AIM: To learn:

* + to display TEXT on screen using a Debug.Log() statement

**Preparation**

* Create a new, empty Unity project
* Create folder “Scenes”, to store your scenes
  + Save into this folder the current (empty) scene as “scene1”
* Create folder “Scripts”
  + You’ll be storing your program files in here !

**Notes**

* When referring to C# files, the term “CSharp” or “Csharp” may also be used
  + The terms mean the same thing – the C# computer programming language
* Unity will add (and hide from you) the “.cs” extension
  + So although in the Unity Project panel you’ll see “HelloWorld” or “NewBehaviourScript”, Unity has actually created the file “HelloWorld.cs” or “NewBehaviourScript.cs”
* Methods (functions) in Csharp are named in UpperCamelCase, just like class names
  + Just get used to it – every language has its own conventions and this is one for Csharp
  + Note, you could name your methods starting with a lower case letter, but then you’d have to mix that with Unity methods that start upper case, and also you’d be programming in a different way to other Csharp programmers
  + So just learn the Csharp way from day one
    - Class names are UpperCamelCase, e.g.
      * class HelloWorld
    - Method names are UpperCamelCase, e.g.
      * Start()
    - Variable names are lowerCamelCase (just like Java), e.g.
      * int score = 0;
* Csharp is very similar to Java
  + One difference is the use of the colon : rather than the word “extends” to indicate a sub-class
  + So in Csharp we write:

class HelloWorld : MonoBehaviour

* + in Java we would have written:

class HelloWorld extends MonoBehaviour

* Mono is the Microsoft Open standard .net compiler
  + It allows several languages to all compile into .net applications, including Csharp, JScript (Microsoft’s version of JavaScript), and mono (a scripting language, a bit like Python)

# Create a “scripts” folder in Project panel

Do the following:

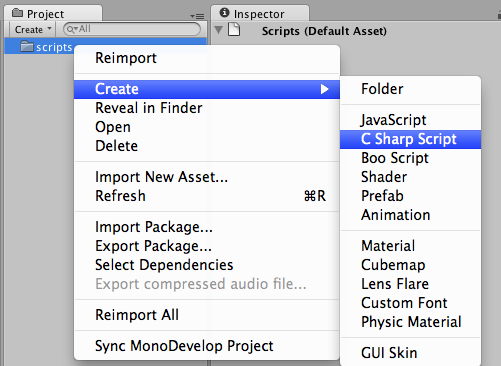
* Create a new folder in the Project panel & and rename it “Scripts”

macbookpro:Users:matt_smith:Dropbox:TEACHING:immedia_2011:10_units:02_hello_world_print:images:reanmed_folder_to_scripts.png

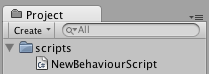
# Create a new CSharp program file “HelloWorld”

Do the following:

* Create a new folder in the Project panel
  + With the mouse over your “scripts” folder, Right mouse click and select menu:
    - Create ⏐C Sharp Script



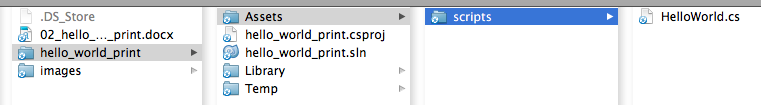
* You should now see a new file named “NewBehaviourScript” (and its graphical document icon should have a little “C#” logo)



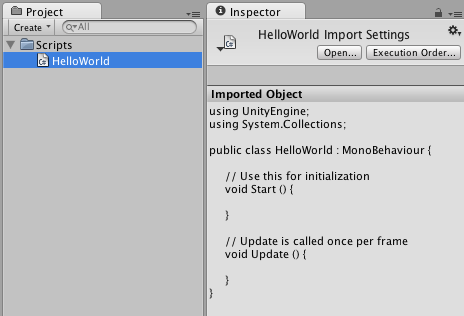
* Select (long mouse click) this file and rename it “HelloWorld” (then press Return – otherwise the name may revert back to NewBehaviourScript)



* You can now see the folder and file you have created in the computer’s file system
  + NOTE: Unity has added the “.cs” file extension for you, even though this file extension is hidden in the Unity Project panel

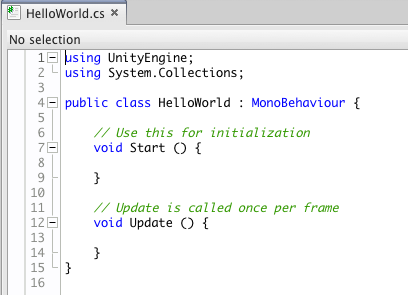


* You can now see a preview of the contents of the program file when the file is selected in the Project panel – the code preview is displayed in the Inspector panel



# Edit your HelloWorld Class

Do the following:

* Open up your HelloWorld.cs file in the MonoDevelop program editor
  + Just double click the file in the Project panel  
    or   
    click the “Edit…” button in the Inspector panel when the file is selected
  + the MonoDevelop application should automatically start, and open your HelloWorld.cs program file after a few seconds …
  + 

1. Remove the **Update() method**
   * + We’ll learn about this soon …
2. Add to the body of the **Start() method** this console text display statement:

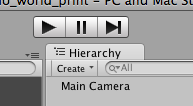
**Debug.Log(“Hello World!”);**

***HINT:  
See the Appendix for a full listing of how your HelloWorld.cs program should look…***

# Save and run your program

Do the following:

* Save your edited file in MonoDevelop
* Change back to the Unity application, and click the run button
  + This is the standard triangle pointing to the right “playhead” controller to run/play something (the one on the LEFT of the group of 3 buttons!)
  + In this case, it runs your application from the beginning



Nothing happened !!!!!!!

