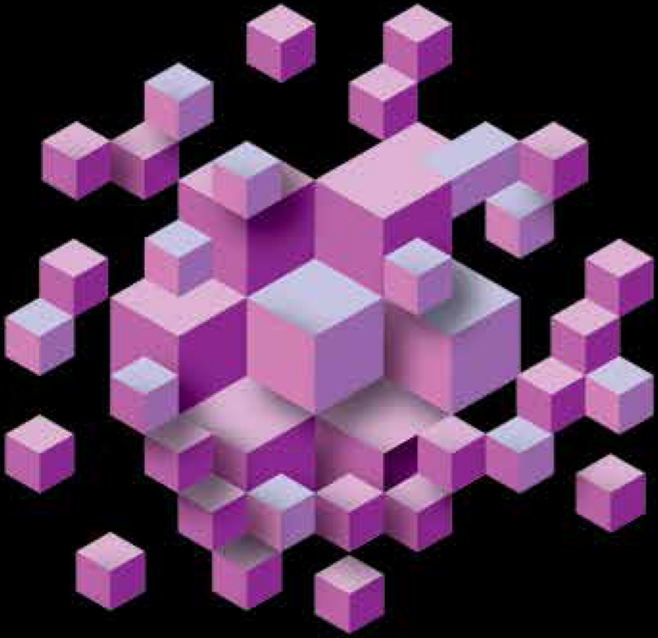
Master the art of creating authentic mobile applications on Microsoft's newest platform, Windows Phone 8



Pro Windows Phone App Development

THIRD EDITION

Falafel Software

Apress®

Contents at a Glance

About the Author xvii

About the Technical Reviewer xix

■Chapter 1: Introduction 1

■Chapter 2: Getting Started 7

■Chapter 3: Building the UI with XAML 35

■Chapter 4: Phone Controls 95

■Chapter 5: Navigation 179

■Chapter 6: Application Lifecycle 217

■Chapter 7: Gestures 233

■Chapter 8: Device Capabilities 263

■Chapter 9: Mapping 311

■Chapter 10: Live Tiles and Notifications 343

■Chapter 11: Data Sources 385

■Chapter 12: Using Data in the Cloud 421

■Chapter 13: Designing In Blend 449

■Chapter 14: Taking Your App to Market 517

Index 529

CHAPTER 1

Introduction

This chapter describes the groundwork you need to perform before writing Windows Phone 8 applications including:

- The particular skill sets in Windows 8, Visual Studio, and Windows Phone 8 that you need to have to get the most from this book.
- PC and phone hardware requirements to support the development environment.
- Software requirements for the operating system and the Visual Studio development environment.

Why Upgrade to Windows Phone 8?

Microsoft's intent is to lead and not follow in the race to create a superior phone operating system. Windows Phone 8 takes advantages of all the latest hardware features, such as Near Field Communications (NFC), while Windows Phone 7.x applications will still run "out of the box." The clean user interface favors content over "chrome" and is easy to learn. From the developer's perspective, Windows Phone 8 is simply a lot of fun to develop on and is quite productive. Having worked with the Android platform and having a nodding aquintanceship with iOS, Windows Phone 8 is clearly my platform of choice.

What You Need to Know

You should be able to navigate the Windows 8 environment, at least to the point where you can get to the Start screen, search for applications on the PC, and run applications.

You should be familiar with the Visual Studio environment, be able to create projects, add project items, and assembly references. You should know how to code in either Visual Basic .NET or C#. The code for this book is presented in C#. If you already use XAML, you will have a good head start learning Windows Phone 8 development.

In particular, this book will not specifically include `using` statements in code listings except for unusual circumstances. When Visual Studio encounters a member that it cannot resolve, the editor underlines the member with a wavy red line. You should right-click the member and choose the `Resolve` context menu option to add the `using` statement automatically. In cases where the namespace includes extensions, such as `System.Linq`, Visual Studio will not offer the Resolve context menu. In these cases, the instructions or code listings will include the specific `using` statements.

Spend some time learning the features and operation of Windows Phone 8 devices, including unlocking the screen, navigating from the Start screen, and "pinning" tiles. The link below should get you started:

`www.windowsphone.com/en-us/how-to/wp8/start/get-started-with-windows-phone-8`

What You Need to Have

To get started with Windows Phone 8 development, you will need some specific hardware and software. In this section, we'll cover the requirements for:

- PC Hardware requirements
- Phone Hardware Requirements
- Operating system requirements
- The Windows Phone 8 SDK
- Visual Studio

PC Hardware Requirements

Your PC must have *Hyper V* and *SLAT* support. Hyper V is required to run virtual machines, namely the Windows Phone 8 emulator. SLAT (Second Level Address Translation) is used by Hyper V to translate virtualized guest addresses to real physical addresses. The bottom line is that you need to enable virtualization in your PC BIOS settings for Hyper V and SLAT. The specific settings will vary according to your PC's BIOS setup. Older machines may not have these settings available.

How do you know if you're good to go? There are a number of software utilities that assess hardware and firmware. Microsoft offers a command line tool, Coreinfo, that lists virtualization capabilities. Find Coreinfo for download at Technet:

`http://technet.microsoft.com/en-us/sysinternals/cc835722.aspx`

Run CoreInfo from the developer command line as an administrator:

1. Navigate to the Windows 8 Start screen.
2. Type "Command" to search on and select the available command line applications.
3. Locate the "Developer Command Prompt for VS2012" application in the results list, right-click, and select Run as administrator from the app bar.
4. Change your directory (`cd`) to the folder that contains `Coreinfo.exe`.
5. Run the command `coreinfo -v`.

The command will list information relating to virtualization capabilities. The output below shows a sample run of Coreinfo on a laptop that is ready for Windows Phone 8 development. The hyphen in the HYPERVISOR line indicates that the Hypervisor is present but not enabled. Asterisks in the VMX and EPT lines indicate that hardware-assisted virtualization and SLAT support are supported.

```
C:\WINDOWS\system32>cd C:\Download\Coreinfo

C:\Download\Coreinfo>coreinfo -v

Coreinfo v3.2 - Dump information on system CPU and memory topology
Copyright (C) 2008-2012 Mark Russinovich
Sysinternals - www.sysinternals.com
```

```
Intel(R) Core(TM) i7 CPU       Q 720  @ 1.60GHz
Intel64 Family 6 Model 30 Stepping 5, GenuineIntel
HYPERVISOR      -       Hypervisor is present
VMX             *       Supports Intel hardware-assisted virtualization
EPT             *       Supports Intel extended page tables (SLAT)
```

The *Microsoft Assessment and Planning Toolkit* (http://technet.microsoft.com/en-us/library/bb977556.aspx) verifies that your machine supports Hyper V. The Microsoft Assessment and Planning Toolkit may be overkill compared with CoreInfo. This toolkit is a complete planning utility that checks for capabilities over cloud, desktop, server, and more. In Figure 1-1, the Microsoft Assessment and Planning tool indicates support for Hyper V support and Windows 8 readiness.

