User interface design

for Window Phone 8, using C#

# Lab 3 – Making an app fluid and beautiful

## Functional Goals

Take a functional Windows Phone 8 app that displays bandwidth usage data from Rose-Hulman’s network usage tool and add animations, live tiles, and loading indicators where appropriate.

## Learning Goals

* Understand Windows Phone UI paradigms, including built-in animations and Live Tiles

## 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) 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.
* Code from Lab2 (a complete version may be available from your instructor)

## 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.

## Be impatient: Add a loading indicator to the scraper

Right now, your bandwidth app doesn’t do anything to show the user that it’s loading, which is less than ideal. Sometimes connecting to and scraping the Rose-Hulman network usage tool takes upwards of 5 seconds. In those times, many of your users will leave the app assuming it’s broken unless you let them know it’s not. To do this:

1. In MainPage.xaml, in the <phone:PhoneApplicationPage tag, make sure that the system tray is visible: shell:SystemTray.IsVisible="True".
2. In MainPage.xaml.cs, right before new Thread(Scraper.Scrape).Start(this);, create a new progress indicator:

if (fromNetwork)

SystemTray.ProgressIndicator.IsVisible = false;

var indicator = new ProgressIndicator

{

IsVisible = true,

Text = "Updating usage...",

IsIndeterminate = true

};

SystemTray.ProgressIndicator = indicator;

1. At the end of the UpdateUi method, stop the progress indicator:

Question 1: With regards to ProgressIndicator, what does indeterminate mean? Why is this indicator indeterminate? (12 points)

## Be fluid: Animate your BandwidthMeter

Right now, your bandwidth meters “jump” after data is received, which is visually jarring for the user. To give a better experience, add an animation:

1. In BandwidthMeter.xaml, add a new storyboard between the <UserControl tag and the <Border x:Name="LayoutRoot" tag:

public void UpdateBorder(double value, double gridHeight)

{

var sb = (Storyboard)Resources["ShowUsageStoryboard"];

var to = value / 5000 \* gridHeight;

((DoubleAnimation)sb.Children[0]).To = to > 40 ? to : 40;

sb.Begin();

}

<UserControl.Resources>

<Storyboard x:Name="ShowUsageStoryboard">

<DoubleAnimation Storyboard.TargetName="UsageBorder"

Storyboard.TargetProperty="Height"

Duration="0:0:2">

</DoubleAnimation>

</Storyboard>

</UserControl.Resources>

1. Update the UpdateBorder method to trigger the storyboard:

Run your app to see how it looks with some added fluidity.

## Expose what matters: Make your app a Live Tile app

One of the features that makes Windows Phone unique is Live Tiles. The core philosophy of those tiles is to bring what matters – people, content, and information – to the foreground so the user doesn’t have to waste time jumping in and out of apps. So, with that in mind, add a live tile to your app:

1. In MainPage.xaml.cs, between the update code and the progress bar stop code, add the following code:

private static String GetBandwidthStringForTile(BandwidthResults results)

{

var received = Convert.ToInt32(GetBandwidthNumberFromString(results.PolicyReceived)) + " MB";

var sent = Convert.ToInt32(GetBandwidthNumberFromString(results.PolicySent)) + " MB";

return results.BandwidthClass + "\r\nD: " + received + "\r\nU: " + sent;

}

var tileData = new FlipTileData()

{

BackContent = GetBandwidthStringForTile(bandwidthResults),

Title = "Bandwidth Monitor"

};

var primaryTile = ShellTile.ActiveTiles.First();

primaryTile.Update(tileData);

1. Add the following helper method:
2. In the Assets folder, edit the FlipCycleTileMedium, FlipCycleTileSmall, and ApplicationIcon images to reflect the app’s function and differentiate it on the apps list and the home screen.

If you use an image editor other than Visual Studio, be aware that there are some strange issues around low bit depth PNGs on Windows Phone. If your custom images don’t appear, try increasing to a 32-bit PNG.

Protip

1. In the Properties folder, open WMAppManifest.xml.
   1. Ensure that the Tile Template is TemplateFlip, and Support for large Tiles is not checked.

Question 2: Windows Phone has 3 live tile “templates” (including the one you just used). List them. (9 points)

Question 3: How many tile sizes are there? (3 points)

## Congratulations

You’re done; run your app and check it out! 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: Enhancing your animations

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

Right now, your bandwidth meter animation is perfectly ordinary; it looks nice, but it doesn’t “pop”. Windows Phone (and WPF) have some easing functions that can supply that pop.

Question 4: Add an EasingFunction to your bandwidth meter. Submit the XAML code. 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. (6 points)