**GAMEDEVREJECTS()**

**C# Programming**

**Orlando Unity3d Development Meetup**

**Date: 27-Apr-22**

**All Copyright GAMEDEVREJECTS©2022**

**Contents**

[**8.0** **So, What Is A Loop ? & Why Do We Need Loops ?** 3](#_Toc101955109)

[**8.1** **While Loops** 3](#_Toc101955110)

[**8.1.1 Create A New Project** 3](#_Toc101955111)

[**8.2** **do While Loops** 4](#_Toc101955112)

[**8.3** **For Loop** 5](#_Toc101955113)

[**8.4** **For Each Loop** 6](#_Toc101955114)

[**Glossary** 9](#_Toc101955115)

[**References** 10](#_Toc101955116)

# **8.0 So, What Is A Loop ? & Why Do We Need Loops ?**

We require loops in programming, when we need to repeat certain tasks. We W.O.R.M. (We WRITE ONCE RUN MULTIPLE (W.O.R.M.) times until a specified condition is met). Rather than constantly copying and pasting blocks of code, every time we need to action something. Loops are Control Statements. There are 4 main types of loops as follows:

1. **WHILE**
2. **DO WHILE**
3. **FOR**
4. **FOREACH**

Loops are more efficient and save on real estate in our script.

# **8.1 While Loops**

While loops are **“pre-test loops”**. Meaning they check if a specific condition (or expression) is met first, before executing the loop. This is important in a game if you want to check if a condition is being met, before executing a task or tasks.

# **8.1.1 Create A New Project**

Create a new Project, Scene then attach a Script. See Chapter 00.0 Create A Scene.

Or create an empty object, then attach a new script, since we are just using the Console Output anyway 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**.(Drag and Drop) – HIERARCHY => / (Onto).**GameObject** /

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

Ok inside of the void Start() function. The first thing to do is to declare and define a loop counter, to track the number of iterations of the loop. int count = 0;

Then...we are going to write the statement for the pre-test condition > while (counter < 10) run what is inside the loop. Meaning every time, the condition is true inside the parentheses then we always execute or run the code whatever is inside the curly braces. Which is going to be Debug.Log. Where we just print the values of the counter to the console from 0 to 9. We are also going to increment the counter by every time for each pass of the loop.

Using the post increment operator. Which basically means increase the loop counter by 1 > takes the current integer value stored in memory increases it by 1 then places a new loop counter in memory. So, after the first pass 0 becomes 1. 2nd loop pass 2 becomes 3, etc.,

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class NewBehaviourScript : MonoBehaviour

{

    int counter=0;

    // Start is called before the first frame update

    void Start()

    {

    }

    // Update is called once per frame

    void Update()

    {

        while(counter<5)

        {

            Debug.Log($"Count = {counter}");

            counter++;

        }

    }

}

Graphical user interface, application

Description automatically generated

# **8.2 do While Loops**

The next thing I want to talk about are do loops. A Do Loop is a “**post**” test loop, which means the script will execute the inner code first, before checking the condition at the bottom. If the condition is true. i.e. if the loop counter on the 2nd pass is still less than 10. It makes another loop pass again. If it is still true. Then it will execute the code again. The reason why we need the loop counter is, so we don't get an infinite loop. Which will “crash unity” Lol ! And that sucks !

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class NewBehaviourScript : MonoBehaviour

{

    int counter=0;

    // Start is called before the first frame update

    void Start()

    {

        do

        {

            Debug.Log($"Count = {counter}");

            counter++;

        } while (counter <= 5);

    }

    // Update is called once per frame

    void Update()

    {

    }

}

Graphical user interface

Description automatically generated

# **8.3 For Loop**

Last, but not least. Let's check the for Loop. The for Loop is really useful for counting, and inputting data into an array for example. The “**for**” Loop is like a **while** loop. A **pre-test** loop. You have your declaration of a variable, at the beginning.

Usually this is going to be a single character like x,y,z,c,I private variable. Of course, you can give a non-single character or better name if you want to. The next code fragment is the condition, we specify the upper limit of the loop. i.e. c less than 10. => So, while c is still less than 10. It will still continue to run what's inside the for loop. So once the 1st loop is completed. It is going to increment c by 1. Save then. Let's play this and see what happens.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class NewBehaviourScript : MonoBehaviour

{

    int counter=0;

    // Start is called before the first frame update

    void Start()

    {

        for(int c=0;c<10;c++)

        {

            Debug.Log($"Loop pass = {c}");

        }

    }

    // Update is called once per frame

    void Update()

    {

    }

}

Graphical user interface

Description automatically generated

# **8.4 For Each Loop**

The **For Each** loop is very similar to the for loop we just discussed. But it is different in the sense that it does a lot of the heavy lifting for you. Foreach states that for each item in an object. Remember that everything in .NET is an object. Objects can be made from other objects. So, if we create a string variable equal to New York City => **string name ="New York City";**

Notice the green squiggly. It is just a warning, to say the variable is not being used. Then we write…

**foreach(char a in name)**

**{**

**}**

Remember name is a string. The string is a group of characters. Each one of the letters in the name New York City is a character. We are just going to echo output or print the characters to the console. By typing:

**Debug.log(a);**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class NewBehaviourScript : MonoBehaviour

{

    int counter=0;

    // Start is called before the first frame update

    void Start()

    {

        int c=0;

        string name="Orlando City";

        foreach(char a in name)

        {

            Debug.Log($"Letter {c} = {a}");

            c++;

        }

    }

    // Update is called once per frame

    void Update()

    {

    }

}

Graphical user interface, application

Description automatically generated

# **Glossary**

# **References**

/End