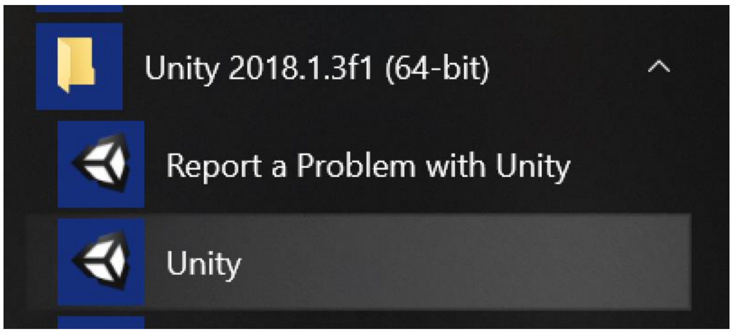
Immersive Environments Tutorial

Unity

Goals:

- Collisions
- Triggered Events
- Changing Scenes
- Simple NPCs

Load up unity

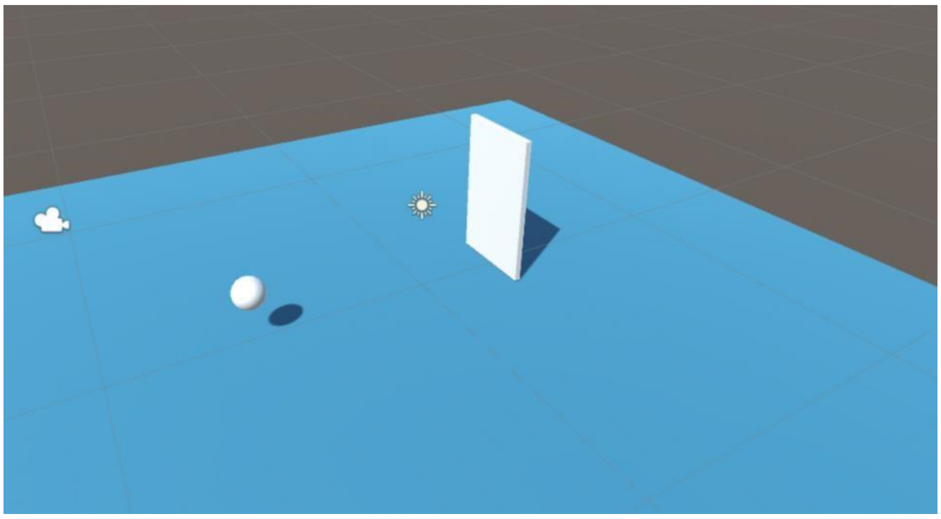


Build Object: Collisions in Unity

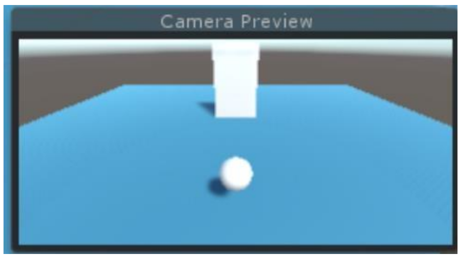
Aim: Build various elements and test differing collision types.

Door Open

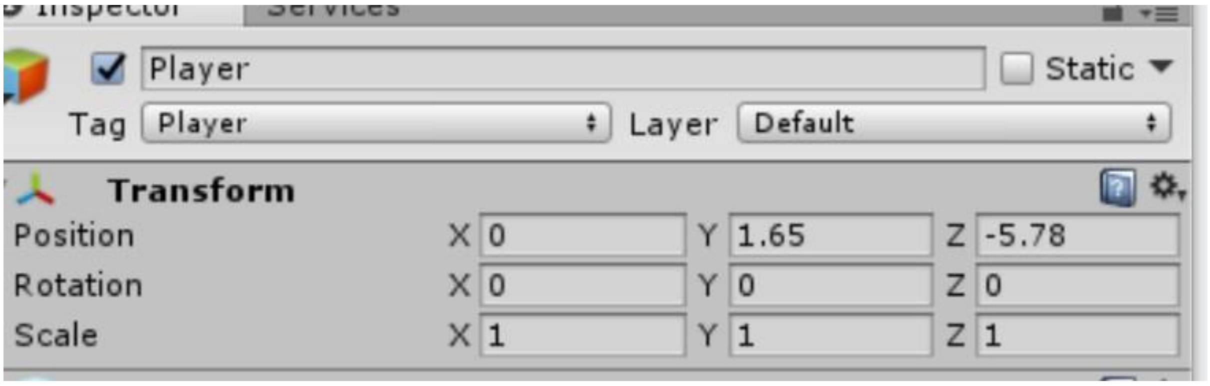
Build the following scene



The main camera is lined up like this:



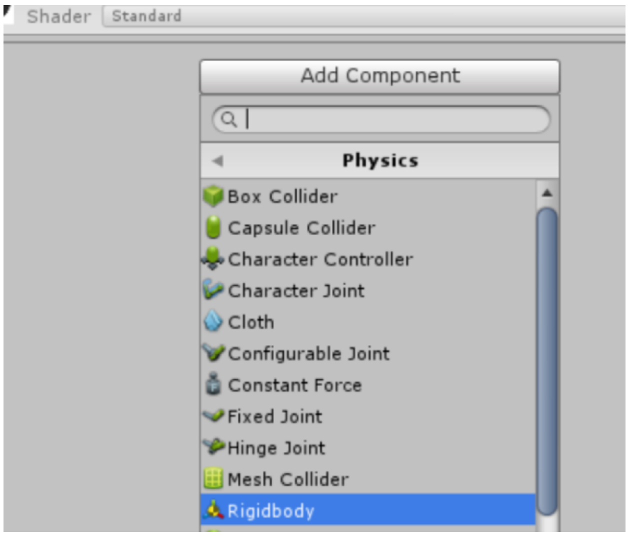
The sphere has been called player and has the following attributes



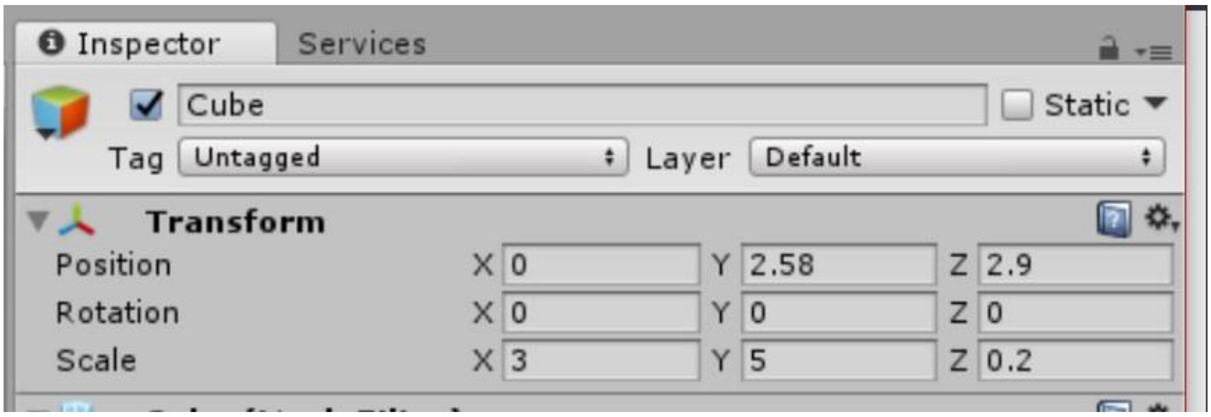
Player
Tag: Player Layer: Default

	X	Y	Z
Position	0	1.65	-5.78
Rotation	0	0	0
Scale	1	1	1

Add a Rigidbody to the Player. Component→ Physics → Rigidody



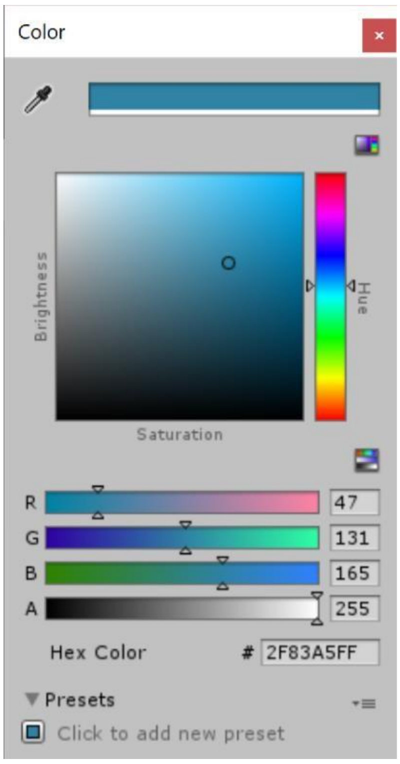
The cube has been shaped like a door, and has the following attributes



Cube
Tag: Untagged Layer: Default

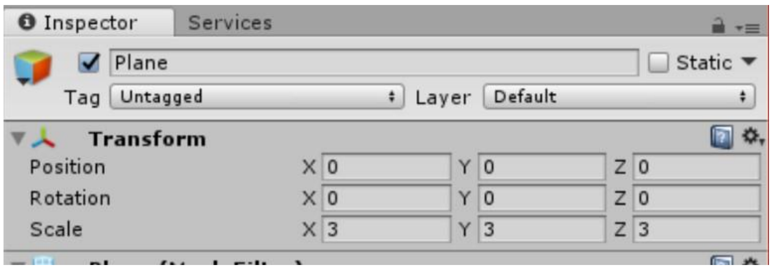
	X	Y	Z
Position	0	2.58	2.9
Rotation	0	0	0
Scale	3	5	0.2

The floor plane has the following material applied to it



R 47
G 131
B 165
A 255
Hex Color # 2F83A5FF

And the following information

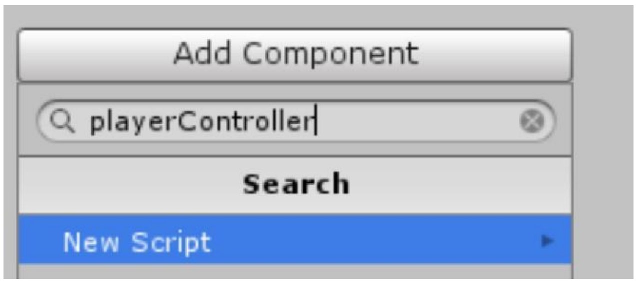


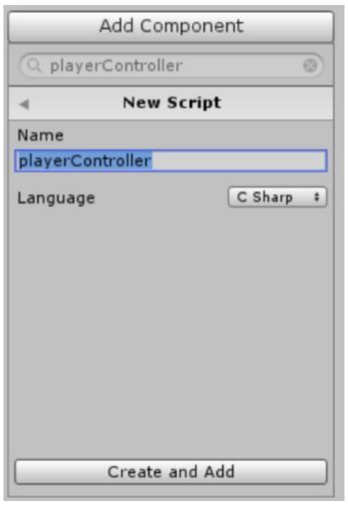
Plane
Static
Tag Untagged
Layer Default

Position	X	0	Y	0	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	3	Y	3	Z	3

Now that we have the basic layout, we will add the code to move the player around.

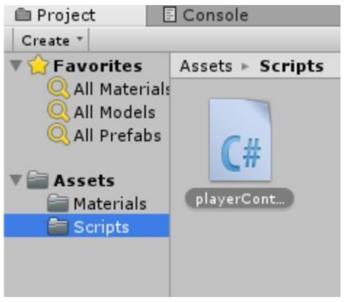
Create a new script called playerController and apply it to the player.

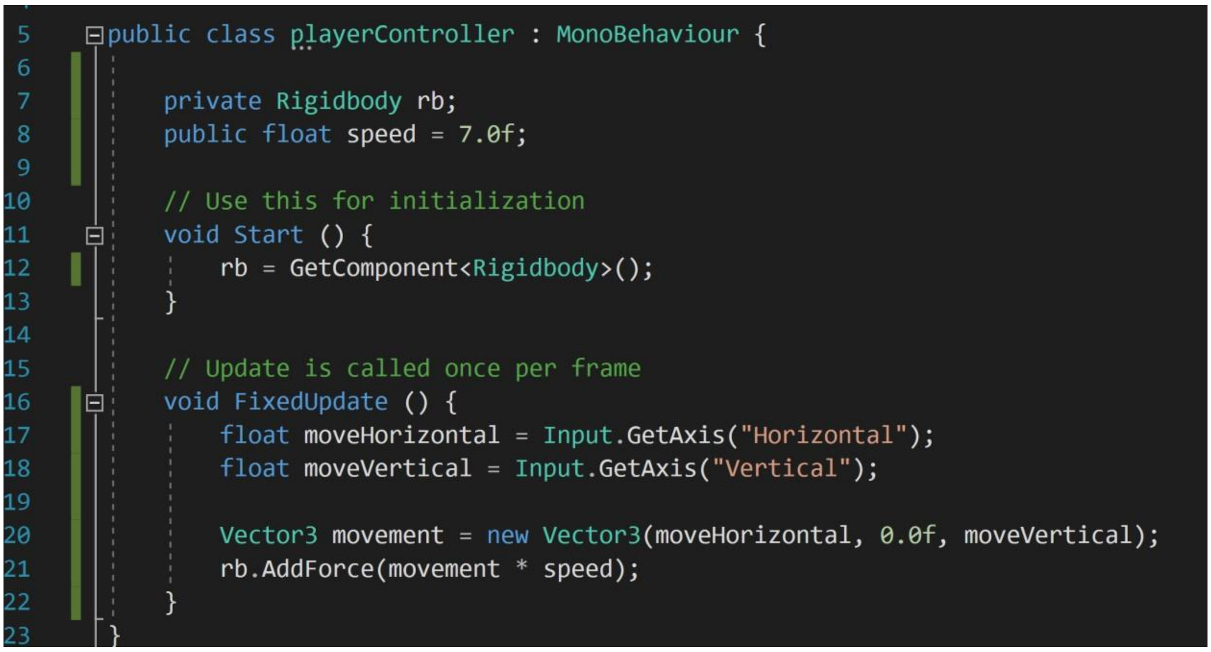




Name playerController
Language C Sharp

Place it in the scripts folder and then open it up in visual studio and apply the following code

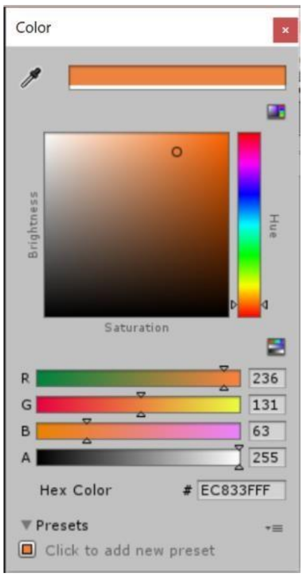



```
public class playerController : MonoBehaviour {

    private Rigidbody rb;
    public float speed = 7.0f;

    // Use this for initialization
    void Start () {
        rb = GetComponent<Rigidbody>();
    }

    // Update is called once per frame
    void FixedUpdate () {
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");

        Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);
        rb.AddForce(movement * speed);
    }
}
```

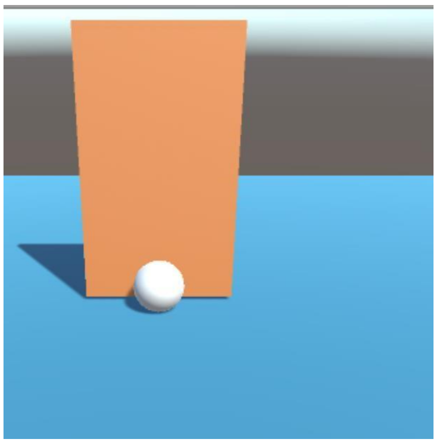
Save and test in Unity.

