**Lab 5: - Unity – Scripting – event Functions**

**Aim of this lab session:**

To continue learning the Unity scripts using C#. This session will cover typical Unity event functions, including those for initialization (e.g. Awake(), Start()), regular updates (such as FixedUpdate()) and physics related events such as OnCollisionEnter(), OnTriggerExit(), etc.

1. Start the Unity and create a new project, called Lab5\_Unity.
2. Create a ‘Sphere’ object and then a ‘C# Script’ (which should be put into a subfolder under the ‘Assets’ folder). Rename this script as “*SphereController*” and attach it to the ‘Sphere’.
3. Double click the “SphereController” icon will open it in Monodevelop. Edit the script so it looks like:

using UnityEngine;

using System.Collections;

public class SphereController : MonoBehaviour {

// This is called when the scene is loaded

void Awake () {

Debug.Log(“Awake is called.”);

}

// Use this for initialization on any component which is enabled

void Start () {

Debug.Log(“Start is called”);

}

}

1. Back to Unity and ‘Play’. From the message displayed (in the Console window), which method is called first?
2. Create a few more objects such a, planes, and attach each of them a script like above.
3. ‘Play’ the game and see if Start() is called only after all objects’ Awake() functions are called.
4. What are the main differences between Awake() and Start()? What can/should be done in the method Awake()? (Task 5.1)

**Practice on regular update events**

1. Add a Cube into the scene.
2. Create a script called ‘CubeUpdateTesting.cs’ and attach it to the Cube.
3. Add a ‘Rigidbody’ component to the Cube (‘Add Component 🡪 Physics 🡪 Rigidbody’).
4. Modify the anatomy of the script as follows:

public class CubeUpdateTesting : MonoBehaviour {

//private float speed = 0.2f;

public float speed = 0.02f;

*private Rigidbody rb;*

void Start () {

rb = GetComponent<Rigidbody> ();

}

void Update () {

//ControlByArrowKeys(speed);

}

void FixedUpdate() {

ControlByArrowKeys(speed);

}

void ControlByArrowKeys(float speed)

{

float xMove = Input.GetAxis("Horizontal") \* speed;

float zMove = Input.GetAxis("Vertical") \* speed;

// use this method only for an object with Rigidbody component attached

// and it is not ‘Kinematic’

rb.AddForce (xMove, 0.0f, zMove);

}

1. Save the script and back to Unity to ‘Play’. Adjust the ‘speed’ for different effect.
2. Note that the method ‘*ControlByArrowKeys*(speed)’ should be called from FixedUpdate() as it deals with physics. Modify the codes above by calling it in the Update() and compare the result.
3. Now add another update method, LateUpdate( ) in the script as below:

void LateUpdate() {

}

1. If you move the *ControlByArrowKeys* function to inside this update, you will notice that not much has changed. The difference is that this update occurs after the others (i.e. Update() and FixedUpdate())so is usually reserved for moving the camera based on the other object locations.
2. Refer to the lecture slides and understand the differences between these 3 update methods, in terms of how often they are called, when they should be used, etc.

**Practicing the physics event functions – collision, trigger and colliders**

1. Now add a new method to the CubeUpdateTesting class.

void OnCollisionEnter(Collision coll)

{

Debug.Log("We are colliding");

}

1. You should now see this message in the console when the sphere and cube collides with each other. (Adjust the positions of these two objects if they don’t collide). This is because the RigidBody calls the function OnCollisionEnter whenever it collides with another object. Information about the other object can be obtained by using the Collision parameter which is passed in to the method as a parameter. Position the cursor over the method and then press the key ‘Ctrl’ and single quote ‘, you will be prompted more info about how to use it, plus examples of script.
2. Similarly, you may add other methods, such as OnCollisionStay( ), OnCollisionExit( ) and print messages (Task 5.2) to understand the differences among them.
3. Next step is to practice the three method OnTriggerEnter( ), OnTriggerStay( ) and OnTriggerExit( ). Add the methods into the class above. Do you see any messages printed out?
4. In the Unity editor, tick the box ‘Is Trigger’ in the Inspector for the ‘Cube’. ‘Play’ the game, what happens then? (Task 5.3)
5. Try understanding the difference between ‘trigger’ and ‘collider’.