Platform Dependent Compilation

Unity includes a feature named "Platform Dependent Compilation". This consists of some preprocessor directives that let you *partition* your scripts to compile and execute a section of code exclusively for one of the supported platforms.

Furthermore, you can run this code within the Editor, so you can compile the code specifically for your mobile/console and test it in the Editor!

Platform Defines

The platform defines that Unity supports for your scripts are:

UNITY_EDITOR	Define for calling Unity Editor scripts from your game code.
UNITY_STANDALONE_OSX	Platform define for compiling/executing code specifically for Mac OS (This includes Universal, PPC and Intel architectures).
UNITY_DASHBOARD_WIDGET	Platform define when creating code for Mac OS dashboard widgets.
UNITY_STANDALONE_WIN	Use this when you want to compile/execute code for Windows stand alone applications.
UNITY_WEBPLAYER	Platform define for web player content (this includes Windows and Mac Web player executables).
UNITY_WII	Platform define for compiling/executing code for the Wii console.
UNITY_IPHONE	Platform define for compiling/executing code for the iPhone platform.
UNITY_ANDROID	Platform define for the Android platform.
UNITY_PS3	Platform define for running PlayStation 3 code.
UNITY_XBOX360	Platform define for executing Xbox 360 code.
UNITY_NACL	Platform define when compiling code for Google native client (this will be set additionally to **UNITY_WEBPLAYER**).
UNITY_FLASH	Platform define when compiling code for Adobe Flash.

Note: These defines were introduced at version 3.0.

Also you can compile code selectively depending on the version of the engine you are working on. Currently the supported ones are:

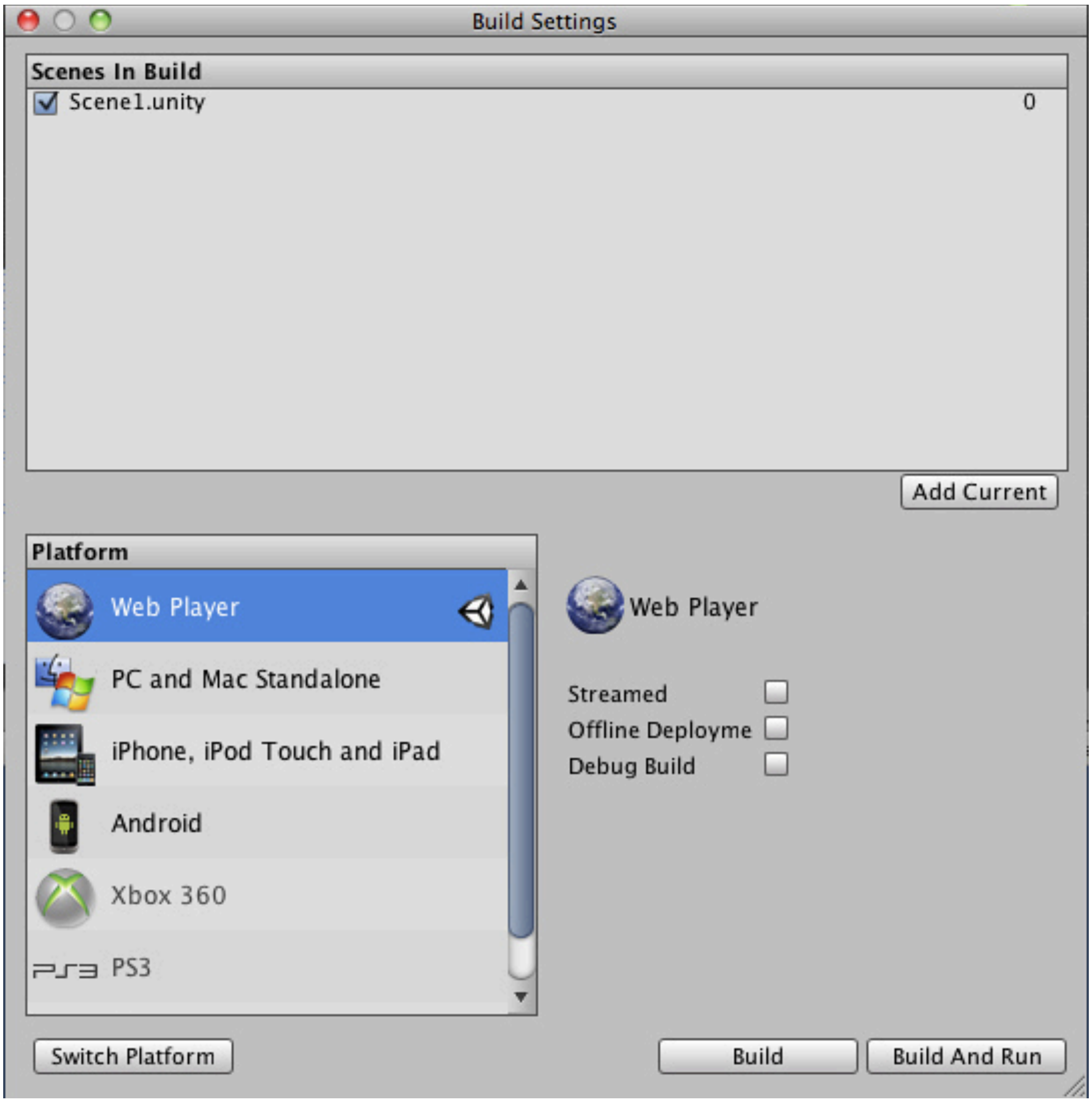
UNITY_2_6	Platform define for the major version of Unity 2.6.
UNITY_2_6_1	Platform define for specific version 1 from the major release 2.6.
UNITY_3_0	Platform define for the major version of Unity 3.0.
UNITY_3_0_0	Platform define for the specific version 0 of Unity 3.0.
UNITY_3_1	Platform define for major version of Unity 3.1.
UNITY_3_2	Platform define for major version of Unity 3.2.
UNITY_3_3	Platform define for major version of Unity 3.3.
UNITY_3_4	Platform define for major version of Unity 3.4.
UNITY_3_5	Platform define for major version of Unity 3.5.