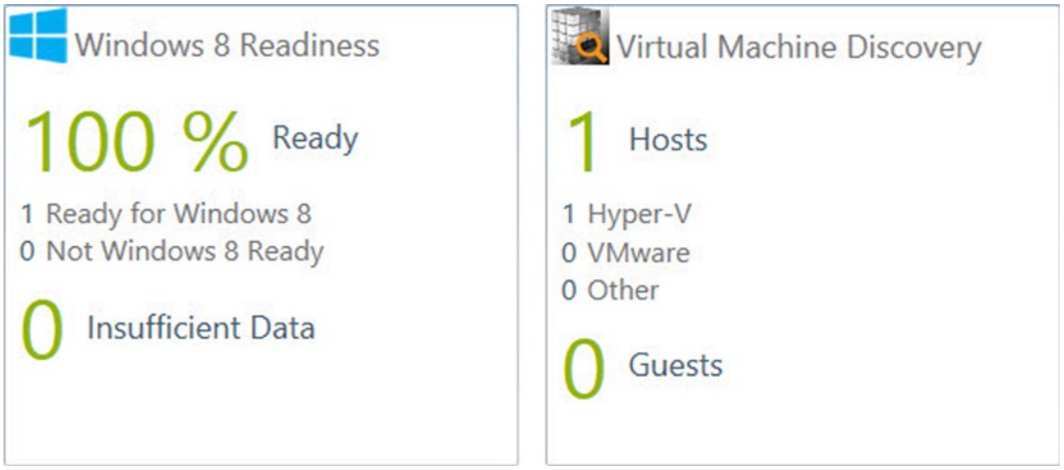


Figure 1-1. *Microsoft Assessment and Planning Toolkit Output*

Phone Hardware Requirements

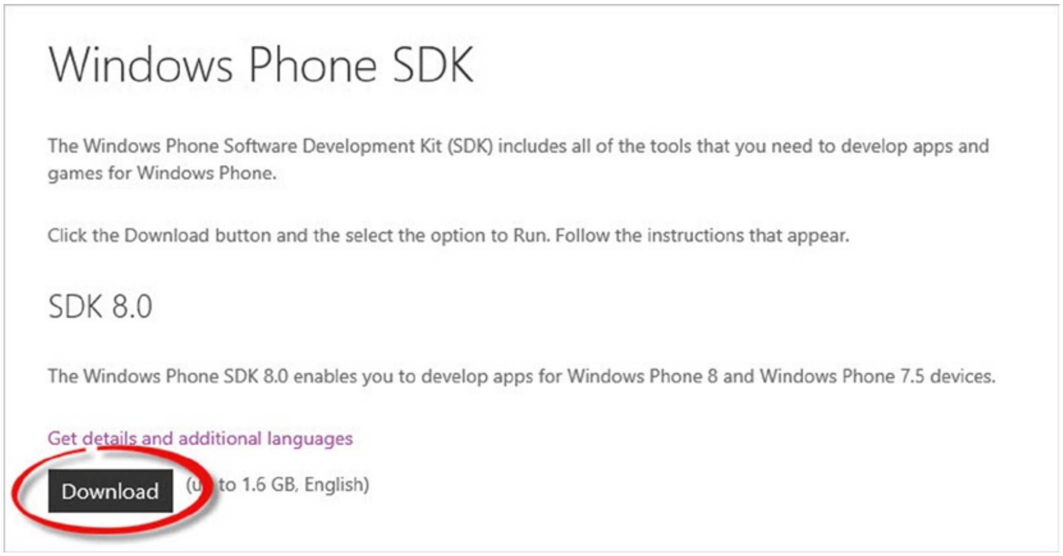
Windows Phone 8 development does not actually require a phone. You can get by using the Emulator application until you require specific device capabilities operating in a physical environment (e.g., the Accelerometer or geographic location). While the Emulator is convenient for initial testing, you should run your application against one or more phone devices for final testing. When choosing a phone, be sure that it runs the Windows Phone 8 operating system, such as the Nokia 92x, HTC Windows Phone 8x, or Samsung ATIV S. *Legacy phones intended for Windows Phone 7.x are not upgradable to Windows Phone 8*. For more information on using a Windows Phone 8 device, see the section “Creating, Configuring, and Testing an Application” in Chapter 2.

Operating System Requirements

Windows Phone 8 development requires the 64-bit version of *Windows 8 Pro or greater*. You can still install and run using only Windows 8, but the emulator will not work.

The Windows Phone 8 SDK

The Windows Phone 8 SDK allows you to create Windows Phone 8 applications in Visual Studio and to test your applications on your PC using an emulator. The most important download is located at the *Windows Phone Dev Center* (`https://dev.windowsphone.com`). Here you'll find a link to the SDK 8 download, as shown in Figure 1-2, and a full list of requirements for install (`www.microsoft.com/download/details.aspx?id=35471`).



Windows Phone SDK

The Windows Phone Software Development Kit (SDK) includes all of the tools that you need to develop apps and games for Windows Phone.

Click the Download button and the select the option to Run. Follow the instructions that appear.

SDK 8.0

The Windows Phone SDK 8.0 enables you to develop apps for Windows Phone 8 and Windows Phone 7.5 devices.

Get details and additional languages

(u to 1.6 GB, English)

Figure 1-2. *The Windows Phone 8 SDK Download Link*

The download consists of `Wpexpress_full.exe` that installs to the `\Program Files (x86)\Windows Phone Kits\8.0` directory by default. Here are some of the goodies that are installed:

- SDK Windows Phone 8 Assemblies
- Windows Phone 8 Emulator
- Visual Studio Express 2012 for Windows Phone
- Adds project and item templates to Visual Studio 2012
- Development Kit Tools for Windows Store Apps
- MS C++ 2012 Compilers and Core Libraries for Windows Phone 8
- Direct X Libraries
- XNA Game Studio 40
- Windows Phone 7.1 Support
- Blend SDK for Windows Phone 8 and 7.1
- Microsoft Advertising SDK

■ **Note** “If you install the Windows Phone SDK 8.0 on a computer with a Gigabyte motherboard and the USB 3.0 host controller enabled in the BIOS, the computer might stop responding when it boots (for details on impacted Gigabyte motherboard models, see Installing Hyper-V on Gigabyte systems). To resolve this, disable USB 3.0 support in computer’s BIOS.” Source: Windows Phone SDK 8 Install Notes (`www.microsoft.com/en-us/download/details.aspx?id=35471`).

Visual Studio

A stand-alone version of *Visual Studio Express 2012 for Windows Phone* is included as part of the Windows Phone 8 SDK install. If you already have Visual Studio Professional, Premium, or Ultimate installed, the SDK will install as an add-in to your existing version. Be aware that the Express version of Visual Studio 2012 will be missing some features you may be expecting—that do appear in the non-express versions—such as the ability to switch between Windows Phone and Web development within the same IDE, the ability to use Add-ins (such as JetBrains’ ReSharper), and some architectural tools.

The screenshots in this book are taken from Visual Studio 2012 Professional.

Summary

The list of prerequisites may look daunting. In fact, if your hardware is too old, you may need to go shopping for new hardware to satisfy the requirements. Try scanning this chapter to get an idea of the overall process, and then work through each item in order. Once you are able to open Visual Studio 2012, open a new project and see Windows Phone 8 templates, you’re good to go!

CHAPTER 2

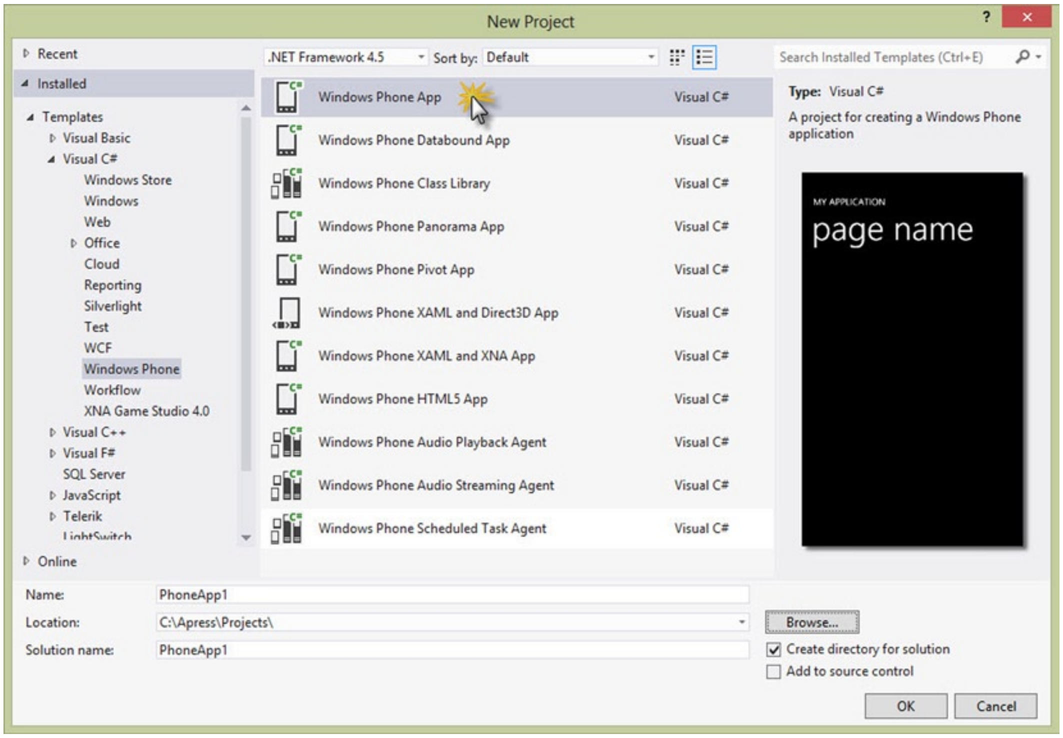
Getting Started

This chapter focuses on learning your way around a Windows Phone 8 application project, including:

- How to create a basic "hello world" Windows Phone 8 project.
- How to navigate the Visual Studio solution, particularly the location and purpose of key files.
- How to run the application in Visual Studio.
- Running your application in the emulator.
- Running your application on your phone device.

Creating a Windows Phone 8 Project

To create Windows Phone 8 projects, select `File` ➤ `New` ➤ `Project` from the Visual Studio menu and choose one of the `Windows Phone` templates (see Figure 2-1). Windows Phone templates can be found under both the Visual Basic and Visual C# nodes; this book will be using C# as the default language. The `Windows Phone App` template creates a "Hello World" type application by default. Click `OK` to have Visual Studio begin creating the new Windows Phone App.


New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows Store
Windows
Web
Office
Cloud
Reporting
Silverlight
Test
WCF
Windows Phone
Workflow
XNA Game Studio 4.0
Visual C++
Visual F#
SQL Server
JavaScript
Telerik
LightSwitch
Online
Sort by:
Search Installed Templates (Ctrl+E)
Windows Phone App Visual C#
Windows Phone Databound App Visual C#
Windows Phone Class Library Visual C#
Windows Phone Panorama App Visual C#
Windows Phone Pivot App Visual C#
Windows Phone XAML and Direct3D App Visual C#
Windows Phone XAML and XNA App Visual C#
Windows Phone HTML5 App Visual C#
Windows Phone Audio Playback Agent Visual C#
Windows Phone Audio Streaming Agent Visual C#
Windows Phone Scheduled Task Agent Visual C#
MY APPLICATION
page name
Browse...
OK
Cancel

.NET Framework 4.5 · Sort by: Default

Type: Visual C#

A project for creating a Windows Phone application

Name: PhoneApp1

Location: C:\Apress\Projects\

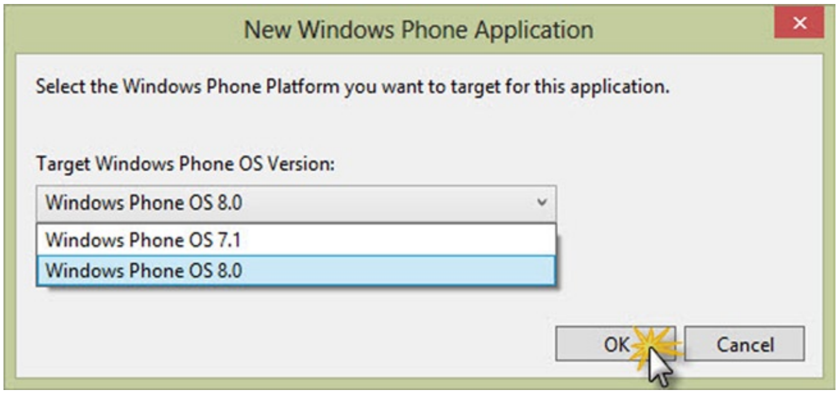
Solution name: PhoneApp1

[✓] Create directory for solution

[ ] Add to source control

Figure 2-1. *Using the Windows Phone App Project Template*

Next, choose the Windows Phone OS version (see Figure 2-2). For the purposes of this book we will choose the `Windows Phone OS 8.0` default exclusively.



