CODE NINJAS®

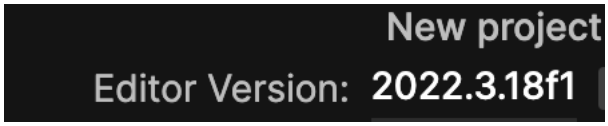
Bronze Belt Ninja Guide

Activity 06: SuperShapes

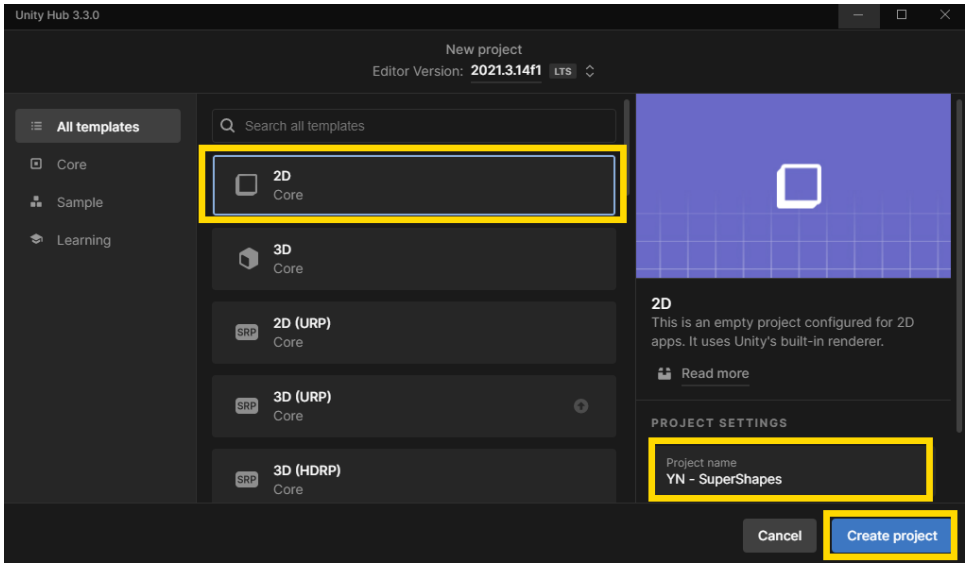
ACTIVITY 6: SUPERSHAPES

In this lesson, you will create a simple game where gravity, colliders, and physics come into play.

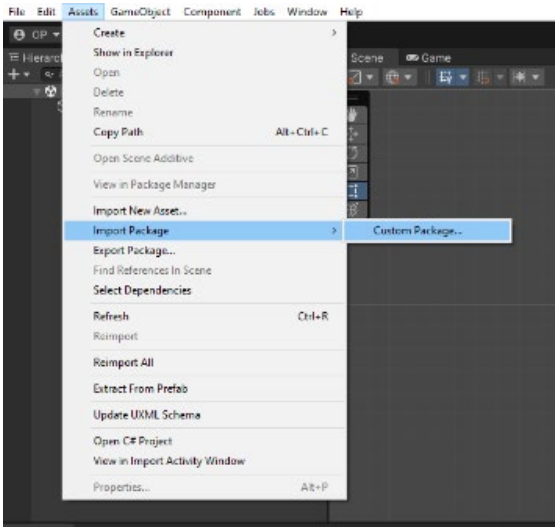
1 Open the Unity Hub application on your computer. Select the **New Project** button in the upper right-hand corner. Make sure the **2022.3 LTS** version is being used.



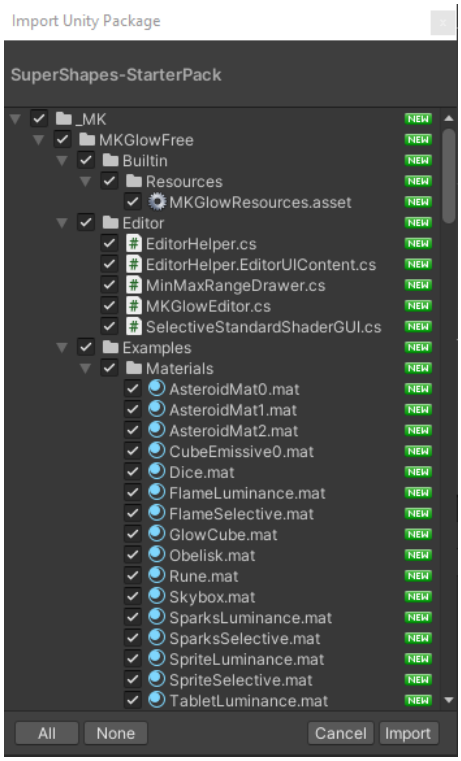
2 Select **2D** in the Templates column. Next, type in your first and last initials with the name of the project. For example, John Smith would save their project as JS-SuperShapes. Select the folder location where you want to save your project. Click the **Create** button.



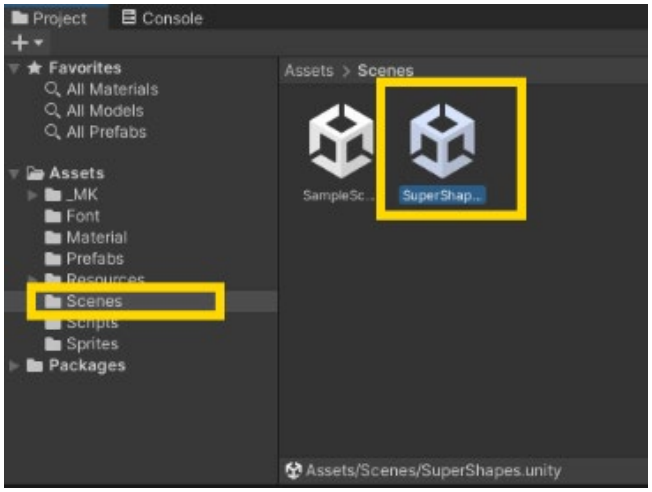
3 Once Unity has opened, just like with previous tasks, we'll want to install a package. Go to **Assets** and select **Import Package** and click on **Custom Package**.



