Creating multi-platform games with Cocos2d-x version 3.0 or later

In this tutorial you'll learn how to create a simple game using the Cocos2d-x framework, version 3.0 or later, in a Windows* development environment and how to compile it to run on Windows and Android*.

What is Cocos2d-x?

Cocos2d-x is a cross-platform framework for games (and other graphical apps, like interactive books) based on the cocos2d for iOS*, but using C++, JavaScript*, or Lua* instead of Objective-C*.

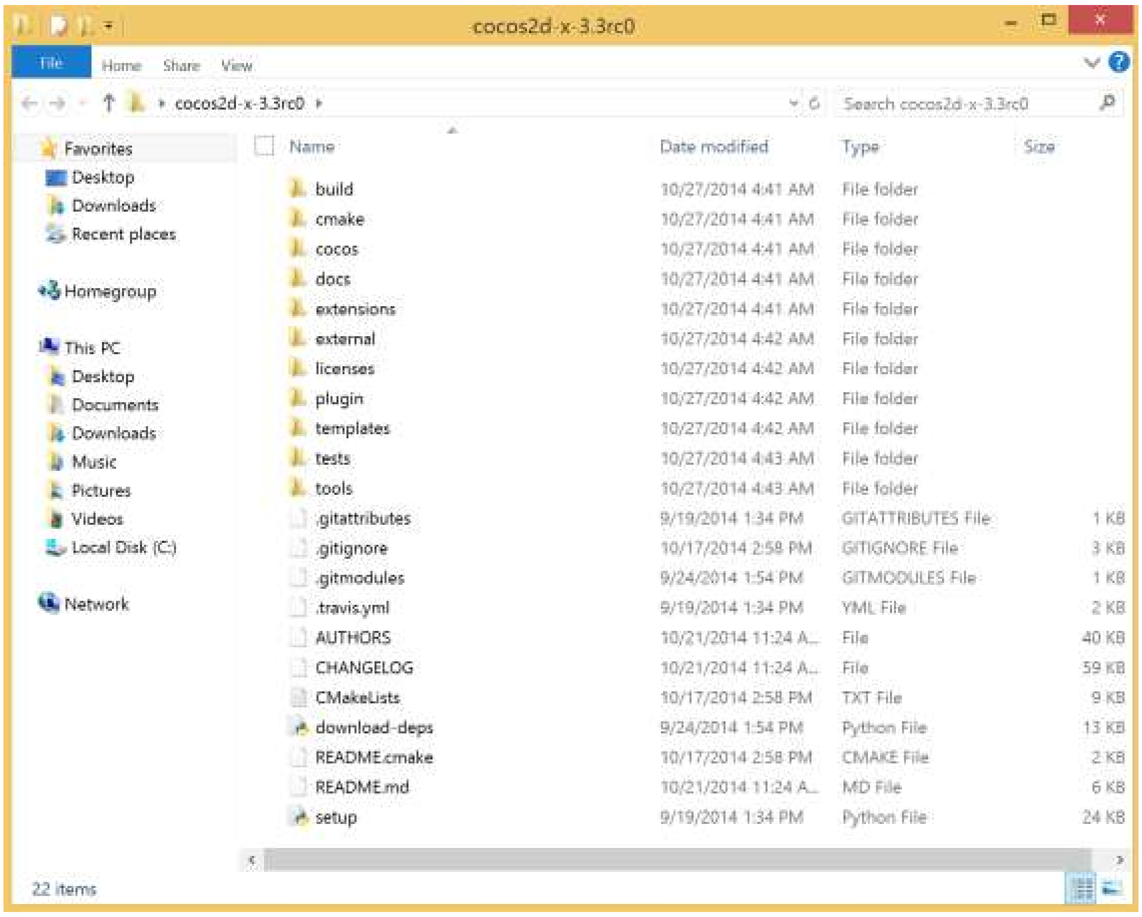
One of the advantages of this framework is to create games that can be deployed on different platforms (Android, iOS, Win32, Windows* Phone, Mac*, Linux*, etc.), keeping the same code base with a few platform-specific adaptations for each one.

Cocos2d-x Console

The cocos2d-console was introduced in the 3.0 version. It is a command line tool that provides some functionality to manage Cocos2d-x or Cocos2d-JS projects like: creation, execution, building, debug, etc.

Creating your first game

1 - Download the latest version of the framework and unzip it in your development environment. In this tutorial the 3.3rc0 version was used, and the framework was unzipped to the desktop (C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0).



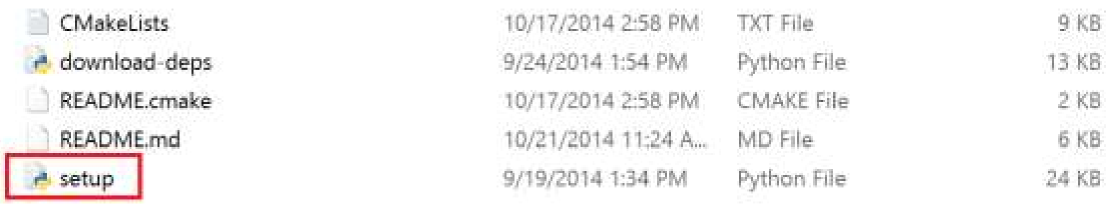
Name	Date modified	Type	Size
build	10/27/2014 4:41 AM	File folder	
cmake	10/27/2014 4:41 AM	File folder	
cocos	10/27/2014 4:41 AM	File folder	
docs	10/27/2014 4:41 AM	File folder	
extensions	10/27/2014 4:41 AM	File folder	
external	10/27/2014 4:42 AM	File folder	
licenses	10/27/2014 4:42 AM	File folder	
plugin	10/27/2014 4:42 AM	File folder	
templates	10/27/2014 4:42 AM	File folder	
tests	10/27/2014 4:43 AM	File folder	
tools	10/27/2014 4:43 AM	File folder	
.gitattributes	9/19/2014 1:34 PM	GITATTRIBUTES File	1 KB
.gitignore	10/17/2014 2:58 PM	GITIGNORE File	3 KB
.gitmodules	9/24/2014 1:54 PM	GITMODULES File	1 KB
.travis.yml	9/19/2014 1:34 PM	YML File	2 KB
AUTHORS	10/21/2014 11:24 A...	File	40 KB
CHANGELOG	10/21/2014 11:24 A...	File	59 KB
CMakeLists	10/17/2014 2:58 PM	TXT File	9 KB
download-deps	9/24/2014 1:54 PM	Python File	13 KB
README.cmake	10/17/2014 2:58 PM	CMAKE File	2 KB
README.md	10/21/2014 11:24 A...	MD File	6 KB
setup	9/19/2014 1:34 PM	Python File	24 KB