Select the Windows Phone Platform you want to target for this application.

Target Windows Phone OS Version: Windows Phone OS 8.0

Figure 2-2. *Choosing the OS Version*

You can run the application by pressing F5 or clicking the green Run arrow to display the application and page titles in the emulator, as shown in Figure 2-3.

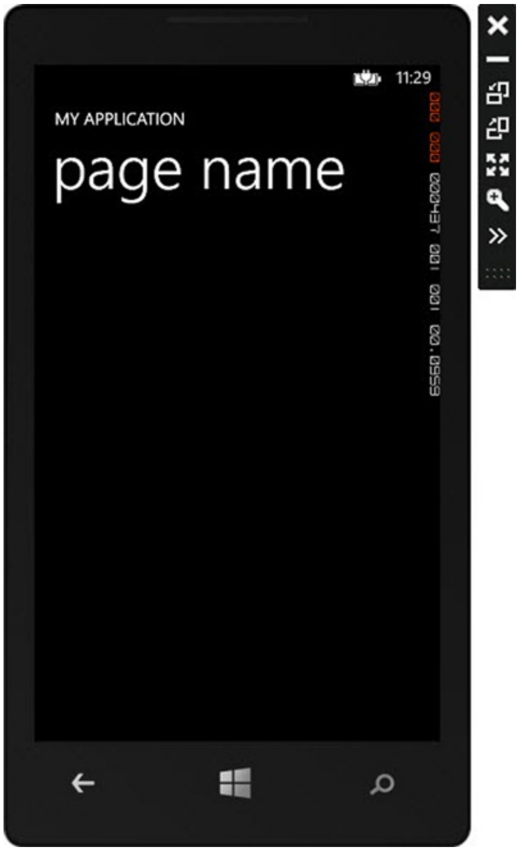
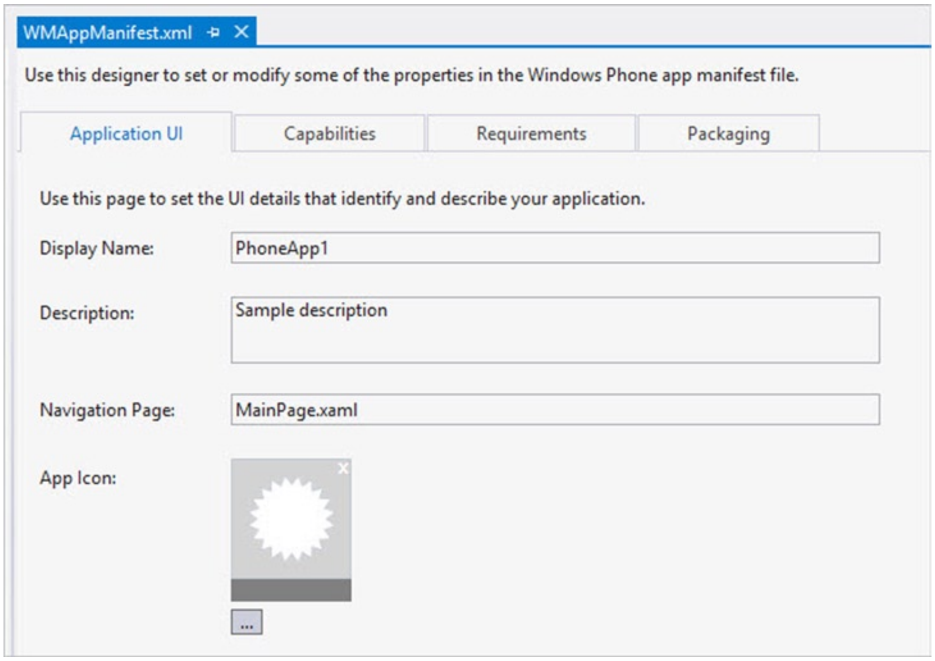


Figure 2-3. *Default Application Running in the Emulator*

Windows Phone 8 Solution Anatomy

When you create a new Windows Phone 8 application in Visual Studio, a single project is created. The key elements are:

- `AppManifest.xml` lists resources to include when deploying the application. Visual Studio takes care of maintaining this file for you. The `AppManifest.xml` file is located in the Properties folder of the project.
- `WMAppManifest.xml` describes the application in more detail including the `Display Name`, starting `Navigation Page`, `App Icon`, a list of `Capabilities` and hardware `Requirements`. The `WMAppManifest.xml` file is located in the Properties folder of the project. The values from the manifest are used by the device and by the Windows Phone Store application submission process. Double-click `WMAppManifest.xml` to see the Visual Studio designer. Figure 2-4 shows the visual tabbed interface where you can tweak settings without having to edit XML directly.



Use this designer to set or modify some of the properties in the Windows Phone app manifest file.

Use this page to set the UI details that identify and describe your application.

Display Name: PhoneApp1

Description: Sample description

Navigation Page: MainPage.xaml

App Icon:

Figure 2-4. *The WMAppManifest Designer*

■ **Tip** If you prefer to work with the XML directly, right-click `WMAppManifest.xml` (located in the Properties folder of the project) and select `View Code` from the context menu.

- The `Assets` folder contains the images for the application icon and the tile images that display in the Windows Phone 8 Start screen.
- The `Resources` folder allows your application to be localized for particular languages and cultures.
- The `LocalizedStrings` class provides access to language resources and can be referenced in your application's user interface.
- The `App` class is defined in files `App.xaml` and `App.xaml.cs` located in the root directory of the project. It is the entry point for the application and handles application startup/shutdown, phone life-cycle events (launching, moving the application to the foreground or background), exceptions and navigation failure. The snippet below shows a small sample of the `App` class.

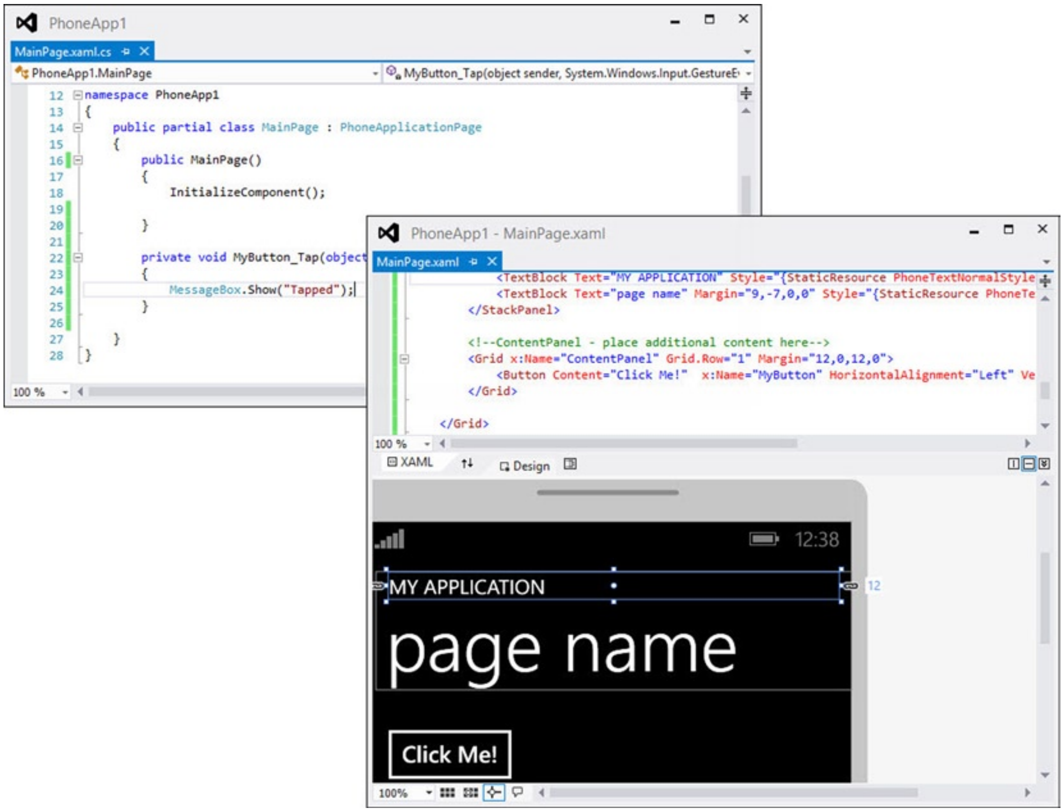
```
public App()
{
    // Global handler for uncaught exceptions.
    UnhandledException += Application_UnhandledException;

    // Standard XAML initialization
    InitializeComponent();
```

```
        // Phone-specific initialization
        InitializePhoneApplication();

        // Language display initialization
        InitializeLanguage();
    ...
```

- The initial main page is created automatically and contains a `.XAML` (Extensible Application Markup Language) file to describe the user interface and a `.XAML.CS` code-behind file to define client logic. Your application will typically have multiple page classes. The Figure 2-5 shows the editors for code behind, the XAML, and the design view of the XAML.



```
namespace PhoneApp1
{
    public partial class MainPage : PhoneApplicationPage
    {
        public MainPage()
        {
            InitializeComponent();

        }

        private void MyButton_Tap(object
        {
            MessageBox.Show("Tapped");
        }

    }
}
```

```
            <TextBlock Text="MY APPLICATION" Style="{StaticResource PhoneTextNormalStyle
            <TextBlock Text="page name" Margin="9,-7,0,0" Style="{StaticResource PhoneTe
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Button Content="Click Me!"  x:Name="MyButton" HorizontalAlignment="Left" Ve
        </Grid>

    </Grid>
```

Figure 2-5. *The MainPage XAML, Designer, and Code Behind*

- The XAP file is simply a compressed file, like a ZIP file, that contains all the files needed to execute the application. This file is created automatically by Visual Studio and placed under the \bin directory. To see this file, you need to depress the "Show All Files" button in the Solution Explorer. It is the XAP file that is deployed to the device or to the Windows Phone Store (see Chapter 11 for more information). Figure 2-6 shows the unzipped contents of the XAP file.