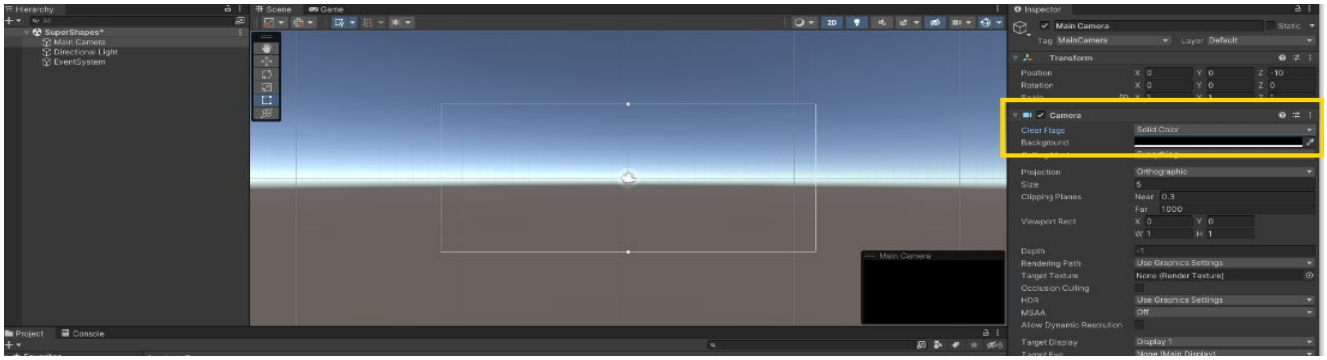
4 Select the **Activity 06 - SuperShapes-StarterPack.unitypackage**. All of the materials inside the package should be selected, if they are not, select **All** and click the **Import** button.



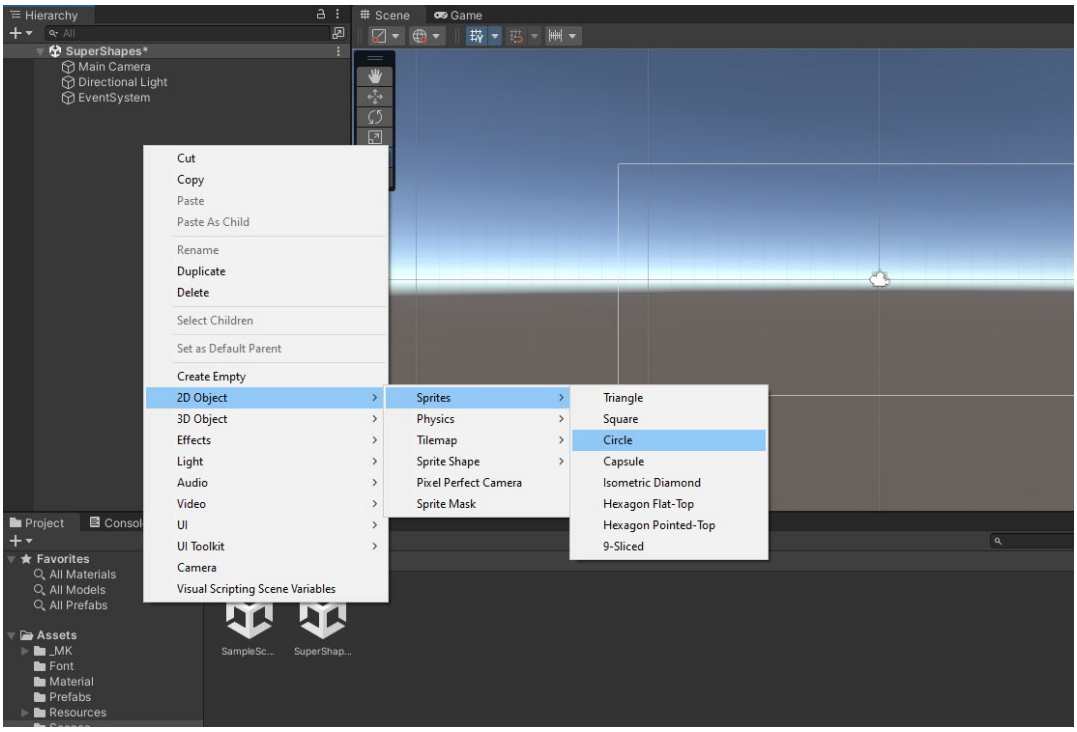
5 With the package installed, we want to load the *Supershapes* scene. It's a blank scene at the moment, but we will add to it. Go to the **Scenes** folder and double-click **SuperShapes**.



6 First, let's change the background color. Select the **Main Camera** gameobject and change the **Clear Flags** at the top from **Skybox** to **Solid Color**. Next, change the **Solid Color** to black.

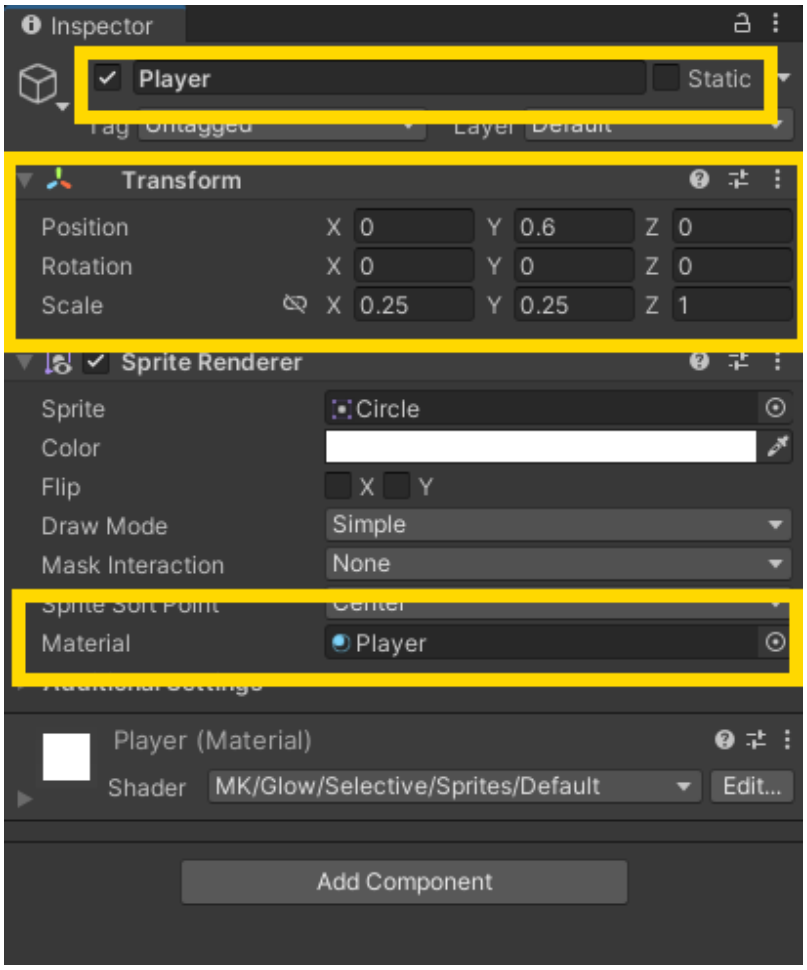


7 We are going to add a player. Instead of creating an empty game object, we are going to go to **2D** to **Sprites** and add a **Circle,** make sure you rename the object to **Player**.

