Creating your own Windows Phone UI Controls using Microsoft Expression Blend

This tutorial provides a step-by-step guide to creating custom UI controls using Expression Blend, using the example of a simple on-off switch.

Introduction

Aren't there times where you need to add a Control in your Form or Page while in designer mode? Yes, the first place to look is your Toolbox. But what if there isn't a Control that fits exactly to your needs? At those times you were relying on your code. Drawing on Canvas on some platforms or extending other Controls and overriding `onPaint` methods on other. A time consuming process including many application runs and debugging until achieving a satisfying result. Not to mention the effort needed to animate the Control.

In our case, Windows Phone, you will be surprised how few Controls are at your disposal. Controls that are often used or found in other applications that came preinstalled with the phone are not in your Toolbox (like the on/off switch of the Settings application). This article's title could more accurately be "Breaking the ice between the developer and Expression Blend". That's because, in this article, I'll try to vanquish the fear of someone else (even a tool) editing your code and the belief that everything is best done when done in code. Expression Blend is a very powerful tool and easy to use in order to quickly achieve very common but also time consuming jobs.

For this article you will need Visual Studio 2010 and Microsoft Expression Blend. We will create a custom Control to use as an on/off switch. So, let's get started.

Getting started

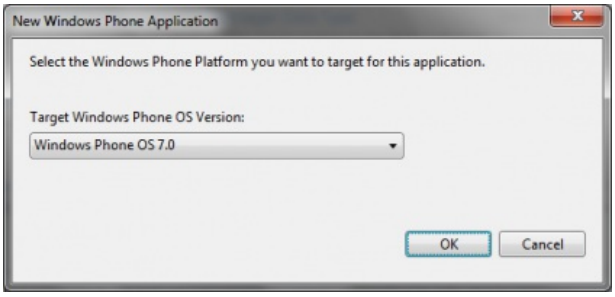
Let us first create a Windows Phone project.

Start Visual Studio then go to **File -> New -> Project...**

Then select Visual **C# -> Silverlight for Windows Phone -> Windows Phone Application**

At the bottom give the Name **SwitchProject** to your project.

If you have both Windows Phone SDK 7.0 and 7.1 installed you will be prompted to select one. Select **Windows Phone OS 7.0** as we won't need something that came with the 7.1 SDK.



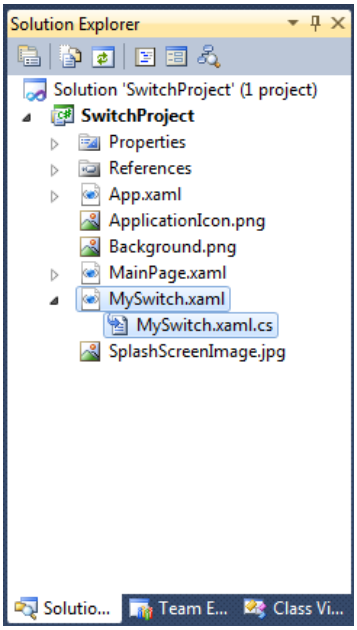
Now you should be able to see the initial project that the Visual Studio has created. Just for verification, you can see that the toolbox doesn't have the Switch control that you so desperately need! No worries, we will create one ourselves.

Right click your project's name on the **Solution Explorer** and select **Add -> New Item...**

Then, from the screen that shows up, select Visual **C# -> Silverlight for Windows Phone -> Windows Phone User Control**.

Let's call it **MySwitch.xaml**.

(C) Copyright Nokia 2013. All rights reserved.



As you can see, two new files have been added in your solution. The **MySwitch.xaml** and the code-behind file **MySwitch.xaml.cs**. On the XAML editing panel you should see the code below and a big 480x480 square in the designing panel waiting for us to draw on it.

```
<UserControl x:Class="SwitchProject.MySwitch"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    d:DesignHeight="480" d:DesignWidth="480">

    <Grid x:Name="LayoutRoot">
    </Grid>
</UserControl>
```

First steps in Expression Blend

Now start Expression Blend. Click **File -> Open Project/Solution...** and select our **SwitchProject.csproj** from the disk.

