Creating a XAML Tile Control

One of the navigation mechanisms used in Windows 8 and Windows Phone is a Tile. A tile is a large rectangle that can have words and pictures that a user can click on. You can build your own version of a Tile in your WPF or Silverlight applications using a User Control. With just a little bit of XAML and a little bit of code-behind you can create a navigation system like that shown in Figure 1.

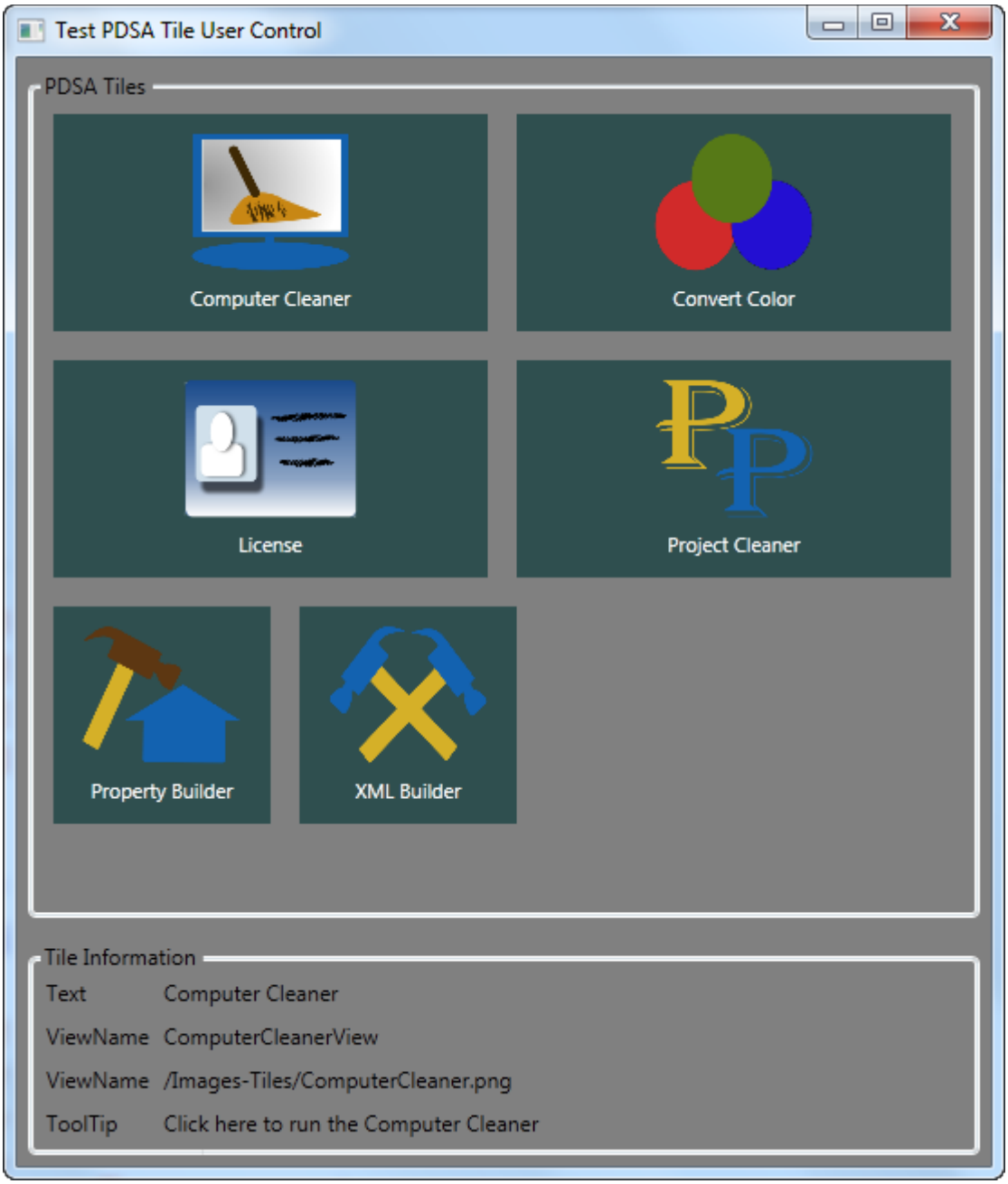

Figure 1: Use a Tile for navigation. You can build a Tile User Control with just a little bit of XAML and code.

The WPF application shown in Figure 1 uses a WrapPanel to display a series of Tile objects. There are two styles defined in this Window to give us a large tile and a small tile. These styles and the usage of the Tile will be shown later, but first let's look at how you can create this tile user control.

The User Control

In a WPF or Silverlight application you can create user controls which are a composite of other controls grouped together as a single unit. This user control can then be dragged and dropped onto a Window or User Control from the Visual Studio Toolbox. To create a “Tile” you need a Border, Grid, Image and a TextBlock control. Of course you will need to style these to get the appearance you saw in Figure 1. You will also need to use a Visual State Manager to highlight the tile the user is currently hovering over. The complete XAML for the tile is shown below:

```
<Border x:Name="borMain"
        Style="{StaticResource pdsaTileBorderStyle}"
        MouseEnter="OnMouseEnter"
        MouseLeave="OnMouseLeave"
        MouseLeftButtonDown="OnMouseLeftButtonDown">
  <VisualStateManager.VisualStateGroups>
    <VisualStateGroup Name="MouseStates">
      <VisualState Name="MouseEnter">
        <Storyboard>
          <ColorAnimation
            To="{StaticResource
                    pdsaTileBorderHighlightColor}"
            Duration="00:00:00"
            Storyboard.TargetName="borMain"
            Storyboard.TargetProperty="BorderBrush.Color" />
        </Storyboard>
      </VisualState>
      <VisualState Name="MouseLeave" />
    </VisualStateGroup>
  </VisualStateManager.VisualStateGroups>
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="*" />
      <RowDefinition Height="Auto" />
    </Grid.RowDefinitions>
    <Image Grid.Row="0"
           Name="imgMain"
           Style="{StaticResource pdsaTileImageStyle}"
           Source="{Binding TileImageUri}" />
    <TextBlock Grid.Row="1"
               Name="tbText"
               Style="{StaticResource
                         pdsaTileTextBlockStyle}"
               Text="{Binding TileText}" />
  </Grid>
</Border>
```

