Archived:Creating a New Cocos2d-x Project for Windows Phone 8

This article explains how to create a new Cocos2d-x project for Windows Phone 8.

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

This article was created for cocos2d-x v2.0.3 beta port for Windows Phone. While the approach for creating new projects remains similar, newer integrations you should follow the official cocos2d-x instructions at cocos2d-x environment setup

Introduction

Cocos2d-x is a very popular 2D game framework. It's based on C++ and it has been ported before to iOS, Android, Windows, Marmalade, Linux, Bada, Blackberry-QNX, and now it's available also for Windows Phone 8 (c++ is supported). Creating games with Cocos2d-x makes sense, since you can easily port the games from one platform to another. This tutorial guides you on how to create a new project for the Windows Phone 8 platform, and after that you can continue by starting to build your own application, or porting an existing one.

You can download the Cocos2d-x Windows Phone 8 variant from their website.

You can import the Cocos2d-x project to Visual Studio, but soon you'll notice that the whole framework is actually built inside a HelloWorld application. The next steps describe how to remove the helloworld-stuff, and start building you own application.

Tip: If you're going to port your existing project then it may be a good idea to import the original Cocos2d-x project first and run the tests. These tests should tell you which features are supported in the Cocos2d-x Windows Phone 8 port and which are not.

Setting up the project

- Download the Cocos2d-x Windows Phone 8 version.
- Rename the **cocos2dx-0.13.0-wp8-0.8** folder properly for your application.
- Go inside that new folder. You can delete the **cocos2dx-wp8.sdf** file. You should also see a folder named **HelloWorld**, you should also rename this folder properly for your own projects.
- Edit the **cocos2dx-wp8.sln** file. The original file should look like this:

```
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "cocos2d",
"HelloWorld\cocos2d.vcxproj", "{1A0D24B4-1DE0-400B-BC35-92A51351477F}"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "test", "tests\tests.vcxproj",
"{A9E507CA-55C4-45D9-A0E4-6450A43176A6}"
EndProject
```

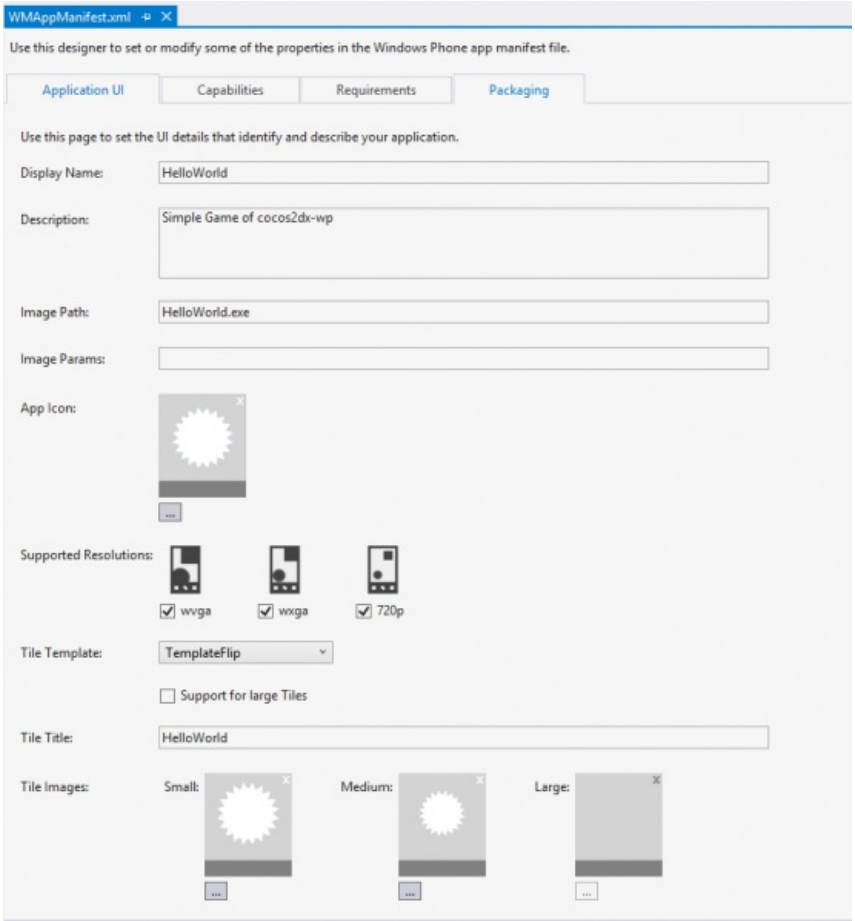
You can remove the tests project, and you should change the name of the **cocos2d** app, and it should point into your own app. It should look like this:

```
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "myapp", "myapp\cocos2d.vcxproj",
"{1A0D24B4-1DE0-400B-BC35-92A51351477F}"
EndProject
```