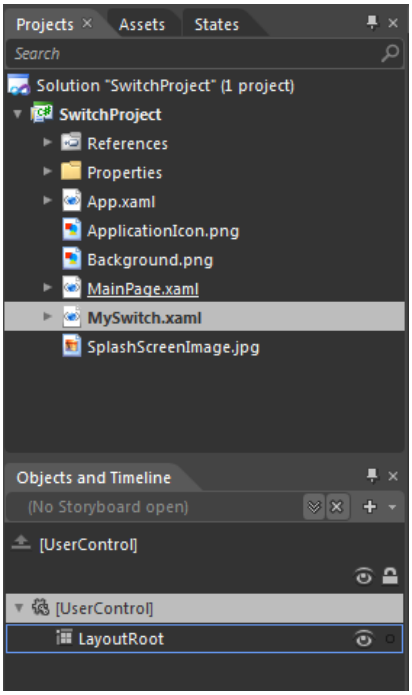
Wait... what? What a design tool has to do with our C# project? Well, Expression Blend is a tool for designing XAML interfaces for applications built with Silverlight and WPF. Silverlight and XAML is our case so it's completely normal for Expression Blend to be able to open it!

So, after opening the project you will notice our project appearing in the Solution Explorer just like we left it back in Visual Studio.

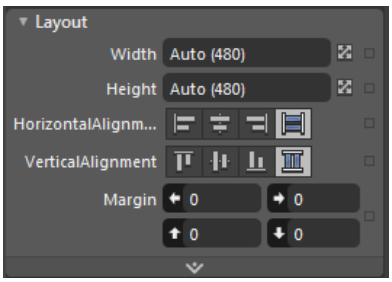
Double click the **MySwitch.xaml** file.

In the center of the screen, the designing panel, you can see the empty 480x480 rectangle that we saw in Visual Studio. On the left you can see a tab with the name **Objects and Timeline**. In this tab we can see the hierarchy of controls and shapes used in our Control. It is the tab that we will use for the rest of the article. For now, our control consists of only a Grid named **LayoutRoot** as we would expect from the xaml code above. The root node in the hierarchy tree represents our control as a whole and it is named `[UserControl]`. Click it.

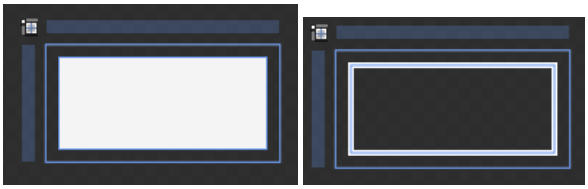
(C) Copyright Nokia 2013. All rights reserved.



We don't want our switch to be that big so we will change its size. To do so we could just drag the edges in the designing panel or we can use the **Layout** area that appears at the right of the screen under ther Properties tab. Let's just edit the width and height from there and set it to Width=160 and Height=80 (that's big enough for presentation reasons).



Now that we are satisfied with our Control's size let's start drawing in it. On the far left of the screen click the **Rectangle** button (or press **M** on your keyboard). Our mouse cursor will be transformed into a cross. Drag the mouse and create a rectangle like shown below. A rectangle was added in our hierarchy. Click it so that we can change some of its properties.



On the **Brushes** area on the right, click on **Fill** and select **No Brush** and then click on **Stroke** and select the white color (**#FFFFFFFF**).

On the **Appearance** area set the `StrokeThickness` property to **4**.

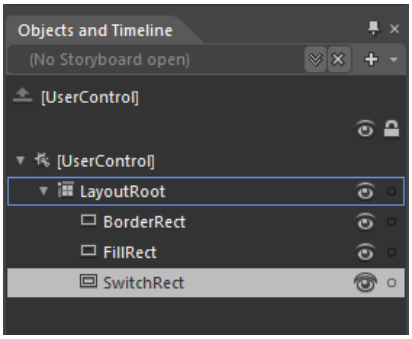
That's all we had to do to create a border for our switch. Give it the name **BorderRect** setting the `Name` property in the **Properties** tab.