To ensure that the player sphere is visible when heading towards the cube, we'll change the colour of the cube.

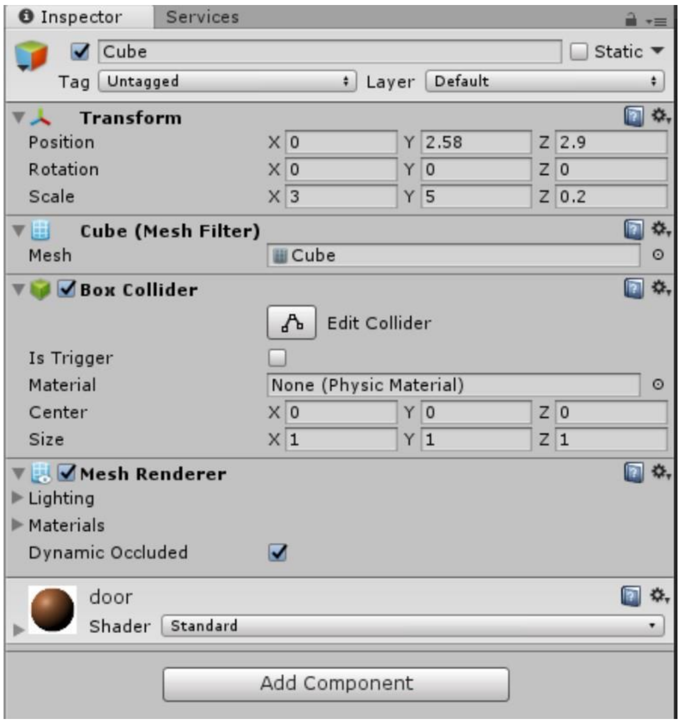
The colour selected was this:



Now that we have the ability to see the sphere when it approaches the door, we need to implement a collision area and an effect that occurs when we get there, in this case, we will just move the door.



Now, let's examine the inspector on the cube.



Inspector | Services

Cube ✓ | Static ☐
Tag Untagged | Layer Default

Transform
Position X 0 Y 2.58 Z 2.9
Rotation X 0 Y 0 Z 0
Scale X 3 Y 5 Z 0.2

Cube (Mesh Filter)
Mesh Cube

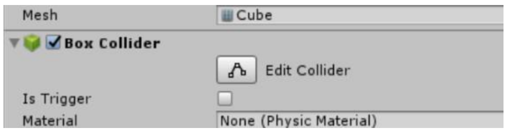
✓ Box Collider
Edit Collider
Is Trigger ☐
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1

✓ Mesh Renderer
Lighting
Materials
Dynamic Occluded ✓

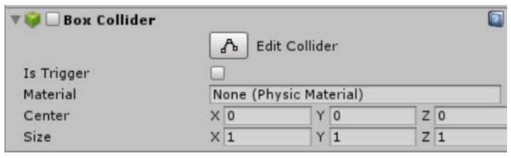
door
Shader Standard

Add Component

The transform is as expected, nothing to out of the ordinary, Cube mesh filter is fine, but notice how it automatically has a box collider on it that is active, if you make the box collider in active, i.e. remove the tick, and play. You will see the player able to move straight through the door object.



Mesh Cube
✓ Box Collider
Edit Collider
Is Trigger ☐
Material None (Physic Material)



☐ Box Collider
Edit Collider
Is Trigger ☐
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1

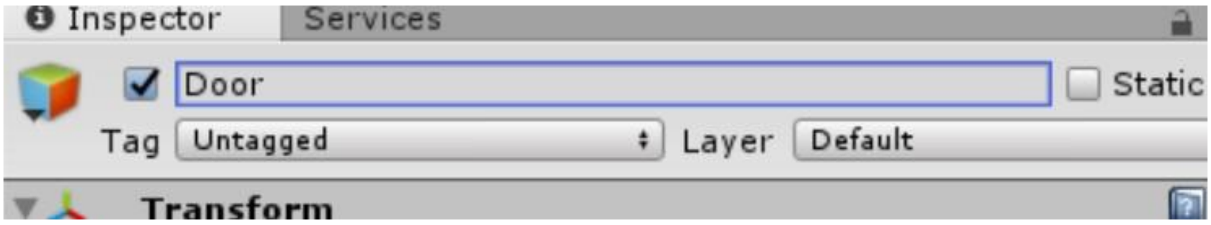
Do this and test.

Turn back on the box collider.

So, the collider is what allows our objects to interact with each other.

The plan is to add an addition collider around the door object (Now is a good time to rename cube to door) and have it trigger an event.

Rename cube to Door.

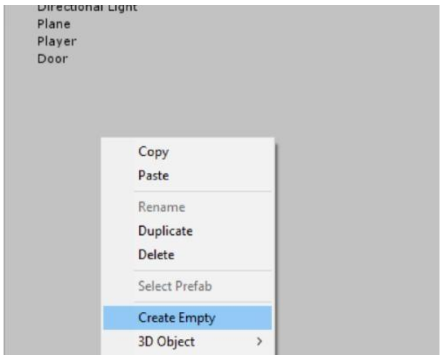


Inspector | Services

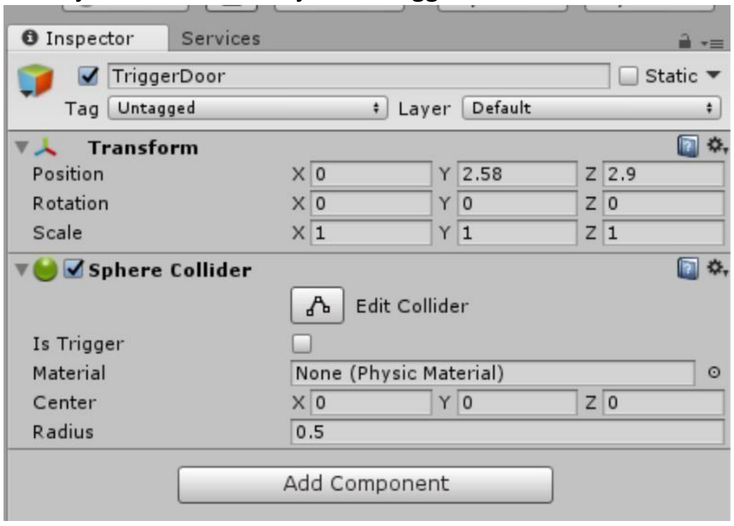
✓ Door | Static ☐
Tag Untagged | Layer Default

Transform

Right click on the hierarchy and add an empty game object.

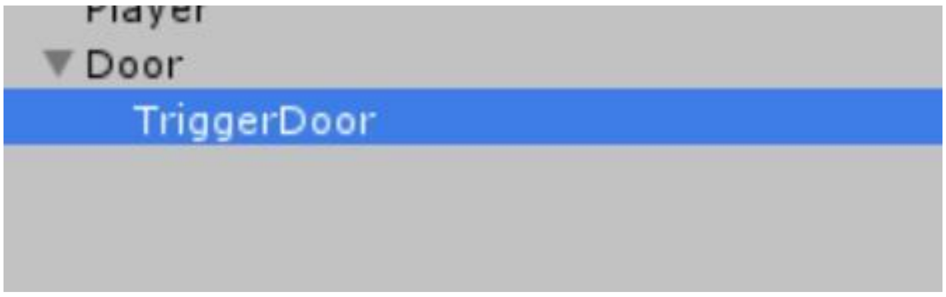


From here, we add a sphere collider onto this object, rename the object from GameObject to TriggerDoor.

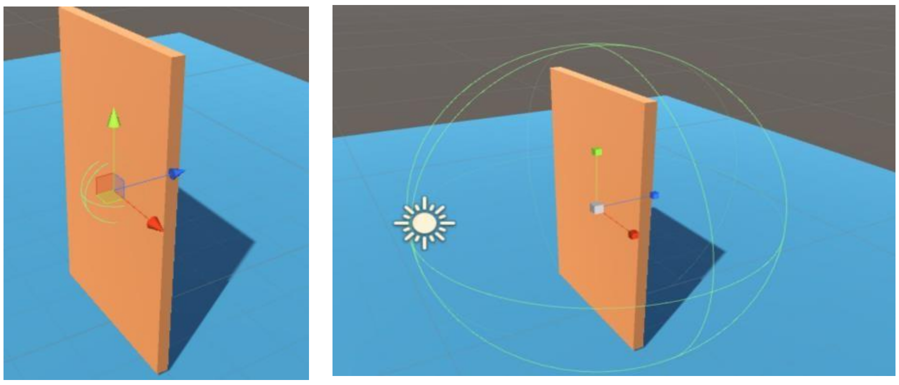


TriggerDoor
Tag Untagged Layer Default
Position X 0 Y 2.58 Z 2.9
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Is Trigger ☐
Material None (Physic Material)
Center X 0 Y 0 Z 0
Radius 0.5

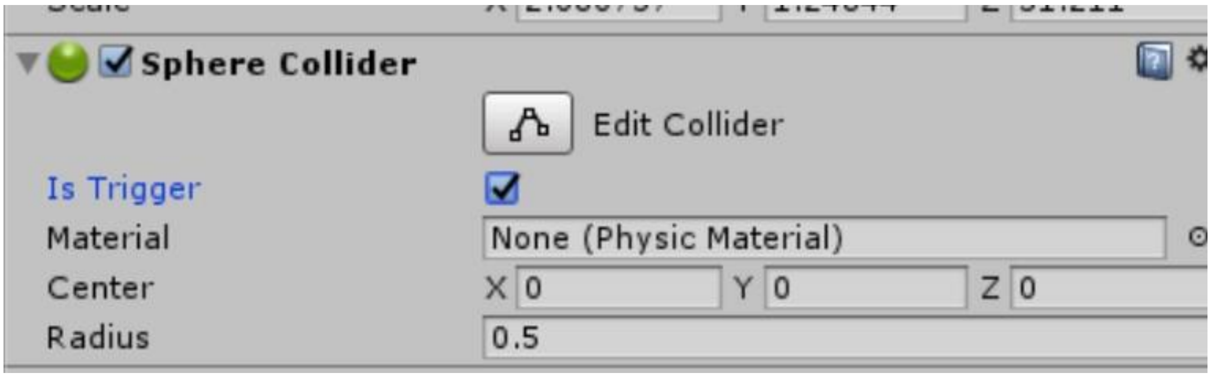
In the hierarchy drag the TriggerDoor on the Door object, in this way creating a parent/Child link.



Now, when looking at the scene, the sphere collider is too small to be of use, as we want it to trigger before the player hits the door, as such, scale it up.

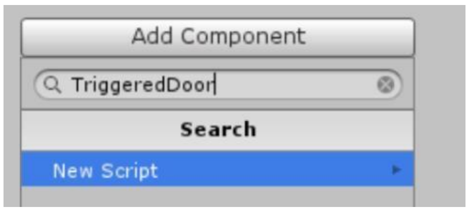


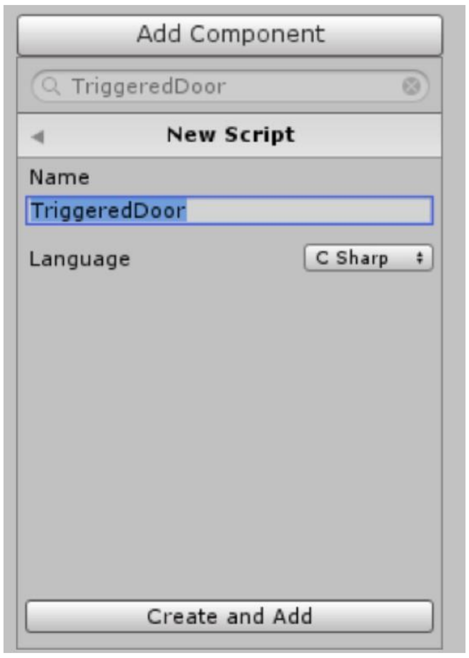
In the inspector, we need to change the sphere collider's attribute, so it works as a trigger and not a collider. Otherwise our player won't be able to pass through the door area. So, on the sphere collider, check the is trigger box.



Is Trigger ✓
Material None (Physic Material)
Center X 0 Y 0 Z 0
Radius 0.5

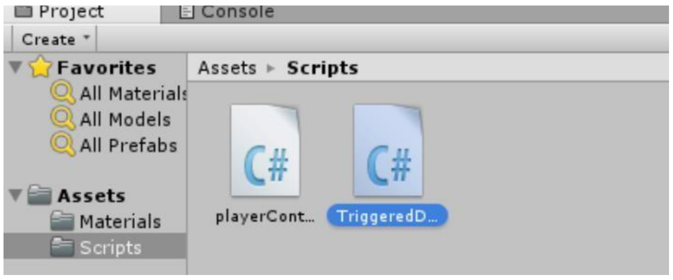
Now that we have prepared, let's add a script to the TriggerDoor object.