Figure 1. *Cocos2d-x version 3.3 RC0 directory structure*

2 – To create a new project in cocos2d-x, use *setup.py* (a Python* script) located in the framework folder to configure all the environment variables to build for Win32 and Android platforms. You will need to download, install and configure the items below before executing setup.py:

- Android* SDK
- Android* NDK
- Apache Ant*

If you haven't installed Python Runtime, download the 2.7.6 version from here: http://www.python.org/download/



CMakeLists	10/17/2014 2:58 PM	TXT File	9 KB
download-deps	9/24/2014 1:54 PM	Python File	13 KB
README.cmake	10/17/2014 2:58 PM	CMAKE File	2 KB
README.md	10/21/2014 11:24 A...	MD File	6 KB
setup	9/19/2014 1:34 PM	Python File	24 KB

Figure 2. *setup.py location*

3 - Open the command prompt (cmd.exe) and execute the following commands:

- Navigate to the script folder (framework folder)
  cd C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0

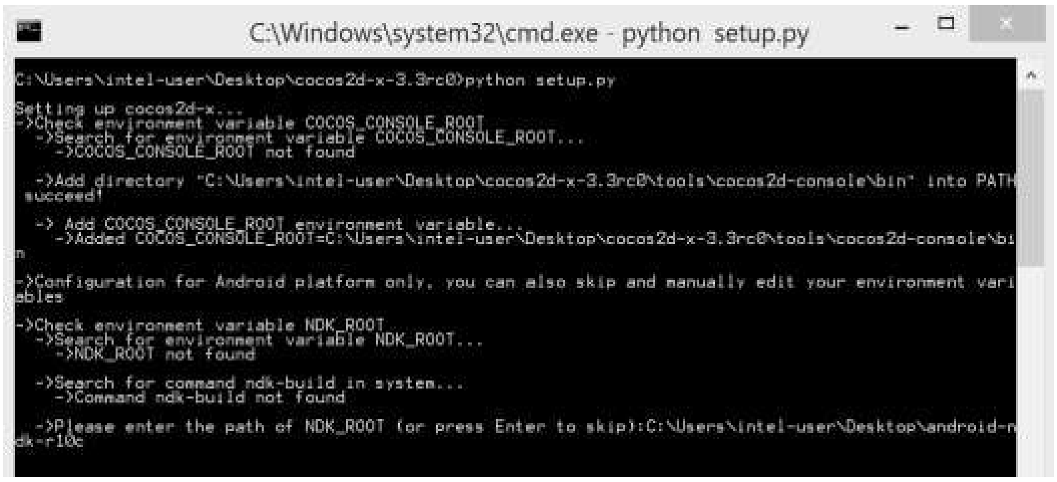
- Run the script *setup.py*:

  python setup.py (or **setup.py** only)

Note: To run the Python command from the command prompt, **add the folder where Python was installed to the environment variable path**.

- The script will request the installation path of Android SDK, Android NDK, and ANT.

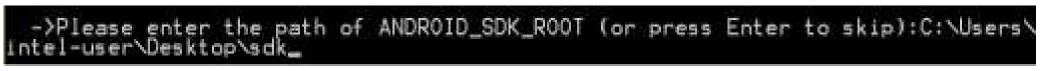
- Android NDK folder path:



```
C:\Windows\system32\cmd.exe - python  setup.py

C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0>python setup.py

Setting up cocos2d-x...
->Check environment variable COCOS_CONSOLE_ROOT
  ->Search for environment variable COCOS_CONSOLE_ROOT...
    ->COCOS_CONSOLE_ROOT not found

  ->Add directory "C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0\tools\cocos2d-console\bin" into PATH
 succeed!

  -> Add COCOS_CONSOLE_ROOT environment variable...
    ->Added COCOS_CONSOLE_ROOT=C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0\tools\cocos2d-console\bi
n

->Configuration for Android platform only, you can also skip and manually edit your environment vari
ables

->Check environment variable NDK_ROOT
  ->Search for environment variable NDK_ROOT...
    ->NDK_ROOT not found

  ->Search for command ndk-build in system...
    ->Command ndk-build not found

  ->Please enter the path of NDK_ROOT (or press Enter to skip):C:\Users\intel-user\Desktop\android-n
dk-r10c
```

Figure 3. *Cocos2d-console requesting NDK folder path*

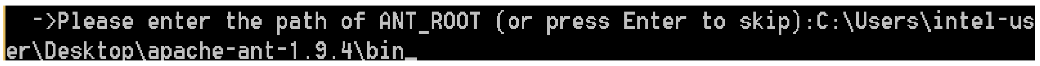
- Android SDK folder path:



```
  ->Please enter the path of ANDROID_SDK_ROOT (or press Enter to skip):C:\Users\
intel-user\Desktop\sdk_
```

