Magnifying Video Brush in Windows Phone

This code example explains how to associate Magnifying functionality with Silverlight Video Brush in Windows Phone.

Introduction

While working with the video brush API, I wanted to see an object in a magnified view and I realized that there is no such API which allows to do video stream zoom in and zoom out. However it is very easy to implement such kind of functionality using managed API in Windows Phone.

This article shows how to use the Silverlight VideoBrush API to magnify the video camera in Windows Phone application using manage API. Here is a video demonstration of what we wish to achieve:

The media player is loading...

Prerequisites

For Windows Phone 7.5,

- Windows 7 OS or above
- Windows Phone 7.5 SDK
- Visual Studio Express Edition 2010 for Windows Phone

For Windows Phone 8,

- Windows 8 OS
- Visual Studio 2012 or
- Windows Phone 8 SDK (If you have)

API used

The main APIs for this solution are:

1. PhotoCamera
2. CameraVideoBrushExtensions
3. CompositeTransform
4. VideoBrush

Implementation

Step 1: Creating Windows Phone application using traditional way.

Create a Windows Phone 7 project, using traditional way using Microsoft Visual Studio -> File -> New Project -> Select template Silverlight for Windows Phone 7 and choose *Windows Phone Application*. Here I have named the application **WmDev_Magnifier**.

Step 2: Layout designing using XAML code; to build the basic functionality of magnifier application, it uses standard controls such as Canvas, Slider and TextBlock in **MainPage.xaml**.

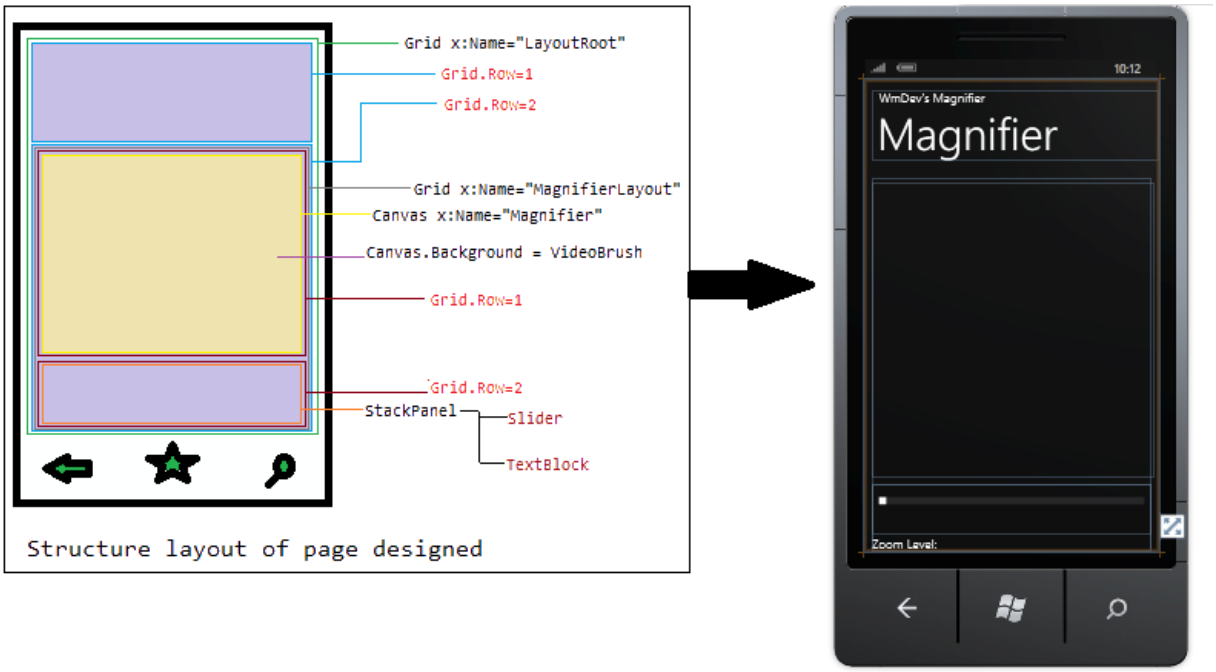
XAML Code Snippet:

```
<Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="WmDev's Magnifier"
Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Magnifier" Margin="9,-7,0,0"
```