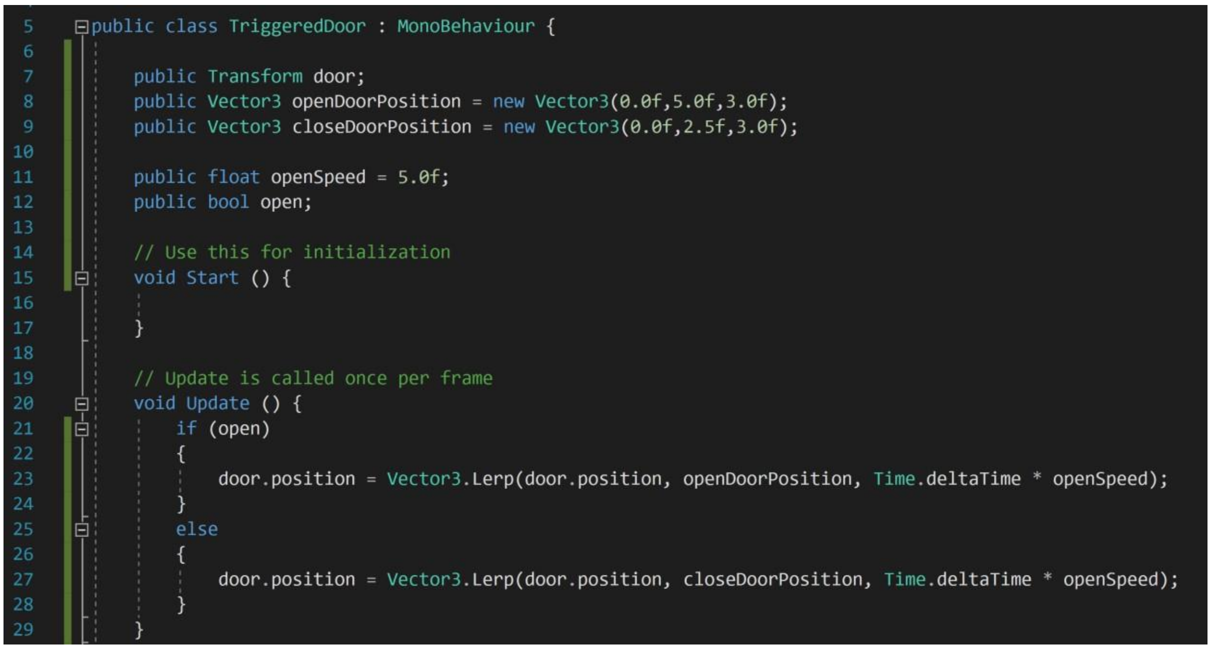




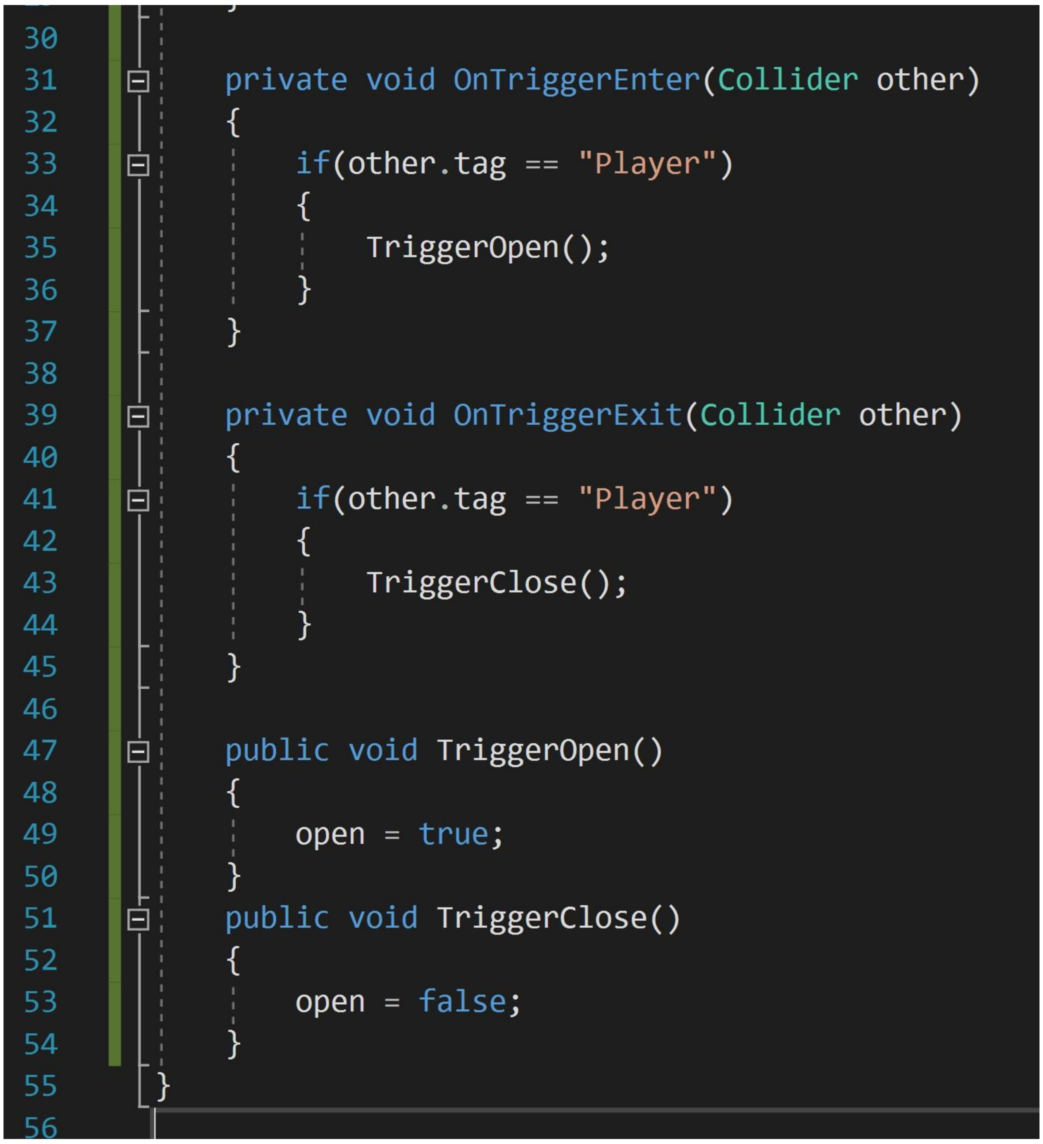
Name TriggeredDoor
Language C Sharp

Move the created script into the script file, then open it up it visual studio and add the following code.





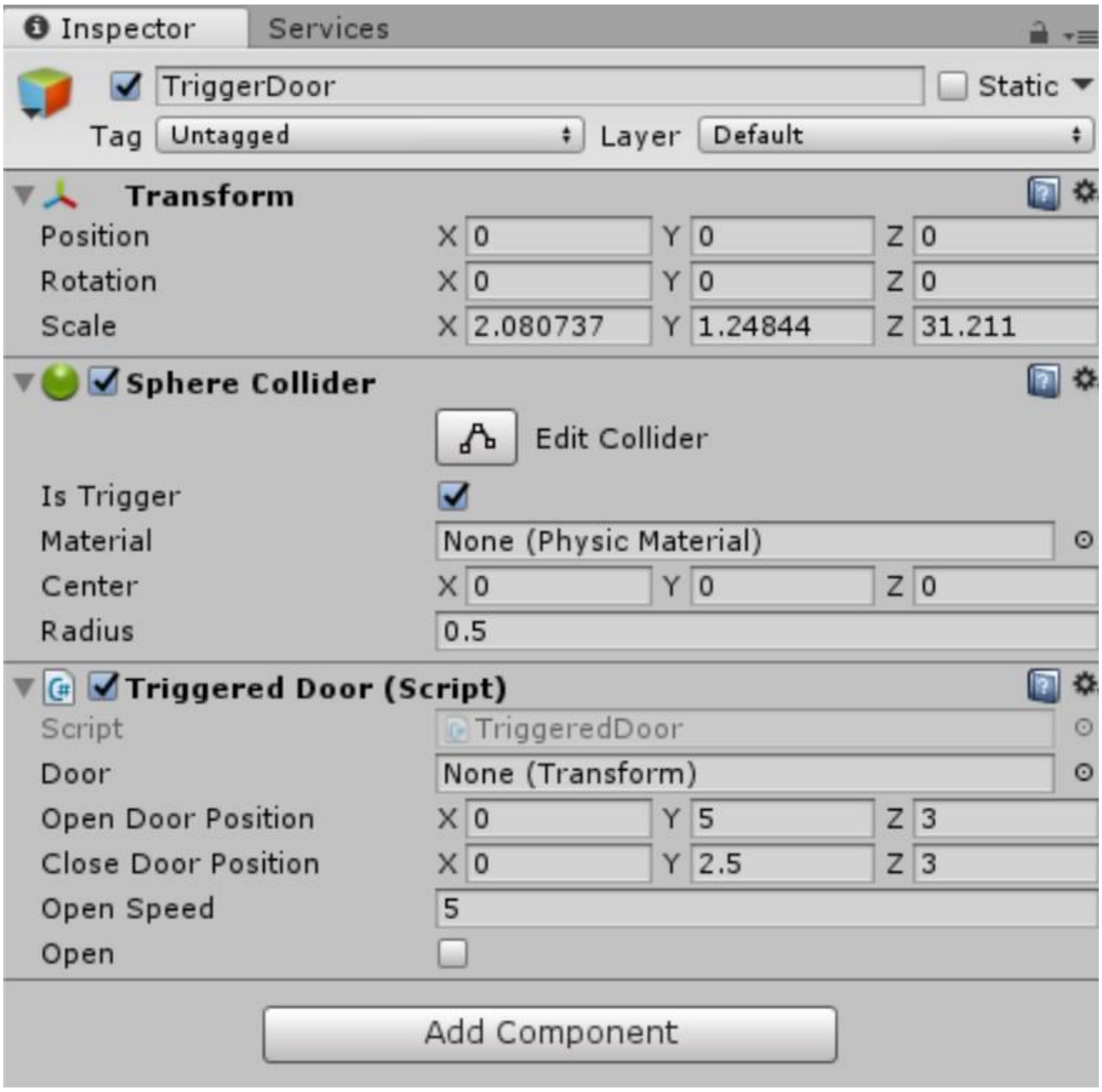
```
public class TriggeredDoor : MonoBehaviour {

    public Transform door;
    public Vector3 openDoorPosition = new Vector3(0.0f,5.0f,3.0f);
    public Vector3 closeDoorPosition = new Vector3(0.0f,2.5f,3.0f);

    public float openSpeed = 5.0f;
    public bool open;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        if (open)
        {
            door.position = Vector3.Lerp(door.position, openDoorPosition, Time.deltaTime * openSpeed);
        }
        else
        {
            door.position = Vector3.Lerp(door.position, closeDoorPosition, Time.deltaTime * openSpeed);
        }
    }
```



```

    private void OnTriggerEnter(Collider other)
    {
        if(other.tag == "Player")
        {
            TriggerOpen();
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if(other.tag == "Player")
        {
            TriggerClose();
        }
    }

    public void TriggerOpen()
    {
        open = true;
    }
    public void TriggerClose()
    {
        open = false;
    }
}

```

Save this and then go back to unity.

Ensure that the script is attached to the TriggerDoor Object



TriggerDoor ☑ · Static ☐
Tag: Untagged · Layer: Default

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 2.080737	Y 1.24844	Z 31.211

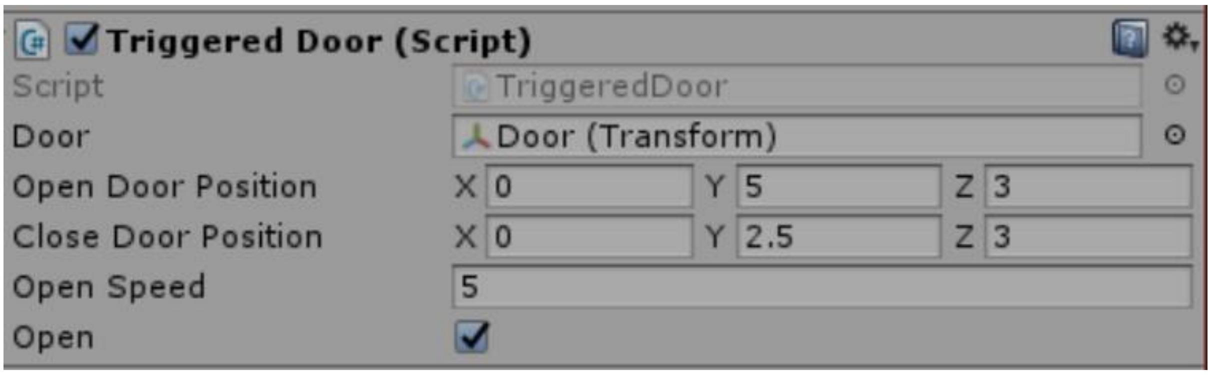
Sphere Collider ☑

Is Trigger	✓
Material	None (Physic Material)
Center	X 0 · Y 0 · Z 0
Radius	0.5

Triggered Door (Script) ☑

Script	TriggeredDoor
Door	None (Transform)
Open Door Position	X 0 · Y 5 · Z 3
Close Door Position	X 0 · Y 2.5 · Z 3
Open Speed	5
Open	☐

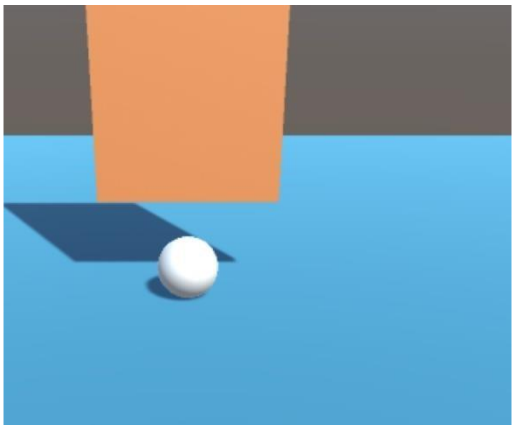
When looking at the script, notice how the Transform door code, aligns with None, this is awaiting a link to the actual object, drag the door object into it or use the target to locate the object.



Triggered Door (Script) ☑

Script	TriggeredDoor
Door	Door (Transform)
Open Door Position	X 0 · Y 5 · Z 3
Close Door Position	X 0 · Y 2.5 · Z 3
Open Speed	5
Open	✓

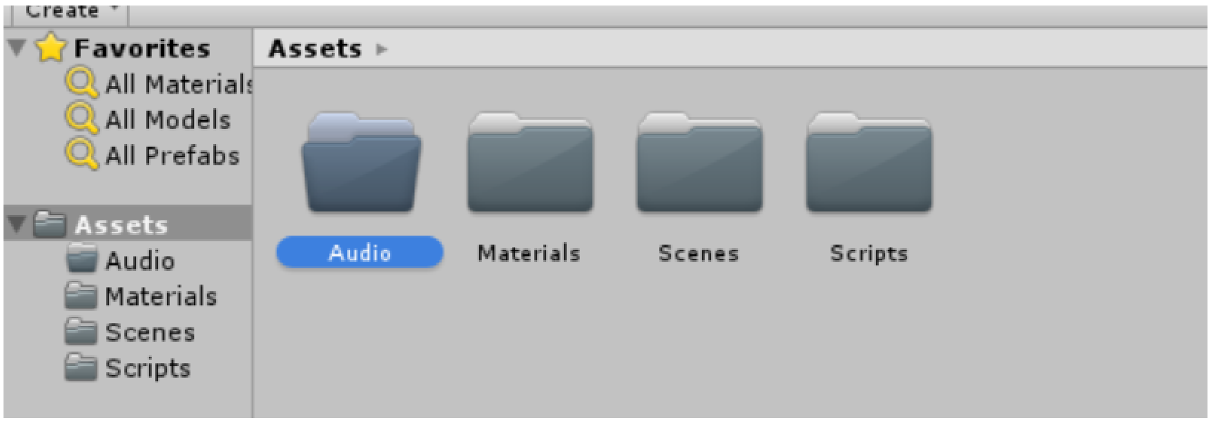
Run the program and you should see the door lifting up to allow the player sphere to travel under it.



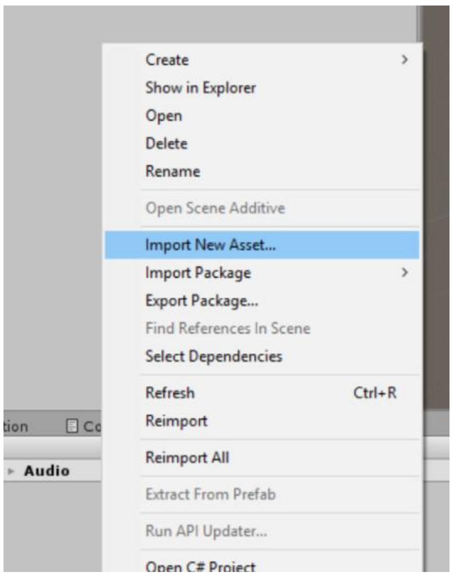
During testing the speed of 5 didn't open the door smoothly, when I dropped the Open Speed to 1, the door opening was much smoother.

