12.08.2019

31263 / 32004 Game Development Lab Week 4

Getting Started

1. Download the corresponding week's zip file from the Lab section of UTSOnline.
2. Unzip the project folder and open it in Unity. If there are any warnings about difference in versions, just continue. If this causes any red errors in the console once the project opens, notify the tutor.
3. Within the Weekly folders there are image and executable files starting with "Status…". These files give you a preview of what is expected for each point percentage below.
    a. If you are on Mac and the Status-100Percent App file won't run, hold control and click (right click) then select Open, if there is a security warning, acknowledge it an press open again.
    b. If you are running the Windows executable on the lab computers, you need to copy the entire executable folder into Windows(C:)/Users/<student number>/AppLockerExceptions. From there you can double run the .exe file.

Tasks

Points	Requirements
40%	Setting your Default Script Editor • Make sure that Unity is set to use Visual Studio (or your IDE of choice) and that the auto-complete functionality in in that IDE is working properly for the Unity API. • This setting is found at: ○ Windows: Edit->Preferences->External Tools->External Script Editor ○ Mac: Unity->Preferences->External Tools->External Script Editor • If you are on a PC in the lab room: ○ Select "Visual Studio" if it is there ○ Otherwise, click "Browse" and find the Visual Studio executable at C:\Program Files (x86)\Microsoft Visual Studio\2017\Professional\Common7\IDE\devenv.exe ○ This change applies to each PC you log into in the lab room. E.g. if you log into the same PC each week, you should not need to change this, but if you log into a different PC you will need to do it again for that PC. • Create a new C# script called Test. ○ In the Start method, start typing "Deb" ○ You should have an auto-complete that says "Debug". Select this, then start typing ".Lo" and select the Debug.Log() method.

	○ Finish this to be Debug.Log("Test"); • Attach this script as a component of the MainCamera by dragging it onto the Camera.
60% (P)	Setting up prefabs and referencing them in scripts • In Unity, create a red material and a blue material and set their albedo accordingly ○ Create two spheres in the scene, one with the red material, the other with the blue material. ○ Now create two prefabs from these called RedPrefab and BluePrefab ○ Delete the two sphere gameobjects from the scene. • Right click in the Project Window and create a new C# script called "LoadAssets" • Open up the LoadAssets script. • Create a public GameObject member variable called redObj of the LoadAssets class (i.e. outside of any method). • Save the script and return to the Unity window to auto-compile it. • In the Hierarchy panel, create a new empty GameObject, rename it to "LoadManager", set all position and rotation values to 0, set all scale values to 1, and add the LoadAssets script as a component of this game object • Assign the RedPrefab prefab to the redObj variable of the LoadAssets component of the LoadManager game object.
70% (C)	Instantiating Prefabs as GameObjects through code • Create a private GameObject member variable called blueObj of the LoadAssets class but make it available to the Inspector View (tip: search **SerializeField** in Unity Docs). • Open the LoadAssets script again. When the game starts, the LoadAssets component should create a redObj at (2,0,0) and a rotation of zero and a blueObj at (-2,0,0) and a rotation of zero (tip: search **Instantiate**, **Vector3**, and **Quaternion.identity** in Unity Docs). • Save the script, assign BluePrefab to blueObj on the LoadManager game object, and press play in Unity to see the results.
80% (D)	Printing GameObject properties • In Unity, create a new script called "ConsolePrint" • Edit the RedPrefab and BluePrefab prefabs to have a ConsolePrint component. • For every frame, the ConsolePrint script should print the following "<gameObject name>: i" where <gameObject name> is the name of the game object that the component is attached to while i is an integer that increments every frame and is set to 0 at the start of the game. • Press play in Unity and observe what happens.
90% (HD)	Renaming files, referencing components, and using tags • Change the name of ConsolePrint in the Project Window to "PrintAndHide" and fix any errors that arise. • In PrintAndHide make the member variable "public Renderer rend". ○ Select RedPrefab in the Project Window, then drag it onto its own PrintAndHide.Rend reference. ○ Repeat the above for BluePrefab. • Create a new Tag called "Red" and assign the RedPrefab Tag to this.

	• Create a new Tag called "Blue" and assign the label of BluePrefab Tag to this.
100% (HD)	Deactivating GameObjects and Disabling Components • In the PrintAndHide script ○ If the Tag of the game object is "Red" and i = 100, deactivate the game object. ○ If the Tag of the game object is "Blue" and i = a random integer between 200 and 250 (inclusive of the numbers 200 and 250) which is generated when the game starts, disable the Renderer component of this object. • Press play in Unity and observer what happens, especially in the console. CLEAN, EFFICIENT, ELEGANT CODE!

Submission

When you complete the activity to the grade threshold that you want, you then need to:

1. Complete the "Status-StudentSubmission.txt" file in the highest level of the project folder.
2. Remove all other "Status-…" files and folders to reduce the size of your project.
3. Zip the entire project folder.
4. Re-name the zip file to **"[student ID]-LabWeek[week number].zip".**
5. Submit the zip file to UTSOnline for the associated link for this week in the Lab **before Monday 9am of the following week**.
6. Failure to follow any of these could result in a 0% mark for that week.
7. You will also **demo your submission to your tutor** at the **start of the following week's lab.**

If you finish the activity early, show it to your tutor before you submit it on UTSOnline so they can help you make some final corrections and mark it at that time.