20.09.2019

31263 / 32004 Game Development Lab Week 9

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%	Scene Set-up • In the Project Window, go to the Scenes folder, right click and create an new scene called "StartScene" • Open StartScene and create a sphere in the Hierarchy Window with a position (-3, 1, 0). • Create a new script in the Scripts folder called MoveSphere and attach it to the sphere. • Create an empty game object and call it "Managers" • Attach the Tweener component to the Managers gameobject • Create a script called "SpeedManager"

50% (P)	Moving the Sphere
	• In SpeedManager
	  ○ Create a private static member variable called speedModifier with a default value of 1.0f;
	  ○ Create a public static property called SpeedModifier with a public getter (returning the speedModifier variable) but no setter.
	• In MoveSphere
	  ○ Create a private member variable called "target" with a value of (-3,1,0).

	○ Create a private member variable called “duration” with a value of 1.5f; ○ Create a public member variable to store a reference to the Tweener attached to Managers add assign it in the Inspector Window ○ In Update(), if !TweenExists(transform), then: ▪ Flip target in the x-axis, so if it was negative, make it positive (e.g. -3 becomes +3, or +3 becomes -3) ▪ Add a new tween with the spheres transform, current transform position, target, and the duration variable divided by the SpeedModifier property in SpeedManager.
60% (P)	<u>Enum States</u> • In SpeedManager ○ Create an enum called GameSpeed with values Slow = 1 and Fast = 3. ○ Create a private static member variable called currentSpeedState of GameSpeed type with a default value of Slow. ○ Create a public static property called CurrentSpeedState with: ▪ A public getter that returns currentSpeedState ▪ A public setter that assigns both currentSpeedState with the new enum (Slow or Fast) and speedModifier with the enum’s value (1 or 3) cast to a float. • Create a new script called InputManager and attach it to the Managers gameobject ○ When space is pressed, the InputManager should toggle the SpeedManager’s currentSpeedState, switching the game from slow to fast or vice versa. You should do this using the C# “ternary operator” (with the ? symbol) ○ During play, you should see the sphere change speed on the next tween after space is pressed.
70% (C)	<u>Loading Scenes</u> • Create a new script called GameManager. ○ Create an enum called GameState with values of Start and WalkingLevel ○ Create a public static member variable called currentGameState with a starting value of Start. • In the InputManager, if the current scene is the StartScene according to the GameManager and the Return key is pressed, then WalkingScene should be loaded using the build index (int) form of the method (hint: see SceneManager.LoadScene(), you will need to import UnityEngine.SceneManagement) • The WalkingScene uses root motion for the animation and is handled in the CharacterMovement script. ○ Modify the animator and the CharacterMovement script so that the speed of the walking animation is affected by the SpeedManager’s speedModifier variable, similarly to how the spheres movement was. • Press play, space, then return. The character should be walking quickly. ○ Press space again, the character’s speed won’t change, why? ○ By only changing the InputManager script, allow the Space key to change the walking animation speed in the WalkingScene.

80% (D)	PlayerPrefs
	• Create a new script called SaveGameManager and attach it to Managers.
	  ○ This component should have a public void SaveSpeed() that writes the currentSpeedState to the PlayerPrefs as an int and saves the changes every time the Space key is pressed. Remember to keep your managers focused on one task, Input should not be read in SaveGameManager at all.
	  ○ This component should also have a public method that loads the speed state from PlayerPrefs (if it is in PlayerPrefs) and assigns it to the SpeedManager. This method should be called from the SaveGameManager when the game starts.
	  ○ Both of these should use a key string that is unchangeable during runtime.

90% - 100% (HD)	Loading Level Asynchronously
	• Make sure that all of the scenes in the Scenes folder are added to the Build Settings
	• Create a new script called AsyncLoader and attach it to an empty GameObject in the WalkingScene
	  ○ Create a public Vector3 pos
	  ○ Create the method public void LoadLevelAsync()
	  ▪ If pos.x > 2.5f, load RightScene (if it isn't loaded) and unload LeftScene (if it is loaded)
	  ▪ If pos.x < -2.5f, load LeftScene (if it isn't loaded) and unload RightScene (if it is loaded)
	  ▪ If pos.z > 2.5f, load TopScene (if it isn't loaded) and unload BottomScene (if it is loaded)
	  ▪ If pos.z < -2.5f, load BottomScene (if it isn't loaded) and unload TopScene (if it is loaded)
	  ▪ Hint: Use SceneManagement.LoadSceneAsync(…) with LoadSceneMode.Additive passed to it.
	• In CharacterMovement
	  ○ Make a reference to the AsyncLoader in the scene
	  ○ In Update(), assign the AsyncLoader.pos variable to the current position of the character.
	• 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.