XNA Games On Windows Phone 8 with Monogame

This tutorial shows how to use the MonoGame open source project to create XNA games on Windows Phone 8.

Introduction

XNA has been an awesome framework for creating games on the Windows Phone 7.x, Windows and XBOX 360 platforms. It provided very powerful features such as a content pipeline, spritebatches, shaders and 3D, sound, music, game controllers and so much more. Many game developers (including myself) have been using XNA for many years now and have significant codebase, tools, UI libraries and more developed with XNA in mind.

Unfortunately, Microsoft has deprecated support for XNA on the Windows Phone 8 platform (as well as on Windows 8 Store apps). While XNA games still work on Windows Phone 8, they work in a "compatibility mode" which essentially means that the developer is targeting WP7.x and cannot use any of the awesome new features WP8 brought to the table.

MonoGame allows developers to target Windows Phone 8 while still using XNA and C#, our favorite combination for making games. If you are a seasoned XNA game developer or a newbie looking to make your first XNA game, MonoGame is an awesome choice. I've used MonoGame myself in a couple of commercially available projects, the latest of which is MonsterUp Memory for Windows 8, currently available at the store.

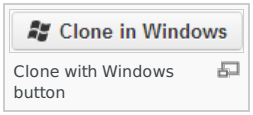
What is MonoGame

MonoGame is an Open Source implementation of the Microsoft XNA 4 Framework. Without disassembling any of the original XNA libraries, the community is trying to build a compatible open source library (using even the same namespaces) that targets multiple platforms. MonoGame utilizes the power of other open source projects, like SharpDX, which delivers the full DirectX API under the .NET platform for 2D and 3D graphics as well as sound and Lidgren.Network which provides networking support for MonoGame. But the good news is that you don't have to worry about these, since in the end MonoGame provides you, the XNA game developer, with a familiar XNA compatible game development framework that can be used to target Windows Phone 8. An amazing aspect of MonoGame is that it opens the doors not only to be able to continue to use XNA, but also to port your games to other platforms.

Getting started

Before getting started, we have to setup our development environment. First of all, you need a Windows 8 x64 Pro based system. If you also want to use the WP8 emulator, you need to make sure your system supports SLAT. If you download the WP8 SDK from the link you can find to the side of this article, you will get Visual Studio 2012 Express installed at the same time. This is enough, but if you want to use the full version of Visual Studio 2012, make sure you install this before installing the SDK. If you as many developers are not yet ready to install Windows 8 on your production machine, you can consider the possibility to install it on a Virtual Machine. If that's the case, take a look at this article. After VS and WP8 SDK are installed, we need to get MonoGame itself. There are a few ways to do this:

- Download the full source code of the project from here and unzip it in a directory of your choice, or
- Since MonoGame is hosted in GitHub, you can use your favourite Git client to clone the repository. I use the official GitHub app which you can download from here for free. Install the client, then go to the MonoGame GitHub page and click on the "Clone with Windows" button.