Figure 4. *Cocos2d-console requesting SDK folder path*

- Apache ANT bin folder path:



```
  ->Please enter the path of ANT_ROOT (or press Enter to skip):C:\Users\intel-us
er\Desktop\apache-ant-1.9.4\bin_
```

Figure 5. *Cocos2d-console requesting ANT bin folder path*

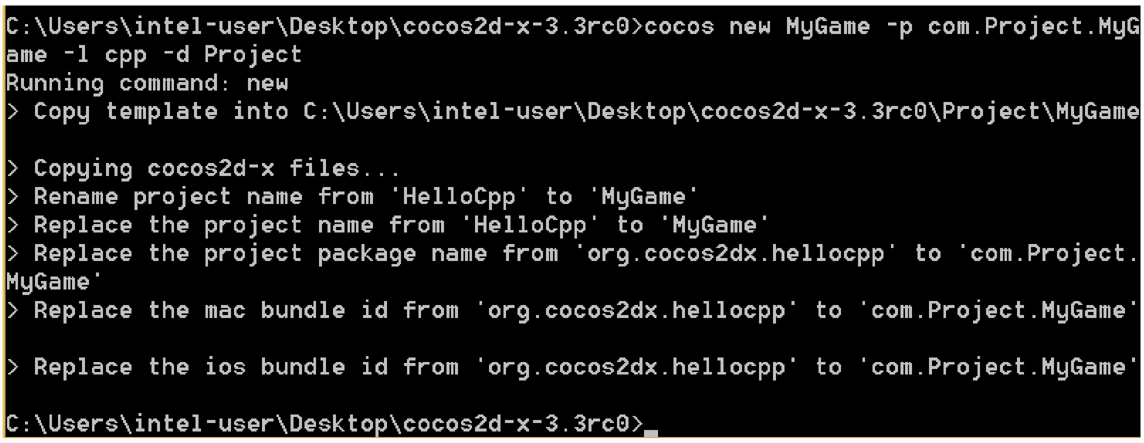
After including the requested paths, reopen the command prompt (cmd.exe). **This action is necessary to use cocos2d-console commands**.

4 – Type cmd.exe to go to the command prompt (cocos2d-console commands can only be issued from here) and open the framework folder again:

  cd C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0

In the step below we will create a new Cocos2d-x project:

```
cocos new MyGame –p com.Project.MyGame –l cpp –d Project
```



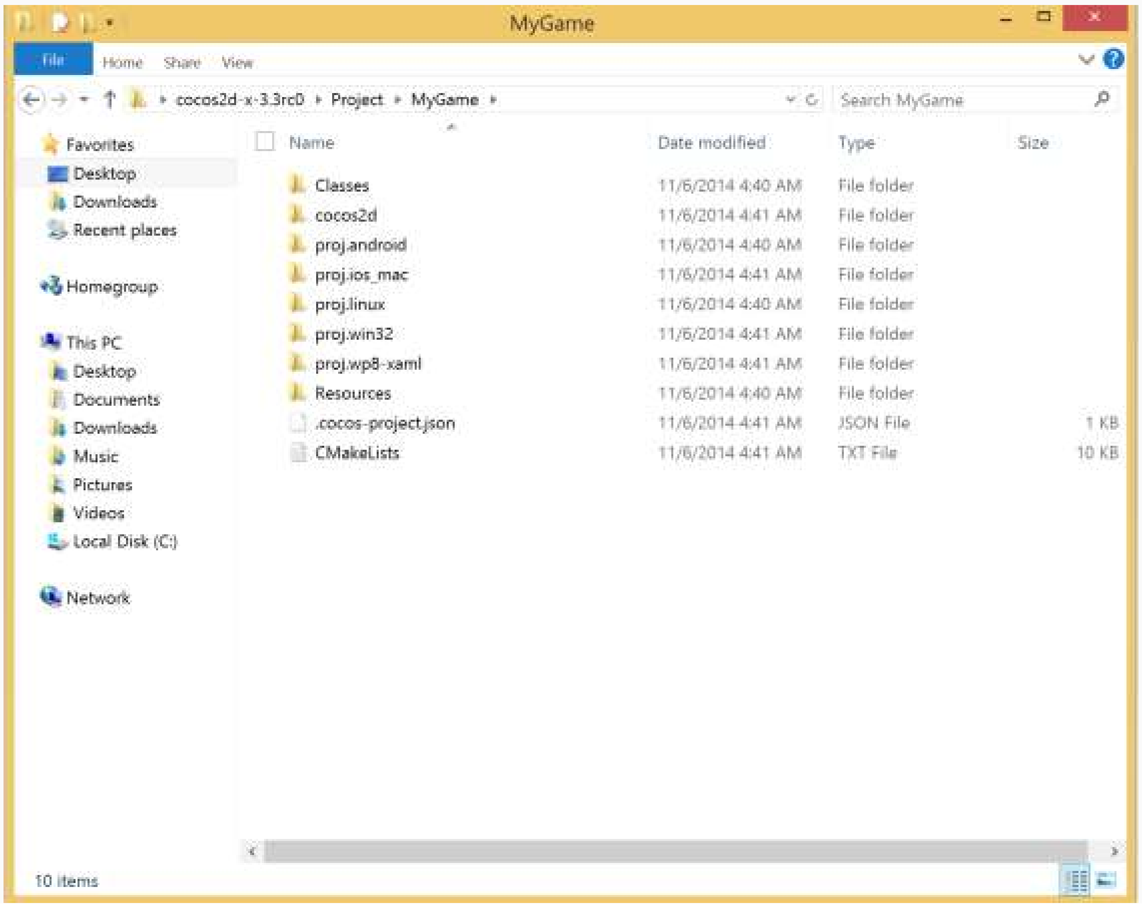
```
C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0>cocos new MyGame -p com.Project.MyG
ame -l cpp -d Project
Running command: new
> Copy template into C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0\Project\MyGame

> Copying cocos2d-x files...
> Rename project name from 'HelloCpp' to 'MyGame'
> Replace the project name from 'HelloCpp' to 'MyGame'
> Replace the project package name from 'org.cocos2dx.hellocpp' to 'com.Project.
MyGame'
> Replace the mac bundle id from 'org.cocos2dx.hellocpp' to 'com.Project.MyGame'

> Replace the ios bundle id from 'org.cocos2dx.hellocpp' to 'com.Project.MyGame'

C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0>_
```

