**GAMEDEVREJECTS:**

**C# PROG IN UNITY : MORE ABOUT ARGUMENTS, FUNCTIONS & PARAMETERS**

**Orlando Unity3d Development Meetup**

**Date: 26-Oct-21**

**All Copyright GAMEDEVREJECTS©2021**

**Contents**

[**3.7.0**  **Custom or User Defined Functions** 3](#_Toc86129736)

[**3.7.1**  **Creating A Function** 3](#_Toc86129737)

[**3.7.2**  **Passing Data by Value to a Function** 3](#_Toc86129738)

[**3.7.3**  **Returning Data** 4](#_Toc86129739)

[**Glossary** 6](#_Toc86129740)

[**Resources** 7](#_Toc86129741)

# **3.7.0 Custom or User Defined Functions**

So far, we have talked about the standard default execution order functions void Awake(), void Start() & void Update() functions. In this section we are going to learn about the **“Custom or User Defined Functions”**. i)Why we need custom functions. ii)How to create custom functions. Also, we need to consider the different iii)**Return** types, iv)**Arguments** and v)**Parameters** of functions.

Firstly, “why do we need a function or functions ?”. We know that if we want to do a task…we just need to write a statement in C# for that. However, suppose now in a game. I want a player character to shoot, every time I press the space button. I must instantiate the bullet, make the bullet translate from the players gun to a point in the 3D space. Check if the enemy has been hit by the bullet. If they have been hit. Then make a blast or particle effect. As one can see a programmer in this instance needs to create a lot of statements, every time they want the player to shoot. So, instead…of re-writing the code again, again and again (I just need to shoot), is write “all-the-the-code-statements-once”...and put it inside of a function called Shoot().The usefulness of doing this, is that we can (W.O.R.A) call the function repeatedly without writing additional code; also, efficiency saving time and effort.

# **3.7.1 Creating A Function**

To create a function. The first thing you need to do is declare the data type of function then write the name of the function. In this case Shoot. Then provide a pair of parentheses () followed by two curly braces. The code inside the curly braces is known as the function body. Very importantly every time we write a function, you must provide a return type. What we mean is. You need to specify the type of data you want to return and output back to the main program i.e. do we want to return the value as a double, float, integer, string (text) or a Boolean (true or false). **Void** means we do not want the function to return anything. We just want the function to perform some tasks like print a menu to the console.

# **3.7.2 Passing Data by Value to a Function**

Parameters are the arguments passed from the function call to the function definition and then “return” some data. Ok let’s look at the parameters. To take inputs into the function, one needs to understand what data types are going to be needed. Let’s say we want to pass some data, like the number of bullets. So int bullets is known as the parameter, and the value 100 in the call is known as the argument. So, calling the function Shoot(100) will pass 100 to the int parameter int bullets. So, the function will print the message I am shooting, then output I have 100 bullets. So, if we run that now.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class FunctionEx1 : MonoBehaviour

{

    void Awake()

    {

        Debug.Log("1st event - I am awake !");

    }

    // Use this for initialization

    void Start ()

    {

        //Debug.Log("2nd Event - I have just started once !");

        Shoot(100);

    }

   // Update is called once per frame

    void Update ()

    {

        Debug.Log("3rd Event - I am updating every frame !");

    }

    void Shoot(int bullets)

    {

        Debug.Log("I am Shooting!");

        Debug.Log("The number of shots left = "+bullets);

    }

}

# **3.7.3 Returning Data**

Now as well as receiving data to the function we can return data back. Now to return data from a function. You must write return then the data Let’s say bullets – 20. Now what we are doing is taking the number of bullets data The Argument = 120, processing it, then returning bullets – 20.

So, I am taking 100 bullets, then returning 80 bullets back to the void start. This is the basic functionality of a function or method So to access the final value, we must store the value inside of a variable.

So, we modify the call Shoot(100) to look like this -> int bulletsLeft=Shoot(100);

Now one important thing to remember is to be consistent in your data types. So, if you are passing an int data value you also need to make sure you return an int value. Or if you pass a float you a return a float. Functions are not mutually exclusive from the call. Every function must have the same data type call.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class FunctionEx2 : MonoBehaviour

{

    private int bulletsLeft;

    void Awake()

    {

        Debug.Log("I am AWAKE");

    }

    // Use this for initialization

    void Start ()

    {

        Debug.Log("I have STARTED!!");

        bulletsLeft = Shoot(120);

        Debug.Log("The bullets I have left now is = "+bulletsLeft);

    }

//Update is called once per frame

    void Update ()

    {

        //Debug.Log("I am UPDATING EVERY FRAME");

        //Shoot();

    }

    int Shoot(int b)

    {

        Debug.Log("The number of bullets I started with was = "+b);

        return b-20;

    }

}

# **Glossary**

# **Resources**

/End