Clone with Windows button

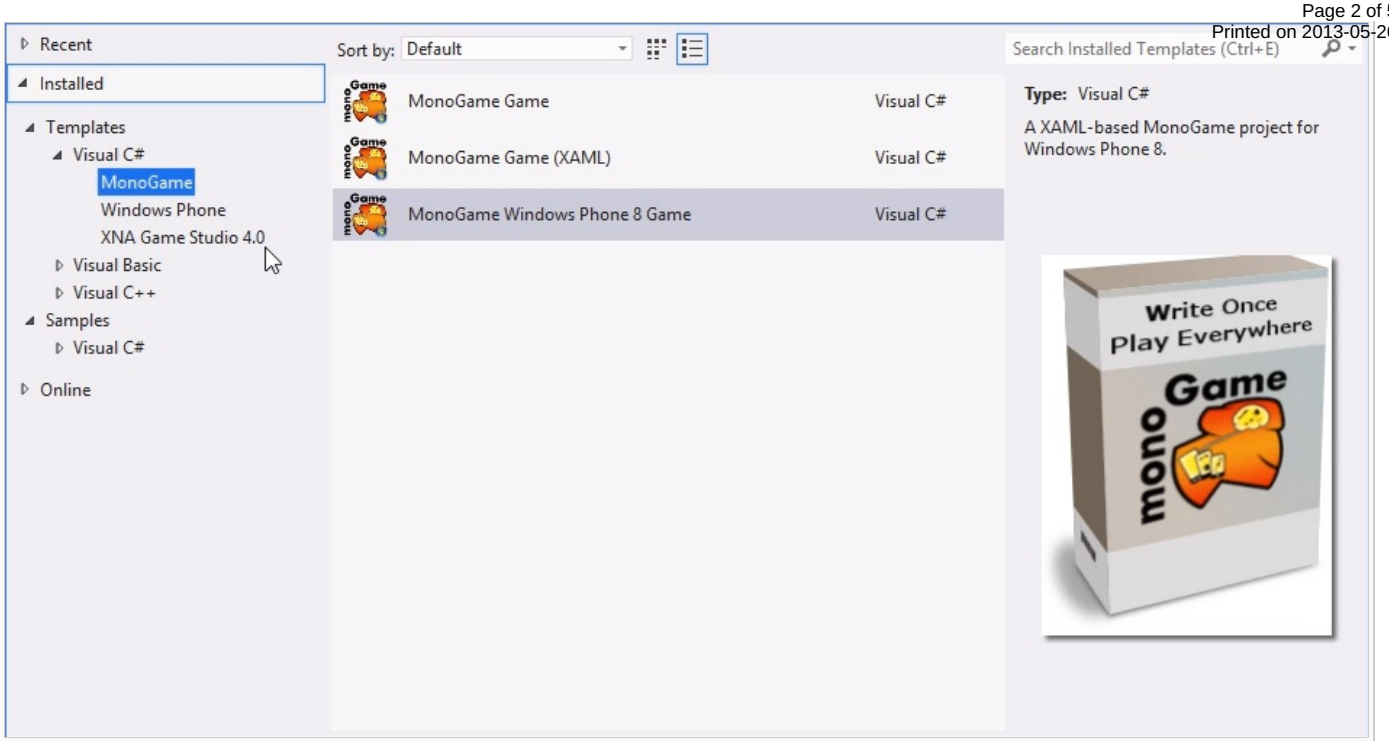
- You can also use Windows PowerShell and GitHub command line interface to achieve the same results. Just go to the directory you want MonoGame to reside and use the following commands.

```
git clone https://github.com/mono/MonoGame.git
cd .\MonoGame
git submodule init
git submodule update
```

Most likely, after you experience the power and simplicity of XNA, you will want to make lots of games with it. So, we should better make a Visual Studio 2012 template for them. Luckily, MonoGame comes with these templates and all you need is to copy them to the correct place.

Go to **C:\Users\[your username]\Documents\Visual Studio 2012\Templates\ProjectTemplates\Visual C#** and create a MonoGame directory. Then go to the place you downloaded the MonoGame project source, open ProjectTemplates and then VisualStudio2012. Copy everything in here to the directory you created before. If you do this right, the next time you open Visual Studio 2012 and click on New Project you will see a new MonoGame category under Visual C#. Well done. Double click on Windows Phone 8 Game to create your first XNA WP8 MonoGame game.

(C) Copyright Nokia 2013. All rights reserved.



The MonoGame templates

Linking to the MonoGame framework

The template is setup and you have created your solution. But it still does not know where you downloaded MonoGame itself! This is very easy to fix. Just follow these steps.

- Right click on your solution
- Choose **Add -> Existing Project**
- Go to the directory your downloaded MonoGame
- Open **MonoGame.FrameWork**
- Choose **MonoGame.Framework.WindowsPhone.csproj** file
- Click Open
  Now you have added the MonoGame framework project to your solution,you need to reference it in your game.
- Open References in your project in solution explorer
- Delete the reference that has a yellow triangle named **MonoGame.Framework.WindowsPhone**
- Right click on References and choose **Add Reference**
- Click on **Projects**
- You should see the MonoGame.Framework project on the list. Tick the box next to it.
- Click **OK**

Now MonoGame is added and linked to your game project.

Fixing minor issues

At this point, you might notice that the project has issues compiling and running. We need to fix a few minor things and we are set to go.

- In the solution explorer, click on the Show All Files button.
- Find Game1.cs, right click and choose Include in Project
- Open GamePage.xaml
- Change
  ```
  x:Class="XamlPhoneGame1.MainPage
  ```
  to
  ```
  x:Class="XamlPhoneGame1.GamePage
  ```
- Open **GamePage.xaml.cs**
- Change
  ```
  public partial class MainPage : PhoneApplicationPage
  ```
  to
  ```
  public partial class GamePage : PhoneApplicationPage
  ```
  and
  ```
  public MainPage()
  ```


(C) Copyright Nokia 2013. All rights reserved.

to

```
public GamePage()
```

- Finally open **Properties -> WMAppManifest.xml** in your game project and change the Navigation Page from **MainPage.xaml** to **GamePage.xaml**

At this point, you have a working XAML, XNA compatible, WP8 MonoGame game project that compiles and runs on the emulators and on awesome Nokia WP8 devices. Congratulations!