Now that we have the animation, let's add some audio to the scene. Grab the audio file from L@G, or if you have a freesound.org account, you can get the same sound from here: https://freesound.org/people/mredig/sounds/120557/

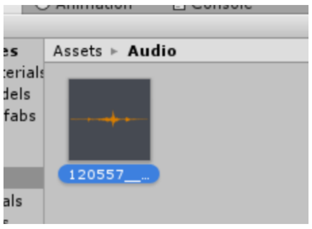
To start with add an audio folder and then import the audio into the system



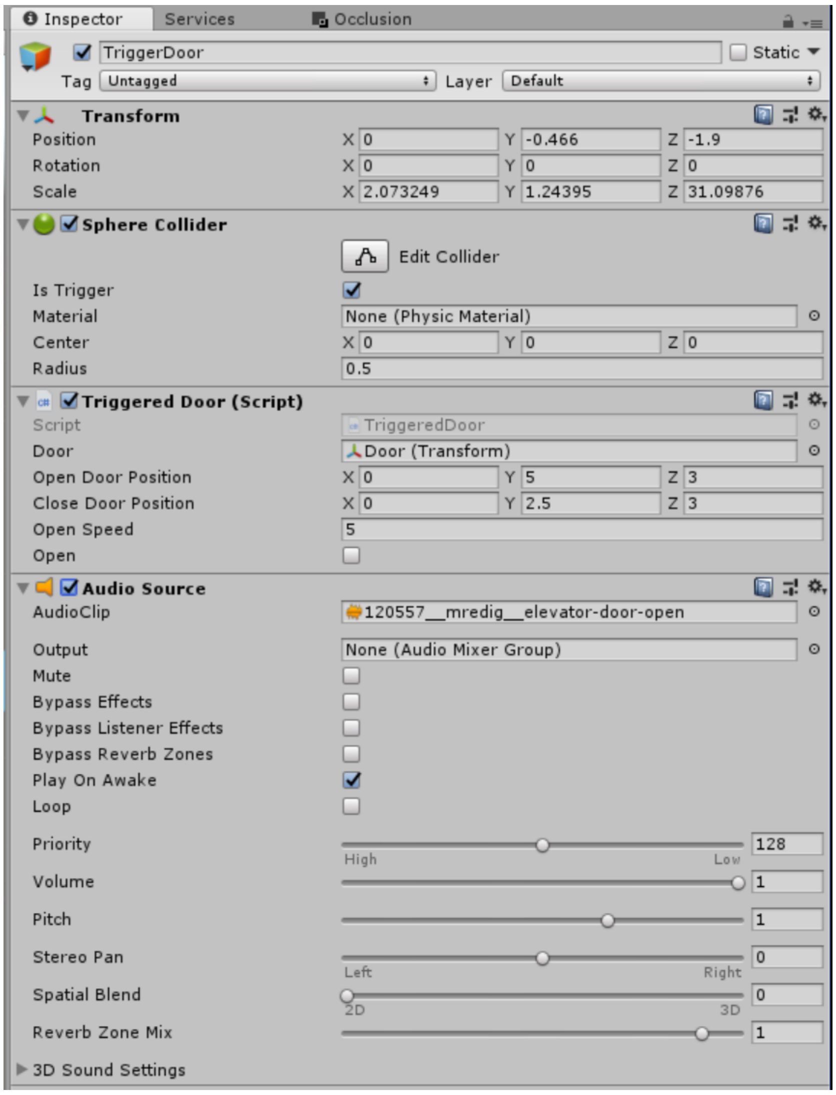
Right click in the audio folder and import new Asset



The audio folder will now look like:



On the TriggerDoor object, drag the audio file, it should look like this



TriggerDoor ☐ Static
Tag Untagged Layer Default

Transform

Position	X 0	Y -0.466	Z -1.9
Rotation	X 0	Y 0	Z 0
Scale	X 2.073249	Y 1.24395	Z 31.09876

Sphere Collider

Is Trigger	✓
Material	None (Physic Material)
Center	X 0 Y 0 Z 0
Radius	0.5

Triggered Door (Script)

Script	TriggeredDoor
Door	Door (Transform)
Open Door Position	X 0 Y 5 Z 3
Close Door Position	X 0 Y 2.5 Z 3
Open Speed	5
Open	☐

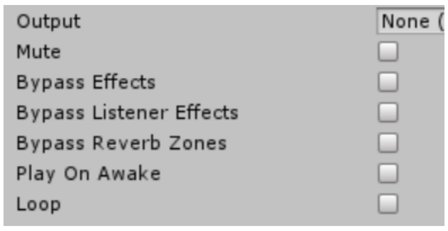
Audio Source

AudioClip	120557__mredig__elevator-door-open
Output	None (Audio Mixer Group)
Mute	☐
Bypass Effects	☐
Bypass Listener Effects	☐
Bypass Reverb Zones	☐
Play On Awake	✓
Loop	☐
Priority (High – Low)	128
Volume	1
Pitch	1
Stereo Pan (Left – Right)	0
Spatial Blend (2D – 3D)	0
Reverb Zone Mix	1

▸ 3D Sound Settings

Now, if you run the game, you will hear the noise the start with, this is because the play on awake is ticked. So, when the game loads, it runs the audio file.

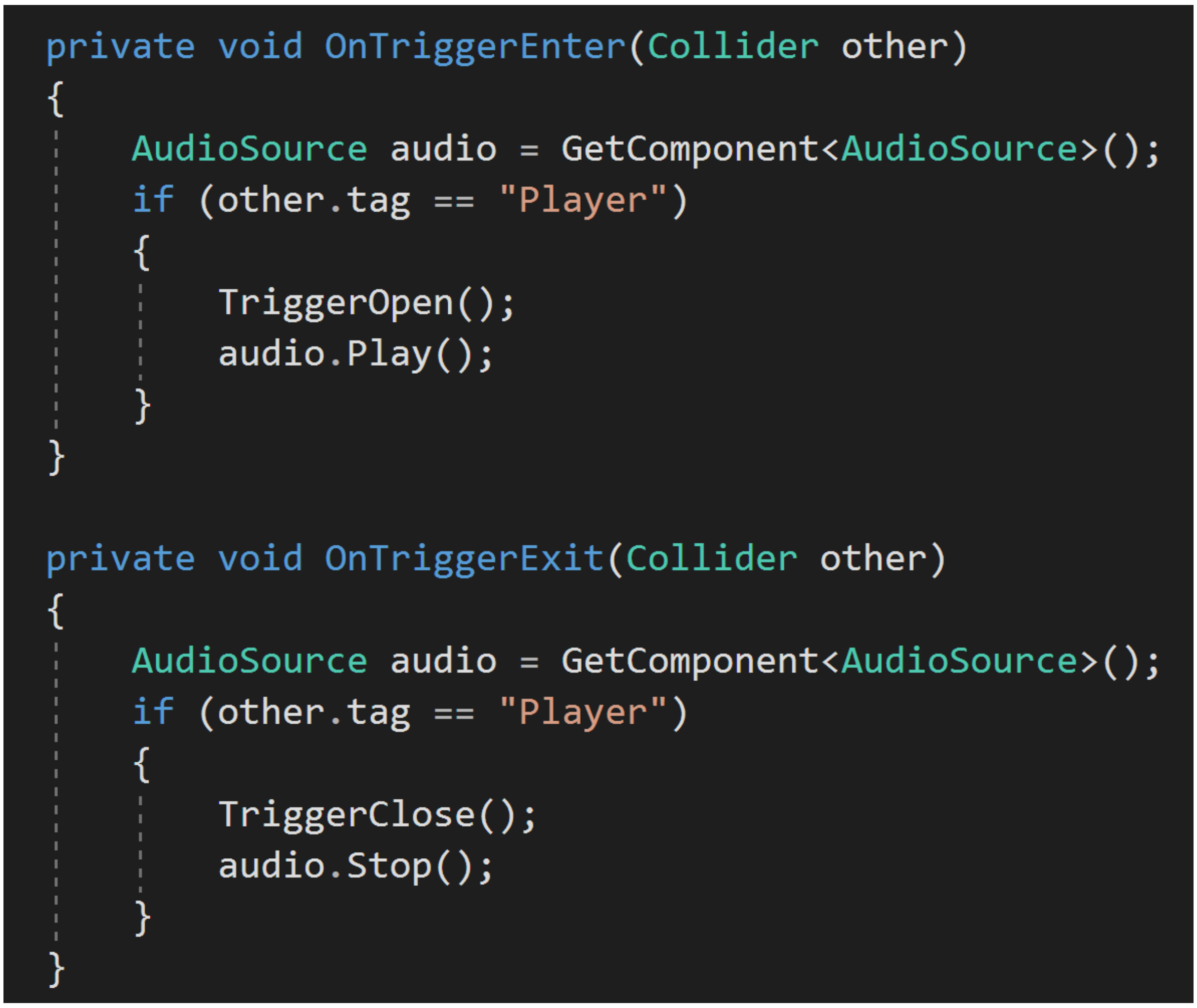
To stop this behaviour, untick play on awake.



Output	None (
Mute	☐
Bypass Effects	☐
Bypass Listener Effects	☐
Bypass Reverb Zones	☐
Play On Awake	☐
Loop	☐

Now, we have to add some code to the TriggeredDoor scrip to enable the audio to activate once we open the door.

To do this we need to access the audio source and make it occur in the correct position, that position being once we start the movement of the door. Modify the code to the following:



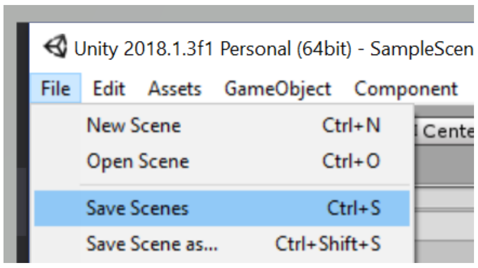
```
private void OnTriggerEnter(Collider other)
{
    AudioSource audio = GetComponent<AudioSource>();
    if (other.tag == "Player")
    {
        TriggerOpen();
        audio.Play();
    }
}

private void OnTriggerExit(Collider other)
{
    AudioSource audio = GetComponent<AudioSource>();
    if (other.tag == "Player")
    {
        TriggerClose();
        audio.Stop();
    }
}
```

The reason we have both Play() and Stop(), is so that we can have more control over the sound, also depending on the audio clip, you may need to trim it to make it occur at the speed you want.

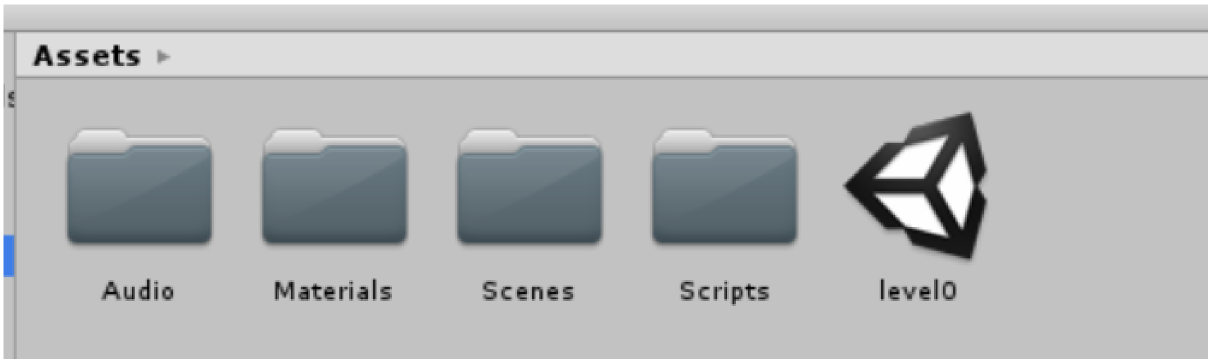
Once coded, save and test.

Using a trigger to change scenes

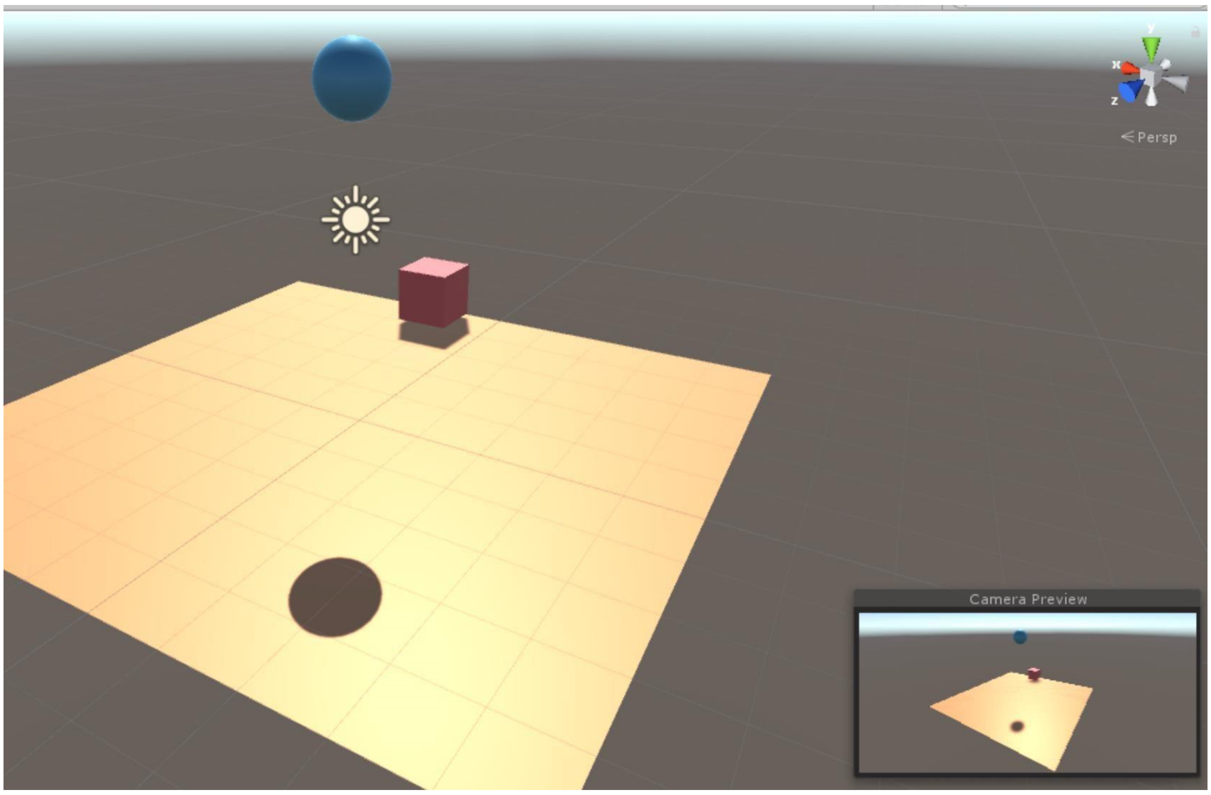
Using the same scene that we have already created, save the scene as level0,



This will default to the assets folder; Move into the Scenes folder.

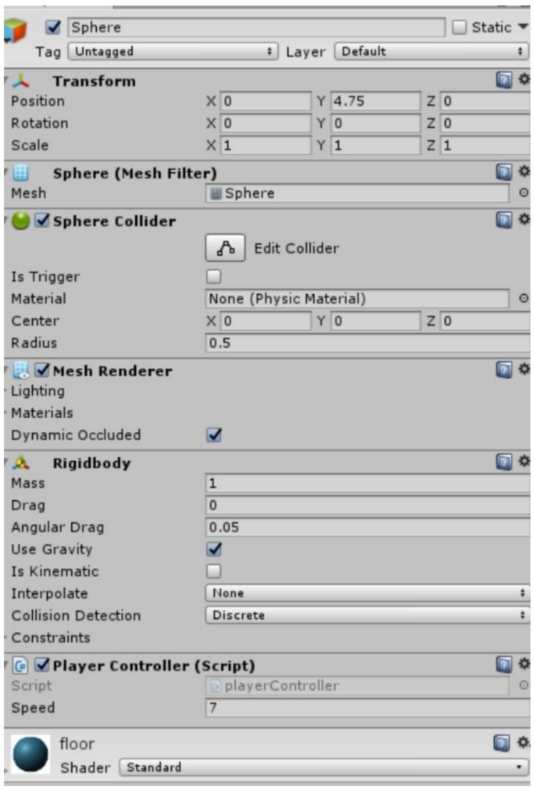


Then create a new scene as below and save as level1.



