Unity - Basic Movement Scripting

In this, we will write code that makes a game Object move up, down, left and right based on the user’s input. This should help us understand the workflow of Unity scripting more easily.

Step 1: Let us start with our movement script. Create a new script, and name it “Movement”.

Now, open the script and you should see the regular stuff which is at your desktop.

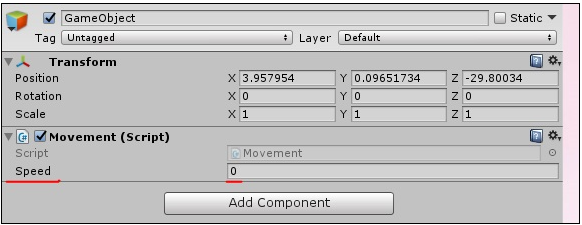
Step 2: Let us create a public float variable named **speed**. Making a variable **public** in Unity has a great advantage –

* The variable shows up as a modifiable field inside the editor, so you don’t have to manually adjust the values in code.



If we save this script without touching the other methods, it should compile in Unity.

(You can see when it is compiling by the Buffericon in the bottom right corner.)

Step 3: Next, **drag and drop** the script from the Assets onto the Game Object. If you do it correctly, this is what you should see in the Game Object’s properties –

Step 4: Since the speed value is adjustable and need not be changed in code all the time, we can use update() method instead of start().

Let us now consider the objectives for the Update method –

* Check for the user input.
* If there is a user input, read the directions of input.
* Change the position values of the object’s transform based on its speed and direction. To do so, we will add the following code –

Code:

void Update() {

float h = Input.GetAxisRaw(“Horizontal”);

float v = Input.GetAxisRaw(“Vertical”);

gameObject.transform.position = new Vector2 (transform.position.x + (h \* speed),

transform.position.y + (v \* speed));

Let us now discuss the code in brief.

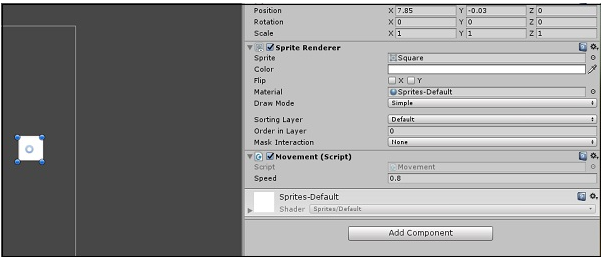
First of all, we make a floating-point variable named **h** (for horizontal), and its value is given by the **Input.GetAxisRaw** method. This method returns -1, 0 or 1 depending on which key the player has pressed on the up/down/left/right arrows.

The Input class is responsible for getting input from the user in the form of key presses, mouse input, controller input, and so on. The GetAxisRaw method is slightly harder to understand, so we’ll get back to that later.

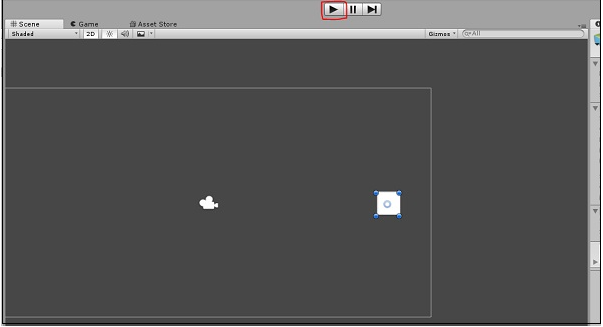
Next, we are **updating** the position of our gameObject to a new position defined by creating a new **Vector2**. The Vector2 takes 2 parameters, which are its **x and y** values respectively. For the x value, we provide the sum of the object’s **current** position and its **speed**, effectively adding some amount every frame the key is pressed to its position.

Save this script and head back to Unity. Unity will automatically update all scripts once it compiles successfully, so you don’t have to reattach the script again and again.

Step 5: Now that you are done, change the value of the **speed** in the GameObject’s properties to say 0.8. This is important because a higher value will make the player move too fast.



Step 6: Now, click **Play** and see your first small game in action!



Try pressing the arrow keys and moving around. To stop the game, simply press Play again. You can even adjust the speed in real-time so you do not have to stop and start it all the time.

Ronak Baser