User interface design

for Window Phone 8, using C#

# Lab 1 – Creating a basic application

## Functional Goals

Create a functional Windows Phone 8 app that contains the necessary controls to display bandwidth usage data from Rose-Hulman’s network usage tool.

## Learning Goals

* Understand Windows Phone UI paradigms, including the Panorama control and the use of “Metro” design to create a straightforward, attractive Windows Phone application
* Understand the basics of the User Control extension paradigm of C# and Windows Phone

## Prerequisites

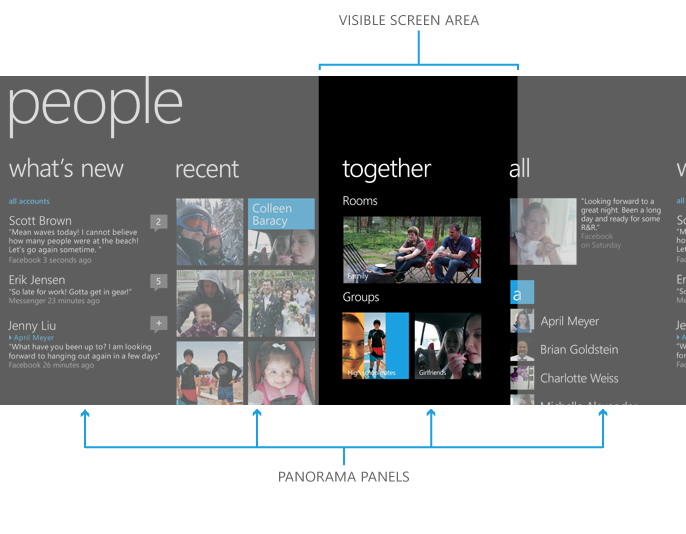
* You’ll need to install Visual Studio (2012 Ultimate was used to create this lab) from the MSDN/DreamSpark service on ANGEL’s RosePortal.
* You’ll also need to download and install the Windows Phone SDK (8.0 was used to create this lab) from https://dev.windowsphone.com/en-us/downloadsdk
  + To properly run the Windows Phone emulator, you’ll need to ensure that **second-level address translation (SLAT)** and **hardware Data Execution Prevention** (Execute Disable on Intel systems; No Execute on AMD) are enabled in your BIOS settings.
* A basic understanding of Visual Studio and C# development, such as that gained from the **User interface design in C#, using WPF** series in this document’s repository.

## Submission Instructions

Submit answers to the **3** (or **4**, with extra credit)questions in this lab as a .pdf to the appropriate Moodle submission form.

## Get started: Create a project

Open Visual Studio and create a new Windows Phone Panorama App.

The Windows Phone SDK comes with some built-in controls that can help you make a native-looking app very quickly; the one you’ll use for this lab is called the Panorama control. Look at the image below and notice how the app is really just one big horizontal page that the user can swipe to see additional content.

The app you just created in Visual Studio contains one of these controls – we just need to add content to it.

## Make it yours: Customizing a Panorama

1. In the <phone:Panorama opening tag, change the Title to “rhit bandwidth monitor” (minus the quotes). Note that WP panoramas are generally given *all lower-case* titles.
2. Find a picture on the Internet that evokes Rose-Hulman and looks good with white text on top (i.e. a darker image). In your favorite photo editor, crop it down to 2160 x 1280 and place it in the Assets folder of your project as BackgroundDark.{png,jpg}.
3. In Visual Studio, add the picture to the Assets folder using the Add Existing Item right-click menu item.
4. Change the <ImageBrush of the <phone:Panorama.Background> tag to point to BackgroundDark using the Properties panel, or by typing in the correct relative path.

Run the project – I recommend using the 720P emulator as this is both the largest and (arguably) the most popular Windows Phone resolution, and so it will give you the most accurate representation of what your interface will look like on phones like the Nokia Lumia 920 and the HTC 8X. Notice that without much work, you’ve created the skeleton of an app that looks like it belongs on Windows Phone, but has its own unique personality and flair.

## Creating a reusable control: User Controls

By the time this app is finished, you’ll have 4 different bandwidth meters: upload and download for both actual and policy usage. You could just design one and then copy-paste your XAML, but that’s really not the most extensible or maintainable solution; so, don’t do that. Instead, create a custom User Control you can reuse.

1. Add a new Windows Phone User Control (call it BandwidthMeter) to the project using the right-click menu.
2. In the <UserControl definition, add DataContext="{Binding RelativeSource={RelativeSource Self}}" and change the d:DesignWidth to 150.
3. Add the following code to your UserControl to get started.

<Border x:Name="LayoutRoot">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height=".5\*"/>

<RowDefinition Height=".5\*"/>

<RowDefinition Height="4\*"/>

</Grid.RowDefinitions>

<Border Grid.Row="0" Background="#88B22222">

<TextBlock HorizontalAlignment="Center" VerticalAlignment="Top" Text="{Binding RedTextBlock}"/>

</Border>

<Border Grid.Row="1" Background="#88FFA500">

<TextBlock HorizontalAlignment="Center" VerticalAlignment="Top" Text="{Binding YellowTextBlock}"/>

</Border>

<Border Grid.Row="2" Background="#8832CD32">

<TextBlock HorizontalAlignment="Center" VerticalAlignment="Top" Text="{Binding GreenTextBlock}"/>

</Border>

<Border Name="UsageBorder" Grid.RowSpan="3" Background="#88B50938" VerticalAlignment="Bottom" Height="0">

<TextBlock Name="UsageTextBlock" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Border>

</Grid>

</Border>

1. Review the code from the previous step and make sure you understand what’s going on. In particular, note that we started the control with a Border instead of a Grid. Don’t worry about the Text bindings (which should be throwing an error at the moment); we’ll get to them.
2. Add two attributes to the Border (using the Properties panel or IntelliSense) to set its brush (color) to {StaticResource PhoneForegroundBrush} and its thickness to 5.
3. Below the last inner Border, add the following code.