Figure 6. *Created Cocos2d-x project*

Here are quick explanations of the parameters:

- new: creates a new project and must be followed by the name of the project (in the example, MyGame)
- -p: defines package name
- -l: select programming language. The value can be cpp or lua
- -d: directory where the framework will create the project structure

If everything went all right, your project will be created in the Project folder, in the directory where framework was extracted.



Name	Date modified	Type	Size
Classes	11/6/2014 4:40 AM	File folder	
cocos2d	11/6/2014 4:41 AM	File folder	
proj.android	11/6/2014 4:40 AM	File folder	
proj.ios_mac	11/6/2014 4:41 AM	File folder	
proj.linux	11/6/2014 4:40 AM	File folder	
proj.win32	11/6/2014 4:41 AM	File folder	
proj.wp8-xaml	11/6/2014 4:41 AM	File folder	
Resources	11/6/2014 4:40 AM	File folder	
.cocos-project.json	11/6/2014 4:41 AM	JSON File	1 KB
CMakeLists	11/6/2014 4:41 AM	TXT File	10 KB

Figure 7. *MyGame directory structure*

The created project contains the base code of the game (Classes), the resources (images, audio, etc.), and one project for each framework-supported platform.

Building Android* Apps

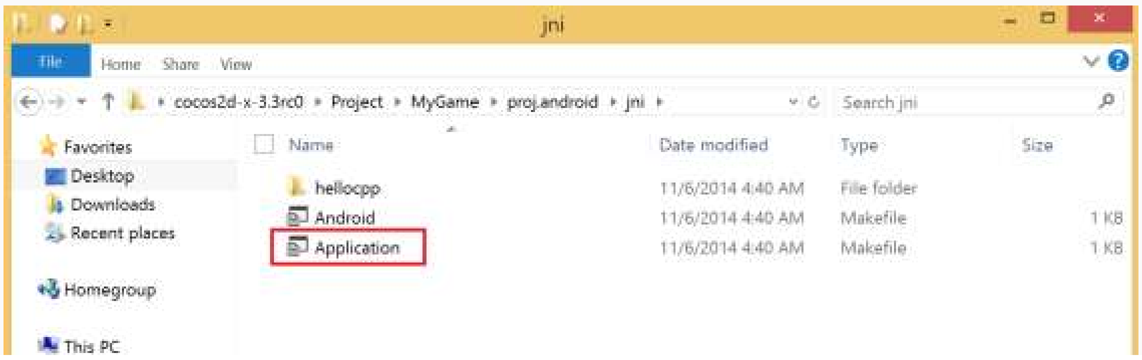
Requirements:

- You need to configure all the environment variables for building Android app games (Android SDK, Android NDK, and ANT). If you have not done this step yet, please see the “Creating your first game” section of this article.
- Java Development Kit (JDK) installed

Note: Cocos2d-console uses the javac command to build for Android, because of this is necessary add JAVA_HOME environment variable (JDK path).

1 – We are going to compile our game for more than one architecture, since the framework doesn’t compile for x86 and armeabi-v7a by default. Edit the Application.mk file located in:

C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0\Project\MyGame\proj.android\jni

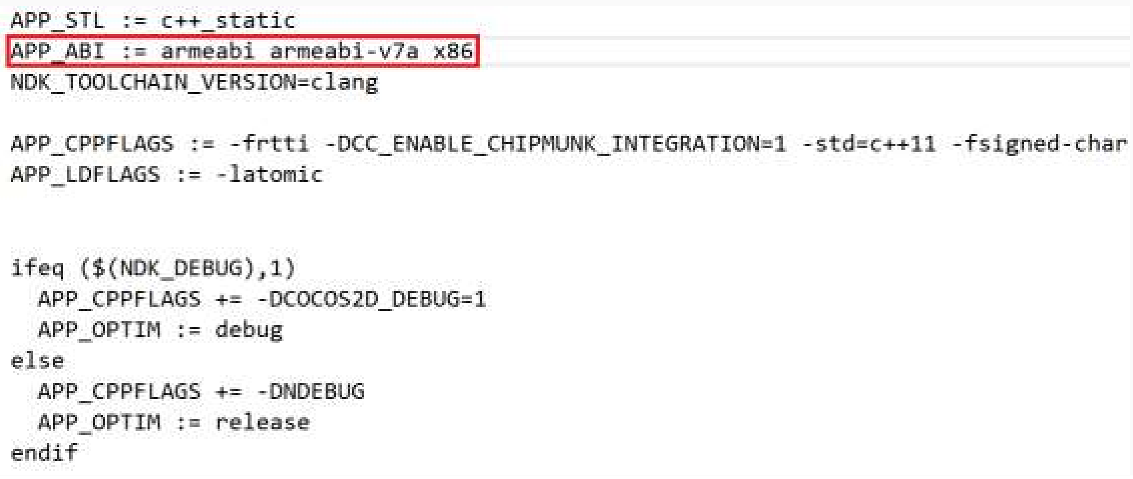


Name	Date modified	Type	Size
hellocpp	11/6/2014 4:40 AM	File folder	
Android	11/6/2014 4:40 AM	Makefile	1 KB
Application	11/6/2014 4:40 AM	Makefile	1 KB