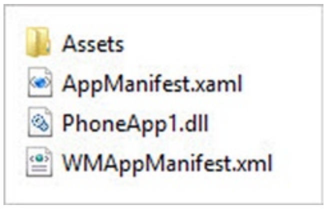


Figure 2-6. *XAP File Contents*

Running Applications from Visual Studio

By default, you run your Windows Phone 8 application in Visual Studio using the emulator. The emulator lets you run an application directly from your PC desktop without a physical device. If you have a Windows Phone 8 device plugged in to your computer via USB cable, you can run the application directly on the device. Use the Visual Studio `Device` drop-down on the Standard toolbar to select where the application should run (see Figure 2-7).

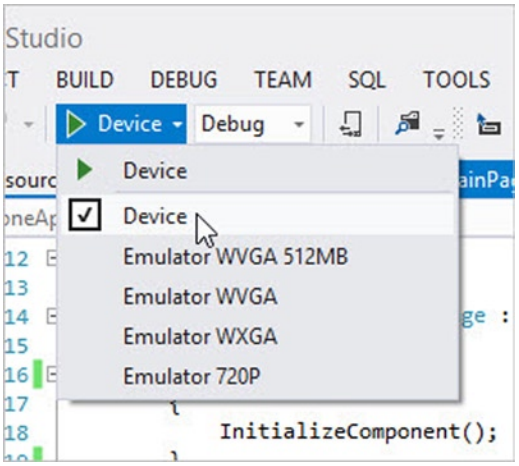


Figure 2-7. *Selecting from the Device Menu*

Using the Emulator

The emulator simulates Windows 8 Phone applications running in three screen sizes:

- `WVGA` - *800* × *480* pixels. An additional 512MB RAM version of WVGA allows you to emulate memory-constrained devices.
- `WXGA` - *1280* × *768* pixels.
- `720p` - *1280* × *720* pixels.

The Emulator user interface allows you to try your application in various orientations (e.g., portrait, landscape, portrait up, etc.), and use additional tools that mimic phone interaction with the external world like Accelerometer and Location. The Emulator also has tools to take screenshots of the current phone screen image and a Network tab that describes the Emulator's connectivity.

The Emulator User Interface

The emulator lets you run your application directly from your PC without needing a physical device. You can use your mouse to "tap" items onscreen, or click the Back, Start, and Search buttons at the bottom of the emulator window (see Figure 2-8). The toolbar to the right of the emulator has commands that manipulate the window. To test your application in each basic orientation, use the Rotate Left and Rotate Right buttons.

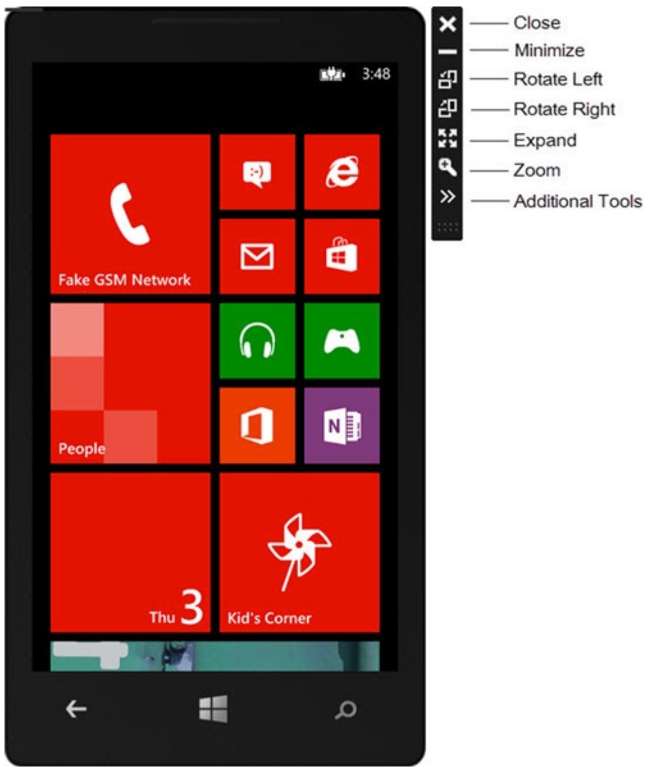


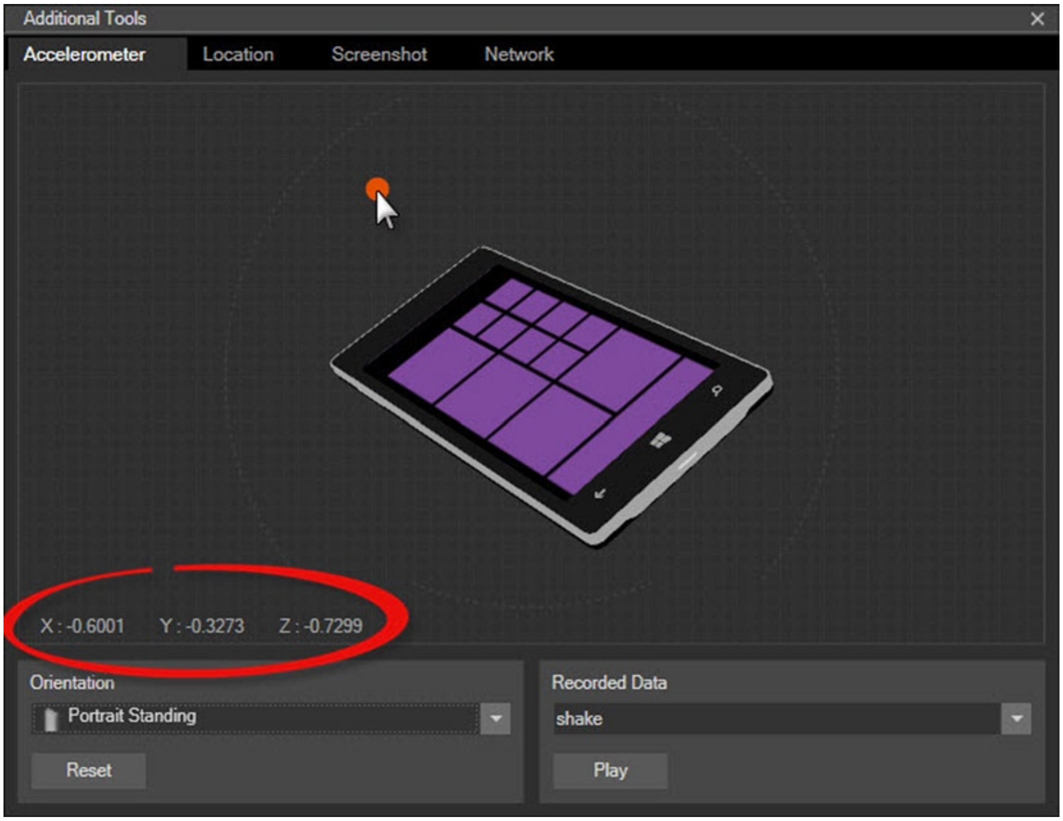
Figure 2-8. *The Emulator User Interface*

Additional Tools

Click the double angle (>>) button on the toolbar of the Emulator to open the `Additional Tools` dialog. This dialog lets you test several real-world scenarios without having an actual phone device, including moving the phone in space and geographically.

Emulator Accelerometer Tab

On the `Accelerometer` tab (see Figure 2-9), drag the red dot and observe the changes to the phone graphic and the X-Y-Z coordinates. Use the `Orientation` drop-down list to set the device to `Portrait Standing`, `Landscape Standing`, `Portrait Flat`, or `Landscape Flat` positions.



X : -0.6001 Y : -0.3273 Z : -0.7299

Orientation: Portrait Standing

Recorded Data: shake

Figure 2-9. *Using the Accelerometer*

The coordinates express gravity values for a given axis. For example, Z will be negative when the device is faced up, positive when faced down. If Z = 1, then the device is lying flat, facing straight down (see Figure 2-10).

