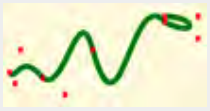
Course 2D_SL: 2D-Computer Graphics with Silverlight
Chapter C6: The Complete Code of the BezierAnimation Project



- Preliminaries
- Page.XAML
- Page.xaml.cs
- Test

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_bezier1` → `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_bezier1` 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="Bezier1.Page"
  Width="400" Height="220">

  <UserControl.Resources>
    <Storyboard x:Name="Storyboard1" Completed="Storyboard1_Completed"/>
  </UserControl.Resources>
```

```
  <Border BorderBrush="Black" BorderThickness="2">
    <StackPanel Orientation="Vertical">
      <Canvas x:Name="canvas" Background="Cornsilk" Height="190">
        <Path Margin="0,50,0,0" Stroke="Green"
              StrokeThickness="5" StrokeStartLineCap="Round"
              x:Name="curve_path" Loaded="curve_path_Loaded">
          <Path.Data>
            <PathGeometry>
              <PathFigure StartPoint ="   0,  0" x:Name="snake">
                <BezierSegment Point1="  30,  0" Point2="  30, 20" Point3="   0, 5"/>
                <BezierSegment Point1=" -30,-30" Point2=" -40,120" Point3=" -65,30"/>
                <BezierSegment Point1=" -78,-20" Point2=" -90, 70" Point3="-110,55"/>
                <BezierSegment Point1="-130, 30" Point2="-135, 60" Point3="-140,50"/>
              </PathFigure>
            </PathGeometry>
          </Path.Data>
        </Path>
        <Path Margin="0,50,0,0" Stroke="Red" StrokeThickness="5"
          x:Name="dotted_path" Loaded="dotted_path_Loaded">
          <Path.Data><PathGeometry/></Path.Data>
        </Path>
      </Canvas>
      <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">
        <Button Content="Start" Click="start_Button_Click"/>
        <Button Content="Step"  Click="step_Button_Click"/>
        <TextBlock Text="Velocity: " VerticalAlignment="Center" Margin="20,0,0,0"/>
        <Slider x:Name="velocity_slider" Width="80" Margin="0,0,20,0"
                Minimum="1" Maximum="500" IsDirectionReversed="True" Value="75"
                ValueChanged="on_velocity_slider_value_changed"/>
        <CheckBox x:Name="show_bezier_points" VerticalAlignment="Center"
                  Content="Show Bezier Points"
                  Checked  ="show_bezier_points_Checked"
                  Unchecked="show_bezier_points_Unchecked"/>
      </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;

namespace Bezier1
{ public partial class Page : UserControl
  { private Point[] p;
    //each point of the snake needs its displacement at each key time
    //no. of points = StartPoint + (no. of segments) * (3 BezierPoints)
    //no. of key times = 3
    private Point[,] dp =
    { { new Point(  20,   0),
        new Point(  20,   0), new Point(  20,   0), new Point(  20,   0),
        new Point(  20, -10), new Point(  20, -10), new Point(  20,   0),
        new Point(   0,   0), new Point(   0,   0), new Point(   0,   0),
        new Point(   0,   0), new Point(   0,   0), new Point(   0,   0) },
      { new Point(   0,   0),
        new Point(   0,   0), new Point(   0,   0), new Point(   0,   0),
        new Point(   0,  10), new Point(   0,  10), new Point(   0,   0),
        new Point(  20,   0), new Point(  20, -20), new Point(  20,   0),
        new Point(   0,   0), new Point(   0,   0), new Point(   0,   0) },
      { new Point(   0,   0),
        new Point(   0,   0), new Point(   0,   0), new Point(   0,   0),
        new Point(   0,   0), new Point(   0,   0), new Point(   0,   0),
        new Point(   0,   0), new Point(   0,  20), new Point(   0,   0),
        new Point(  20,   0), new Point(  20,   0), new Point(  20,   0) }
    };
    //3 consecutive key times defined by day, hour, minute, sec, millisec
    private TimeSpan[] TS = { new TimeSpan( 0, 0, 0, 0,  75 ),
                              new TimeSpan( 0, 0, 0, 0, 150 ),
                              new TimeSpan( 0, 0, 0, 0, 187 ) };
    Boolean animation_run = true;
```

```
public Page()
{ InitializeComponent();
  p = new Point[1+snake.Segments.Count*3]; //no. of Bezier points = 10
  p[0] = snake.StartPoint;  //store the XAML-defined start point of snake to arry p
  int i = 1; //store all XAML-defined Bezier points of all snake segments to arry p
  foreach ( BezierSegment BS in snake.Segments )
  { p[i++] = BS.Point1; p[i++] = BS.Point2; p[i++] = BS.Point3; }
}

//This event occurs only once at first start
//creates the PointAnimations and starts the first move
private void curve_path_Loaded( object sender, RoutedEventArgs e )
{ String s = "(Path.Data).(PathGeometry.Figures)[0].(PathFigure.";
  int j = 0;
  create_animation( j++, s + "StartPoint)" );
  for ( int a=0; a < snake.Segments.Count; a++ )
    for ( int b=1; b <= 3; b++ )
      create_animation( j++, s +    "Segments)[" + a.ToString() +
                        "].(BezierSegment.Point" + b.ToString() + ")" );
  Storyboard1.Begin();
}