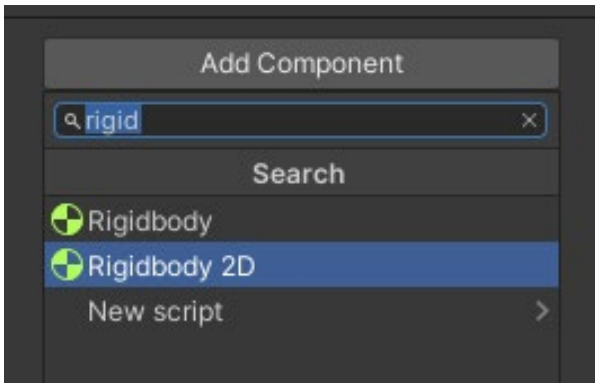


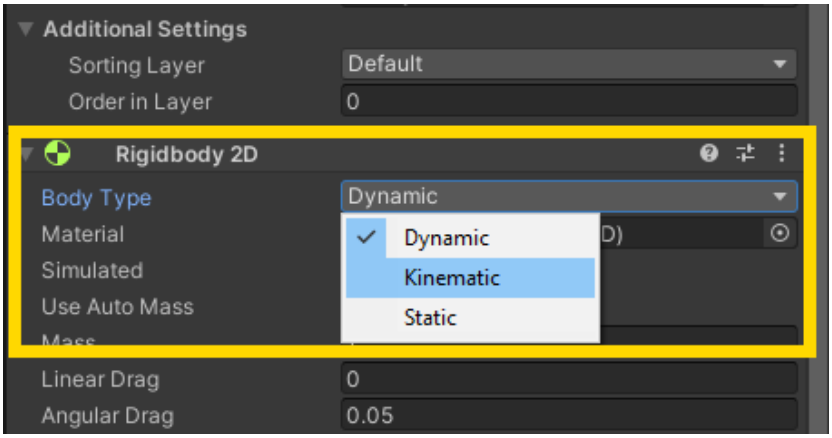
8

Let's move the player up slightly by **0.6**, and shrink its scale to something around **0.25** in both the **x** and **y**. While we are changing things, let's also change the material on the **Sprite Renderer** from **default** to **player.** Your player should currently look like this in the Inspector.

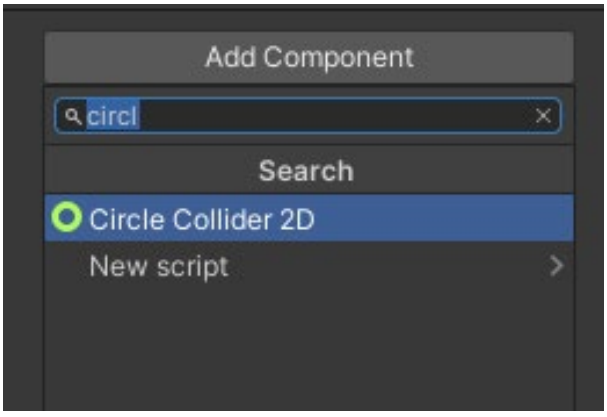


9 Click **Add Component**. Type ***Rigidbody*** in the search box and select the **Rigidbody 2D** component. Change the **Body Type** to **Kinematic**.

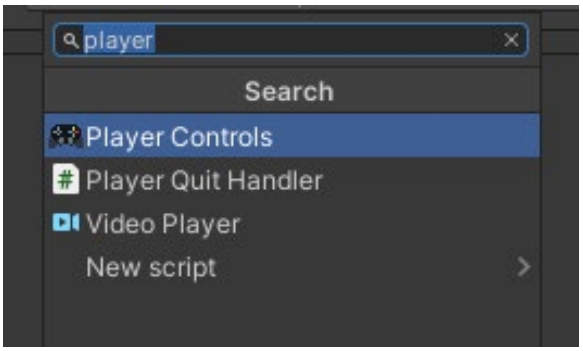




10 Click **Add Component** again. Type ***Circle Collider*** in the search box and select the **Circle Collider 2D** component.



11 Click **Add Component** one last time. Type ***Player Controls*** in the search box. Select the **Player Controls** script.



12 Open the **Player Controls** script and add the following variables to the top, just below the class line.

```
Unity Script | 0 references
public class PlayerControls : MonoBehaviour
{
    //movement speed
    [Header("Default Movement Speed")]
    public float moveSpeed = 200;
    //movement float
    float movement = 0;
```

13 Inside the **void Start** function, we are going to add a line that sets our Time Scale to its default, 1. This will be important later so that when we restart the game, everything is able to move as normal.

```
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        //Timescale is the speed the game runs at
        Time.timeScale = 1;
    }
```

14 Inside the **void Update** function, add the following. This will set our movement variable to either -1, 0 or 1 depending on what the player is inputting. If they input 'Left', then we get –1. If they input, 'Right' we get 1. If they don't input anything, we get 0.

```
20
21      // Update is called once per frame
        Unity Message | 0 references
22      void Update()
23      {
24          //Movement equal to Left and Right input
25          movement = Input.GetAxisRaw("Horizontal");
26      }
27
```

15 Below **void Update**, create a **void FixedUpdate** function. Add the following:

```
27
        Unity Message | 0 references
28      private void FixedUpdate()
29      {
30          //Object transformation rotates around
31          //an object using the settings.
32          transform.RotateAround(Vector3.zero, Vector3.forward, movement * Time.fixedDeltaTime * -moveSpeed);
33      }
34
```

16 Create a **void OnTriggerEnter2D** function, then add the following:

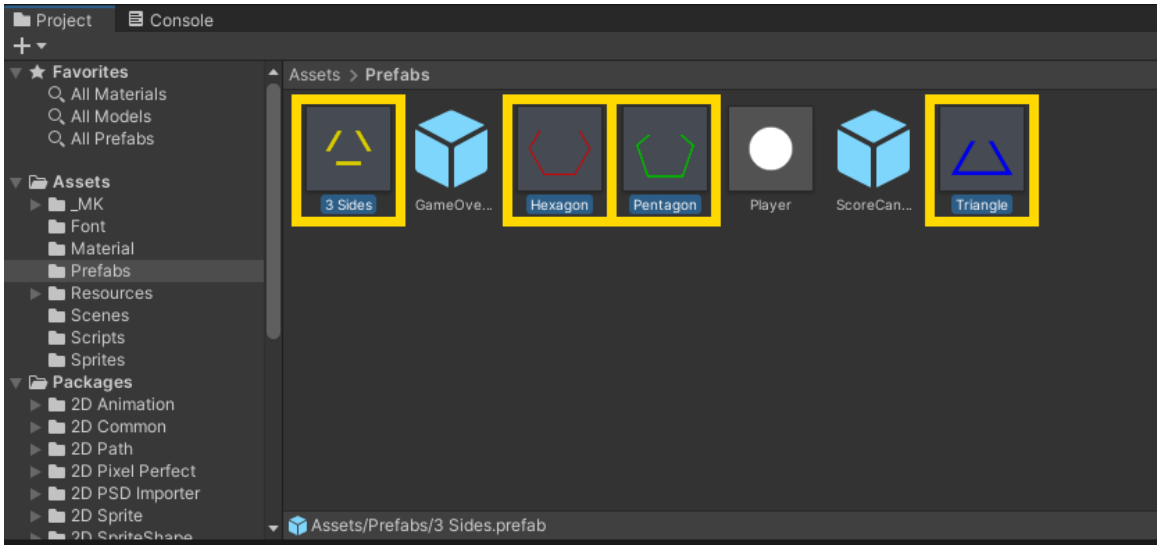
```
34
35      //This function runs if we overlap with a collider set to Trigger
        Unity Message | 0 references
36      private void OnTriggerEnter2D(Collider2D collision)
37      {
38          //Set the game speed to 0
39          Time.timeScale = 0;
40      }
```

Save the script and return to Unity.