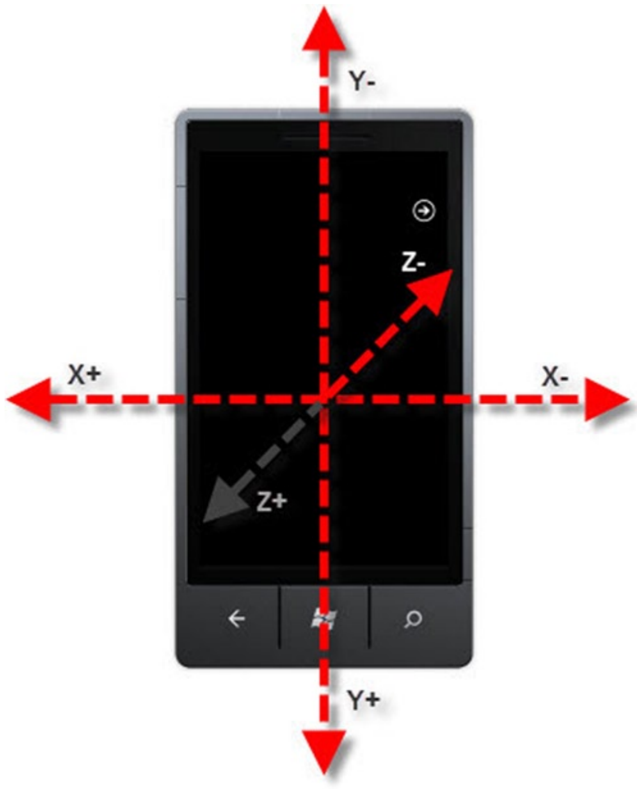


Figure 2-10. *Coordinate Orientation in the Accelerometer*

Select from the `Recorded Data` drop-down and click the `Play` button to play back some set of Accelerometer X-Y-Z settings in a series. The Shake option moves the X, then Y coordinates between negative and positive, back and forth, to simulate the physical shaking of a device (see Figure 2-11).

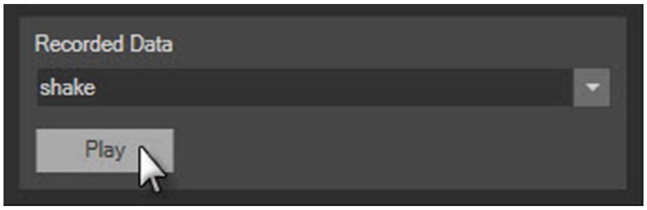


Figure 2-11. *Playing Accelerometer Data from a File*

CREATE CUSTOM ACCELEROMETER DATA

You can actually create your own recorded data. Look for the Shake file located in `<program files>\Microsoft XDE\8.0\sensordata\acc`. The file is in XML format with a series of X-Y-Z coordinates that occur at progressive offsets in time. The snippet below is a subset of the Shake file data.

```
<?xml version="1.0" encoding="utf-8"?>
<WindowsPhoneEmulator
xmlns="http://schemas.microsoft.com/WindowsPhoneEmulator/2009/08/SensorData">
  <SensorData>
    <Header version="1"/>
    <AccData offset="21" x="-00.08400000" y="-01.02100003" z="-00.41700000" />
    <AccData offset="42" x="-00.14200000" y="-00.95099998" z="-00.39700001" />
    <AccData offset="63" x="-00.29400000" y="-00.80199999" z="-00.30399999" />
    ...
```

Copy this file to another name, leave it in the same directory for the emulator to find and it will show up in the drop-down list. The example below is renamed “Shimmy” and shows up in the list (see Figure 2-12). You will need to restart the emulator for the file to display.

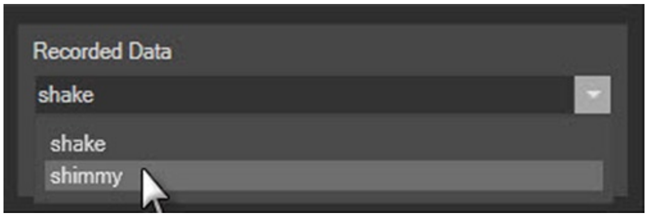
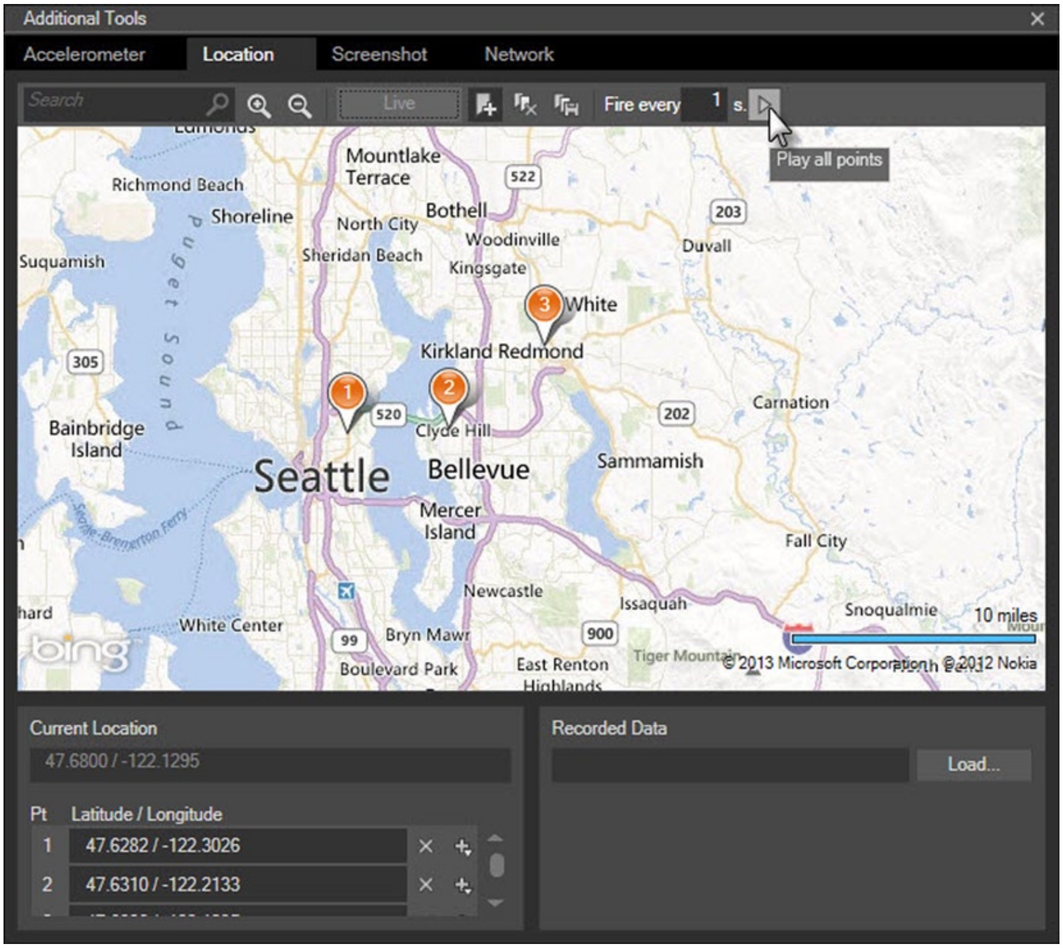


Figure 2-12. *Showing the Custom Recorded Accelerometer Data*

Emulator Location Tab

The `Location` tab lets you mimic geographic position changes. Click anywhere on the map to set a point. Each point will show up in a list of `Latitude/Longitude` at the lower left of the screen. Click the `Live` button and then the Play button to “move” between each location (see Figure 2-13). The map will animate to show each point as the center of the world (which is Redmond, Washington, according to this map).



Current Location: 47.6800 / -122.1295

Pt	Latitude / Longitude
1	47.6282 / -122.3026
2	47.6310 / -122.2133

Figure 2-13. Simulating Location Changes

If you need to restore a set of location points later, click the Save Map Points button (disk icon) from the toolbar at the top of the dialog (see Figure 2-13). The points are saved in `Locations.XML` at `\Users\<user name>\Documents\WindowsPhoneEmulator\Location`. The points are saved as XML and can be restored using the Recorded Data Load... button.

Emulator Screenshot Tab

The `Screenshot` tab makes it easy to capture the current emulator image in a size and format perfect for the Windows Phone Store. The `Capture` button saves the current emulator image without the chrome. That is, the hardware buttons and trim are left out of the image in accordance with requirements for App submissions for Windows Phone Store. Click the `Save` button to persist the image to disk (see Figure 2-14). See "App submission requirements for Windows Phone" – 4.6 – App Screenshots for more information (`http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184844(v=vs.105).aspx`). See Chapter 11 for directions on how to get your application published in the Windows Phone Store.