Figure 8. *Application.mk location*

2 – Add the following line to the file:

APP_ABI := armeabi armeabi-v7a x86



```
APP_STL := c++_static
APP_ABI := armeabi armeabi-v7a x86
NDK_TOOLCHAIN_VERSION=clang

APP_CPPFLAGS := -frtti -DCC_ENABLE_CHIPMUNK_INTEGRATION=1 -std=c++11 -fsigned-char
APP_LDFLAGS := -latomic


ifeq ($(NDK_DEBUG),1)
  APP_CPPFLAGS += -DCOCOS2D_DEBUG=1
  APP_OPTIM := debug
else
  APP_CPPFLAGS += -DNDEBUG
  APP_OPTIM := release
endif
```

Figure 9. *Application.mk after APP_ABI line added*

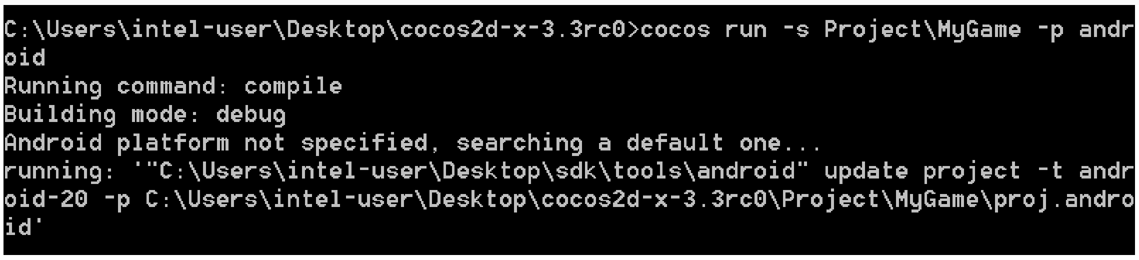
Now that we added our targeted architectures, let's compile our game!

3 – Using the command prompt, go to the framework folder:

cd C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0

4 – Execute the command below to compile and run the game for Android.

cocos run –s Project\MyGame –p android



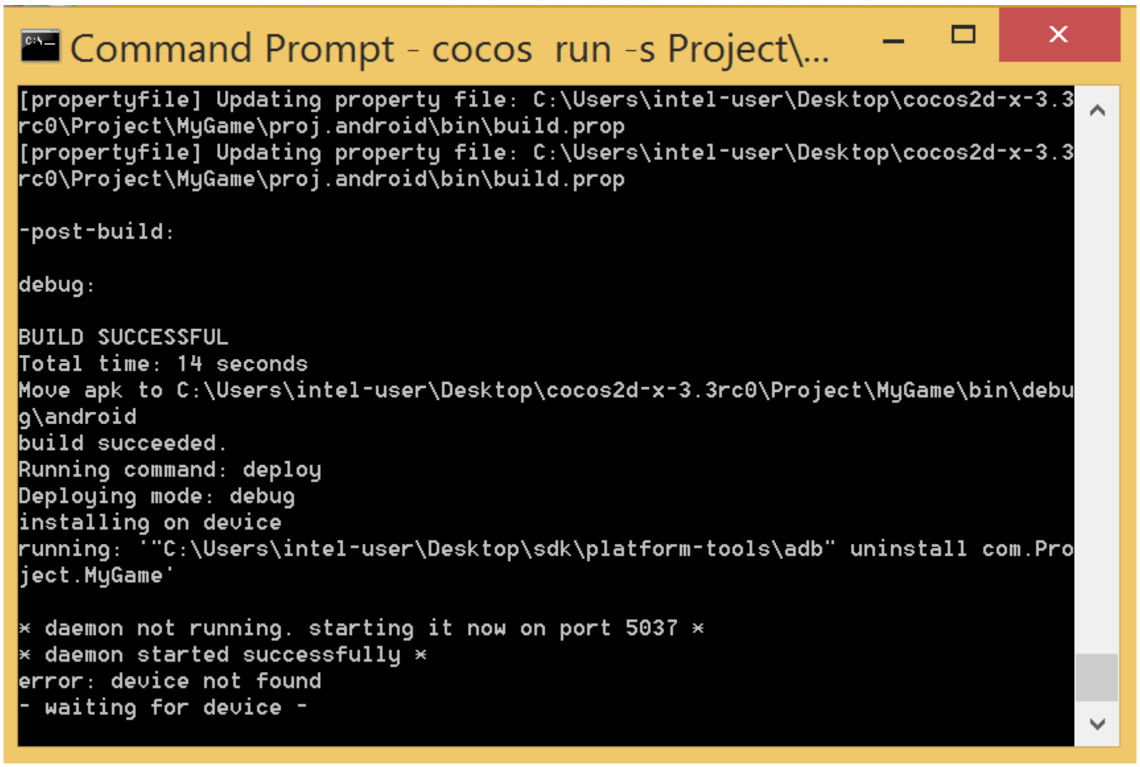
```
C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0>cocos run -s Project\MyGame -p andr
oid
Running command: compile
Building mode: debug
Android platform not specified, searching a default one...
running: '"C:\Users\intel-user\Desktop\sdk\tools\android" update project -t andr
oid-20 -p C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0\Project\MyGame\proj.andro
id'
```

Figure 10. *Executing command to compile and run the game on Android**

- run: compile and run the project
- -s: path to project folder
- -p: selected platform