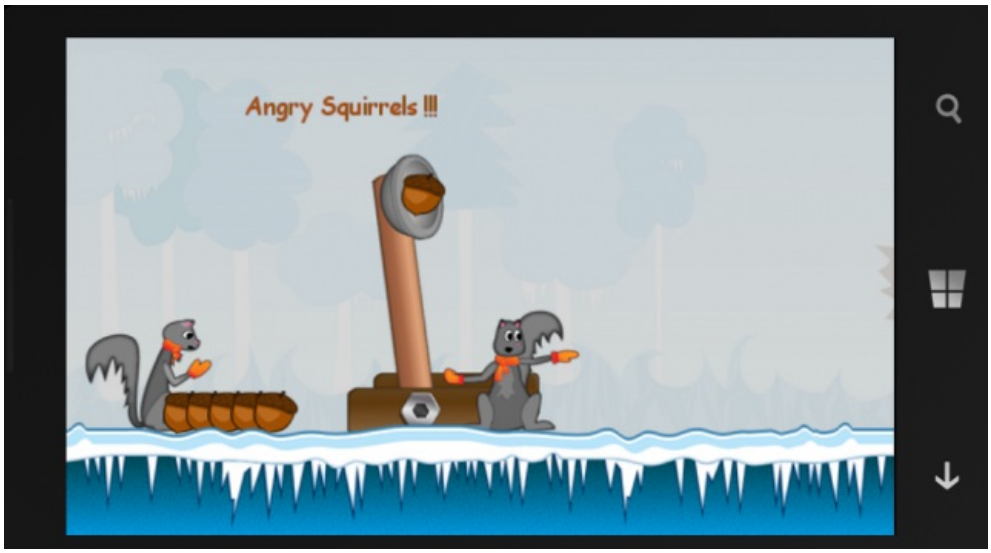
Now load up the project with Visual Studio. Open up **WMAppManifest.xml**, and change the application settings: name, icon, tile images, etc. In the packaging tab you should also change Author & Publisher name, Product Id and Publisher Id. You can easily create new GUIDs with the Visual Studio GUIDGen tool or using this online generator. Right click the **HelloWorld** project name in you project explorer, and choose to rename it. The image below shows example of the HelloWorld project

(C) Copyright Nokia 2014. All rights reserved.

WMAppManifest.xml file, which you should edit for your own application.



Now you should try to build and run the new project. You should see the Angry Squirrels game.



If the game worked on the emulator, then it's time to move forward. Now you can delete the Angry Squirrels stuff and start building your own game. You can safely remove all assets related to the Angry Squirrels game (all the images of Squirrels and blocks etc.). However if you didn't change the app-icon or application tiles from **WMAppManifest.xml** you should keep the Assets/Tiles folder and **ApplicationIcon.png** file.

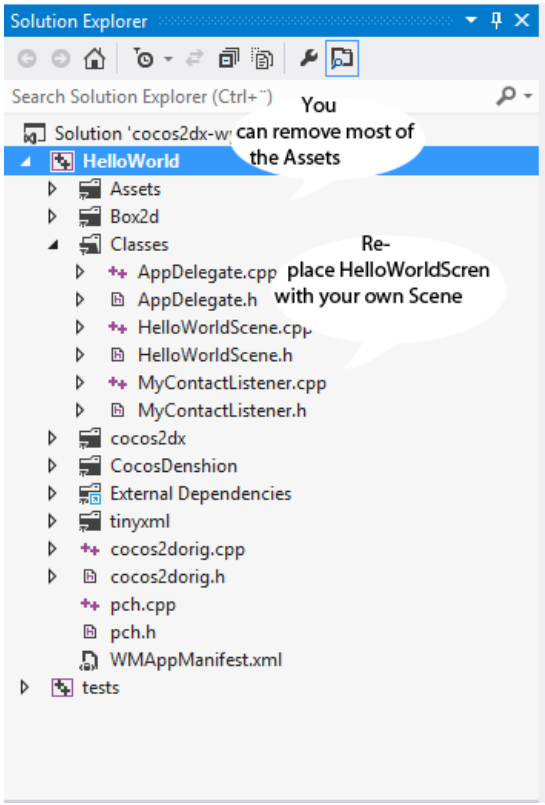
Cocos2d-x Application Structure

The HelloWorld project is basically building the cocos2d-x framework and dependencies within your own application. Here's quick overview of the Project structure:

- **Assets**: This folder has all of your project's asses such as images etc.
- **Box2d**: Box2d physics library. Box2d is part of the cocos2d-x framework and it's build with your project too
- **Classes**: Your own classes related to your own application
- **cocos2dx**: Cocos2d-x itself, contains sprite blitting with DirectX etc.
- **CocosDenshion**: Simple audioengine, part of the Cocos2d-X framework

(C) Copyright Nokia 2014. All rights reserved.

- **tinyxml**: Lightweight xml-parser, also part of the cocos2d-x



Using Cocos2d-x

You should see three classes under the **Classes** selection in Visual studio: `Appdelegate`, `HelloWorldScene`, and `MyContactListener`.

The **Appdelegate.cpp** is responsible for binding the Cocos2d classes into the WP8 platform, and you should keep it in your own project as well. The AppDelete handles all the "magic" between windows phone platform and cocos2-d application. The wp8 port is using DirectX instead of OpenGL, but the wp8 port, and the AppDelegate hides it all form the developer, so unless you're planning to use shaders in your game, you should newer see a difference between wp8 and iOS/Android ports.

The `HelloWorldScene` creates the cocos2d sprites & box2d objects for the game, and the `MyContactListener` is just a helper class, so you can delete both of them, and create your own scene instead. However I recommend keeping the old HelloworldScene file as a reference for the future.

That's it. If you're new to the cocos2d-x, and you're building a new game, you should start with cocos2d-x tutorials at cocos2d-x tutorials

Porting Existing Cocos2-x Applications to Windows Phone 8

For porting existing cocos2d-x applications you should see Porting Cocos2d-x Games for Windows Phone 8

(C) Copyright Nokia 2014. All rights reserved.

 (C) Copyright Nokia 2014. All rights reserved.