How about game content?

One of the many awesome features of XNA is its content pipeline. This allowed you to use many of the popular formats (png, jpg, wav, mp3, x and many more) in your games. The XNA content pipeline took all these files and created .xnb files which in turn could be drawn, played or rendered in your games on any of the supported platforms. For MonoGame, I have bad news and good news. The bad news are that a similar content pipeline is not available but the good news is that there is a pretty simple and straightforward workaround. All you have to do is follow these steps.

- Create a new XNA project targeting WP 7.x.
- Add all your content in the content project of the solution.
- Build the solution.

At this point, you will have all the .xnb files ready to be used in your WP8 game. To make things easier and to avoid adding content files and changing their properties all the time, do the following.

- Go to your WP8 project directory and create a new directory called Content.
- Add this directory to your project.
- Copy all your .xnb files from the bin directory of your WP7.x project into the Content directory of your WP8 project.
- Add only one .xnb file in your WP8 project. Any one will do. Save everything.
- Edit the vsproj file of your game project using your favorite text editor.
- Replace the Content tag with the following

```
<Content Include="Content\**\*.*">
<CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
</Content>
```

This will automatically add any .xnb file you have in the Content directory in your project and set their properties correctly (basically it sets the Copy to Output to Copy if Newer so you don't have to for each file). If you add more .xnb files while developing your game, they will automatically be added too. If you feel more adventurous you may want to use file system soft links to create the .xnb files from the WP7.x project directly inside the Content directory of your WP8 project or even run the XNA content processing automatically every time you build your WP8 MonoGame project but I will not cover these here since they are a bit advanced for a "Getting Started" tutorial.

I've followed the steps, build my first XNA compatible MonoGame game and ran it. Now what?

Now utilize your XNA skills to make awesome game that can use the new awesome features of WP8. Make freemium games using in app purchases (IAP). Make HD games using 720p or WXGA resolutions. Make multiplayer games using NFC, bluetooth or wi-fi. This tutorial does not aim to teach you XNA, there are awesome tutorials out there for that, since XNA is quite mature at this point. But I hope I helped bootstrapping your first XNA game on Windows Phone 8.

I need help! Where do I go?

MonoGame has a very active community. The Windows Phone discussion of MonoGame can be found here. The official pages of the MonoGame projects are here and here.

Files

You can download the Final Template (which is the same you would get if you followed the steps above) from below. Follow the instructions in the ReadMe file to test it out on the emulator or on a real device.

File:TemplateFinal.zip

You can also download a simple sample WP8 game using XNA and MonoGame featuring the green monster from my hit game MonsterUp Adventures :) This simple game runs in ultra high resolution (all 3 supported from WP8) which is impossible using just XNA for WP7.1 which supports only 800x480(wvga). Get the game from below also.

File:SampleGame.zip **COPY-PASTE** the desired platforms into the wiki text and save. All relevant version categories should be applied.
Windows Phone: [[Category:Windows Phone]]
[[Category:Windows Phone 7.5]]
[[Category:Windows Phone 8]]

Nokia Asha: [[Category:Nokia Asha]]
[[Category:Nokia Asha Platform 1.0]]

(C) Copyright Nokia 2013. All rights reserved.

Series 40: [[Category:Series 40]]
[[Category:Series 40 1st Edition]] [[Category:Series 40 2nd Edition]]
[[Category:Series 40 3rd Edition (initial release)]] [[Category:Series 40 3rd Edition FP1]] [[Category:Series 40 3rd Edition FP2]]
[[Category:Series 40 5th Edition (initial release)]] [[Category:Series 40 5th Edition FP1]]
[[Category:Series 40 6th Edition (initial release)]] [[Category:Series 40 6th Edition FP1]] [[Category:Series 40 Developer Platform 1.0]] [[Category:Series 40 Developer Platform 1.1]] [[Category:Series 40 Developer Platform 2.0]]

Symbian: [[Category:Symbian]]
[[Category:S60 1st Edition]] [[Category:S60 2nd Edition (initial release)]] [[Category:S60 2nd Edition FP1]] [[Category:S60 2nd Edition FP2]]
[[Category:S60 2nd Edition FP3]]
[[Category:S60 3rd Edition (initial release)]] [[Category:S60 3rd Edition FP1]] [[Category:S60 3rd Edition FP2]]
[[Category:S60 5th Edition]]
[[Category:Symbian^3]] [[Category:Symbian Anna]] [[Category:Nokia Belle]]

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.