Note: To compile only, enter: **cocos compile –s Project\MyGame –p android**

If everything worked properly, the *cocos2d-console* command will use adb (if configured in environment variables) to install the APK file in a connected device or initialized emulator. If they aren't available, the command will wait for a device or an emulator to become available, like the picture below:



```
[propertyfile] Updating property file: C:\Users\intel-user\Desktop\cocos2d-x-3.3
rc0\Project\MyGame\proj.android\bin\build.prop
[propertyfile] Updating property file: C:\Users\intel-user\Desktop\cocos2d-x-3.3
rc0\Project\MyGame\proj.android\bin\build.prop

-post-build:

debug:

BUILD SUCCESSFUL
Total time: 14 seconds
Move apk to C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0\Project\MyGame\bin\debu
g\android
build succeeded.
Running command: deploy
Deploying mode: debug
installing on device
running: '"C:\Users\intel-user\Desktop\sdk\platform-tools\adb" uninstall com.Pro
ject.MyGame'

* daemon not running. starting it now on port 5037 *
* daemon started successfully *
error: device not found
- waiting for device -
```

Figure 11. *Command waiting for device or initialized emulator*

If you have initialized an emulator or connected a device, this screen will appear:

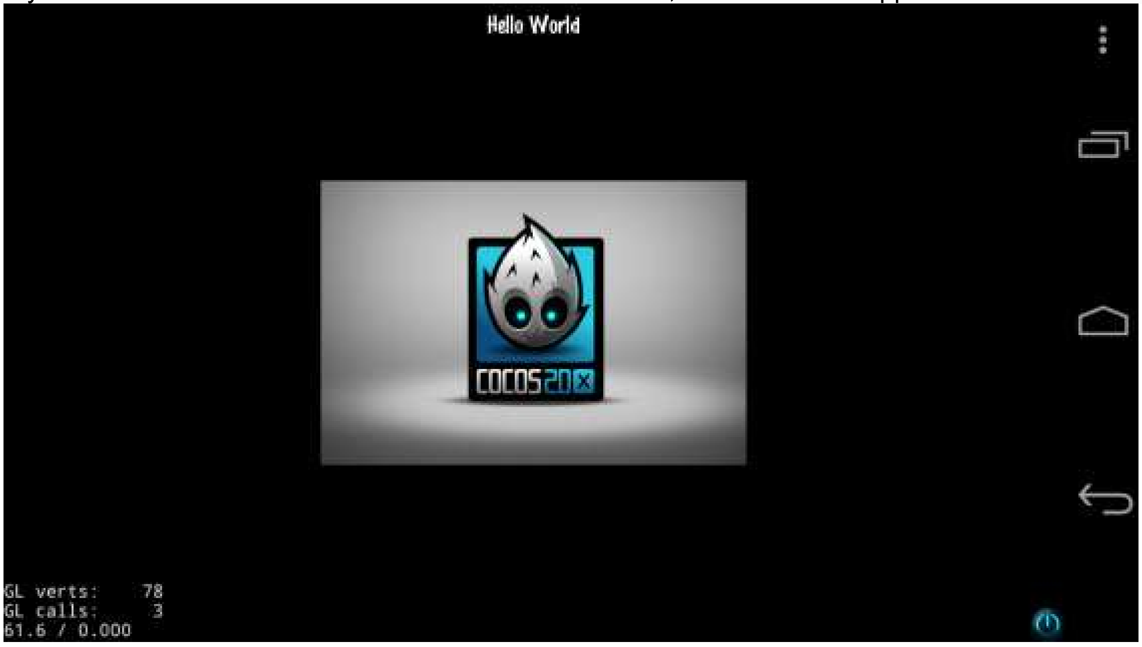


Figure 12. *Game screen on Android* platform*

Building Win32 Apps (for Windows* 7 or Windows* 8 desktop mode)

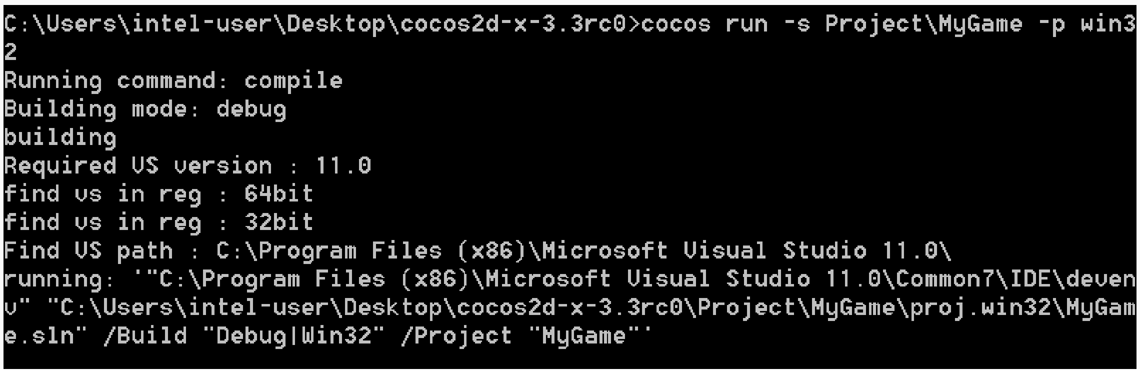
You will need Visual Studio* 2012 or later to build.