<Border Name="UsageBorder" Grid.RowSpan="3" Background="#FFB50938" VerticalAlignment="Bottom" Height="0">

<TextBlock Name="UsageTextBlock" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Border>

1. Notice that you don’t actually see the control you just created, because its height is set to 0.

Question 1: C# uses a somewhat atypical color code schema that looks like hex, but has two extra characters. What is this schema called, and what do the two extra characters control? (Feel free to use MSDN) (6 points)

## Making a reusable control useful: Adding DependencyProperties

Right now, you have a complete UserControl. But, you can’t customize it at all. If you were to add some to MainPage, you’d have a bunch of meters that all showed the same thing. So, let’s change that with some DependencyProperties.

1. In the codebehind, add three DependencyProperties using snippets.
   1. Start typing ‘dependency’ and press Tab. You should get something like:

public static readonly DependencyProperty propertyNameProperty =

DependencyProperty.Register("propertyName", typeof (propertyType), typeof (BandwidthMeter), new PropertyMetadata(default(propertyType)));

public propertyType propertyName

{

get { return (propertyType) GetValue(propertyNameProperty); }

set { SetValue(propertyNameProperty, value); }

}

* 1. Tab through and add a RedTextBlockProperty, and YellowTextBlockProperty, and a GreenTextBlockProperty. Each should be a String. Don’t change the second typeof or the PropertyMetadata.

1. Switch back to the XAML. If you did the previous step correctly, all three Bindings should now resolve properly.

## Using a UserControl

You now have a basic UserControl – congratulations! Now you’re going to use that control back on MainPage.

1. Open MainPage.xaml.
2. Replace all of the code inside <phone:Panorama with the following code:

<phone:PanoramaItem Header="policy usage">

<Grid Name="PolicyUsageGrid" Margin="12,0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width=".05\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Download" HorizontalAlignment="Center"

Style="{StaticResource PhoneTextTitle3Style}"/>

<local:BandwidthMeter Grid.Row="1" Name="PolicyDown" Grid.Column="0" RedTextBlock="5 GB"

YellowTextBlock="4.5 GB" GreenTextBlock="4 GB"/>

<TextBlock Grid.Column="2" Text="Upload" HorizontalAlignment="Center"

Style="{StaticResource PhoneTextTitle3Style}"/>

<local:BandwidthMeter Grid.Row="1" Name="PolicyUp" Grid.Column="2" RedTextBlock="5 GB"

YellowTextBlock="4.5 GB" GreenTextBlock="4 GB"/>

</Grid>

</phone:PanoramaItem>

1. In your PhoneApplicationPage, add the following: xmlns:local="clr-namespace:WP8RHITBandwidth" (change WP8RHITBandwidth to the namespace of your BandwidthMeter).

Question 2: What is an xmlns, and why was it necessary to add one in the previous step? (6 points)

Question 3: Add another PanoramaItem for actual usage. Make sure to change the title and the control properties (e.g. name, TextBlock contents, etc.) where necessary. For actual usage, remember that actual can be up to 4 times the amount of policy usage; make sure your controls reflect that. Submit the XAML code for the item. If you are using a word processor that doesn’t retain Visual Studio’s text formatting on copy/paste, please take a screenshot of your code so that it remains properly formatted and colored. (12 points)

## One more thing: Fill out the panorama

To get an idea of what the panorama control really looks like, add the following code after your other PanoramaItems.

<phone:PanoramaItem Header="campus graph">

<Image Source="http://web.rose-hulman.edu/administration/helpdesk/siteimages/Internet\_Link\_Utilization\_1Day.png"

HorizontalAlignment="Center" VerticalAlignment="Center"/>

</phone:PanoramaItem>

## Congratulations

You’re done; run your app and check it out! It doesn’t do anything just yet – you’ll take care of that in the next lab. Don’t forget: submit answers to the **3** (or **4**, with extra credit)questions in this lab as a .pdf to the appropriate Moodle submission form.

## Above and beyond: Catering to both Windows Phone background themes

From here on is extra credit. It is possible to earn full credit on the lab without doing this section.

In the Windows Phone emulator, head into Settings (swipe/drag left from the main screen to get to the app list) and choose the Theme option. Change to the Light theme.

Relaunch the app from Visual Studio (you may need to stop it first), or by locating it in the app list. (Don’t use the back button to return to the app – changing themes while an app is running can cause some interesting issues.) Notice that, depending on the background image you picked, your text might be a little hard to read with the theme’s color change.

You can fix that by making your background image theme-dependent. Search on Bing or Google – several top results will show you how to do this. Don’t worry about making it work in design-time unless you really want to; just make it work when you actually run the application.

*Note: Make sure to remove d:DataContext="{d:DesignData SampleData/MainViewModelSampleData.xaml}" from the d:DataContext="{d:DesignData SampleData/MainViewModelSampleData.xaml}"; replace it with the RelativeSource context you used in BandwidthMeter. Also remove DataContext = App.ViewModel; from the constructor.*

Question 4: Submit the codebehind code that enables your theme-dependent background image, and two screenshots (the emulator will take screenshots for you – use the double-chevron on the emulator control bar) that showcase your changes. If you are using a word processor that doesn’t retain Visual Studio’s text formatting on copy/paste, please take a screenshot of your code so that it remains properly formatted and colored. (12 points)