, EventArgs e )
    { PathDottedRed.Visibility = Visibility.Visible; }

    private void HideDottedRedPath( object sender, EventArgs e )
    { PathDottedRed.Visibility = Visibility.Collapsed; }

    private void ShowThickBlackPath( object sender, EventArgs e )
    { PathThickBlack.Visibility = Visibility.Visible; }

    private void HideThickBlackPath( object sender, EventArgs e )
    { PathThickBlack.Visibility = Visibility.Collapsed; }

    private void VelocitySliderValueChanged( object sender, EventArgs e )
    { double millisec = 1000;
      try { millisec = velocity_slider.Value; } catch { return; }
      foreach ( PointAnimation PA in Storyboard1.Children )
        PA.Duration = TimeSpan.FromMilliseconds( millisec );
    } //end of private void on_velocity_slider_value_changed( ... )
  }   //end of class Page
}     //end of namespace
```