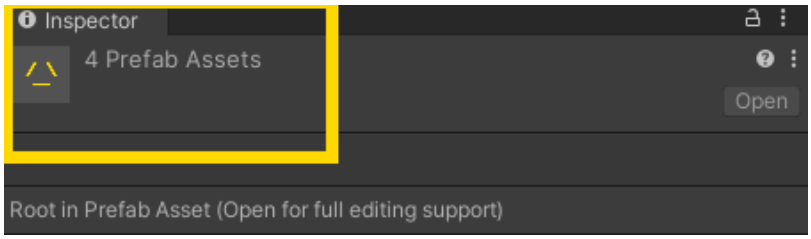
17 Play the game. Use the left and right arrows to move the player. You should see the player rotate in a circle.

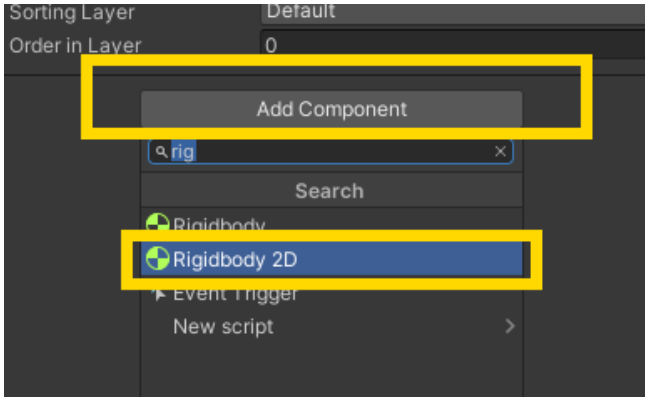
18 Stop the game.

19 Go into the **Prefabs** folder and you will see four shaped polygons: 3 Sides, Hexagon, Pentagon, and Triangle. The next few steps apply to all four of the shapes listed above. Hold down **Ctrl** on your Keyboard and **one by one**, click each shape to select them all.

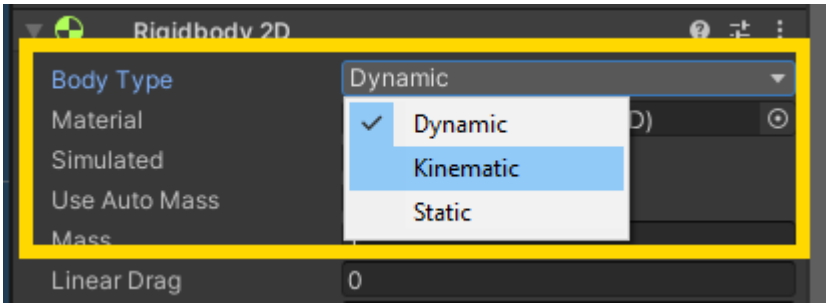


20 With all four shapes selected, click **Add Component**. Type ***Rigidbody2D*** in the search box and select the **Rigidbody 2D** component. To confirm all are selected, you should see "**4 Prefab Assets**" at the top.

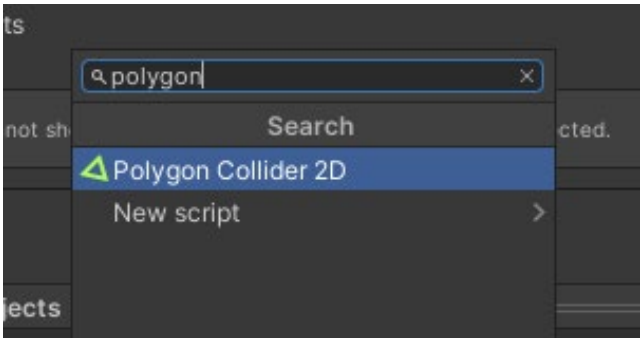




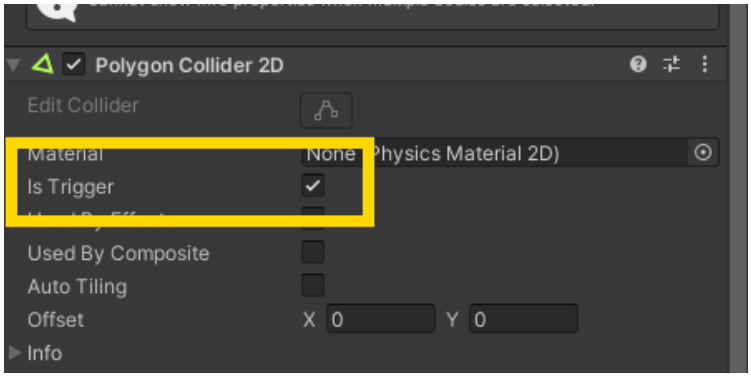
21 Change the Rigidbody's **Body Type** to Kinematic.



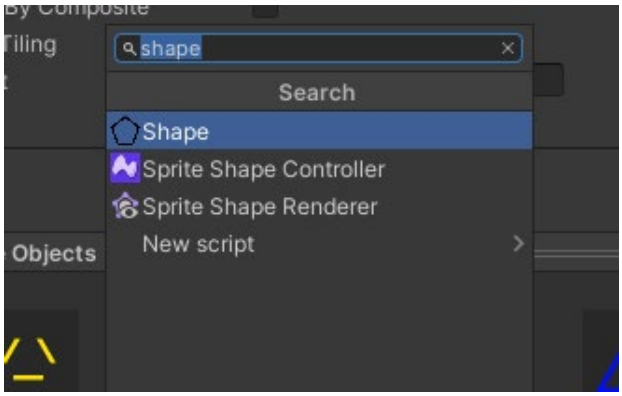
22 Next, with the four shapes still selected, click **Add Component** again. Type ***Polygon Collider 2D*** in the search box and select the **Polygon Collider 2D** component.