What went wrong?

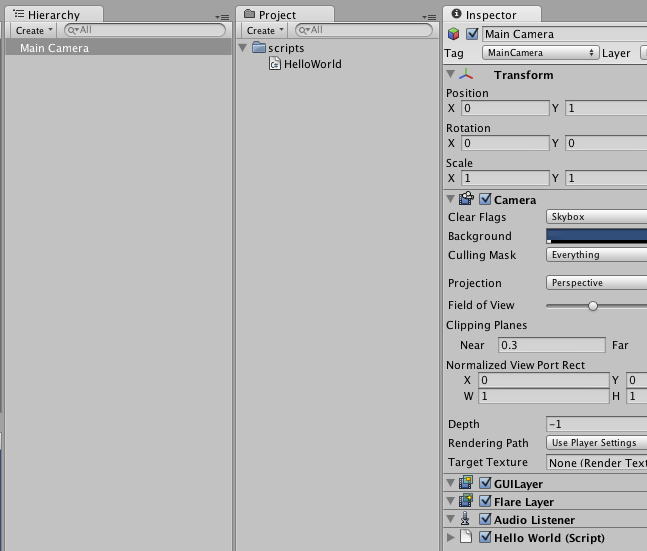
In Unity, **scripts must be attached to Game Objects**, since it is the events that Game Objects receive (like Start() and Update()) that allow Unity to decide which methods of each script to execute at any point in time.

Our code was written in a Start() method, so once the script has been added as a “component” of a game object, when the Unity application is run, a Start() message will be sent to every game object in the scene, and so our Start() method will be executed

# Attach the script to a game object

Do the following:

* In the Unity application, drag your script from the Project panel over the Main Camera object in the Hierarchy
* To confirm you have added the script to the Main Camera, select the Main Camera in the Hierarchy panel, and you should now see the script listed as the last component in the Inspector:



# Run program and see your Hello World! Message displayed

You should now see your “Hello World!” message at the bottom of the Game panel when you run your program:



**Congratulations!**

**You have created your first C# program!**

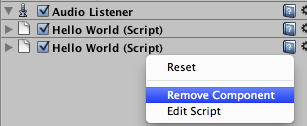
# FAQ: How do I remove a component from a Game Object

You might have this problem – the script has been added TWICE to the Main Camera:



If you have added the wrong script to an object, or added the same script twice, you can remove a component from a Game Object as follows:

* Right mouse click with the mouse pointer over the title bar of the component in the Inspector, and select menu:
  + Remove Component



# FAQ: I have a code error – what do I do?

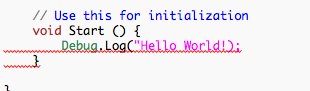
You might have this problem – Unity will not run, displaying a red message that there are errors in your script:



Double click on the RED error message at the bottom of the Game panel, and you will be taken to the offending line of code in MonoDevelop

Then follow standard debugging procedure, e.g.:

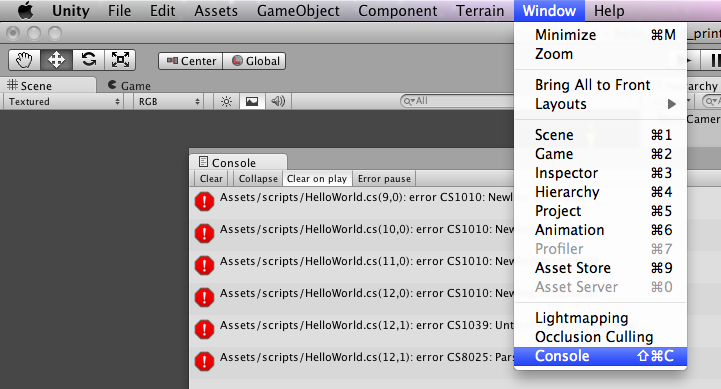
* READ the error message
* Look at the line relating to the error
* Look at the line immediately BEFORE the error line
* Look for SIMPLE things first
  + Is there a missing semi-colon?
  + Is there a missing double quote mark?
  + Is there a missing parenthesis (round bracket) or brace (curly bracket)?
  + Has a method name been spelled wrong?
  + Are the number, order, and type of method arguments correct?
  + About 50% of errors are one of the above …
* In this example, the double quotes were not closed after the World! Text:



# FAQ: How can I see a list of errors (if there is more than 1)?

To see a full list of errors display the Console window. Choose menu:

* Window ⏐Console



**APPENDIX :: Program listing :: HelloWorld.cs**

using UnityEngine;

using System.Collections;

public class HelloWorld : MonoBehaviour {

// Use this for initialization

void Start ()

{

Debug.Log("Hello World!");

}

}