Now let's add a bit of color so that we know when our switch is on or off. We will add another rectangle for that. From the **Layout** area set the rectangle's `Width`, `Height` and `Margin` properties to make it exactly fit into `BorderRect`. Also on the **Brushes** area, click on **Fill** and set it to red (`#FFFF0000`) and then click on **Stroke** and select `No Brush`. Finally set the `StrokeThickness` property to `0`. Let's name it **FillRect**.

One more rectancle and our switch is done. The last rectangle will represent the actual switch. So place it at the right and allow its size to exceed the `BorderRect`. I set it to Width=40 and Height=80. Set white (`#FFFFFFFF`) for **Fill**, `No Brush` for **Stroke** and set `StrokeThickness` to 0. Finally let's name it `SwitchRect`.

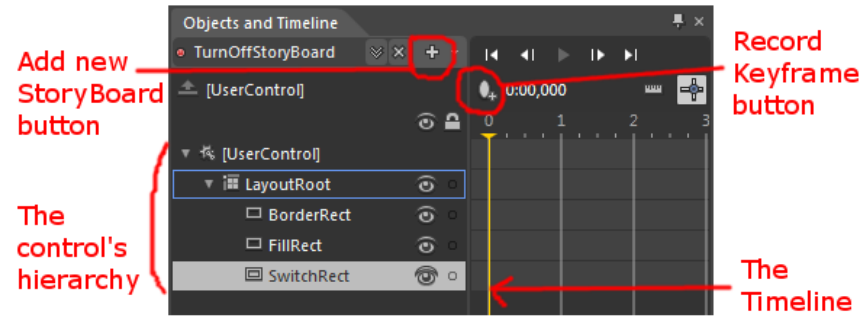
(C) Copyright Nokia 2013. All rights reserved.

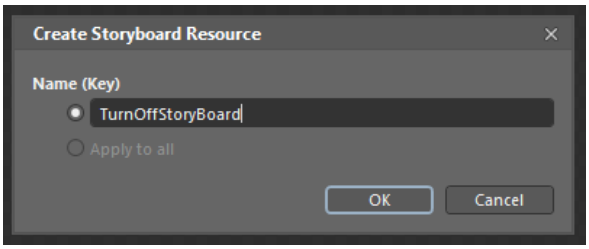
Save our work! The drawing is done and our hierarchy should look like below.



Turning on and off the switch with animation

The drawing is done but we need to give our switch some life by animating its on and off functions. We'll do this with **StoryBoards**.



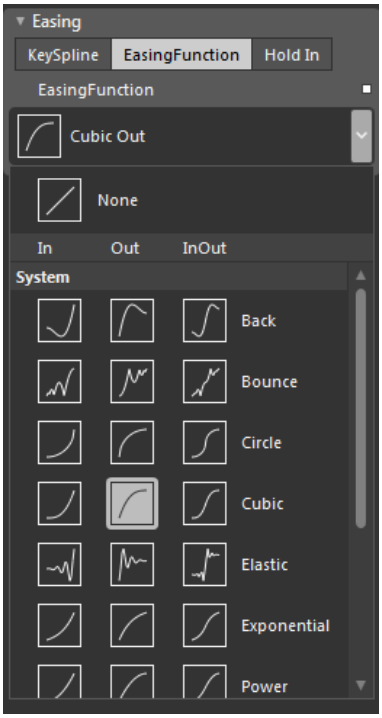


Click on the plus sign at the top right of the **Objects and Timeline** tab. A window will appear asking us to give the new StoryBoard a name. Type **TurnOffStoryBoard**. Now our **Objects and Timeline** tab should look like the above. On the right side is where we will set the states that the rectangles we drew before will have.

To turn off the switch we need to turn our `FillRect` to transparent and place our `SwitchRect` to the left. So follow these steps:

1. Click the ***FillRect***.
2. Set the **Timeline** at time **0** and click the **Record Keyframe** button.
3. On the **Appearance** area at the properties ensure the **Opacity** property is set at **100%**.
4. Now set the **Timeline** at **1**. That would be the duration of our storyboard. It's a bit slow for a switch to take one second to turn off but it's fine for our tutorial.
5. Click the **Record Keyframe** button again to set the state of our ***FillRect*** at that moment. Now change the **Opacity** property to **0%**.
6. Now we will define how the transition from one Keyframe to the other will happen. To do so click somewhere in the shapes representing the Keyframes and in its **Properties** tabs choose an easing function to use. Use the play button of the StoryBoard to actually see the animation we created. I think the **Cubic out** is fine.

