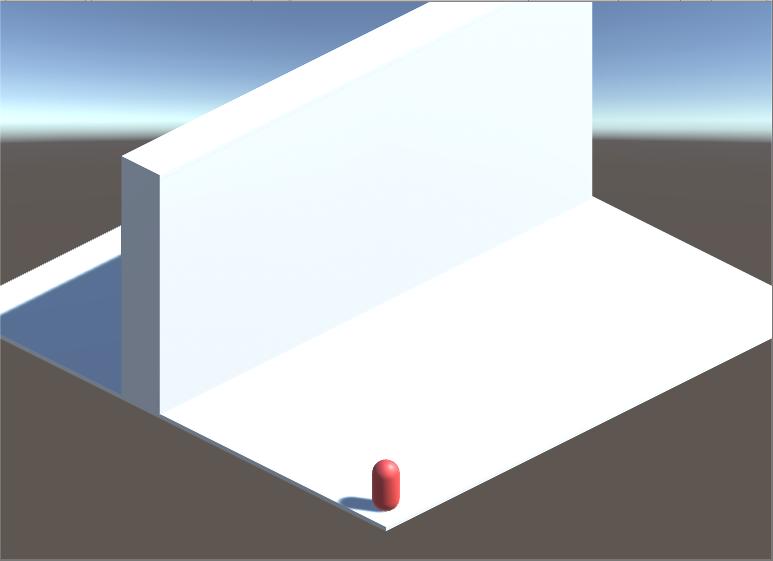
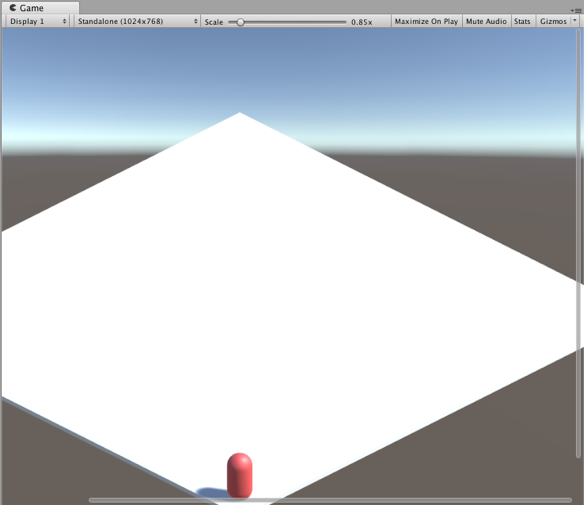
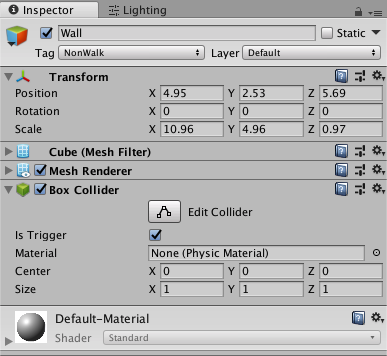
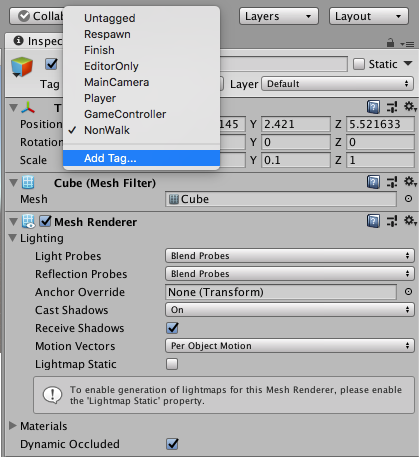
**Character collision and Scene transitions**

**Setting up boundaries**

To set up wall collision and boundaries, we need to create an obstacle or object that the player won’t be able to move through. For this example, I’ve created a wall using a cube and the transform scale.



As we’ll be using tags to identify what game objects can’t be traversed, create a new tag for the wall and select it in the drop down menu of the inspector; for this example, I’m using the tag “NonWalk”. Make sure that the box collider for the wall is also set to “Is Trigger”.

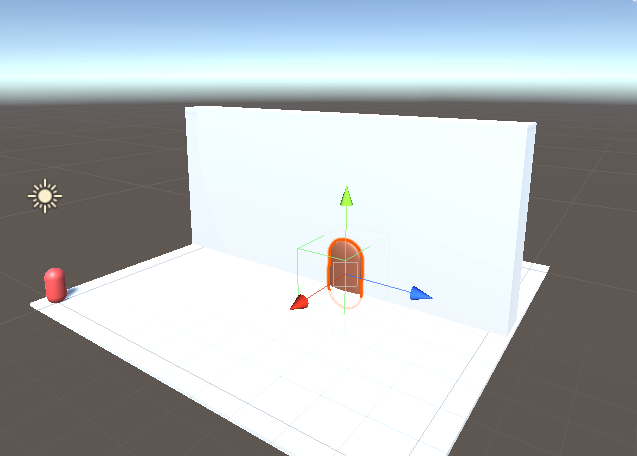
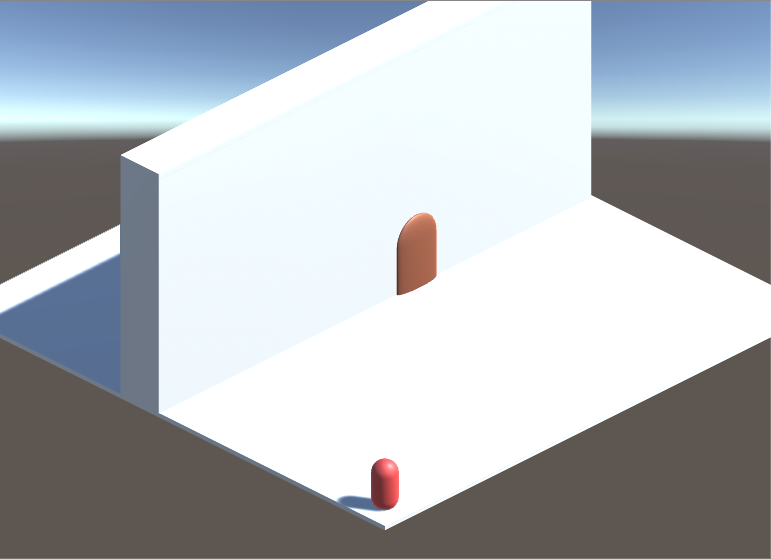