The floor material was applied to the player, the door material was applied to the plane, and a simple red was applied to a new cube.

Attach the script, playerController to the player to ensure that we can move the sphere.

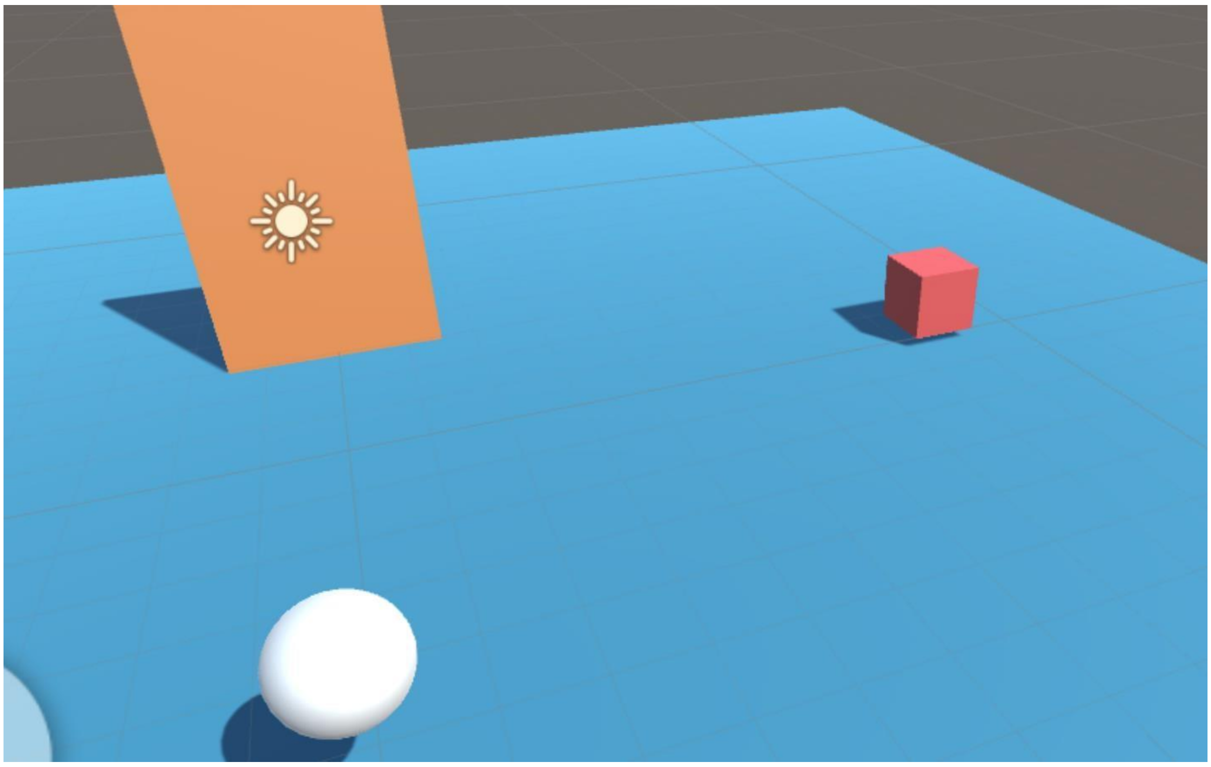


```
☑ Sphere                                   ☐ Static
Tag  Untagged            Layer  Default
Transform
Position          X 0      Y 4.75    Z 0
Rotation          X 0      Y 0       Z 0
Scale             X 1      Y 1       Z 1
Sphere (Mesh Filter)
Mesh              Sphere
☑ Sphere Collider
                  Edit Collider
Is Trigger        ☐
Material          None (Physic Material)
Center            X 0      Y 0       Z 0
Radius            0.5
☑ Mesh Renderer
Lighting
Materials
Dynamic Occluded  ☑
Rigidbody
Mass              1
Drag              0
Angular Drag      0.05
Use Gravity       ☑
Is Kinematic      ☐
Interpolate       None
Collision Detection Discrete
Constraints
☑ Player Controller (Script)
Script            playerController
Speed             7
floor
Shader  Standard
```

Save and test, if the rotation is off for the camera and the controls seem inverted, it's ok, this is just for testing purposes.

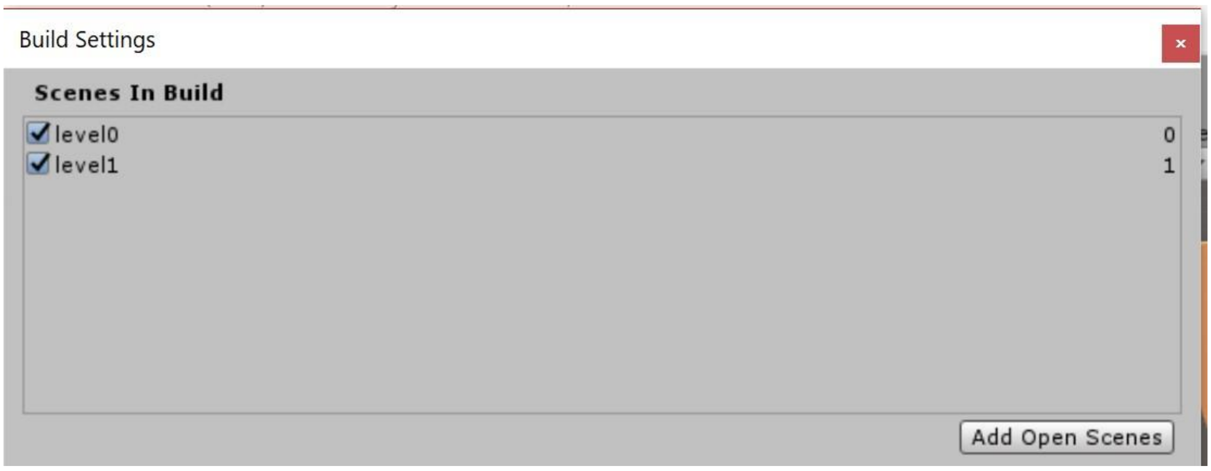
Save this scene and go back to level0.

On level0, create a new cube. And position it like so.



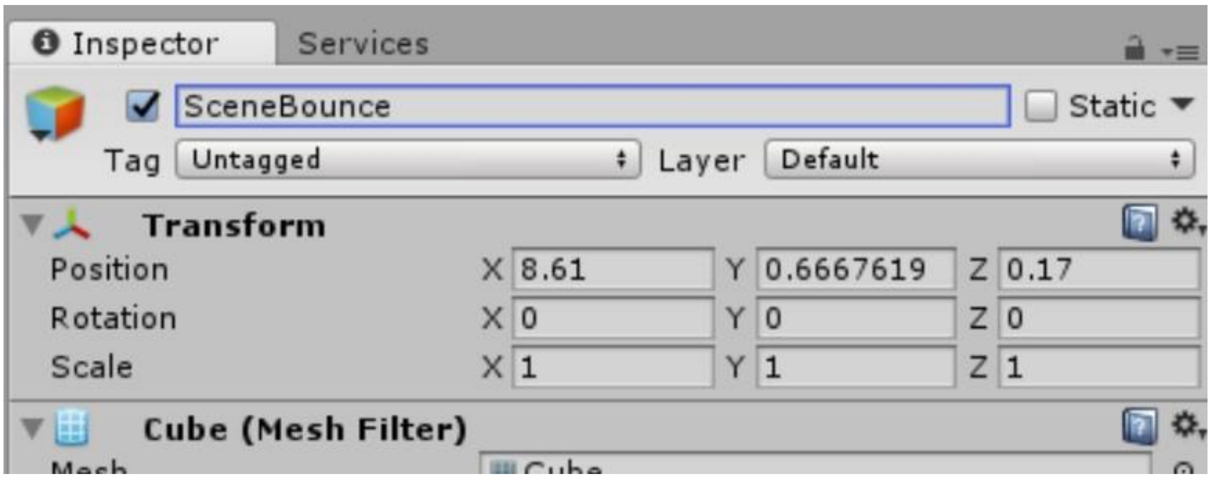
The goal is to assign a collider, just like with the door, except instead of moving the object we will jump between scenes.

To start with add both scenes to the build settings.



✓ level0	0
✓ level1	1

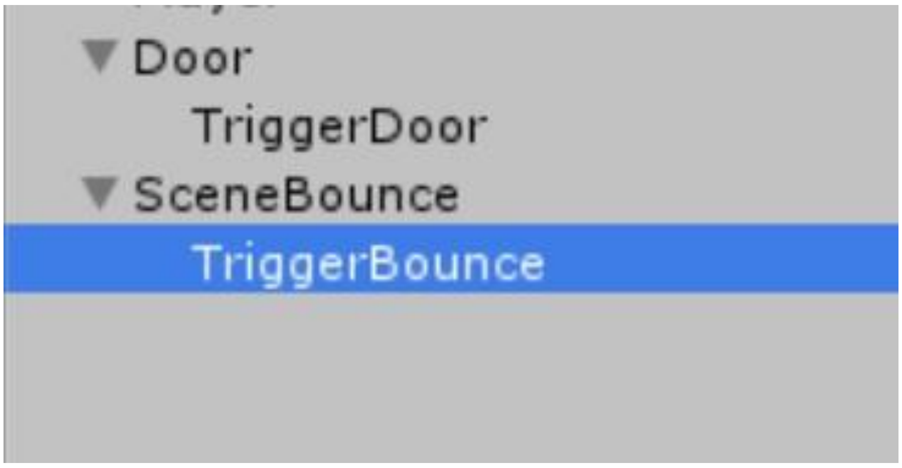
Now, let’s create a new Empty Game object as we did before, name it TriggerBounce and place it on the cube, then and add a sphere collider to the object. Rename the cube to SceneBounce.



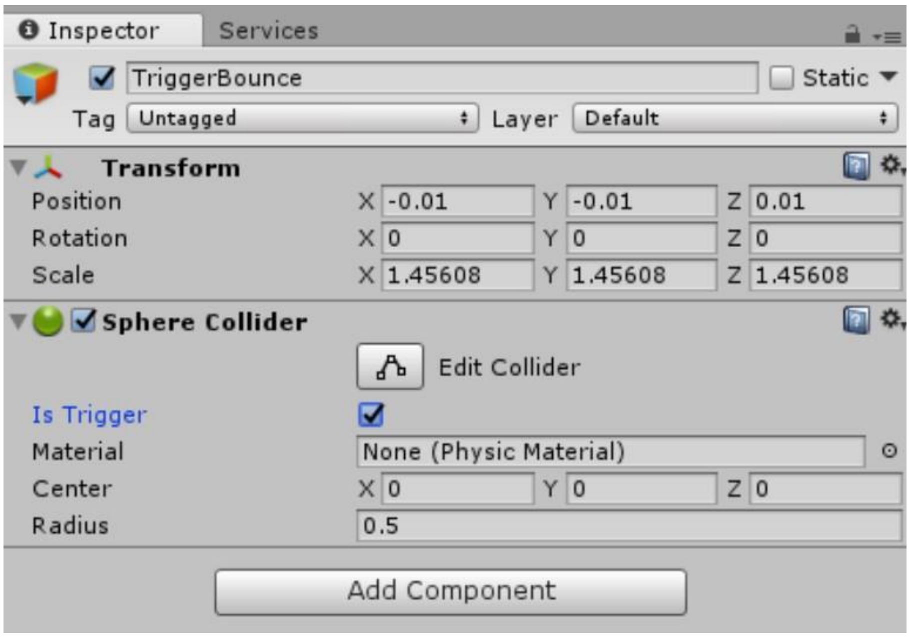
SceneBounce
Tag Untagged Layer Default

Position	X 8.61	Y 0.6667619	Z 0.17
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Drag the TriggerBounce onto SceneBounce to make it a child.



Ensure you have ticked Is Trigger on TriggerBounce


Inspector | Services

TriggerBounce ☐ Static

Tag: Untagged | Layer: Default

Transform

	X	Y	Z
Position	-0.01	-0.01	0.01
Rotation	0	0	0
Scale	1.45608	1.45608	1.45608

Sphere Collider

Edit Collider

Is Trigger ✓

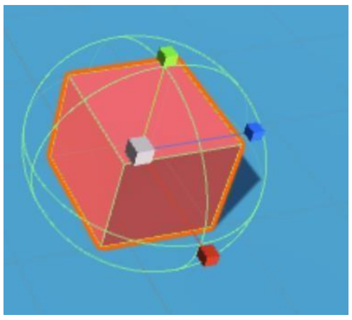
Material: None (Physic Material)

Center: X 0 Y 0 Z 0

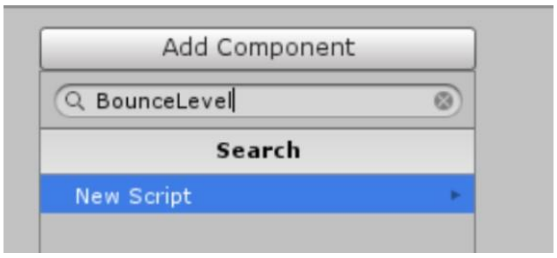
Radius: 0.5

Add Component

Ensure the sphere collider encapsulates the cube.