(C) Copyright Nokia 2013. All rights reserved.



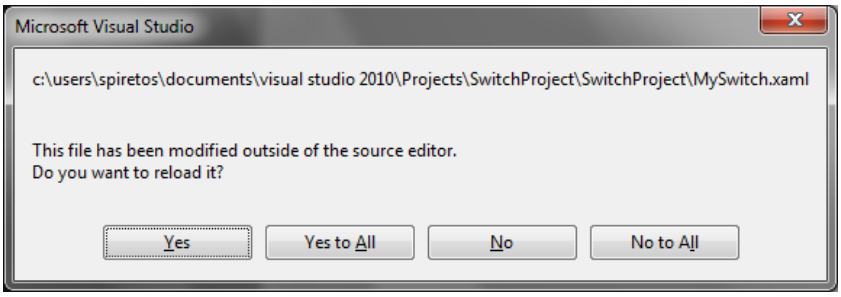
1. Time to animate the ***SwitchRect*** now. So click it, set the Timeline back to 0 and click the **Record Keyframe** button. Our ***SwitchRect*** is in position so let's move on.
2. Set the Timeline to 1 and press the **Record Keyframe** one last time.
3. Now we need to send the ***SwitchRect*** to the left side of the control. We won't do that by changing its position or margin or size as these properties cannot be changed in a StoryBoard. We will use the **Transform** area of the properties. So, select the **Translate** tab and set the property X=-120 (move back 120 pixels) and leave Y=0.
4. Finally, select the same easing curve for this animation.
5. You can now click the play button to see the animation in its full glory!

Now, to turn on the switch we need another StoryBoard named **TurnOnStoryboard**. The process of creating it is similar to the **TurOffStoryboard** and is left as an exercise to the reader. If you find it dificult you can always download the project's source code and play with it in Expression Blend.

So far so good. We drew our switch and we made it move. Now save everything and let's go back to Visual Studio to use it on our application!

Using our control in code

When switching back to Visual Studio you will probably see a message informing you that someone has edited your **MySwitch.xaml** file without using the Visual Studio. That's right, Expression Blend did. Click "Yes" to reload it.



As you can see a new element in our xaml has been added named **UserControl.Resources** and our **LayoutRoot** Grid now contains our three rectangles. Finally you can see the shape of our switch on the designer panel.

Now we need our control to handle the user interaction. For that reason we will add an event listener to our Grid to get user touches. We will register to the Grid's `MouseLeftButtonDown` event and our Grid should look like below in xaml.

```
<Grid x:Name="LayoutRoot"
          MouseLeftButtonDown="LayoutRoot_MouseLeftButtonDown" >
     <Rectangle x:Name="BorderRect" Margin="8" Stroke="White" StrokeThickness="4"/>
     <Rectangle x:Name="FillRect" Margin="12" StrokeThickness="0" Fill="Red"/>
     <Rectangle x:Name="SwitchRect" Fill="White" HorizontalAlignment="Right" StrokeThickness="0" Width="40" Height="80" RenderTransf
      <Rectangle.RenderTransform>
       <CompositeTransform/>
      </Rectangle.RenderTransform>
     </Rectangle>
</Grid>
```

A function named **LayoutRoot_MouseLeftButtonDown** was created in our code-behind **MySwitch.xaml.cs** file if we added the event listener from the xaml, looking like the following

```
private void LayoutRoot_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
```


(C) Copyright Nokia 2013. All rights reserved.

```
{

}
```

Or else you can paste that function to your code and add the below line of code in your `MySwitch` constructor under the `InitializeComponent()` call.

```
LayoutRoot.MouseLeftButtonDown+=new MouseButtonEventHandler(LayoutRoot_MouseLeftButtonDown);
```