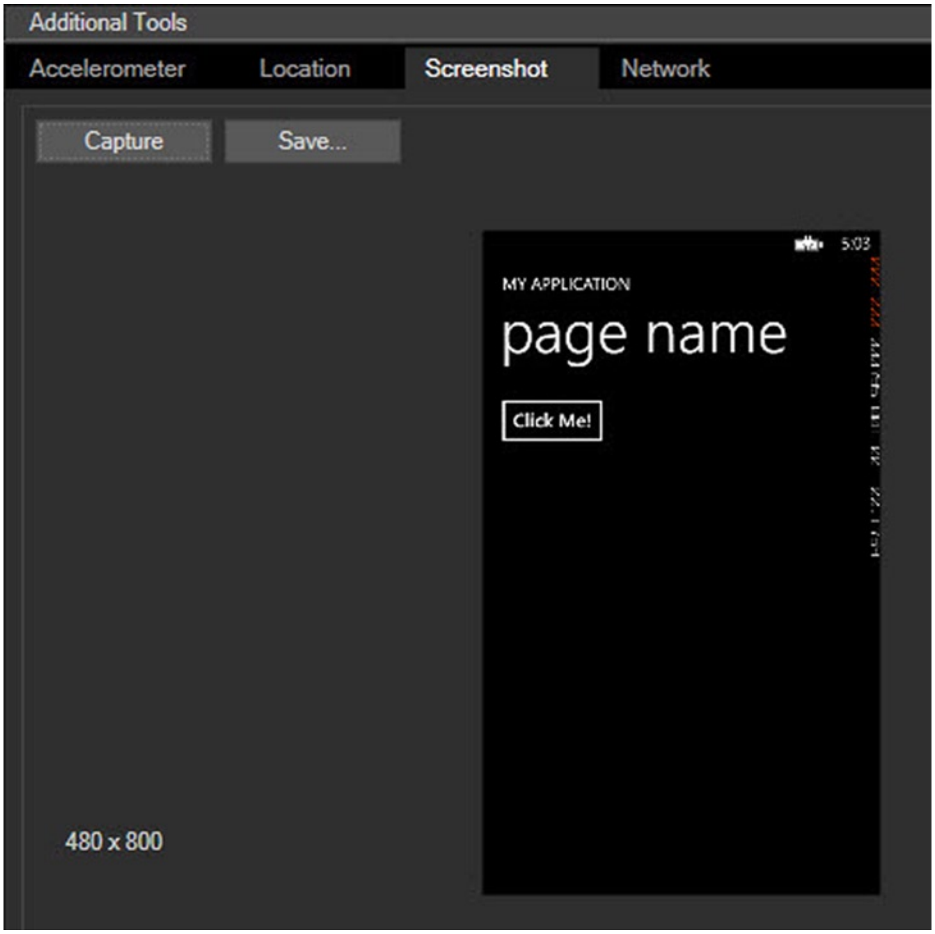
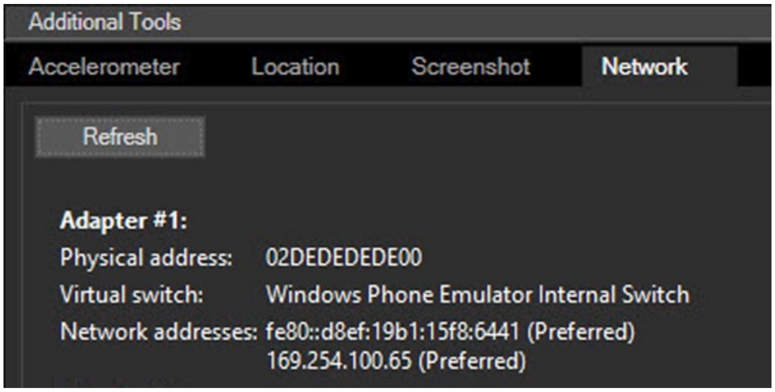


Figure 2-14. *Emulator Screen Capture*

Emulator Network Tab

The `Network` tab lists connections used by the emulator. The emulator comes with network support out-of-the-box. The emulator shows up on the network as a separate device with its own IP address. You don't need to install additional software (other than Windows 8) to get the emulator network connection. The connection will *not* be joined to a Windows domain, so if your development computer must log in to a domain or requires special client networking software, the emulator won't reach those network resources. Figure 2-15 shows the emulator's connection and IP address (the network address can be "pinged" from the command line).



Adapter #1:
Physical address: 02DEDEDEDE00
Virtual switch: Windows Phone Emulator Internal Switch
Network addresses: fe80::d8ef:19b1:15f8:6441 (Preferred)
169.254.100.65 (Preferred)

Figure 2-15. *List of Network Connections*

If your phone can access network resources over Wi-Fi, the emulator should be able to access those same resources. Figure 2-16 shows the emulator with a live internet connection to Google.

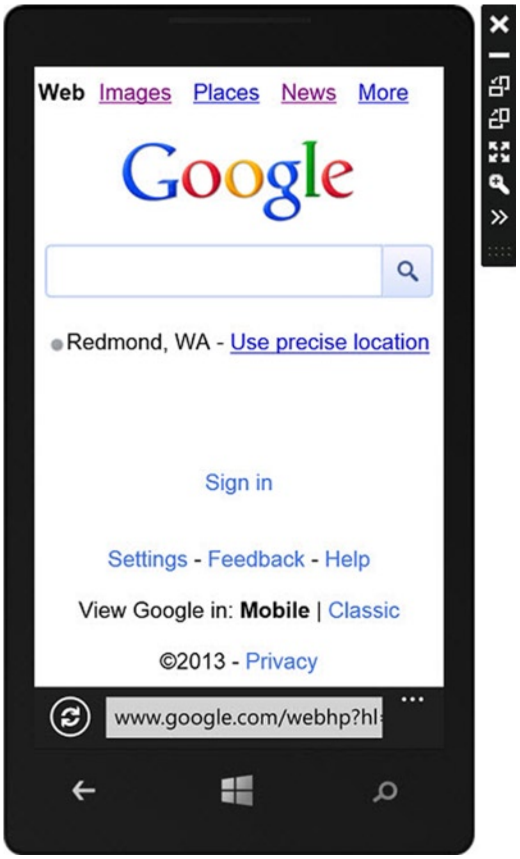

Web
Images
Places
News
More
Google
Redmond, WA - Use precise location
Sign in
Settings - Feedback - Help
View Google in: Mobile | Classic
©2013 - Privacy
www.google.com/webhp?hl

Figure 2-16. *The Emulator with Live Internet Connection*

Creating, Configuring, and Testing an Application

The following series of walk-throughs demonstrate:

- How to satisfy the prerequisites of deploying Windows Phone 8 applications to a phone, such as registering as a developer and unlocking the phone for development.
- How to build a simple Windows Phone 8 application and run the application in the Emulator.
- How to customize the application title and icons.
- How to customize elements on the page.

Prerequisites

There are a handful of prerequisites to deploying Windows Phone 8 applications to your phone device. The first is registering as a developer at `dev.windowsphone.com`. This step will require you to have a Microsoft (Windows Live ID) account.

1. Click the `Sign in` link to enter your existing Microsoft account credentials or follow the links to create a new Microsoft account.
2. Click the `Dashboard` link (see Figure 2-17) at the top of the page (`https://dev.windowsphone.com/join`).

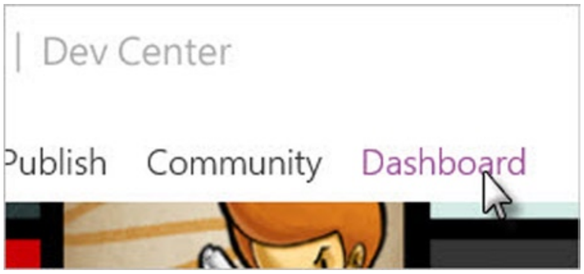
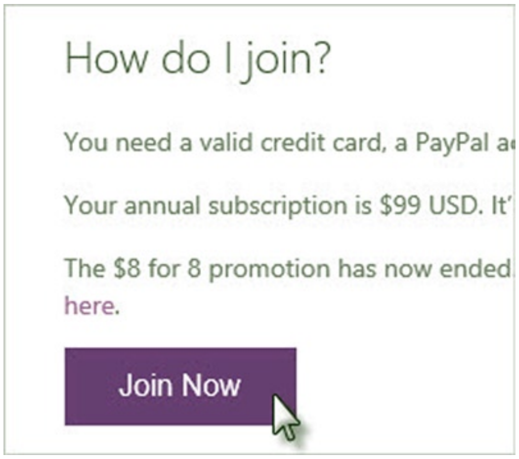


Figure 2-17. *Navigating to the Dashboard*

3. Click `Join Now`, as shown in Figure 2-18, and follow the prompts to complete. There is a subscription cost of $99 USD at the time of this writing.



How do I join?

You need a valid credit card, a PayPal a

Your annual subscription is $99 USD. It'

The $8 for 8 promotion has now ended
here.

Figure 2-18. *Creating a developer account*

Next, register your phone device to "unlock" the phone for development. The phone must have a cell or Wi-Fi connection. The USB connection alone is not enough.

1. Turn on the phone and unlock the screen.
2. Verify the date and time on the phone. The `Settings` app lets you configure the date and time for the phone. Typically, this will be synced automatically with the correct date and time.
3. Connect the phone device USB to the development PC.
4. On the development PC, navigate to the `Start` screen, type "Windows Phone Developer Registration" to search for the registration application, and click the entry in the Apps list to run it (see Figure 2-19).