//This is a subroutine of curve_path_Loaded
//parameters: j = point no., s = target string
//1. create an animation
//2. connect it to curve_path
//3. connect it to a specific Bezier Point of curve_path via string s
//4. add it to Storyboard1
//5. create 3 KeyFrames per animation
//6. set the destinations and durations of the first moves
//7. add the KeyFrames to the animation
private void create_animation( int j, string s )
{ PointAnimationUsingKeyFrames  PAUKF = new PointAnimationUsingKeyFrames();
  Storyboard.SetTarget        ( PAUKF, curve_path );
  Storyboard.SetTargetProperty( PAUKF, new PropertyPath( s ) );
  Storyboard1.Children.Add    ( PAUKF );
  for ( int i=0; i < TS.Length; i++ )  //i between 0 and 2
  { LinearPointKeyFrame LPKF = new LinearPointKeyFrame(); //3 new LPKFs
    PAUKF.KeyFrames.Add( LPKF ); //attach this LPKF to the animation
    p[j].X += dp[i,j].X;         //first x-step of this point
    p[j].Y += dp[i,j].Y;         //first y-step of this point
    LPKF.Value   = p[j];         //first movement
    LPKF.KeyTime = TS[i];        //set movement duration
  }
}

//This event occurs only once at first start. It creates
//a dotted red line with 10 fragments looking like 10 small red rectangles
private void dotted_path_Loaded( object sender, RoutedEventArgs e )
{ //Fill the empty points_path with empty PFs each containing an empty LS.
  //Doing this here in a for-loop is much shorter than writing them many times in XAML.
  PathGeometry PG = (PathGeometry)dotted_path.Data;
  for ( int j=0; j < p.Length; j++ )
  { PathFigure  PF = new PathFigure (); PG.Figures .Add( PF ); //1 PathFigure per point
    LineSegment LS = new LineSegment(); PF.Segments.Add( LS ); //1 LineSegm/PathFigure
  }
}
```

```
    //This event occurs after any complete move of the snake and another move is started
    private void Storyboard1_Completed( object sender, EventArgs e )
    { //fill all PathFigures of dotted_path with a StartPoint and
        a horizontal 5-pixel LineSegment
      if ( (bool)show_bezier_points.IsChecked )
      { PathGeometry PG = (PathGeometry)dotted_path.Data;
        for ( int j=0; j < p.Length; j++ )
        { PathFigure PF = PG.Figures[j];
          PF.StartPoint                          = new Point( p[j].X-2, p[j].Y );
          ( (LineSegment)PF.Segments[0] ).Point = new Point( p[j].X+2, p[j].Y );
        }
      }
      for ( int j=0; j < p.Length; j++ )
      { PointAnimationUsingKeyFrames PAUKF =
         (PointAnimationUsingKeyFrames)Storyboard1.Children[j];
        for ( int i=0; i < TS.Length; i++ )          //i between 0 and 2
        { LinearPointKeyFrame LPKF = (LinearPointKeyFrame)PAUKF.KeyFrames[i];
          p[j].X += dp[i,j].X;                       //x-step
          p[j].Y += dp[i,j].Y;                       //y-step
          if ( p[j].X > canvas.ActualWidth )         //right margin of canvas reached ?
          { p[j].X = 2* canvas.ActualWidth - p[j].X; //turn left
            dp[i,j].X *= -1;                         //reverse direction
          }
          else if ( p[j].X < 0 && dp[i,j].X < 0 )    //left margin of canvas reached ?
          { p[j].X    *= -1;                         //turn right
            dp[i,j].X *= -1;                         //reverse direction
          }
          LPKF.Value = p[j];
        }
      }
      if ( animation_run ) Storyboard1.Begin();
    }

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

    private void step_Button_Click              ( object sender, EventArgs e )
    { animation_run = false; Storyboard1.Begin(); }

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

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

    private void on_velocity_slider_value_changed( object sender, EventArgs e )
    { double millisec = 75;
      try { millisec = velocity_slider.Value; } catch { return; }
      TS[0] = TimeSpan.FromMilliseconds(     millisec );
      TS[1] = TimeSpan.FromMilliseconds( 2  *millisec );
      TS[2] = TimeSpan.FromMilliseconds( 2.5*millisec );
      for ( int j=0; j < Storyboard1.Children.Count; j++ )
      { PointAnimationUsingKeyFrames PAUKF =
         (PointAnimationUsingKeyFrames)Storyboard1.Children[j];
        for (int i=0; i < TS.Length; i++ )      //i between 0 and 2
        { LinearPointKeyFrame LPKF = (LinearPointKeyFrame)PAUKF.KeyFrames[i];
          LPKF.KeyTime = TS[i];                 //set movement duration
        }
      }
    } //end of private void on_velocity_slider_value_changed( ... )
  }   //end of class Page
}     //end of namespace
```