**GAMEDEVEREJECTS:**

**C# Programming for Unity**

**VARIABLES & DATATYPES**

**Date: 18-Apr-22**

**All Copyright GAMEDEVREJECTS©2022**

**Contents**

[**3.0** **Variables & Data Types** 3](#_Toc100916708)

[**3.1** **Difference between Public v Private Keywords** 3](#_Toc100916709)

[**3.2** **Primitive Data Types** 3](#_Toc100916710)

[**3.3** **Non-Primitive Data Types** 3](#_Toc100916711)

[**3.4** **Create, declare a Variable** 3](#_Toc100916712)

[**Glossary** 6](#_Toc100916713)

[**Resources** 7](#_Toc100916714)

# **Variables & Data Types**

Variables & Data Types are a definite need-to-know if you want to write scripts or computer programs in unity. So, what is a variable ? **A variable is a space inside the computer’s memory to store values**. It is called a variable because the information stored in the memory’s location can be changed or varied when the program is running.

Now depending on the type of value we want to store as a variable (i.e. is it a whole or decimal number ?), determines **1)the data type**, and **2)the amount of memory space** we need to allocate to store that value. So for example:

* **a whole number** => use the **integer type,**
* **a decimal number** => use **float or double** ,
* **a word or text** => we use a **string**,
* a single **Character** => we use **char,**

In Unity a **“Variable”** => is aka a **“field declaration statement”.** **A field is a variable that exists inside of a class…but outside of the scope of a method.**

A **method** **is a function that exists inside of a class**. Now in unity (when coding in C#) when declaring a variable, you are always required to “**explicitly declare”** or state what the data type is for your value before you can use it in your program. Now you might think this is a real pain to constantly have to state the data types; unlike other languages like python for example which (automatically) implicitly recognizes the different data types you enter, but believe you me - this will save you so much "heartache" down the line…when you need to debug or review your code, because you will be able to speed read through your code a lot more quickly to find the error.

# **3.1 Difference between Public v Private Keywords**

Public and Private are known as “member access modifiers”. For now, **public** variables can define as variables that can accessed/retrieved (or called) **from “outside” of the method (or function) or class where it was created or resides.**  Whereas **private** indicates that variables **can only be accessed by objects inside of the same method or class**.

# **3.2 Primitive Data Types**

At this point, it is worth discussing what are known as the **primitive data types** in C# for Unity. They are called “primitive” because they are **the main built-in types, that come with the compiler** and can be used to build other data types. The most famous primitive data types are: **int, float, double, string, object, short, char, bool.**

In C#, primitive data types are **objects**, it means when you write the following code, variable foo is an Object. int foo = 10. If you are coming from other languages like Java, C++, JavaScript, …etc. this might sound confusing. However, as the story goes, data types like int, short, char, float, double & bool are structs. So, when you create a variable, you are creating an instance from its structure.

# **3.3 Non-Primitive Data Types**

Non-Primitive data types **are data types created by the programmer and not defined by C#.** They are sometimes called **reference variables** or **object references** since they reference a memory location to store data.

Another important point to remember about modifying code inside of Unity. Is that any value you update or change in Game Play mode, will automatically revert to their default value/s when you exit the gameplay view, back into edit mode.

# **3.4 Create, declare a Variable**

So, let's create, declare a variable to summarize everything we’ve learned so far. For example, suppose we want to store the values/texts inside of a bunch of variables and then display them to the console. First we create a script and attach it to an Empty Game Object as follows:

(NavigateTo).SCENE -- (IN).HIERARCHY -- / (Right Click).**Create Empty** /

(IN).INSPECTOR – Transform – (3 dots) => / **(Select).Reset** /

(IN).PROJECT -- Assets -- \_SCRIPTS -- (Right Click).Create.C# Script => / **NewBehaviourScript** /

(IN).\_SCRIPTS -- (Select).**NewBehaviourScript**.(DragandDrop) – HIERARCHY => / (DropOnto).**GameObject** /

(IN).\_SCRIPTS => / (Select).**NewBehaviourScript**.(Double Left Click) – VSCODE /

There are at least 4 to 5 key steps as follows:

* **Decide to make it public or private (i.e. visible to the other classes in our script)**
* **Declare the data type is it an int or a float. E.g. If it is a whole number then assign it to an int. Or if it has a decimal point then make it a float.**
* **Give it a name. Call it anything you want if it does not start with 1)a number or 2)a special character, let’s just call it num**
* **Define, assign the value to the declared number**
* **Add a semi-colon. The semi-colon is like a full-stop or period. It tells the compiler it’s reached the end of the program instruction for that line or statement.**

Another important point to remember is, in unity if I wanted to store a number with a decimal number in it. **You must use the float or double data type, and you must add a lower-case f at the end of the number**. Thirdly now suppose I want to store my name, for any text or words - we generally use **string** data types. Type string (in lower case) = "John Smith" end with a semi-colon.

Ok now we've allocated the variables to memory. Let’s retrieve those values stored in memory, by outputting them to the console. We do this by using the Debug.Log method as per the script as follows:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class NewBehaviourScript : MonoBehaviour

{

    public int num = 1;

    public float speed = 5f;

    public string c = "in the z-axis !";

    // Start is called before the first frame update

    void Start()

    {

      Debug.Log($"Player = {num} is moving {c} with speed = {speed}");

    }

    // Update is called once per frame

    void Update()

    {

      //Ditto

    }

}

Graphical user interface

Description automatically generated

# **Glossary**

Field A field is a variable that exists inside of a class, but outside of the scope of a method.

Method A method is a function or block of repeatable/executable code that defines behaviours for objects.

Primitive Types Primitive data types are **the main built-in types, that come with the compiler.** E.g. **int, float, double, string, object, short, char, bool.**

**Non-Primitive Non-Primitive data types are data types created by the programmer and not defined by C#. They are**

**Types sometimes called reference variables or object references** since they reference a memory location to store data.

Public Public before a method or a variable means the value or data can be: 1)changed outside the script, 2)edited in the Inspector 3)All other classes can access the items outside of its class.

Private Private before a method or a variable means the value or data is hidden from other classes in your program.

The method or variable 1)can only be accessed by the script or class you are in. 2)It’s not available in the inspector 3)Nobody (or other classes) can see it

Variable A variable is a space inside the computer’s memory to store values. A variable is also known as a field

declaration statement

# **Resources**

Scripting API: MonoBehaviour.StartCoroutine - Unity – Manual

<https://docs.unity3d.com/ScriptReference/MonoBehaviour.StartCoroutine.html>

/End