**Unity Certification Preparation:**

**C# Programming**

**Orlando Unity3d Development Meetup**

**Date: 3 June 2018**

**All Copyright Kovert Studios © 2018**

**Contents**

[**2.0** **So, WHAT IS SCRIPTING?** 3](#_Toc515784303)

[**2.1** **DOUBLE CHECK your IDE SETTINGS** 3](#_Toc515784309)

[**2.2** **How to CREATE a SCRIPT ?** 3](#_Toc515784310)

[**2.3** **The DEFAULT CODE : NAMESPACES, INHERITANCE** 3](#_Toc515784311)

[**2.4** **The CLASS DECLARATION STATEMENT, Introduction to INHERITANCE principle** 4](#_Toc515784312)

# **So, WHAT IS SCRIPTING?**

# Reference to Unity, you will often hear people talking about scripting, what they really mean is computer programming => In other words, giving “instructions” to the game objects in the scene to do something.

# For example, if we create an object in the scene. Naturally we want to be able to interact and control that object. E.g. Let's say move the object from point A in the scene to a point B. To do this in unity we write a “script” (**the source code**) then attach it to the object in the scene window that we want to move.

# *The strict technical definition of scripting means =>* ***the program does not need to be “complied” before running it. Rather your instructions/the program are interpreted at runtime****. In other words, you do not have the “hassle” of creating an executable .exe file before experiencing it.*

# *Whereas compiling means the code is directly executed directly by the computer's CPU. i.e. the* ***source code you have written in your IDE is converted to the CPU's native assembly language before it is run (before runtime)****. Whereas the source code for interpreted languages however must be translated at run-time (i.e. the source code is processed at the same time your program is being run) from any format to CPU machine instructions.*

# Effective from 01 Apr 2017 Unity now only supports 1 programming language for scripting: C#. They have deprecated JavaScript (UnityScript).

# **DOUBLE CHECK your IDE SETTINGS**

Now before we create the 1st script. First, double check your **1) IDE settin**gs are set up Correctly **2)Asset folders** **(\_MATERIALS, \_PREFABS, \_SCENES, \_SCRIPTS**) have been created. Navigate to the **Ribbon/toolbar Top left.**

* **CLICK** on **EDIT > PREFERENCES > EXTERNAL TOOLS > EXTERNAL SCRIPT EDITOR**.
* In the **EXTERNAL SCRIPT EDITOR** drop down list. **CHECK** you have **(or SELECT)** **VS2015/2017** as your default IDE.

# **How to CREATE a SCRIPT ?**

Ok In unity there are **2 key ways** to create a script 1)From inside the **project panel** via the asset folder, or 2)Adding a script as a **component** to the object inside of the inspector.

* **NAVIGATE** to **the PROJECT Panel tab > Right Click CREATE > SELECT C# SCRIPT**….you should see a new script icon pop up called NewBehaviourScript.
* **RENAME** NewBehaviourScript > **HIGHLIGHT** it so you see a blue selection bar > **LEFT CLICK** at the End of Name OR PRESS F2 > then **LEFT CLICK** again (DON'T DOUBLE CLICK) ...**WAIT** it 1-2 seconds > then **RETYPE,RENAME** etc. -> NewScript1
* The 2nd WAY (to create a script) is to **NAVIGATE** to the HIERARCHY tab > **SELECT** the gameObject > **NAVIGATE** to the (Objects) INSPECTOR > **ADD COMPONENT** > NEW SCRIPT > **RENAME** (Change NewBehaviourScript --> NewScript1, **CHECK LANGUAGE** -> C# > CREATE & ADD

If you double click on the "NewScript1" > It should open into VISUAL STUDIO > And you should see some default code.

# **The DEFAULT CODE : NAMESPACES, INHERITANCE**

It is very important to understand what the default code is and does, before charging headlong into your game, because that will help narrow down what code you need. Another point. Whenever you create a new script in unity. It automatically creates a new class with the same name. At the beginning of your code. The very first lines you see are:

**using System.Collections;**

**using System.Collections.Generic;**

**using UnityEngine;**

…In unity C# these are called **namespaces** or **code libraries**. Anytime you see a statement with the command "using" --> it means you can access the **commands, classes, methods & objects** from those libraries for use in your code. ***A namespace is used to avoid naming conflicts***. When you start to add libraries from other programmers it is highly likely you are going to use classes, methods & objects with the same names. Therefore, we create a **namespace** as a unique group name identifier for the collection of classes, methods & objects that it contains.

# **The CLASS DECLARATION STATEMENT, Introduction to INHERITANCE principle**

In **OOP** there is a principle called inheritance. Which means that the class NewScript "inherits" all the **commands, classes, methods and data** from the Monobehaviour class. The **class declaration statement** is as follows:

**public class NewScript1 : MonoBehaviour**

{

//Ditto

}

The **MonoBehaviour class** = **is aka the Parent (Base) class to derive your C# script** -> It contains all the **commands, methods, classes** and components from the unity game engine to use in your C# script. MonoBehaviour is an **API.** For example, using JavaScript in unity every script **automatically or implicitly** derives the code from the MonoBehaviour class. However, when using C# or Boo you must **explicitly** derive your code from the MonoBehaviour class. MonoBehaviour is also part of the UnityEngine namespace (i.e. using UnityEngine). So, if we delete the UnityEngine namespace > We will get an error. When you create a new script; by default, UNITY creates 2 METHODS or functions (between the two curly braces inside the body of the class) called **void Start()** and **void Update()** aka **The Default Execution Order** **functions.** If you do not inherit from MonoBehaviour you “WILL NOT” be able to use these two functions as they are derived from the Monobehaviour class. void start() function is called only once for the duration of your script. Also, void update() is called repeatedly at each frame. OK so how do we use the code. Inside the start function type Debug.Log.

***Debug.Log*** *= Echo outputs or prints the string contents as a message to the Unity Console Window.*

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class NewScript1 : MonoBehaviour

{

// Use this for initialization

**void Start ()**

**{**

**Debug.Log("Start of program");**

**}**

// Update is called once per frame

**void Update ()**

**{**

**Debug.Log("Frame updated !");**

**}**

}//End of HelloScript