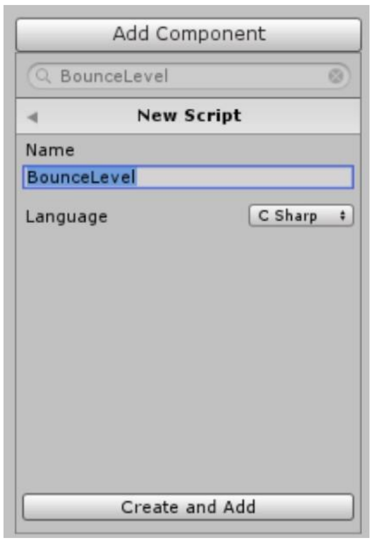
On the SceneBounce add a new script called BounceLevel


Add Component

BounceLevel

Search

New Script


Add Component

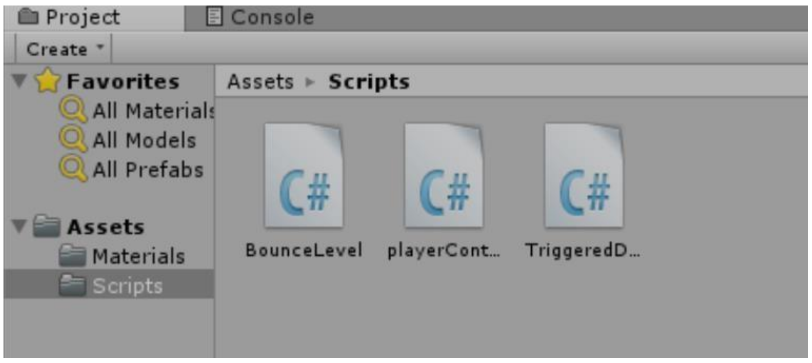
BounceLevel

New Script

Name: BounceLevel

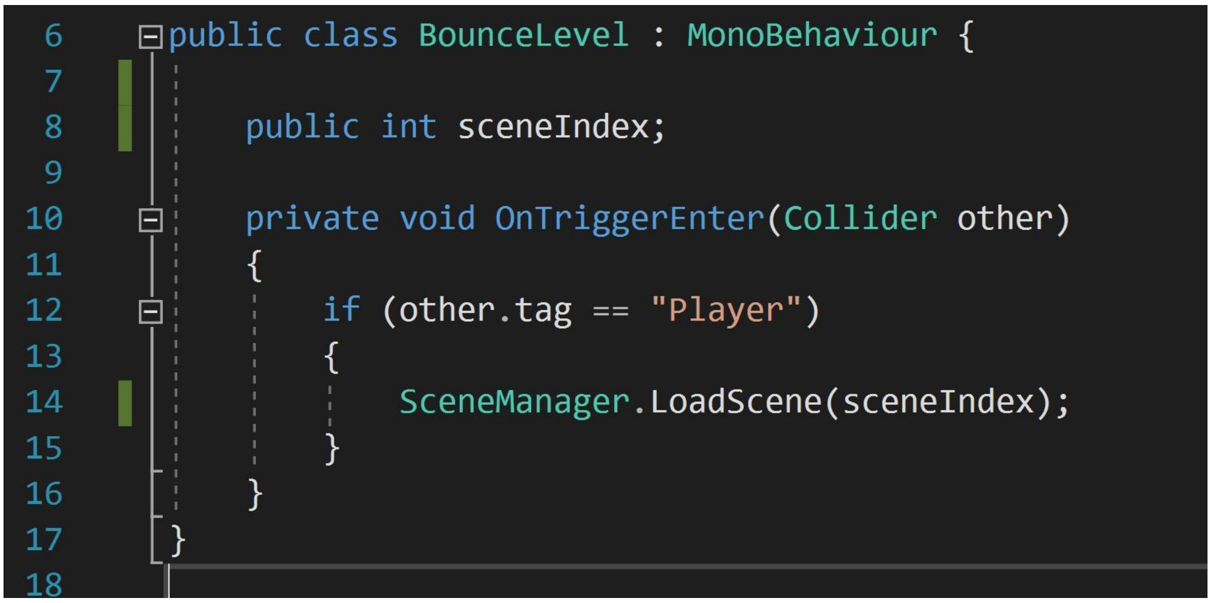
Language: C Sharp

Create and Add



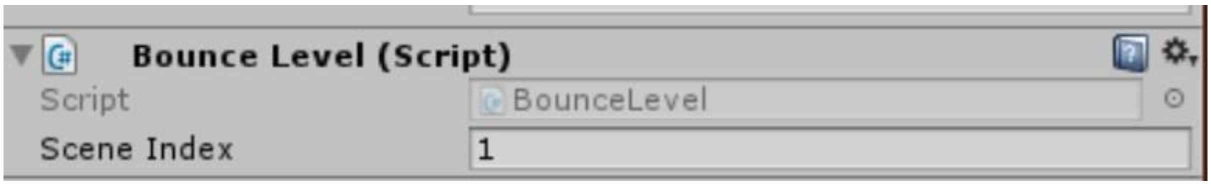
Once you have done this, move the script into the script folder and then open it up in Visual Studio.

Add the following code.



```
public class BounceLevel : MonoBehaviour {

    public int sceneIndex;

    private void OnTriggerEnter(Collider other)
    {
        if (other.tag == "Player")
        {
            SceneManager.LoadScene(sceneIndex);
        }
    }
}
```

Back in Unity, make a change to the inspector to ensure that we are sending the correct correct scene index to change to.



```
Bounce Level (Script)
Script          BounceLevel
Scene Index     1
```

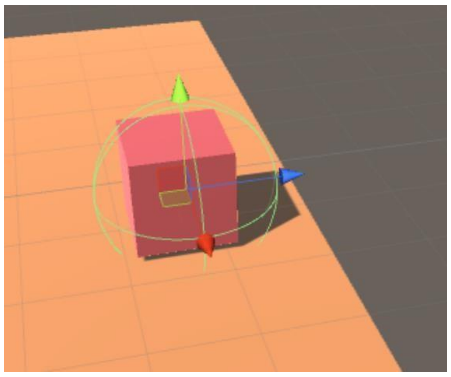
Save and Test, you should be able to walk the player into the cube and bounce to the next scene.

Repeat the process on level1 for the other cube.

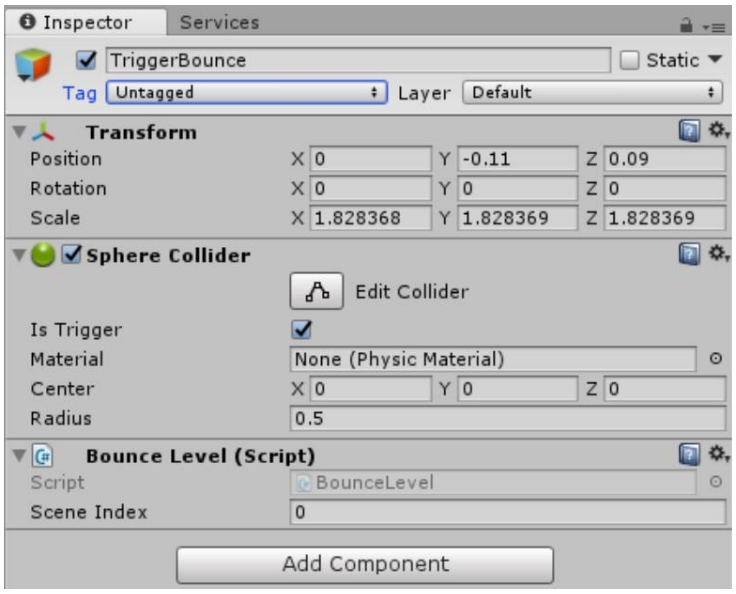
Steps are:

- Empty GameObject
- Apply Sphere collider
- Ensure is trigger is checked
- Link BounceLevel Script
- Ensure the correct Scene Index is being applied.

In the scene



In the Inspector



Inspector Services

TriggerBounce Static

Tag Untagged Layer Default

Transform

Position	X 0	Y -0.11	Z 0.09
Rotation	X 0	Y 0	Z 0
Scale	X 1.828368	Y 1.828369	Z 1.828369

Sphere Collider

Edit Collider

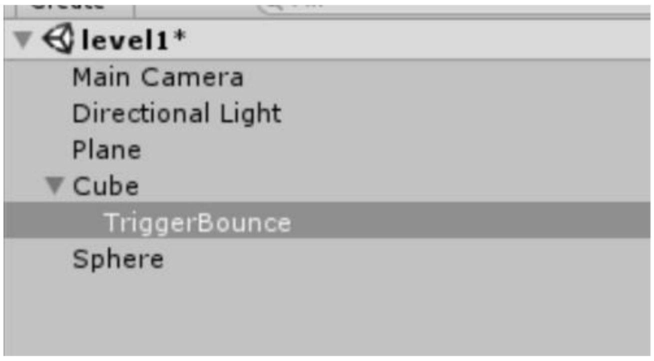
Is Trigger	✓
Material	None (Physic Material)
Center	X 0 Y 0 Z 0
Radius	0.5

Bounce Level (Script)

Script	BounceLevel
Scene Index	0

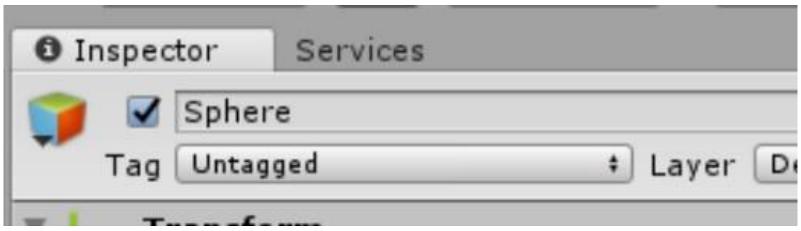
Add Component

In the hierarchy



```
level1*
  Main Camera
  Directional Light
  Plane
  Cube
    TriggerBounce
  Sphere
```

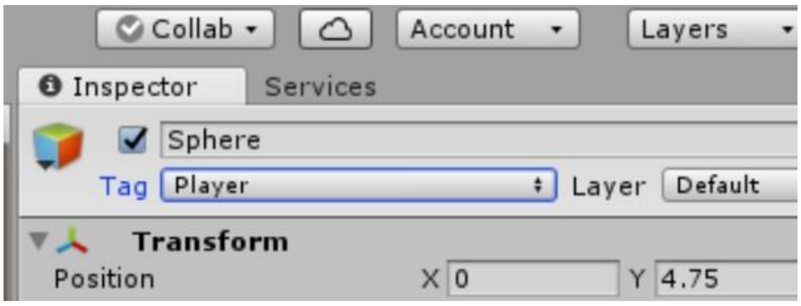
If it doesn't work, ensure that the sphere is tagged as Player.



Inspector Services

Sphere

Tag Untagged Layer



Collab Account Layers

Inspector Services

Sphere

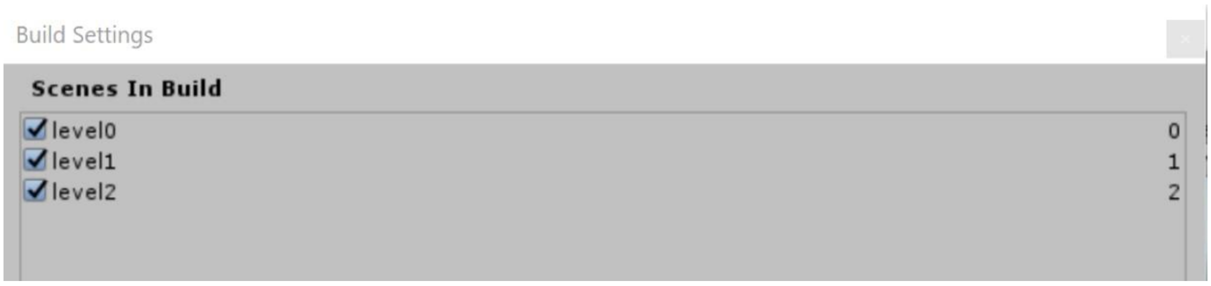
Tag Player Layer Default

Transform

Position X 0 Y 4.75

Adding simple NPCs

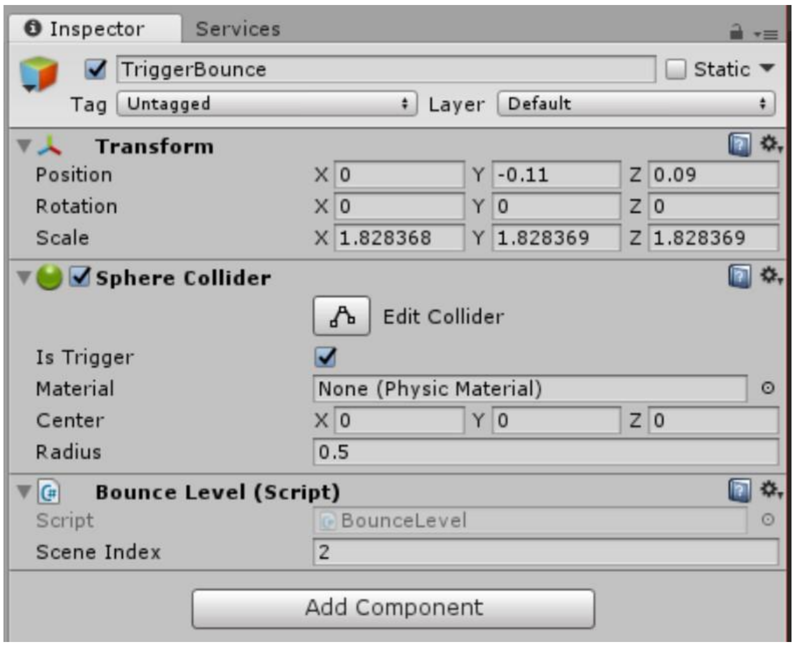
Using the same project, create a new scene and save it as level 2, then add it to the build settings and change the bounce level from 0 to 2 in the level1 scene. This way we can move from level 0->1>2



Build Settings

Scenes In Build

✓ level0	0
✓ level1	1
✓ level2	2



TriggerBounce ✓ — Static ☐
Tag Untagged — Layer Default

Transform

	X	Y	Z
Position	0	-0.11	0.09
Rotation	0	0	0
Scale	1.828368	1.828369	1.828369

Sphere Collider ✓

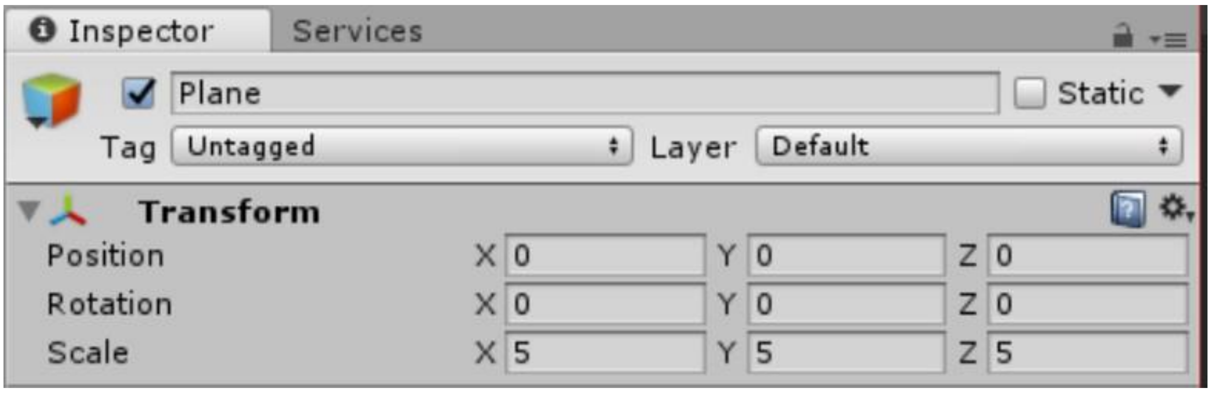
Is Trigger ✓
Material None (Physic Material)
Center X 0 Y 0 Z 0
Radius 0.5

Bounce Level (Script)

Script BounceLevel
Scene Index 2

On level2 do the following:

Add a plane, scale it up 5 and use the same door material on it.

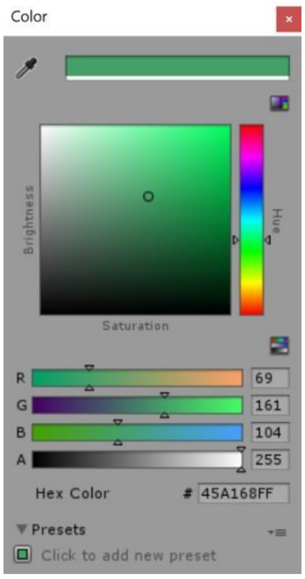


Plane ✓ — Static ☐
Tag Untagged — Layer Default

Transform

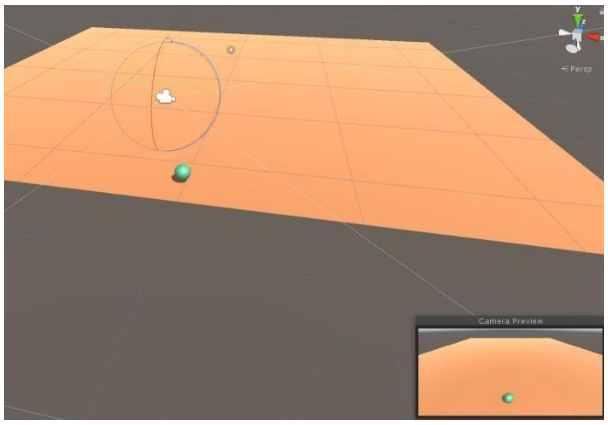
	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	5	5	5

Create sphere, assign it a new green Material.