The Border, the Image and TextBlock all have a style applied to them. A set of default styles are contained in a resource dictionary that comes with the user control. The user control and the resource dictionary are located in a

DLL named PDSA.WPF. You can override the default resource dictionary with one of your own to create a different look and feel for your tiles. You only need to keep the names of the styles the same.

The Visual State Manager has just a single ColorAnimation when the mouse enters the Border. This ColorAnimation will change the border brush color to the value specified in the style named **pdsaTileBorderHighlightColor**. The Border will respond to the MouseEnter and MouseLeave events and call the Visual State Manager to move to the states defined in the XAML as shown in the code below:

```
private void OnMouseEnter(object sender, MouseEventArgs e)
{
  VisualStateManager.GoToState(this, "MouseEnter", true);
}

private void OnMouseLeave(object sender, MouseEventArgs e)
{
  VisualStateManager.GoToState(this, "MouseLeave", true);
}
```

Creating the Click Event

In addition to the MouseEnter and MouseLeave events, the user control must also raise a Click event. The MouseLeftButtonDown event is defined on the Border control. When this event procedure is fired an instance of a class called PDSATileEventArgs is created and a Click event is raised. Here is the code for the MouseLeftButtonDown event.

```
private void OnMouseLeftButtonDown(object sender,
 MouseButtonEventArgs e)
{
  PDSATileEventArgs args = new PDSATileEventArgs();

  args.Text = this.Text;
  args.ViewName = this.ViewName;
  if(ImageUri != null)
    args.ImageUri = this.ImageUri.ToString();
  if(ToolTip != null)
    args.ToolTip = this.ToolTip.ToString();

  RaiseClick(args);
}

public delegate void TileClickEventHandler(object sender,
  PDSATileEventArgs e);

public event TileClickEventHandler Click;

protected void RaiseClick(PDSATileEventArgs e)
{
  if (null != Click)
    Click(this, e);
}
```

As you can see in the MouseLeftButtonDown event you create a new instance of a PDSATileEventArgs class. You gather the dependency properties from the user control and place those into this new PDSATileEventArgs object. Next, you call the RaiseClick method passing in this object. The Click event is raised from this method passing in the current tile object and the instance of the PDSATileEventArgs class. The PDSTileEventArgs class is shown below:

```
public class PDSATileEventArgs : EventArgs
{
  public PDSATileEventArgs() : base()
  {
    ViewName = string.Empty;
    Text = string.Empty;
    ImageUri = string.Empty;
    ToolTip = string.Empty;
  }

  public string ViewName { get; set; }
  public string Text { get; set; }
  public string ImageUri { get; set; }
  public string ToolTip { get; set; }
}
```

Create a Tile in your Application

After you have built this user control you can add a reference to the DLL that contains the user control. This user control will now show up in the Visual Studio Toolbox. Drag and drop a Tile control onto a window and set the appropriate properties via the Property Window or directly in the XAML. Below is the XAML for the “Computer Cleaner” tile shown in the upper left hand corner of Figure 1.

```
<my:PDSAucTile
        Name="tileComputerCleaner"
        Text="Computer Cleaner"
        ViewName="ComputerCleanerView"
        ToolTip="Click here to run the Computer Cleaner"
        ImageUri="/Images-Tiles/ComputerCleaner.png"
        Click="tile_Click"
        Style="{StaticResource tileLarge}" />
```

Responding to the Click Event

When you click on a tile a Click event will fire. This event has a normal event procedure signature where you are passed the object that fired the event and an event argument object. The event argument object is an instance of the PDSATileEventArgs class. This event argument object contains the Text, ViewName, ImageUri and the ToolTip properties that you set in the XAML. In the sample code below these values are simply displayed in text blocks on the main window.

```
private void tile_Click(object sender, PDSATileEventArgs e)
{
  tbText.Text = e.Text;
  tbViewName.Text = e.ViewName;
  tbImageUri.Text = e.ImageUri;
  tbToolTip.Text = e.ToolTip;
}
```

In your application you might use a switch statement on the ViewName property to figure out which view to display as shown below:

```
private void tile_Click(object sender, PDSATileEventArgs e)
{
  switch (e.ViewName)
  {
     Case "ComputerCleanerView":
       // Display the Computer Cleaner View
       break;

     Case "LicenseView":
       // Display the License View
       break;

      ...  etc.
  }
}
```

You should assign a unique ViewName to each tile on your window in order to easily determine which tile was clicked upon and thus what action your program needs to take.

Summary

A Windows Phone or Windows 8 tile is very easy to create in XAML. In this blog post you learned how just a few lines of XAML and some event wire-ups make short work of creating a list of Tile objects. In the sample that comes with this blog post a WrapPanel is used to allow the tiles to be moved around fairly easy. You could put a ScrollViewer control around the WrapPanel to allow the set of Tiles to grow in any direction you wish.

NOTE: You can download the sample code for this article by visiting my website at http://www.pdsa.com/downloads. Select “Tips & Tricks”, then select “Creating a XAML Tile Control” from the drop down list.