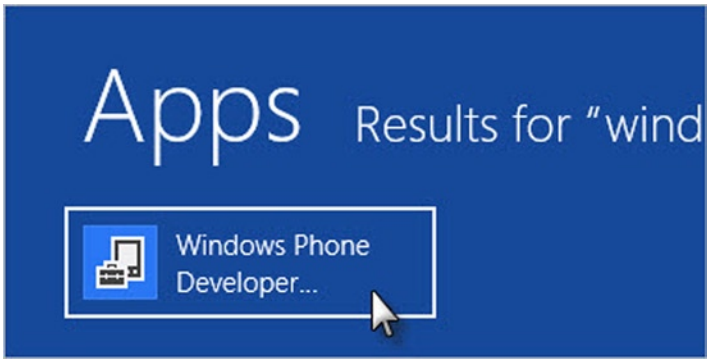
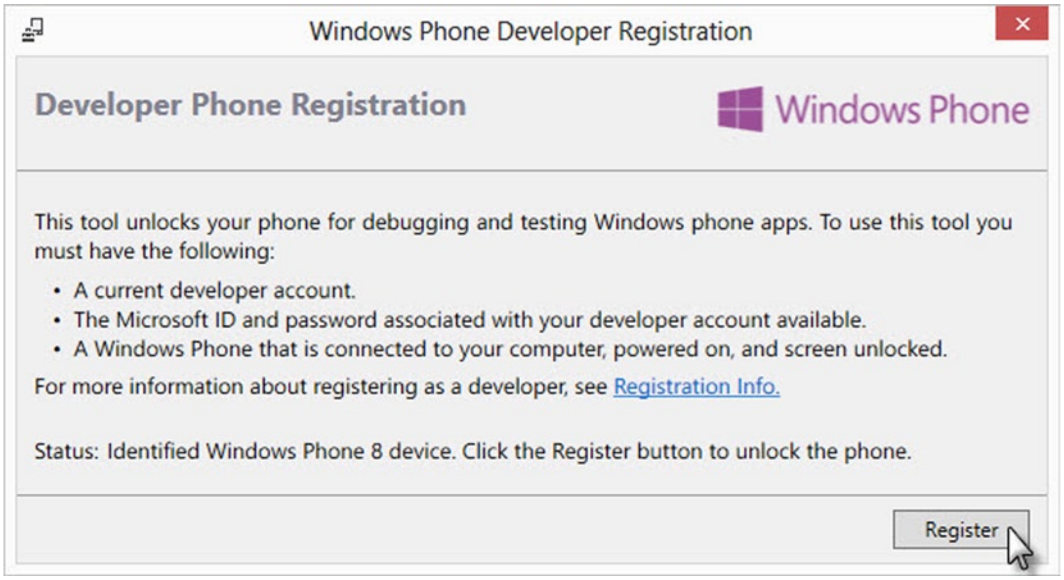


Figure 2-19. *Running the Windows Phone Developer Registration Application*

5. In the Windows Phone Developer Registration screen, verify the `Status` message. The Status will indicate if the phone is ready to be registered and will flag any problems, such as your device is unconnected, before continuing.

6. Click the `Register` button (see Figure 2-20).



Windows Phone Developer Registration
Developer Phone Registration
Windows Phone
Register

This tool unlocks your phone for debugging and testing Windows phone apps. To use this tool you must have the following:

- A current developer account.
- The Microsoft ID and password associated with your developer account available.
- A Windows Phone that is connected to your computer, powered on, and screen unlocked.

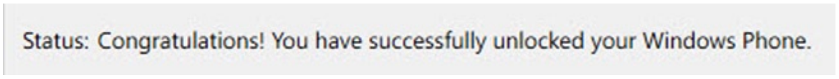
For more information about registering as a developer, see Registration Info.

Status: Identified Windows Phone 8 device. Click the Register button to unlock the phone.

Figure 2-20. *Registering the Phone Device*

7. Sign in with your Windows Live ID connected to your registered developer's account.

8. Check the `Status` again to verify you have successfully unlocked your Windows Phone, as shown in Figure 2-21.



Status: Congratulations! You have successfully unlocked your Windows Phone.

Figure 2-21. *Success Status Message*

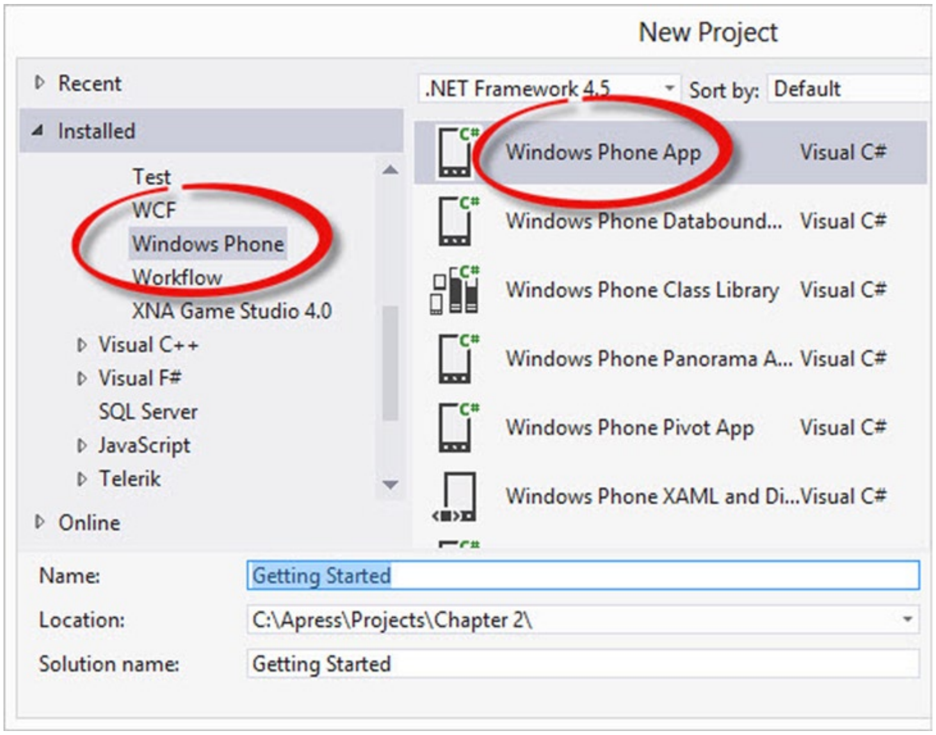
Well done, now you can deploy Windows Phone 8 applications directly onto the phone!

■ **Note** If you receive an error message "Not registered with the marketplace 80043001," this means that the Windows Live ID you used is not associated with an active developer account. You need to go to the Dashboard at `dev.windowsphone.com` and join to get a developer account.

Building the Example Application

The steps that follow demonstrate building and running a simple “hello world” application. In the process, we will change some superficial things like the application icon, tile icon, and titling.

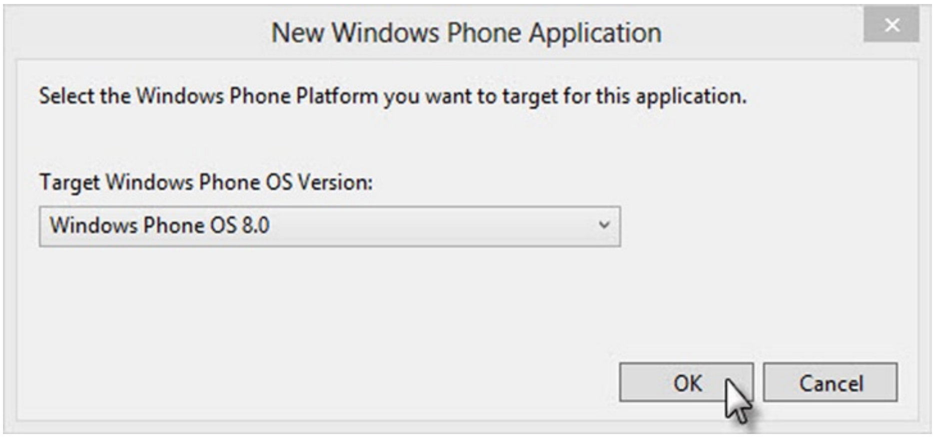
1. From the Visual Studio menu select `File` ➤ `New` ➤ `Project`. In the `New Project` dialog, shown in Figure 2-22, select `Windows Phone` from the Installed Templates list. Select the `Windows Phone App` template. Set the Name of the project to “Getting Started.” Click the `OK` button to continue.


New Project
Recent
Installed
Test
WCF
Windows Phone
Workflow
XNA Game Studio 4.0
Visual C++
Visual F#
SQL Server
JavaScript
Telerik
Online
.NET Framework 4.5
Sort by: Default
Windows Phone App Visual C#
Windows Phone Databound... Visual C#
Windows Phone Class Library Visual C#
Windows Phone Panorama A... Visual C#
Windows Phone Pivot App Visual C#
Windows Phone XAML and Di... Visual C#

Name: Getting Started
Location: C:\Apress\Projects\Chapter 2\
Solution name: Getting Started

Figure 2-22. *Creating a new Windows Phone App*

2. Next, the New Windows Phone Application dialog prompts for the `Target Windows Phone OS Version`. Leave the default Windows Phone OS 8.0 selected and click the OK button to continue (see Figure 2-23).


New Windows Phone Application

Select the Windows Phone Platform you want to target for this application.

Target Windows Phone OS Version:

Windows Phone OS 8.0

Figure 2-23. *Selecting the Target Windows Phone OS Version*

3. Locate the deployment device drop-down list in the Visual Studio toolbar and verify that `Emulator WVGA` is selected (see Figure 2-24).

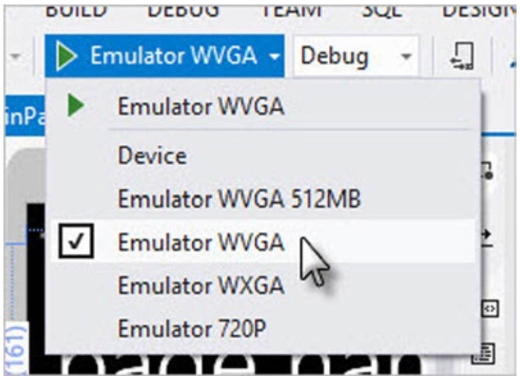


Figure 2-24. *Selecting the Deployment Device*

4. Click the green play button to run the emulator. The emulator should display and then load your application. By default, the application title reads "MY APPLICATION" and the page title is "page name" as shown in Figure 2-25.