1 – Using the command prompt (cmd.exe), go to the folder where framework was extracted:

cd C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0

2 – Execute the following command to compile and run the game on Windows:

cocos run –s Project\MyGame –p win32



```
C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0>cocos run -s Project\MyGame -p win3
2
Running command: compile
Building mode: debug
building
Required VS version : 11.0
find vs in reg : 64bit
find vs in reg : 32bit
Find VS path : C:\Program Files (x86)\Microsoft Visual Studio 11.0\
running: '"C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\IDE\deven
v" "C:\Users\intel-user\Desktop\cocos2d-x-3.3rc0\Project\MyGame\proj.win32\MyGam
e.sln" /Build "Debug|Win32" /Project "MyGame"'
```

Figure 13. *Executing command to compile and run the game on Windows**

Here are quick explanations of the parameters:

- run: compile and run the selected project
- -s: path to project folder
- -p: selected platform

Note: To compile only, use "compile" instead of "run", as follows:
cocos compile –s Project\MyGame –p win32

After executing the run command, you will see the following screen if everything worked ok:

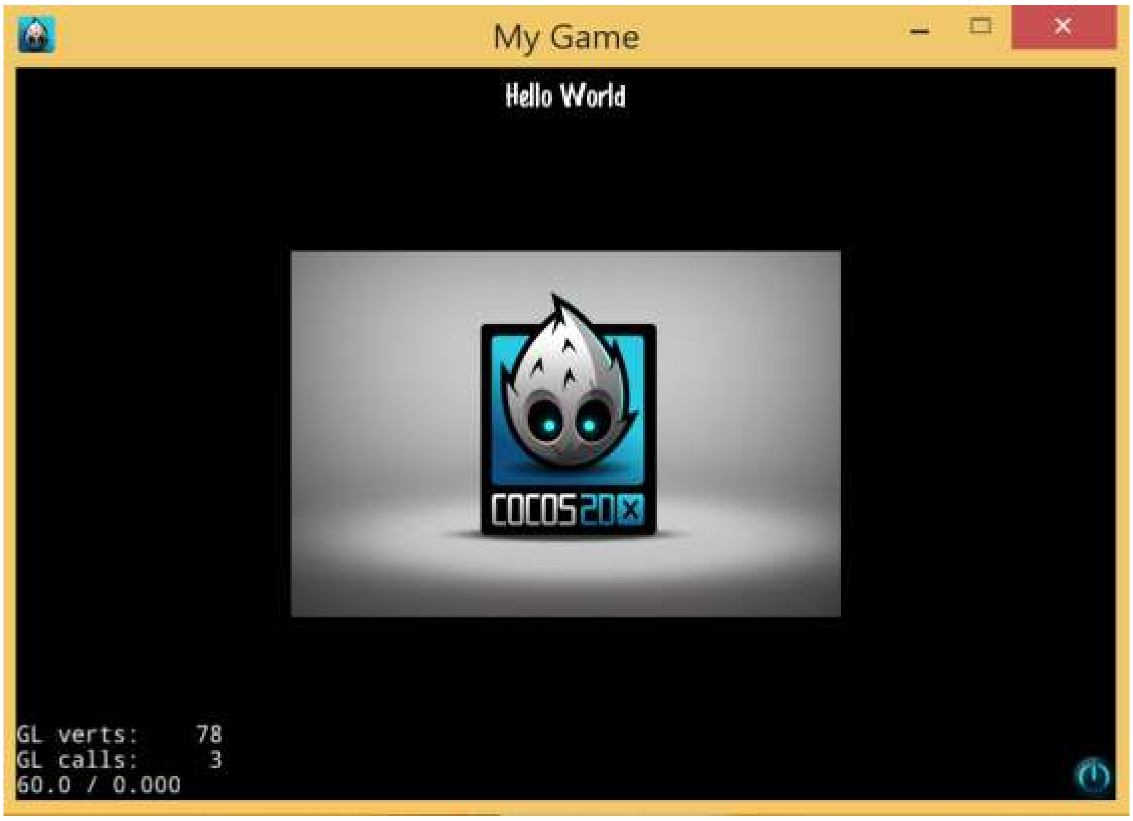
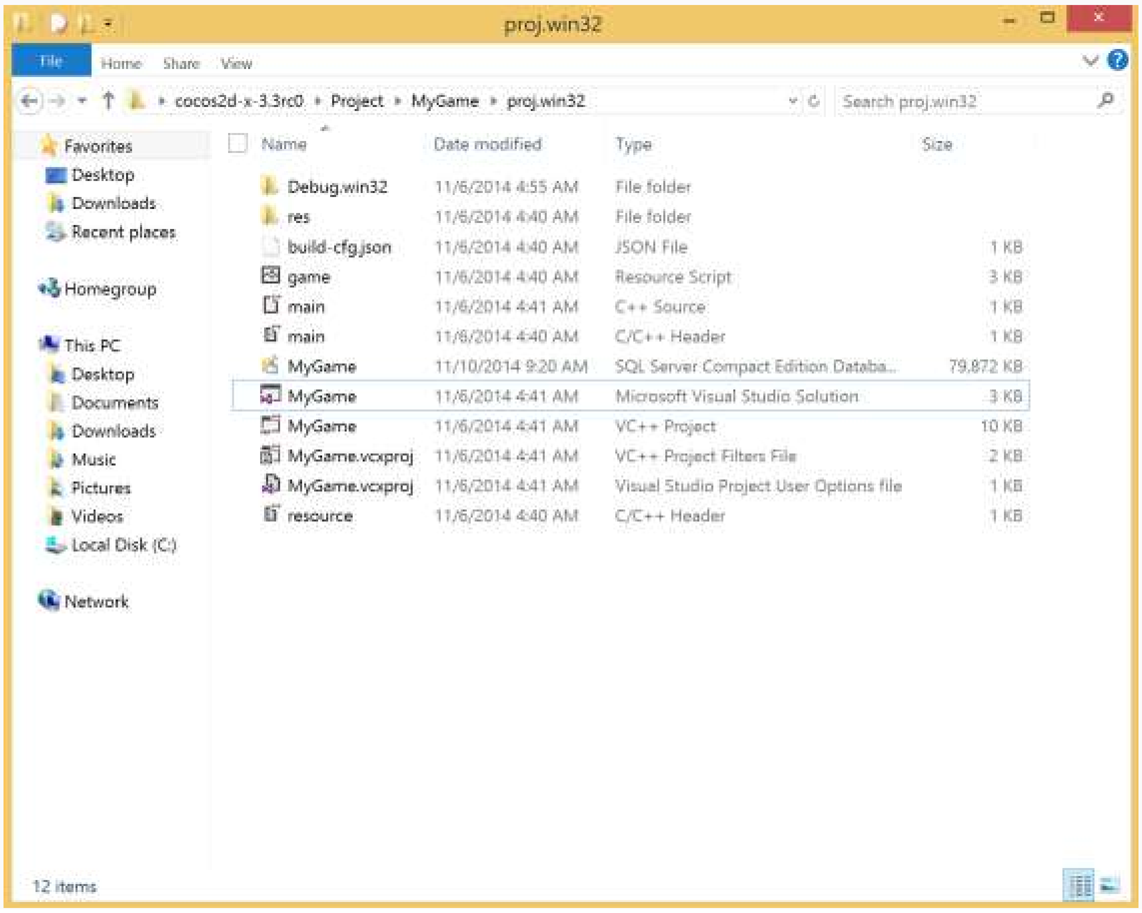