In the player movement script, add a private void for *OnTriggerEnter(Collider collision)*. This ensures that the colliders, if set to *Is Trigger* (in this instance, the wall’s box collider), is being checked. Using an If statement, we’ll detect the *NonWalk* tag using *collision.gameObject.tag == “NonWalk”* and set the next direction (*nextDir*) for X, Y and Z to 0. This means that if the player walks in to an object tagged *NonWalk*, the player will be moved back to their previous position rather than being able to walk through the wall.



**Scene transitions**

I now want my player to able to move through a door – we can do this using the scene manager in unity. To start, I created a capsule with a brown coloured material and reduced the X scale to flatten it. I then moved the capsule up against the wall to make it look like a door. As we’ll need the player so be able to stand within range of the door to open it, I deleted the default capsule collider and added a box collider, altering the X and Z size values.



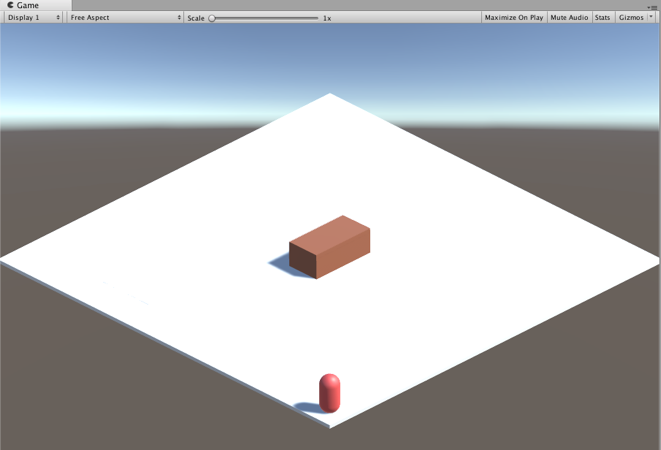
Next, create a new C# script (in this instance, I’ve called it DoorOpen). Ensure you add using *UnityEngine.SceneManagement* to the beginning of the script in order to access the scene manager.

In the public class field, we’ll create a serialized field called *LoadScene* in the form of a string. As no scene index is set, the *Loadscene* index will default to null – this will allow us to input which scene to transition to from the current scene. Next, create a *GameObject SceneTrigger* and a *bool* for *canTrigger*. Using *OnTriggerEnter (Collider other)*, we’ll set the scene trigger to the current *gameObject* (the door in this case) and the *canTrigger* bool to true if the the collider detects an other game object with the tag “Player”.

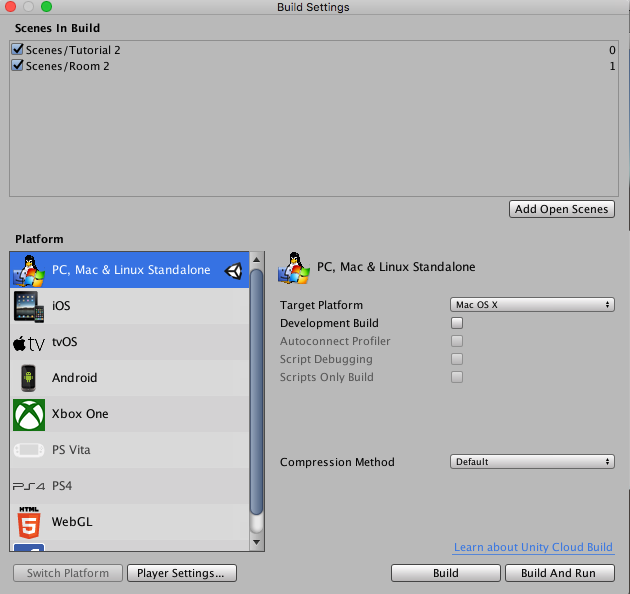
Using *OntriggerExit*, we’ll set canTrigger to false if the object tagged “Player” exits the collider – this prevents the key input that we’ll add from triggering the scene continuously. In Update, we’ll use an if statement to detect the Key input of “E” and if *canTrigger* is set to true – if both are fired, the scene manager will load the next scene and *canTrigger* is reset to false. This script can now be applied to the door object.



Next, duplicate the scene and create a new object or scene layout so that the scene change is evident. I’ve created a chest by adding a cube with a brown material to the scene.



Next, go to file, build settings and drag the two scenes to the “Scenes in build” section.



Finally, on the door game object, type the name of the scene you wish to load in to the “Load scene” field (in this instance “Room 2”)

../../11.png.

If working correctly, the player should be able to walk within range of the door, press E and load the next scene.

In the next tutorial, I’ll be creating enemies, a health system with functioning UI and a death screen.