23 Under the **Polygon Collider 2D**, select the checkbox for **Is Trigger**.



24 Once more, click **Add Component**. Type ***Shape*** in the search box and select the **Shape** script component.



25 Open the **Shape** script, then add the following variables just below the bracket for the class line.

```
Unity Script (12 asset references) | 0 references
public class Shape : MonoBehaviour
{
    //Rigidbody for the object
    [Header("Rigidbody Object")]
    public Rigidbody2D rb;
    //Shrinking Speed
    [Header("Default Shrinking Speed")]
    public float shrinkSpeed = 3;

```

26 Inside the **void Start** function, add the following:

```
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    //Rotation of the rigidbody
    //at a random range
    rb.rotation = Random.Range(0, 360);
    //local scale for the Hexagon
    //equals one for all axes (x,y,z) times ten
    //meaning - localScale = (1,1,1) * 10
    transform.localScale = Vector3.one * 10;
}
```

27

Inside the **void Update** function, add the following:

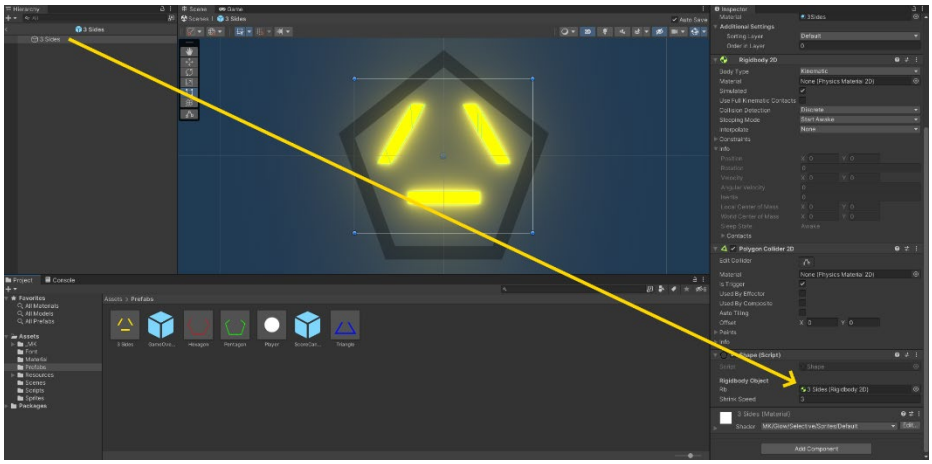
```
28
29      // Update is called once per frame
        Unity Message | 0 references
30      void Update()
31      {
32          //slowly shrink our localScale by
33          //our shrinkSpeed multiplied by game rate
34          transform.localScale -= Vector3.one * shrinkSpeed * Time.deltaTime;
35
36          //When the object gets too small
37          if (transform.localScale.x < 0.05f)
38          {
39              //Destroy Object
40              Destroy(gameObject);
41          }
42      }
43  }
44
```

Save the script and return to Unity.

28

Go back into the **prefabs** folder and select one of the shape prefabs. Although we have been editing them together, we need to do them one at a time for this next step.

Double click into a shape prefab and place the **RigidBody2D** into the **Inspector** field by dragging it.



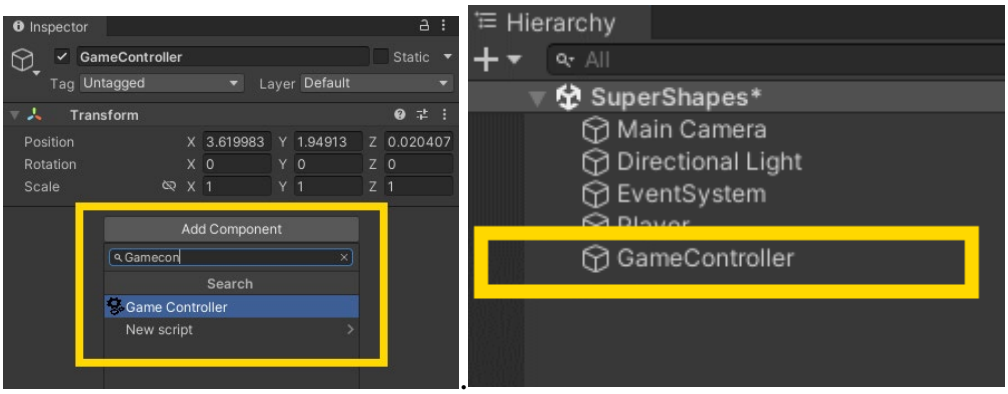
29

Repeat step 28 with the remaining shapes.

All your shapes should have a **Ridigbody2D** set to **Kinematic**, a **Polygon Collider** set to **Is Trigger**, and the **Shapes** Script with the correct **Rigidbody2D** in the **RB** field.

30

Before you play the game, add a Spawner to spawn multiple random shapes. See if you can add a new GameObject on your own and rename it **GameController**. Click **Add Component**. Type ***Game Controller*** in the search box and select the **Game Controller** script component.



31

Open the **Game Controller** script and add the following variables:

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

Unity Script (1 asset reference) | 0 references
public class GameController : MonoBehaviour
{
    //The [] tells Unity that we want an Array
    [Header("Shape Objects")]
    public GameObject[] shapePrefabs;
    //The first object will spawn after
    //the spawnDelay and then every spawnTime
    [Header("Default Spawn Delay Time")]
    public float spawnDelay = 2;
    [Header("Default Spawn Time")]
    public float spawnTime = 3;
```

32

Inside the **void Start** function, add the following:

```
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    InvokeRepeating("Spawn", spawnDelay, spawnTime);
}
```

33

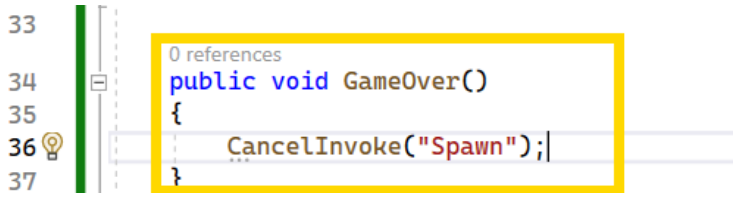
Create a **void Spawn** function just below **void Start** by adding the following:

```
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    InvokeRepeating("Spawn", spawnDelay, spawnTime);
}