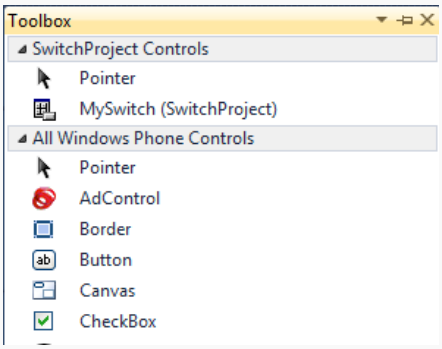
We also need to create a boolean property for our control in order to hold its state. Let's call it `SwitchIsON`, initialize it to `true` in our constructor and set its body to the following:

```
public bool SwitchIsON
{
     get;
     private set;
}
```

We finally need to change this property every time the user touches our switch and use the StoryBoards we created in Expression Blend to animate the appropriate action. Also we need to fire an event so that others know when our control's state is changed. So our code should look like the following:

```
public partial class MySwitch : UserControl
{
    public bool SwitchIsON
    {
        get;
        private set;
    }

    public event EventHandler StateChanged;

    public MySwitch()
    {
        InitializeComponent();
        SwitchIsON = true;
    }

    private void LayoutRoot_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
    {
        if (SwitchIsON)
            TurnOffStoryBoard.Begin();
        else
            TurnOnStoryBoard.Begin();
        SwitchIsON = !SwitchIsON;

        if (StateChanged != null)
            StateChanged(this, null);
    }
}
```

Before we run our application only one thing remains. To add our brand new switch to the MainPage of our project. If you check your Toolbox again you will see our `MySwitch` control in it! So drag it in the Page and run our application.



Our control can now be found in the Toolbox

Our application's first run

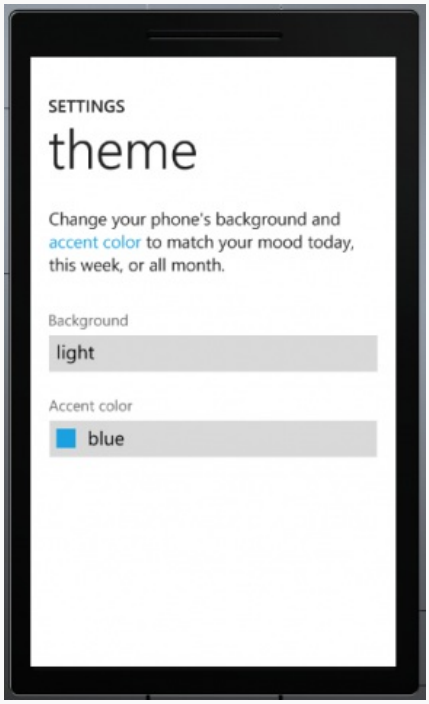
That's great isn't it? We can actually use our own Control that was created to fit exactly our needs! So, are we done? We drew our control and animated it to fit our needs. We let it handle the user interaction by its own and we created an event to let our code know when its state is changed.

Well... let us just check one last thing.

(C) Copyright Nokia 2013. All rights reserved.

Being user theme friendly

What if the user of the application changes the theme preferences of the phone? Let's say to "light" and "blue". Let's see what will happen to our application?



The user changes the phone's theme

Our switch is gone!

That's right... disaster. Where is our switch? We did so much to make it work with one theme only? Oh no, should I fill my code with if-then-else and switch-case statements to achieve my goal? Calm down! We will fix our control in a matter of seconds.

Go back to Expression Blend and open our project again. We have to change the colors we used to match the user's theme. To do so we will replace the white color we used in our rectangles with the **PhoneContrastBackgroundBrush** and the red color with the **PhoneAccentBrush**. You will find those brushes under the **Brush Resources** tab of the **Brushes** area. Save everything and lets run our application again.

We did it!

Yes, now we are done!

Summary

(C) Copyright Nokia 2013. All rights reserved.

I hope this tutorial made you take the first step into using Expression Blend. It is a very powerful tool and we used only a very small portion of its full potential. Though that portion is enough to allow us create our own UI Controls that fit our specific needs and not only that. Create user interfaces that add value to the application, contribute to the total User Experience and make our code from just functional to a real application.

Related links

- A project using a version of the switch, four other custom UI Controls and many other StoryBoards: SheepClock project
- Everything you need to know about Expression Blend, with tons of information in videos and tutorials: Learn Expression Blend

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.