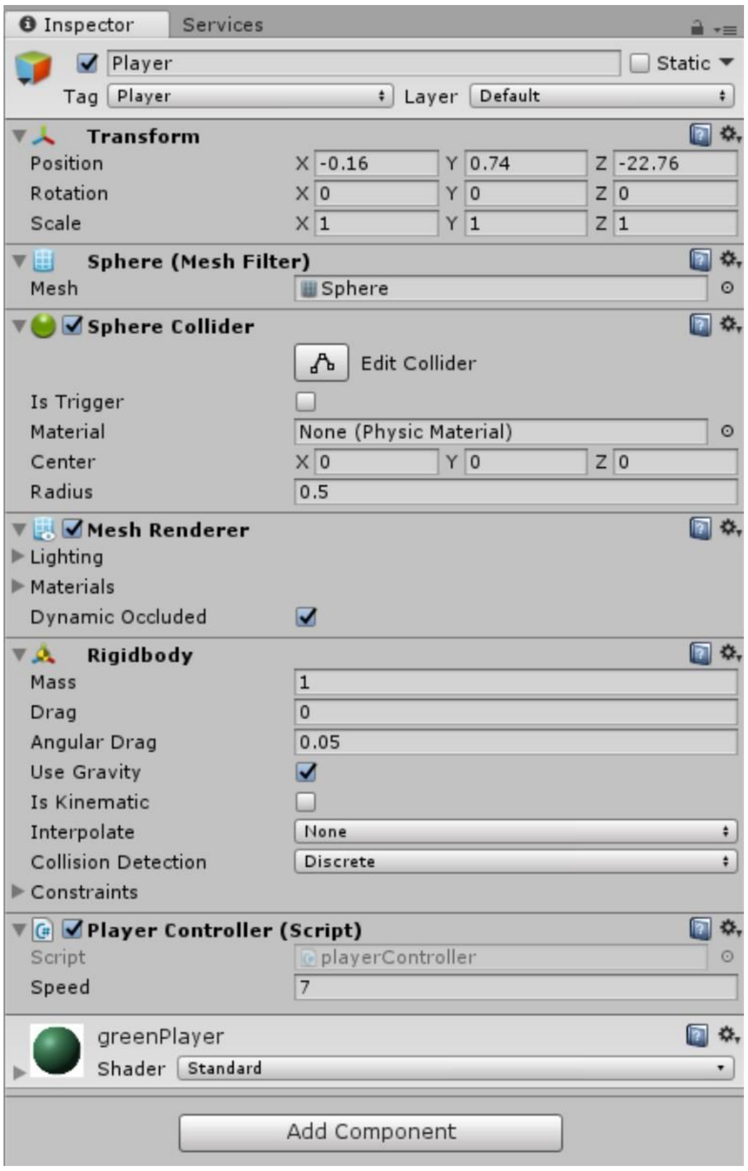


R 69
G 161
B 104
A 255
Hex Color # 45A168FF

Position the camera above and behind the player so we can see what is going to occur.



Apply a rigid body and movement script to the player and ensure the sphere is tagged as player.


Inspector | Services

Player — Static

Tag: Player — Layer: Default

Transform
Position: X -0.16, Y 0.74, Z -22.76
Rotation: X 0, Y 0, Z 0
Scale: X 1, Y 1, Z 1

Sphere (Mesh Filter)
Mesh: Sphere

Sphere Collider
Edit Collider
Is Trigger: ☐
Material: None (Physic Material)
Center: X 0, Y 0, Z 0
Radius: 0.5

Mesh Renderer
Lighting
Materials
Dynamic Occluded: ✓

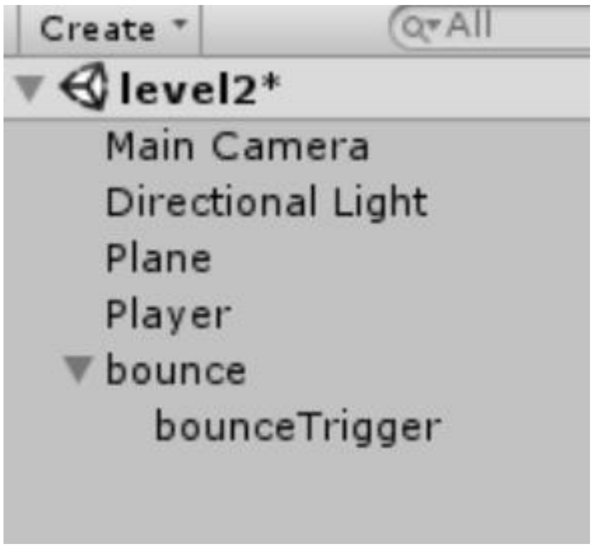
Rigidbody
Mass: 1
Drag: 0
Angular Drag: 0.05
Use Gravity: ✓
Is Kinematic: ☐
Interpolate: None
Collision Detection: Discrete
Constraints

Player Controller (Script)
Script: playerController
Speed: 7

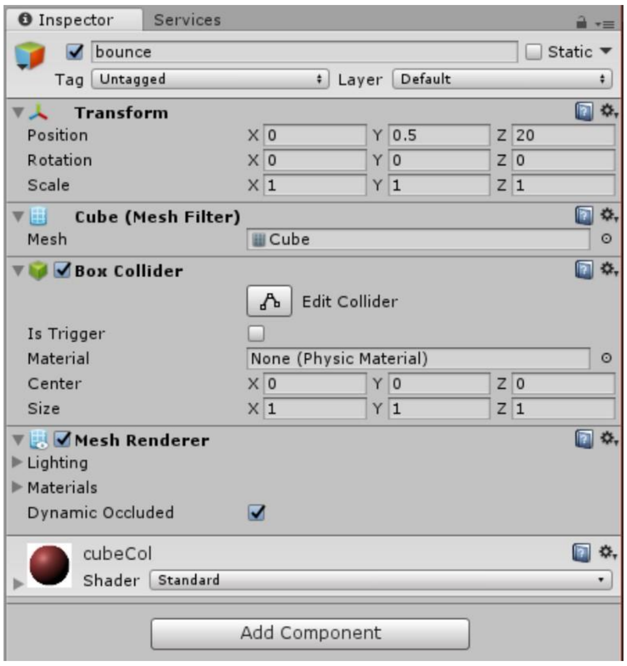
greenPlayer
Shader: Standard

Add Component

Next create a cube, call it bounce and apply an empty game object to it, apply the scripts to make it so if the player runs into it, it will bounce them to level0. Exactly what we have already done so far.


Create ▾ | All

level2*
- Main Camera
- Directional Light
- Plane
- Player
- bounce
  - bounceTrigger


Inspector | Services

☑ bounce ☐ Static
Tag Untagged Layer Default

Transform

Position	X	0	Y	0.5	Z	20
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Cube (Mesh Filter)
Mesh Cube

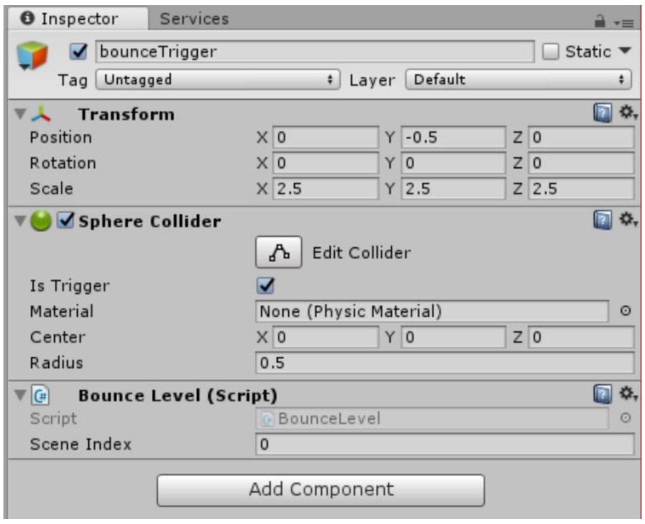
☑ Box Collider
Edit Collider
Is Trigger ☐
Material None (Physic Material)

Center	X	0	Y	0	Z	0
Size	X	1	Y	1	Z	1

☑ Mesh Renderer
▸ Lighting
▸ Materials
Dynamic Occluded ☑

cubeCol
Shader Standard

Add Component


Inspector | Services

☑ bounceTrigger ☐ Static
Tag Untagged Layer Default

Transform

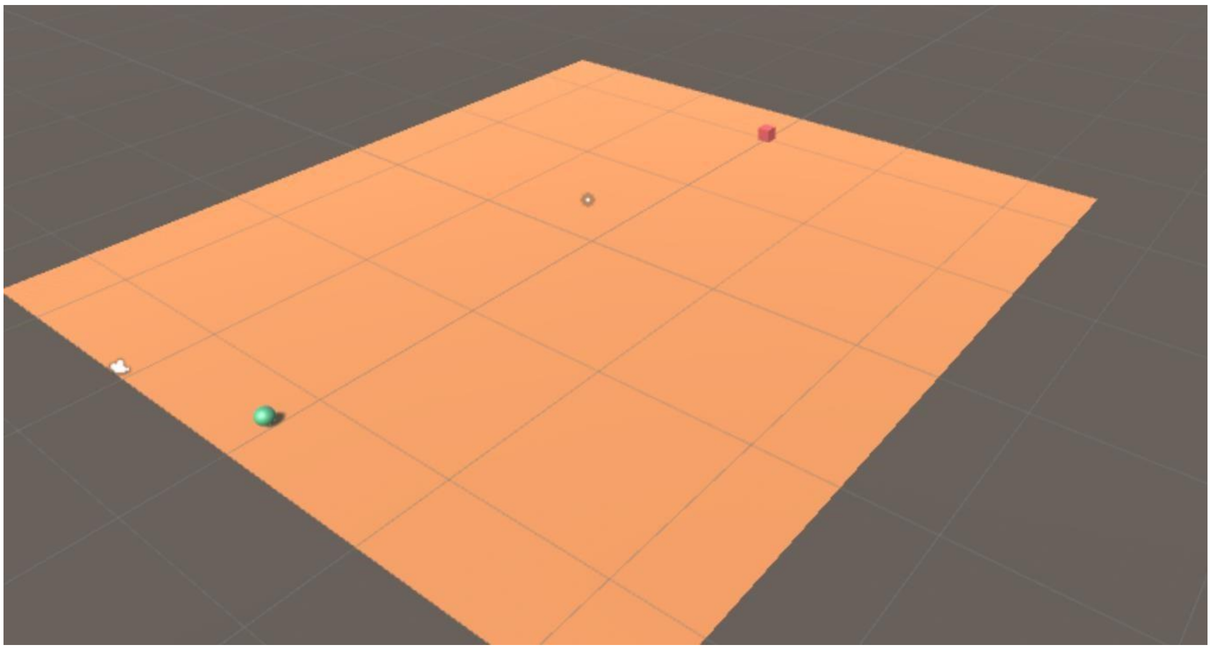
Position	X	0	Y	-0.5	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	2.5	Y	2.5	Z	2.5

☑ Sphere Collider
Edit Collider
Is Trigger ☑
Material None (Physic Material)
Center X 0 Y 0 Z 0
Radius 0.5

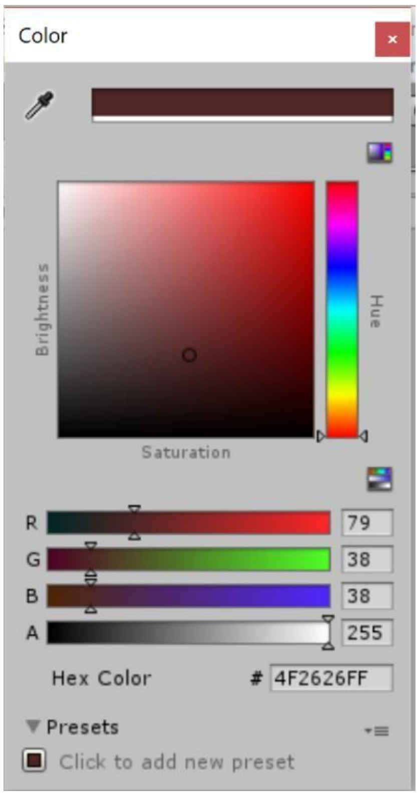
Bounce Level (Script)
Script BounceLevel
Scene Index 0

Add Component

You have the following scene

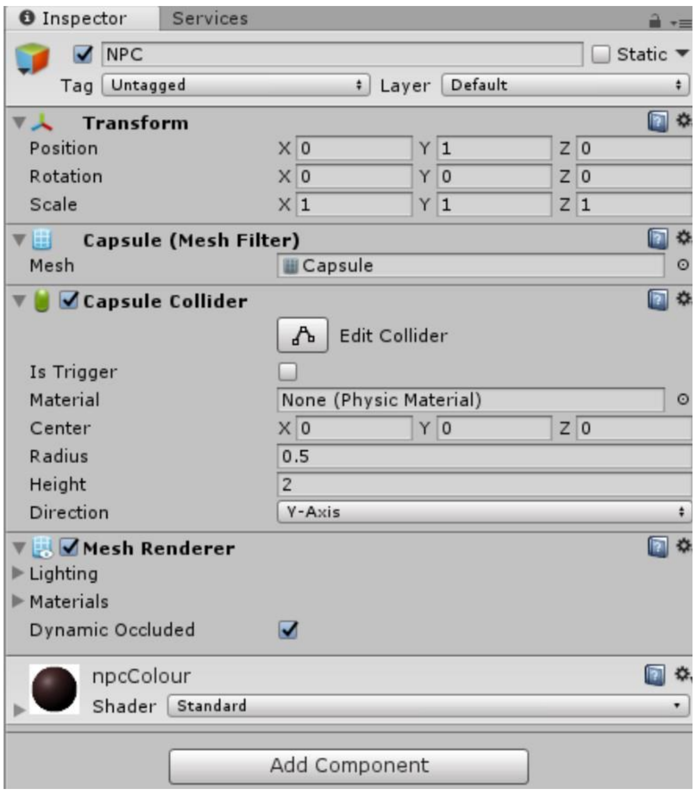


Next, we add a NPC, as we are not bringing in fully fleshed out characters for this, create a capsule, name it NPC and apply a new colour to it.