0 references
void Spawn()
{
    //Get a random number between the range of 0 and our
    //array length
    int randomInt = Random.Range(0, shapePrefabs.Length);
    //spawn new hexagon that was picked randomly
    Instantiate(shapePrefabs[randomInt],Vector3.zero,Quaternion.identity);
}
```

34

Create a **void Game Over** function, add the following function just below the end of **void Spawn**:



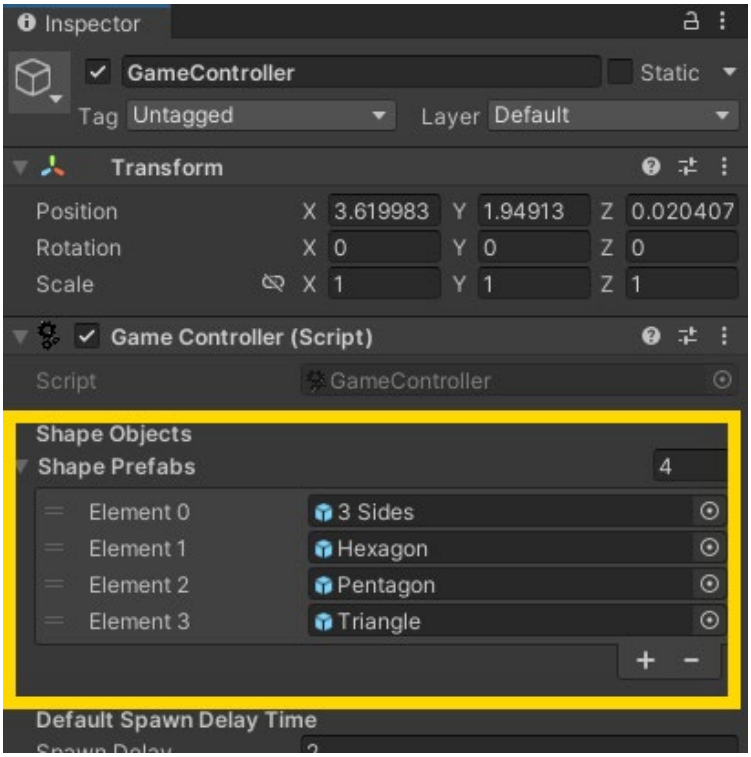
```
0 references
public void GameOver()
{
    CancelInvoke("Spawn");
}
```

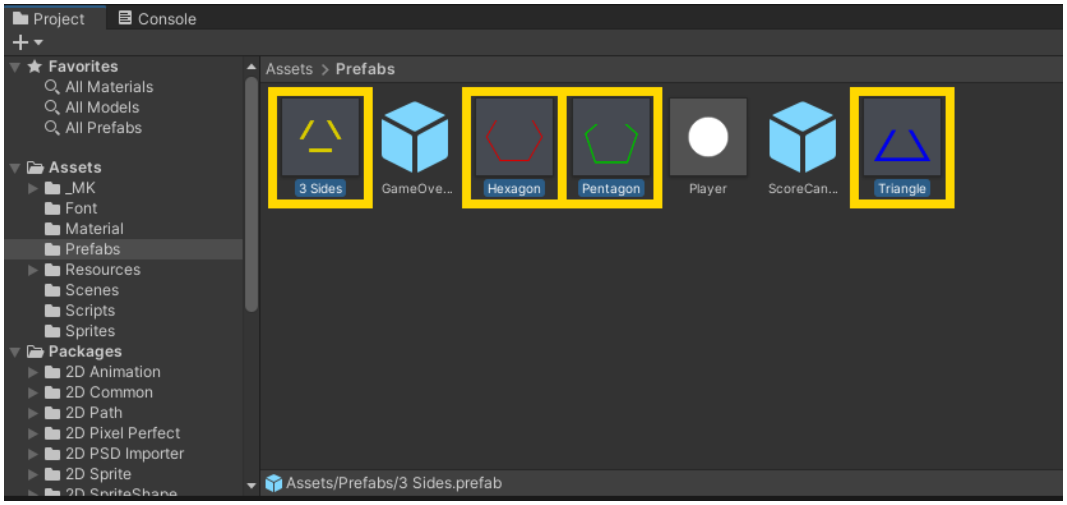
Save the script and return to Unity.

35

Inside the Project window, you will find the **prefabs** folder that contains the four Prefabs we want to add to our array. Next, select the **Game Controller** and one by one drag the prefabs into the **Shape Prefabs Array**.

If you are having trouble with this, you can also type the number **4** into the array **Size** and drag them in that way, or search for them with the little dot next to each element.



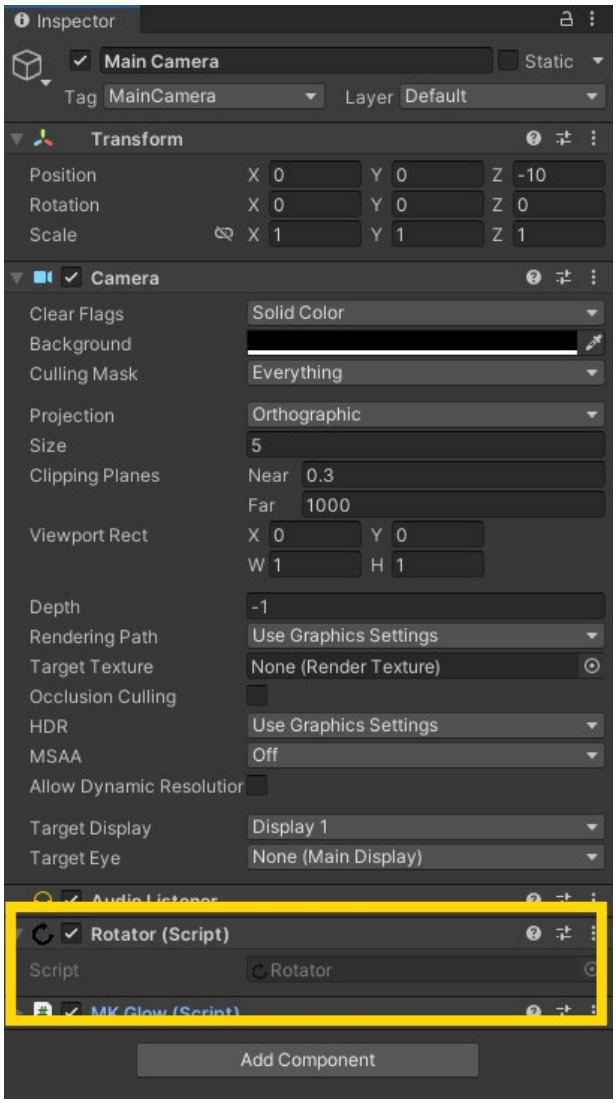


36 Play the game. All of the shapes will begin to appear and shrink. If you touch one of the objects, the game will end.

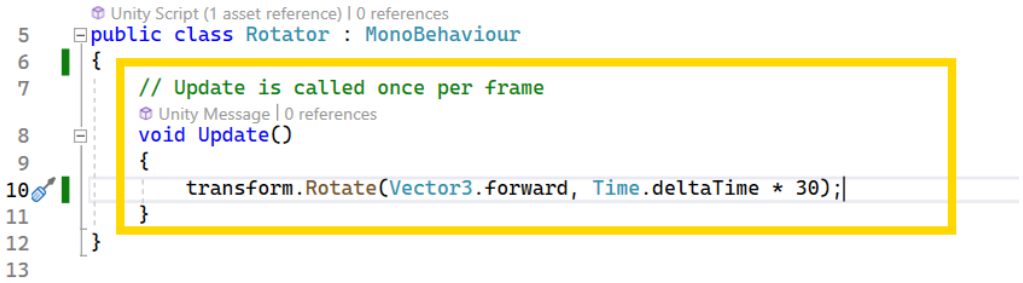
37 Stop the game.

38 Now to add more difficulty to the game. Start by selecting the **Main Camera** and loading the **Rotator** script component attached.

If you don't see the **Rotator** script, you may need to add it by pressing **Add Component** and typing ***Rotator***.



39 Delete the **void Start** function. Inside the **void Update** function, add the following:

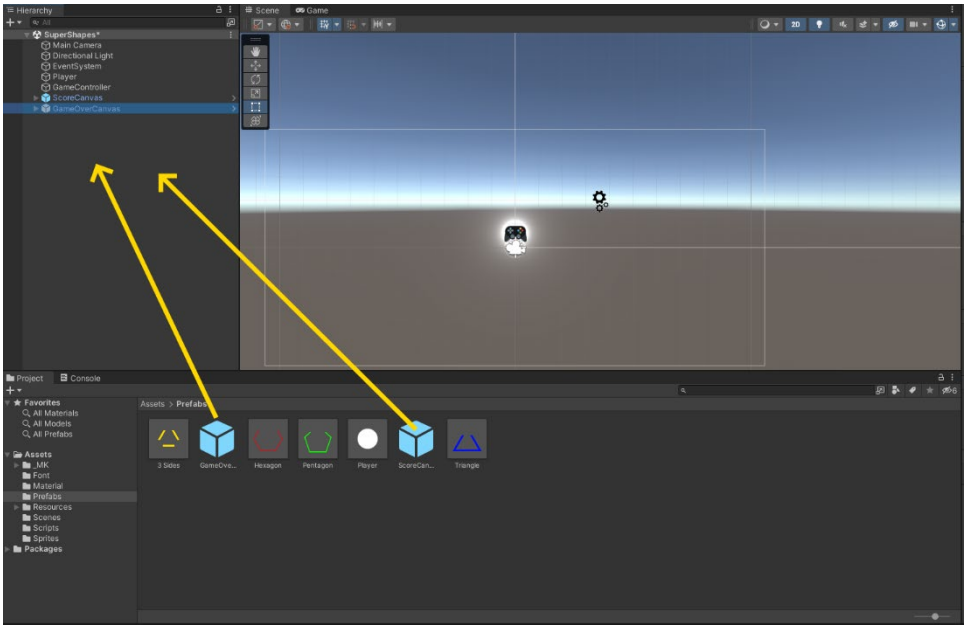


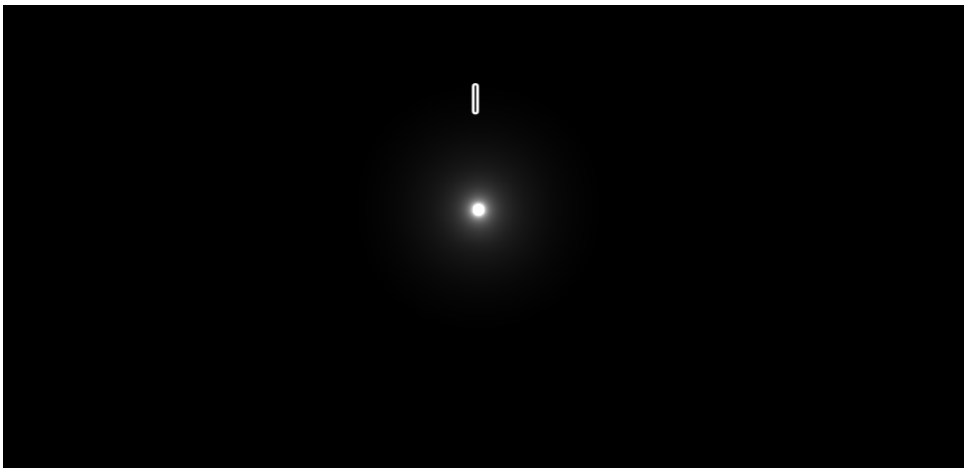
40 Save the script and return to Unity. Play the game. The **Main Camera** will begin to rotate, which will add more difficulty to the game.

41 Stop the game.

42 Time to add scoring and the Game Over screen. To do so, add both the **ScoreCanvas** and **GameOverCanvas** to the **Hierarchy** from the **Prefabs** folder. If you play the game, you'll notice that the score is now seen in the Game window.

If you cannot see the score. Double click it in the Hierarchy, it may be too high and outside of the canvas's white lines.





43

Reopen the **Game Controller** script and add the following variable:

```
Unity Script (1 asset reference) | 0 references
public class GameController : MonoBehaviour
{
    //The [] tells Unity that we want an Array
    [Header("Shape Objects")]
    public GameObject[] shapePrefabs;
    //The first object will spawn after
    //the spawnDelay and then every spawnTime
    [Header("Default Spawn Delay Time")]
    public float spawnDelay = 2;
    [Header("Default Spawn Time")]
    public float spawnTime = 3;

    [Header("Game Over UI Canvas")]
    public GameObject gameOverCanvas;
```

44

Inside the **void GameOver** function, add the following:

```
0 references
public void GameOver()
{
    CancelInvoke("Spawn");
    //Set the gameOverCanvas to be visible
    gameOverCanvas.SetActive(true);
}
```

Save the script and return to Unity.

45

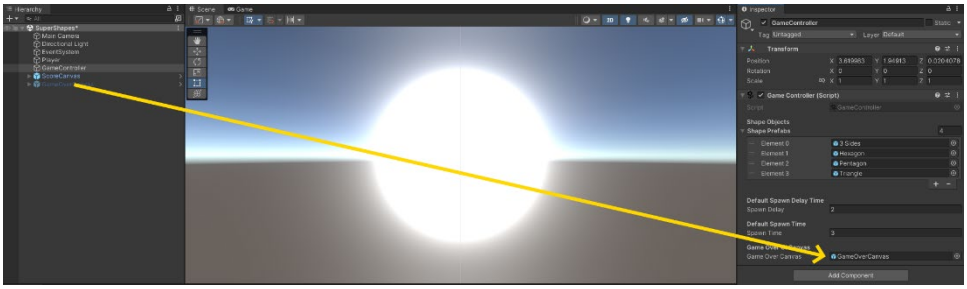
Reopen the **Shapes** script and inside the **void Update** function, add the following inside the **if** statement:

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    //slowly shrink our localScale by
    //our shrinkSpeed multiplied by game rate
    transform.localScale -= Vector3.one * shrinkSpeed * Time.deltaTime;

    //When the object gets too small
    if (transform.localScale.x < 0.05f)
    {
        //Destroy Object
        Destroy(gameObject);
        Score.score++;
    }
}
```

Save the script and return to Unity.

46 With the **GameController** selected, search and select the **GameOverCanvas** game object in the input field of the **Game Controller** script component. You can also drag it from the Hierarchy.



