Welcome to Week 8: Unity Concepts - C#

This week is all about getting comfortable setting up, writing, and testing C# scripts in Unity. Becoming familiar with C# scripts is vital for understanding how to create, test, and debug Unity applications. Even if you are not comfortable writing C# code, simply reading and identifying what each line of code actually does is a useful exercise for fixing and improving your applications. The Unity Scripting API and the C# Language Specification are definitive resources for researching and identifying C# code.

Table of Contents

Part 1 - Setup Your Development Environment

1.1 - Visual Studio Setup

- If you have not made one already, go to the Visual Studio webpage and create a user account.
- Download Visual Studio 2022 Community Edition from the same website you created an account.
- Visual Studio is considered an industry standard by many software professionals, so it is a good practice to get familiar with it.

1.2 - Notepad++ Setup

- Download a recent version of Notepad++.
- Notepad does not require a user account and usually loads more quickly than Visual Studio.

1.3 - Configure Unity for IDE

- Open Unity Hub and start a new Unity project.
- Set the location of the project to somewhere that is easy to find in the directory.
- Once the new project is open, select the **Edit** tab on the top left.
- Unfold the menu and select **Preferences**.
- Choose the **External Tools** option.
- Unfold the **External Script Editor** menu and select your preferred IDE.
- If this is set up correctly, Unity will automatically open C# scripts in your preferred IDE when editing.

Part 2 - Creating C# Scripts in Unity

2.1 - Creating a New Script

- Right click in the **Assets** area and select **Create > C# Script**.

- Name your script **RotateObject**, capitalizing each word and leaving out spaces.
- Select and open the script from the **Inspector**.

- The following "boilerplate" script will be generated as the foundation for your new C# script.

```
1using System.Collections;
2using System.Collections.Generic;
3using UnityEngine;
4
5public class RotateObject : MonoBehaviour
6{
7    // Start is called before the first frame update
8    void Start()
9    {
10
11    }
12
13    // Update is called once per frame
14    void Update()
15    {
16
17    }
18}
19
```

2.2 Base Script Overview

- **MonoBehavior** is the base class from which every Unity script derives.
- When an application starts, runs, or stops; Unity calls events in the following order.
- For the following, we will concentrate on adding code to `void Update()`

Part 3 - Manipulating Game Objects with Scripts

3.1 - Add Public Variables to the Script

- Above `void Update()` type the following:

```
1public float speed = 20f;
```

- This line of code establishes:
  - **public** - viewable in the Hierarchy
  - **float** - whole or decimal number value, followed by **f**
  - **speed** - name of the variable

3.2 - Code the Object Rotation

- Under `void Update()`, type the following line of code inside of the brackets.

```
1 transform.Rotate(Vector3.right * speed * Time.deltaTime);
```

- This line of code establishes
  - **transform** - used to store and manipulate the position, rotation and scale of the object
  - **Rotate** - specifies that the transform will be a rotation
  - **Vector3** - establishes that the transform is in 3D space
  - **right, left, forward, back** - indicates the direction of the rotation
  - *** (asterisk)** - arithmetic operator for multiply
  - **speed** - refers to the established variable
  - **Time.deltaTime** - completion time in seconds since the last frame

- Save your script and exit the code editor.

3.3 - Create 3D Object and Attach Script

- In the **Hierarchy**, right click and select **3D Object > Cube** from the menu.
- Enter **Play Mode** and verify that the cube is visible in the viewport.
- Exit **Play Mode** and select the Cube to view the **Inspector**.
- Drag your new script into the area below **Add Component.**
- Now your script is attached to the **Cube** game object.
- Notice that the speed variable is visible.
- If you remove the 20f value after public float speed, you can alter the speed directly from the **Inspector**.
- Enter play mode and watch your cube rotate.

---

4 - Write a Simple Object Visibility Off Script

- Create a new script the same way as before and title it **ObjectVisibility**

```
1using System.Collections;
2using System.Collections.Generic;
3using UnityEngine;
4
5public class ObjectVisibility : MonoBehaviour
6{
7    // Start is called before the first frame update
8    void Start()
9    {
10        GetComponent<Renderer>().enabled =
!GetComponent<Renderer>().enabled;
11    }
```

---

5 - Commit and Push to Repository

- Save your Unity project and exit out of Unity.
- Create a new repository in Bitbucket or Github.
- Give it the same name as your current project.
- Open Sourcetree or Gitbash.
- Clone your new repository from Bitbucket or Github and save it in your directory.
- After exiting Unity, “Stage” your changes by adding your project folder to the cloned repository.
- Add a “Comment,” and “Commit” your changes.

- Double check that you have selected the correct repository and branch.
- “Push” your project changes.
- Open Github or Bitbucket and verify that the repository updated successfully.