Note: For versions before 2.6.0 there are no platform defines as this feature was first introduced in that version.

Testing precompiled code.

We are going to show a small example of how to use the precompiled code. This will simply print a message that depends on the platform you have selected to build your target.

First of all, select the platform you want to test your code against by clicking on **File -> Build Settings**. This will bring the build settings window to select your target platform.



Scene1.unity 0

Web Player
PC and Mac Standalone
iPhone, iPod Touch and iPad
Android
Xbox 360
PS3

Web Player

Streamed ☐
Offline Deployme ☐
Debug Build ☐

Build Settings window with the WebPlayer Selected as Target platform.

Select the platform you want to test your precompiled code against and press the **Switch Editor** button to tell Unity which platform you are targeting.

Create a script and copy/paste this code:

JavaScript Example:

```
function Awake() {
  #if UNITY_EDITOR
    Debug.Log("Unity Editor");
  #endif

  #if UNITY_IPHONE
    Debug.Log("Iphone");
  #endif

  #if UNITY_STANDALONE_OSX
    Debug.Log("Stand Alone OSX");
  #endif

  #if UNITY_STANDALONE_WIN
    Debug.Log("Stand Alone Windows");
  #endif
}
```

C# Example:

```
using UnityEngine;
using System.Collections;

public class PlatformDefines : MonoBehaviour {
  void Start () {

    #if UNITY_EDITOR
      Debug.Log("Unity Editor");
    #endif

    #if UNITY_IPHONE
      Debug.Log("Iphone");
    #endif

    #if UNITY_STANDALONE_OSX
        Debug.Log("Stand Alone OSX");
    #endif

    #if UNITY_STANDALONE_WIN
      Debug.Log("Stand Alone Windows");
    #endif

  }
}
```

Boo Example:

```
import UnityEngine

class PlatformDefines (MonoBehaviour):

        def Start ():
                ifdef UNITY_EDITOR:
                        Debug.Log("Unity Editor")

                ifdef UNITY_IPHONE:
                        Debug.Log("IPhone")

                ifdef UNITY_STANDALONE_OSX:
                        Debug.Log("Stand Alone OSX")

                ifdef not UNITY_IPHONE:
                        Debug.Log("not an iPhone")
```

Then, depending on which platform you selected, one of the messages will get printed on the Unity console when you press play.

In addition to the basic *#if* compiler directive, you can also use a multiway test in C# and JavaScript:-

```
#if UNITY_EDITOR
    Debug.Log("Unity Editor");
#elif UNITY_IPHONE
    Debug.Log("Unity iPhone");
#else
    Debug.Log("Any other platform");
#endif
```

However, Boo currently supports only the *ifdef* directive.

Page last updated: 2012-06-27