47 In the **PlayerControls** script, replace the code inside the **OnTriggerEnter2D** function with the following:

```
34
35
36      //This function runs if we overlap with a collider set to Trigger
        Unity Message | 0 references
37      private void OnTriggerEnter2D(Collider2D collision)
38      {
39          GameObject.Find("GameController").GetComponent<GameController>().GameOver();
40      }
```

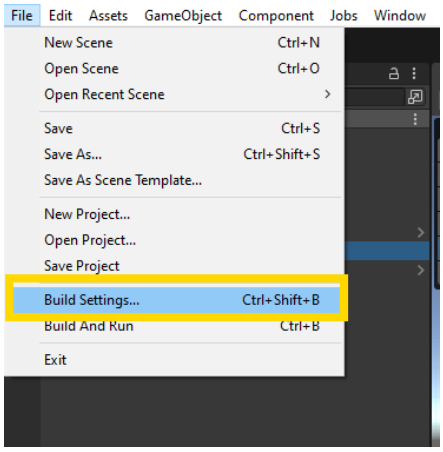
48 Save the script and return to Unity.

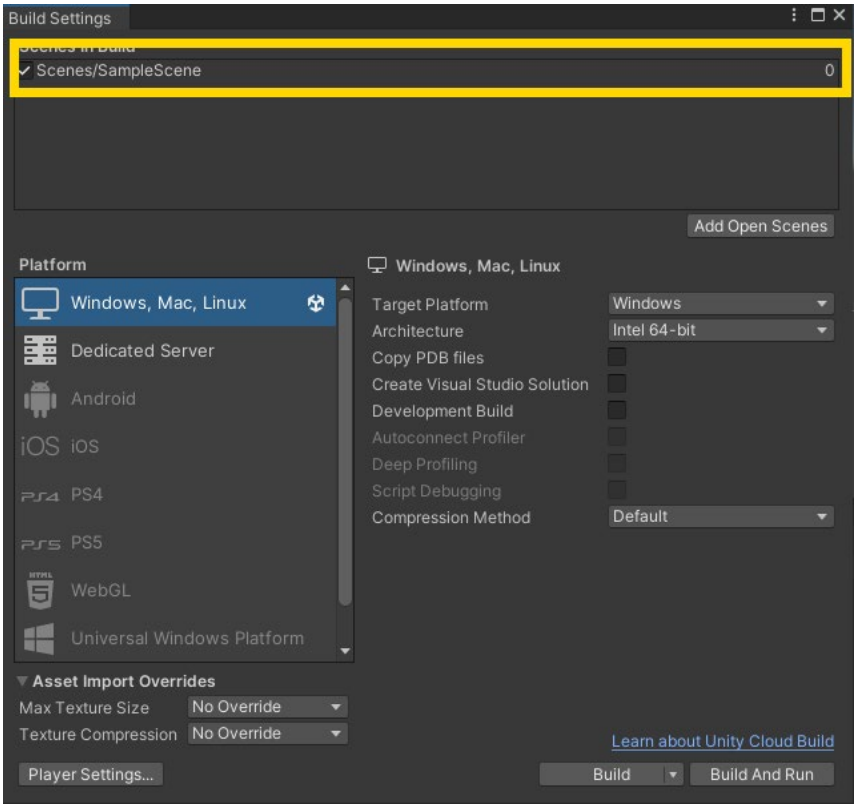
49 Reopen the **GameController** script and add the following to the **GameOver** function:

```
36
        1 reference
37      public void GameOver()
38      {
39          CancelInvoke("Spawn");
40          //Set the gameOverCanvas to be visible
41          gameOverCanvas.SetActive(true);
42          Time.timeScale = 0;
43      }
```

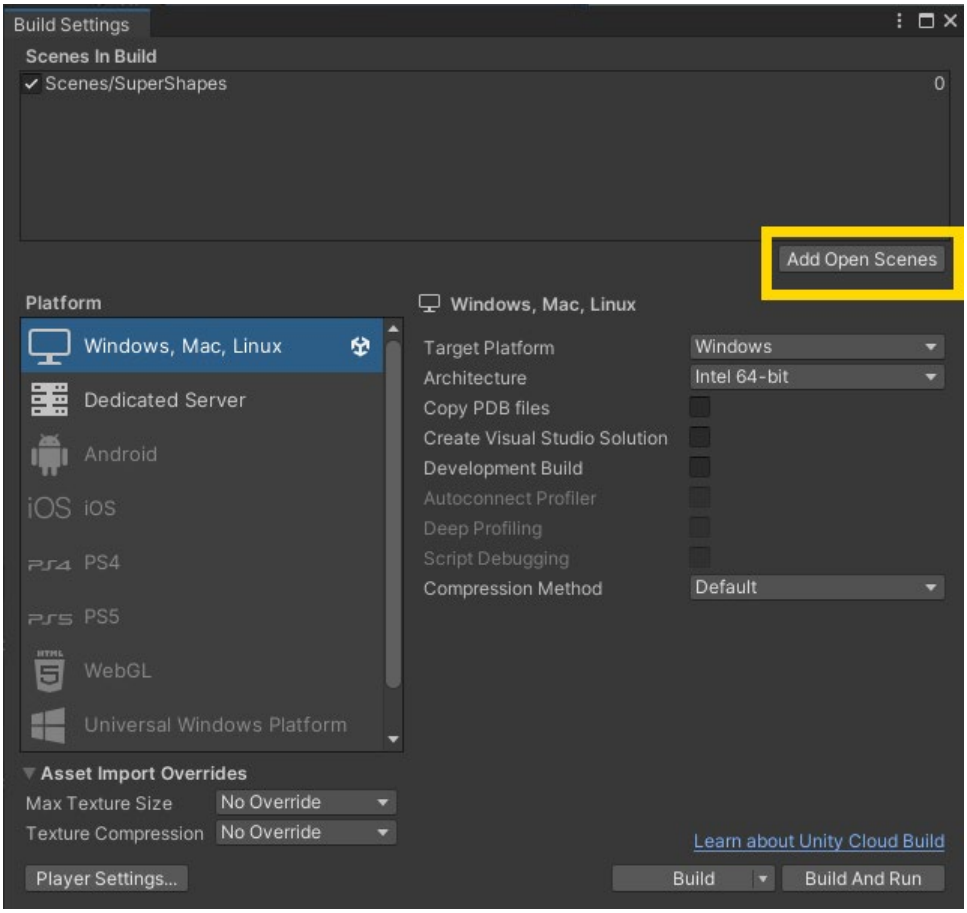
50 Save the script and return to Unity.

51 In order to get the play again button to work, we will have to adjust the build settings. Click the **File** tab then **Build Settings**. Select the Scenes/SampleScene and delete it.





52 Once deleted, click **Add Open Scenes**. This will allow for the game to restart.



53 Play your game and see what you get. **Save** your game and **Export** it. **Submit** your game.