Figure 14. *Game screen in Windows* platform*

It’s possible to use Visual Studio to compile and run the game project:

1 – Inside the Project directory, open with Visual Studio *MyGame.sln* file in the “proj.win32” folder.



Name	Date modified	Type	Size
Debug.win32	11/6/2014 4:55 AM	File folder	
res	11/6/2014 4:40 AM	File folder	
build-cfg.json	11/6/2014 4:40 AM	JSON File	1 KB
game	11/6/2014 4:40 AM	Resource Script	3 KB
main	11/6/2014 4:41 AM	C++ Source	1 KB
main	11/6/2014 4:40 AM	C/C++ Header	1 KB
MyGame	11/10/2014 9:20 AM	SQL Server Compact Edition Databa...	79,872 KB
MyGame	11/6/2014 4:41 AM	Microsoft Visual Studio Solution	3 KB
MyGame	11/6/2014 4:41 AM	VC++ Project	10 KB
MyGame.vcxproj	11/6/2014 4:41 AM	VC++ Project Filters File	2 KB
MyGame.vcxproj	11/6/2014 4:41 AM	Visual Studio Project User Options file	1 KB
resource	11/6/2014 4:40 AM	C/C++ Header	1 KB

Figure 15. *Win32 project directory structure*

2 – To compile the project, press F6 (or Build menu -> Build Solution) and press F5 to execute it (or Debug menu -> Start Debugging). After building and executing, you should see the same screen presented after the console steps.

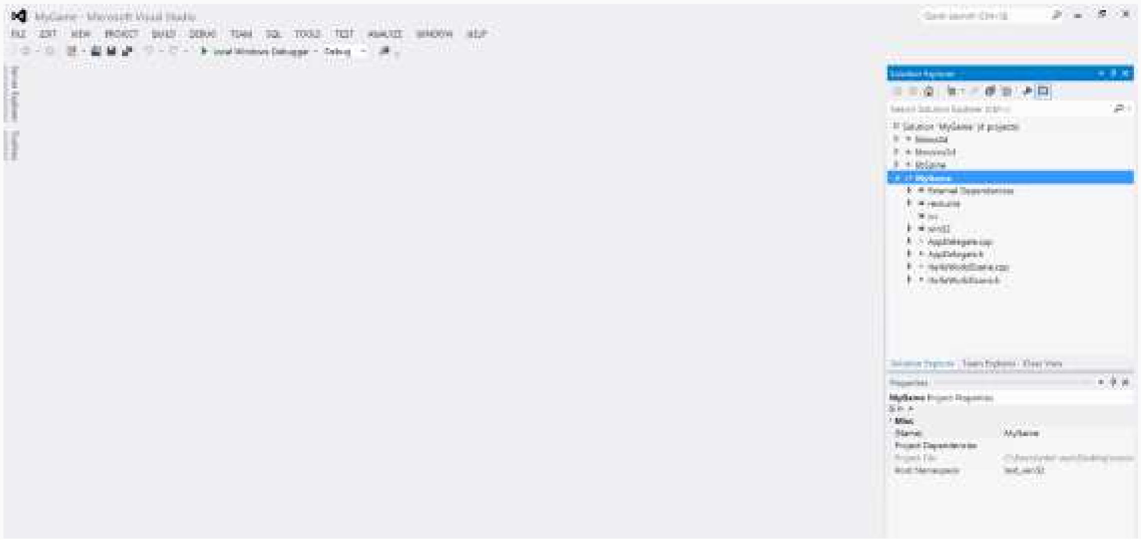


Figure 16. *Win32 project opened in Visual Studio**

Ok, now you know how to create your game and compile it for Android (x86 and ARM*), Windows 7, and Windows 8 (desktop mode)!

References

The source code of Cocos2d-x framework is granted under the MIT License, and it be can be found here.

Cocos2d-x and its documentation: http://www.cocos2d-x.org/

cocos2d-console: http://www.cocos2d-x.org/wiki/Cocos2d-console

Note

At this time, the release version of Cocos2d-x 3.3 has an issue during project creation which doesn’t allow users to create projects (details by clicking here). The problem has been fixed in the latest pre-release, but is not available on the Cocos2d-x latest release yet.