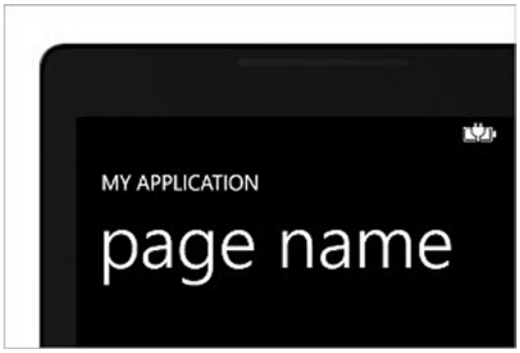


Figure 2-25. *The Default Page Running in the Emulator*

5. Click the back button shown in Figure 2-26 to return to the home page. The emulator will stay open.

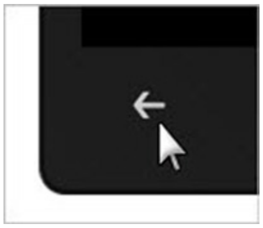


Figure 2-26. *Navigating with the Back Button*

6. The Start screen displays *tiles* that display important live information and allow direct navigation to run your favorite apps. Scroll to the bottom of the Start screen and click the forward button shown in Figure 2-27 to display the *App List*.

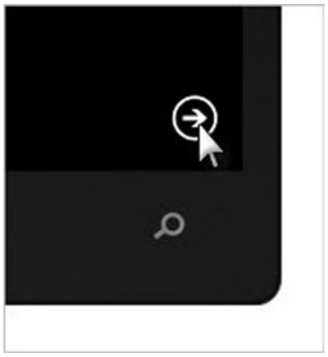


Figure 2-27. *Navigating with the Forward Button*

7. The App List shows “Getting_Started” is listed along with the other applications available on the emulator. Click and hold the “Getting_Started” item. This is equivalent to a right-click on a PC and displays a context menu. Click the `pin to start` item from the menu (see Figure 2-28).

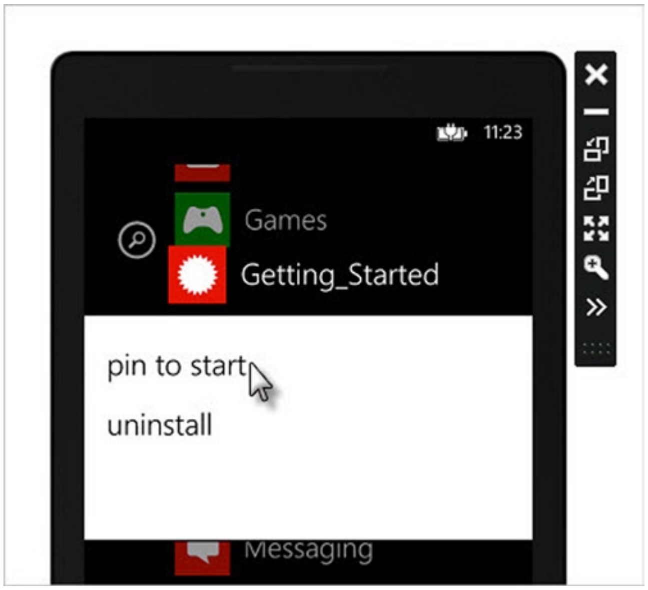


Figure 2-28. *Pinning the Application to the Start Screen*

8. The “Getting_Started” application now has a tile on the Start screen (see Figure 2-29).

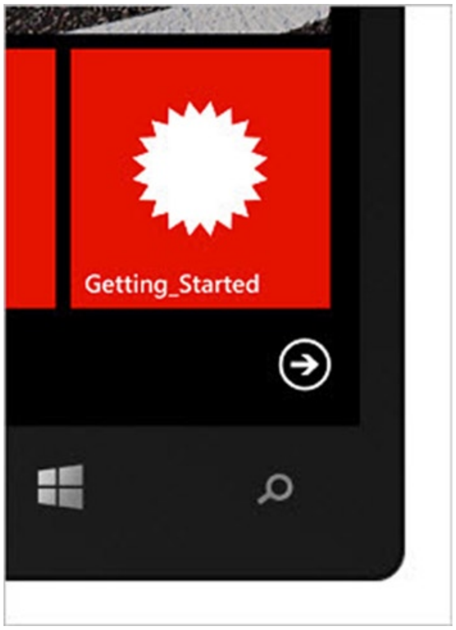


Figure 2-29. *The Application Pinned in the Start Screen*

Click and hold the tile and select the “unpin” icon to remove the app tile. You will need it unpinned for the next exercise.

Customizing Application Settings

This extends the previous set of steps by customizing the title and icons for the application and tile graphics.

The initial Windows Phone 8 project created by default in Visual Studio uses three icons:

- `IconicTileSmall.png` and `IconicTileMediumLarge.png`, in the `\Assets\Tiles` directory, are displayed on the Start screen in two sizes the user can configure.
- `ApplicationIcon.png`, in the `\Assets` directory, shows up on the App List.

The following steps assign these icons and provide titling and captions:

1. Open the project from the previous exercise in Visual Studio.
2. In the Solution Explorer copy the three image files mentioned above and rename them. Figure 2-30 shows the three new files in the Solution Explorer.

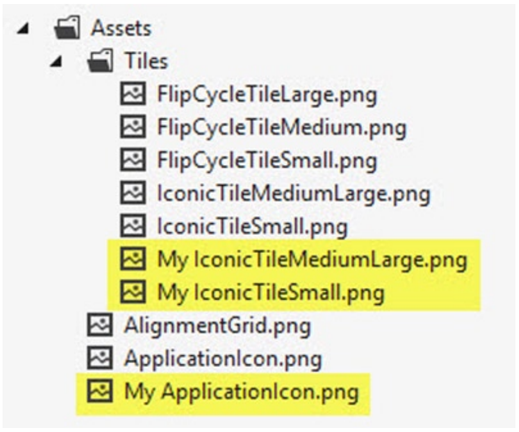


Figure 2-30. *Copied and Renamed Graphics in the Solution Explorer*

3. In the Solution Explorer, double-click each of the three images and edit each image using the Visual Studio 2012 icon editor. You can also use any other drawing tools you are familiar with. The point is to simply tweak the image enough that you will recognize the change when it’s displayed in the running application. Figure 2-31 shows the addition of text using the icon editor.

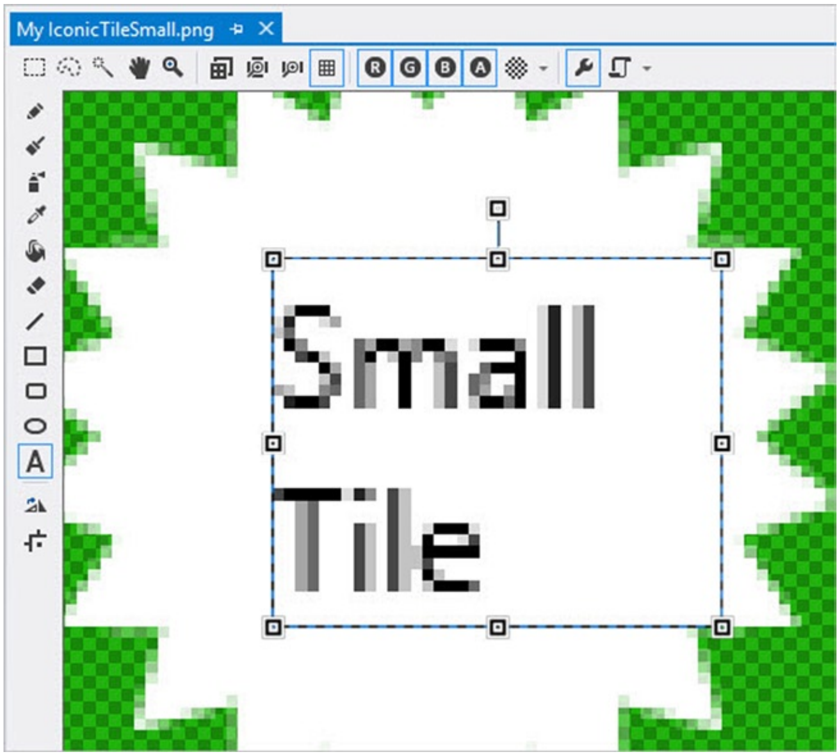
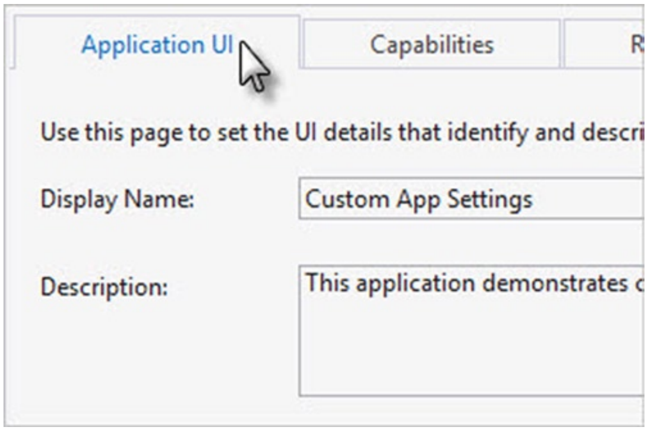


Figure 2-31. *Editing Tile Graphic in the Icon Editor*

4. In the Solution Explorer, double-click the `\Properties\WMAppManifest.xml` file. This will open up the editor for the project settings. Click on the `Application UI` tab shown in Figure 2-32.



Use this page to set the UI details that identify and descri

Display Name: Custom App Settings

Description: This application demonstrates c

Figure 2-32. *Selecting the Application UI tab*