(C) Copyright Nokia 2014. All rights reserved.

```
Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->

        <Grid x:Name="MagnifierLayout" Grid.Row="1" Margin="12,0,12,0"
Background="Transparent">
            <Grid.RowDefinitions>
                <RowDefinition Height="*"/>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
            </Grid.RowDefinitions>

            <Canvas x:Name="Magnifier" Width="460" Height="480" Grid.Row="0"
Tap="Magnifier_Tap">
                <Canvas.Background>
                    <VideoBrush x:Name="vBrush" >
                        <VideoBrush.RelativeTransform>
                            <CompositeTransform x:Name="viewCompositeTransform"
                            CenterX=".5"
                            CenterY=".5" />
                        </VideoBrush.RelativeTransform>
                    </VideoBrush>
                </Canvas.Background>
            </Canvas>

            <!--ContentPanel - place additional content here-->
            <StackPanel x:Name="ControlPanel" Grid.Row="1" Margin="0,0,0,0">
                <Slider x:Name="MagnifierSlider"
ValueChanged="MagnifierSlider_ValueChanged" Minimum="1" Maximum="4" IsEnabled="False"
Orientation="Horizontal"></Slider>
                <TextBlock x:Name="txtZoom" Text="Zoom Level:" />
            </StackPanel>
        </Grid>
    </Grid>
```

Design output in below figure.



(C) Copyright Nokia 2014. All rights reserved.

Step 3: Attaching two important event handlers to the specific components to get a specific functionality in the application. `Tap` event to Canvas, to set the Auto focus operation on specific points in the viewfinder and `ValueChanged` event to Slider, for Magnifying view using `viewCompositeTransform` and scale the x axis factor.

Step 4: Next initialize the PhotoCamera class' object in the `OnNavigatedTo` of page. And attaching `Initialized` event to `PhotoCamera` object and set the `VideoBrush` source to the `CameraVideoBrushExtensions` using set source method and also setting rotation property to the `viewCompositeTransform` from the `PhotoCamera` object Orientation. The following snippet illustrates this.

```
 private PhotoCamera oCamera;

        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            if (Camera.IsCameraTypeSupported(CameraType.Primary))
            {
                // Use standard camera on back of device.
                this.oCamera = new PhotoCamera(CameraType.Primary);

                // Event is fired when the PhotoCamera object has been initialized.
                this.oCamera.Initialized += new
EventHandler<CameraOperationCompletedEventArgs>(cam_Initialized);

                //Set the VideoBrush source to the camera.
                CameraVideoBrushExtensions.SetSource(this.vBrush, this.oCamera);
                this.viewCompositeTransform.Rotation = this.oCamera.Orientation;
            }
        }

        protected override void OnNavigatingFrom(NavigatingCancelEventArgs e)
        {
            if (this.oCamera != null)
            {
                this.oCamera.Dispose();
                this.oCamera.Initialized -= new
EventHandler<CameraOperationCompletedEventArgs>(cam_Initialized);
            }
        }

        private void cam_Initialized(object sender, CameraOperationCompletedEventArgs e)
        {
            base.Dispatcher.BeginInvoke(delegate
            {
                this.MagnifierSlider.IsEnabled = true;
                this.MagnifierSlider.Value = 1.0;
                IEnumerable<Size> source = this.oCamera.AvailableResolutions;
                int num = source.Count<Size>();
                this.oCamera.Resolution = source.ElementAt<Size>(num - 1);
            });
        }

        private void Magnifier_Tap(object sender, System.Windows.Input.GestureEventArgs
```


(C) Copyright Nokia 2014. All rights reserved.

```
e)
        {
            if (this.oCamera != null)
            {
                if (this.oCamera.IsFocusAtPointSupported)
                {
                    try
                    {
                        Point position = e.GetPosition(this.Magnifier);
                        double num = position.X / this.Magnifier.Width;
                        double num2 = position.Y / this.Magnifier.Height;
                        this.oCamera.FocusAtPoint(num, num2);
                    }
                    catch { }
                }
                else if (this.oCamera.IsFocusSupported)
                {
                    try
                    {
                        this.oCamera.Focus();
                    }
                    catch { }
                }
            }
        }

        private void MagnifierSlider_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            if (this.MagnifierSlider != null)
            {
                this.viewCompositeTransform.ScaleX = this.MagnifierSlider.Value;
                this.viewCompositeTransform.ScaleY = this.MagnifierSlider.Value;
                this.txtZoom.Text = string.Format("Zoom level : {0:0.00}",
this.MagnifierSlider.Value);
            }
        }
```

Step-5: Now run the application by hitting F5 Key.

Source code

You can download source codes here: File:WmDev Magnifier.zip

References

1. PhotoCamera
2. CameraVideoBrushExtensions
3. CompositeTransform
4. VideoBrush

Happy Coding

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.