Pop, Snap, and Slide

In this lab, we will code in bursts. Get the starting source code and follow along. I will commit changes after every burst, so you can get caught up if you get lost.

## Gitting the Source Code

If you have a git client installed, then clone the repository. Create a working branch. For example:

cd c:\projects

git clone git://github.com/dallasxaml/PopSnapAndSlide.git

cd PopSnapAndSlide

git checkout –b take1

After each successful burst, commit your changes. If you ever get lost, commit that branch, go back to master, and create a new one:

git add –A

git commit –m "I missed that."

git checkout master

git pull

git checkout –b take2

## Downloading the Source Code

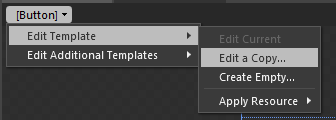
If you don’t have git installed, then go to the following URL and click on the “ZIP” button:

https://github.com/dallasxaml/PopSnapAndSlide

Unzip to your project folder. If you ever get lost, go back to the web page and download the zip again.

# Make buttons pop

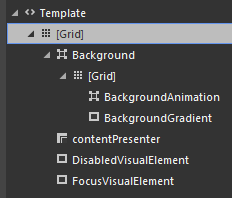
The visual state manager was born in Silverlight. Create a Silverlight project. No need to host the project in a new web site. Switch to Blend. Add a button to your main page. Then select the button, and use the template menu to edit a copy of the button template. Accept the default name (ButtonStyle1) and location (this document).



Now switch to the States tab and click through the visual states. To see the transition animation, click the transitions preview button.



There are several objects in the tree that are used just to provide visual cues.



These are always there. The visual states just change their color and opacity. Change the color of the BackgroundAnimation border and see how that affects the button’s visual behavior.

# Differences in WPF

Now do the same thing in WPF. What is different?

When you create a copy of the template in WPF, you can see that it’s not using the visual state manager. It’s using triggers. Change the background color in the MouseOver and Pressed states. The visual states override the triggers.

<Trigger Property="IsMouseOver" Value="true">

<Setter Property="Background" TargetName="border" Value="{StaticResource Button.MouseOver.Background}"/>

<Setter Property="BorderBrush" TargetName="border" Value="{StaticResource Button.MouseOver.Border}"/>

</Trigger>

Delete the triggers and then add a transition time to the states. Does it work as expected? The base style uses a brush resource for the background.

The WPF template does not have all of the objects that the Silverlight one had. So the visual states are more limited. If you want to add more objects, you could get buttons like Sliverlight. If you like that sort of thing.

# Navigation

The visual state manager can help the user navigate through their data. Put all of the controls on the page, and then use the visual state manager to control which ones are visible.

Drop a couple of buttons onto the WPF application. Then add a couple of rectangles. The rectangles represent views that you want to navigate between. Color one of them blue and the other red.

Overlay the two rectangles on the right, and stack the buttons on the left. Set their opacity to 0 and their visual transform x to 150. Then create a couple of visual states called RedState and BlueState. In each visual state, set the opacity to 100 and the transform x to 0.

Now wire up the buttons so that they switch to the selected visual state.

    private void Red\_Click(object sender, System.Windows.RoutedEventArgs e)

    {

        VisualStateManager.GoToElementState(LayoutRoot, "RedState", true);

    }

    private void Blue\_Click(object sender, System.Windows.RoutedEventArgs e)

    {

        VisualStateManager.GoToElementState(LayoutRoot, "BlueState", true);

    }