Color

R 79
G 38
B 38
A 255
Hex Color # 4F2626FF
Presets
Click to add new preset



Inspector | Services

NPC ☐ Static
Tag Untagged | Layer Default

Transform
Position X 0 Y 1 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Capsule (Mesh Filter)
Mesh Capsule

✓ Capsule Collider
Edit Collider
Is Trigger ☐
Material None (Physic Material)
Center X 0 Y 0 Z 0
Radius 0.5
Height 2
Direction Y-Axis

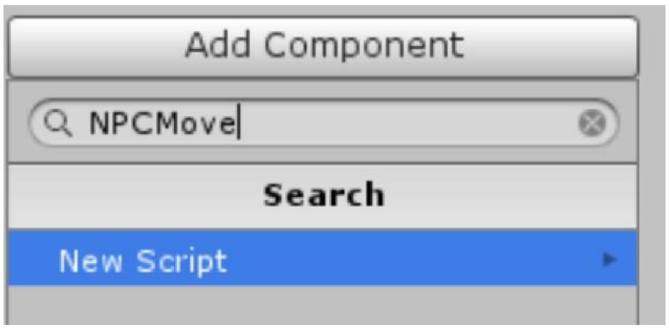
✓ Mesh Renderer
Lighting
Materials
Dynamic Occluded ✓

npcColour
Shader Standard

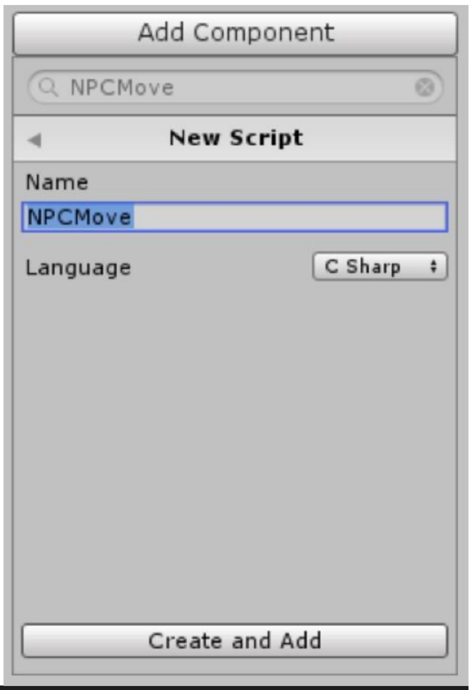
Add Component

Our NPC will go from side to side and if he collides with the player, the player will be sent back to the start scene.

So, the first thing we need to do, is to create a script in which the NPC can move.



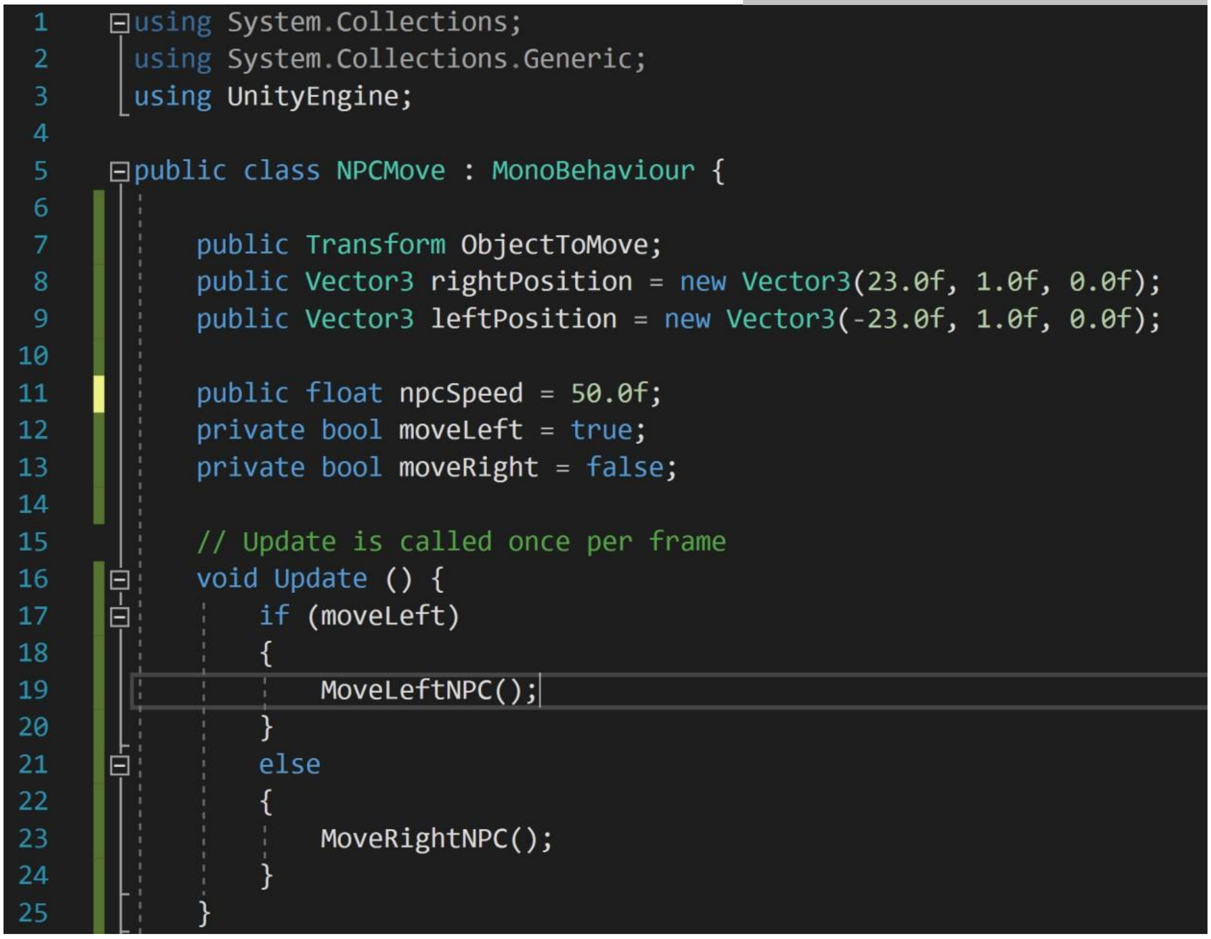
Move it into the scripts folder.



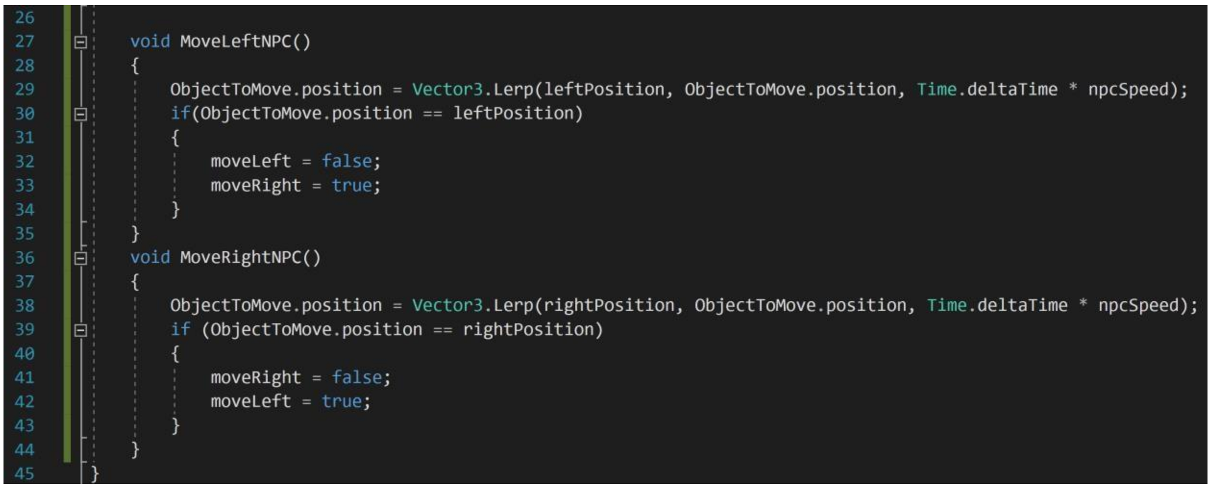
Name: NPCMove

Language: C Sharp

Add the following code

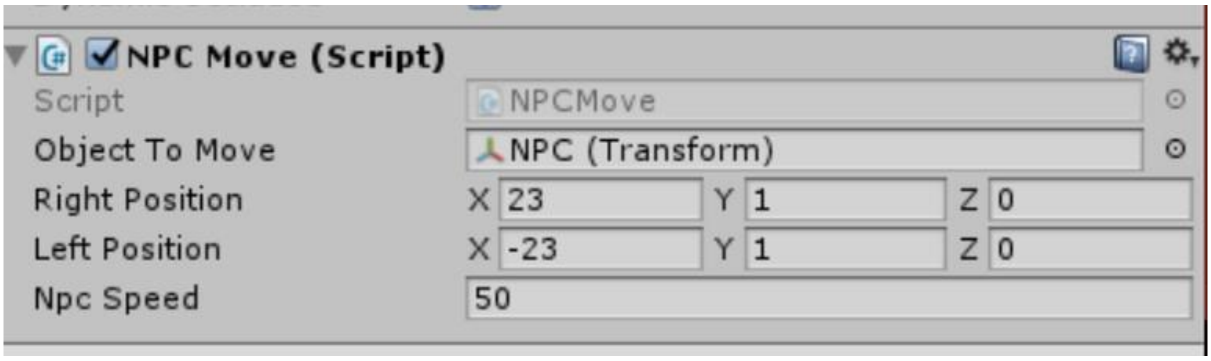


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class NPCMove : MonoBehaviour {

    public Transform ObjectToMove;
    public Vector3 rightPosition = new Vector3(23.0f, 1.0f, 0.0f);
    public Vector3 leftPosition = new Vector3(-23.0f, 1.0f, 0.0f);

    public float npcSpeed = 50.0f;
    private bool moveLeft = true;
    private bool moveRight = false;

    // Update is called once per frame
    void Update () {
        if (moveLeft)
        {
            MoveLeftNPC();
        }
        else
        {
            MoveRightNPC();
        }
    }
```


```
    void MoveLeftNPC()
    {
        ObjectToMove.position = Vector3.Lerp(leftPosition, ObjectToMove.position, Time.deltaTime * npcSpeed);
        if(ObjectToMove.position == leftPosition)
        {
            moveLeft = false;
            moveRight = true;
        }
    }
    void MoveRightNPC()
    {
        ObjectToMove.position = Vector3.Lerp(rightPosition, ObjectToMove.position, Time.deltaTime * npcSpeed);
        if (ObjectToMove.position == rightPosition)
        {
            moveRight = false;
            moveLeft = true;
        }
    }
}
```

Once done, save and then go back to unity.

In the inspector, assign the NPC object to the Object To Move section of the script and test. You might need to change the speed value.


NPC Move (Script)

Script	NPCMove
Object To Move	NPC (Transform)
Right Position	X 23 Y 1 Z 0
Left Position	X -23 Y 1 Z 0
Npc Speed	50

You should have the capsule moving from side to side.

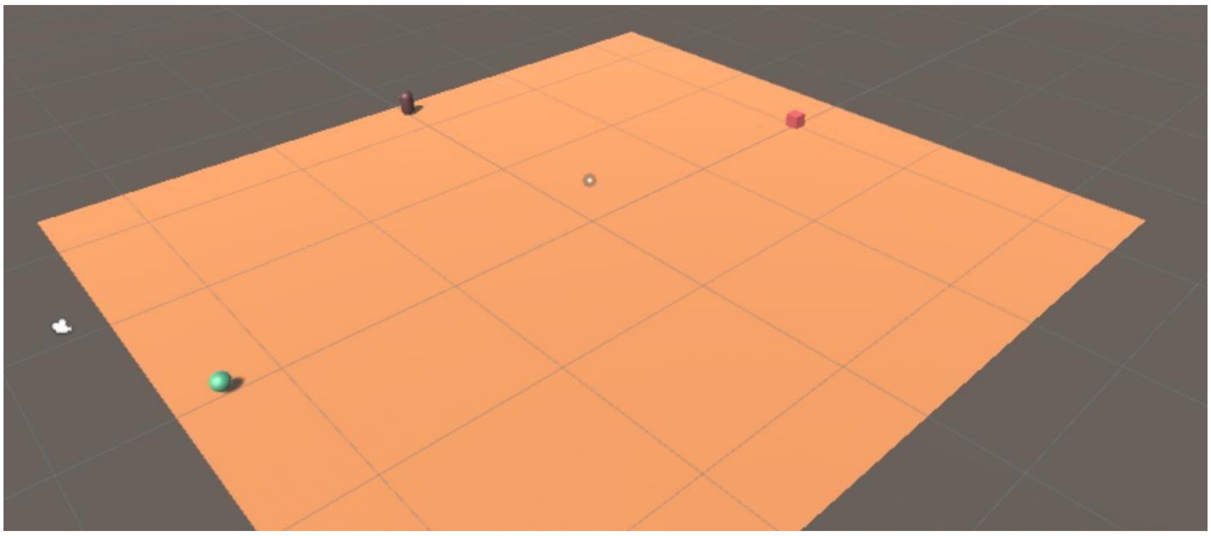
Next, apply the collider to the NPC.

Steps are:

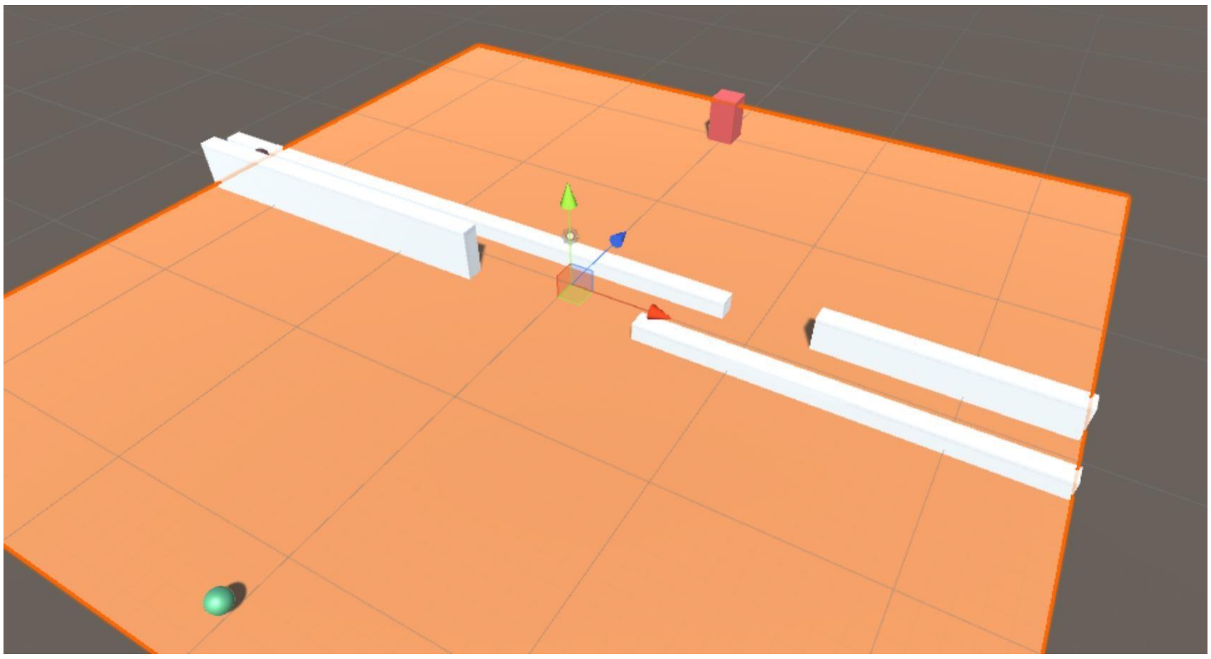
- Empty GameObject
- Apply Sphere collider
- Ensure is trigger is checked
- Link BounceLevel Script
- Ensure the correct Scene Index is being applied.

Run the program, you should be forced back to the main stage if you hit the npc, on either side of the board. You can modify the code, to make the npc move up and down the board as desired, but in this case, just to make it a little more challenging, we'll add some cube walls.

Before walls



After Walls



Play with the speed, the size of the colliders and so forth, this simple layout can stop the user from reaching the final bounce.

Which of course, you could have as taking the player to a new scene which congratulates them on getting through the levels.