CODE NINJAS®

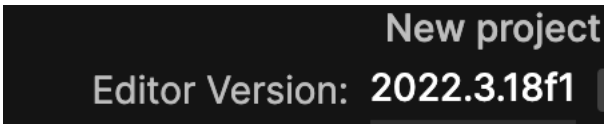
Bronze Belt Ninja Guide

Activity 07: PolyRun

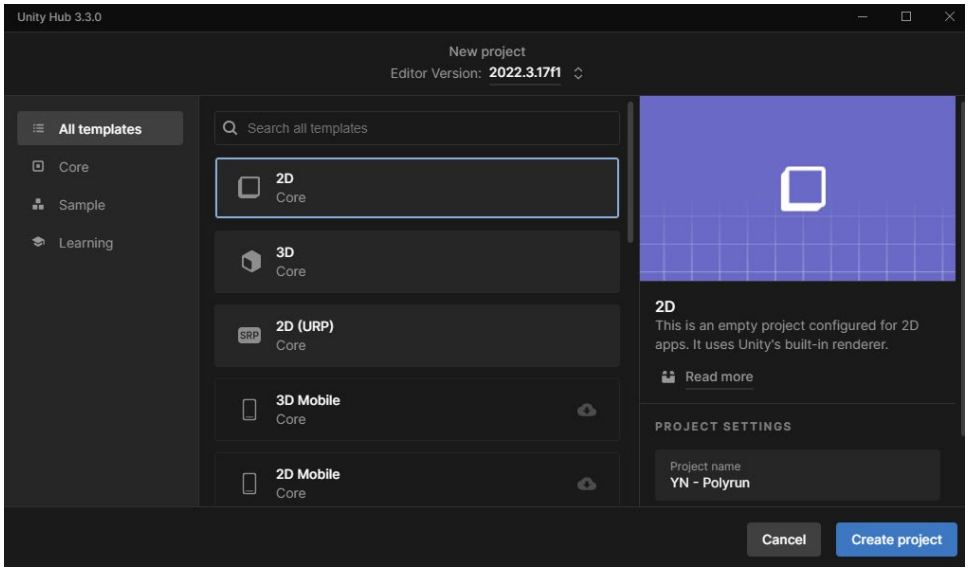
Activity 7: PolyRun

It's time to make another "runner" game!

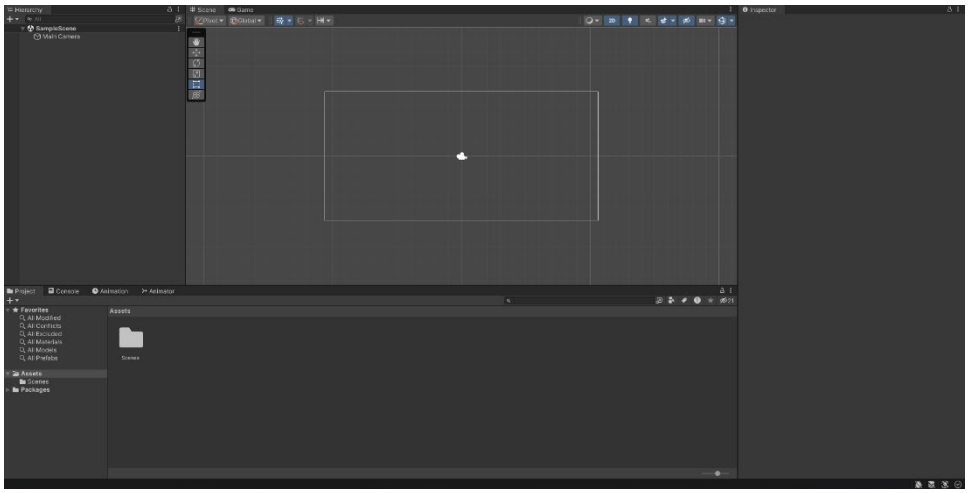
1 Open the Unity Hub application on your computer. Select the **New Project** button in the upper right-hand corner. Make sure the **2022.3 LTS** version is being used.



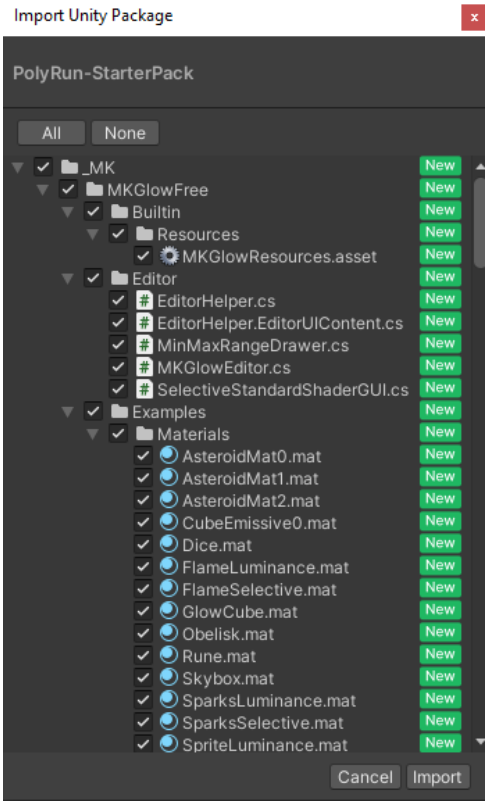
2 Select **2D core** in the Templates column. Next, type your first and last initials with the name of the project. For example, John Smith would save their project as **JS-PolyRun**. In the image below YN is used for "Your Name". Select the folder location where you want to save your project. Click the **Create** button.



3 As soon as Unity has loaded all the necessary assets into the program, this is what you'll see. Notice that we have no assets or scripts in the project. Let's fix that.

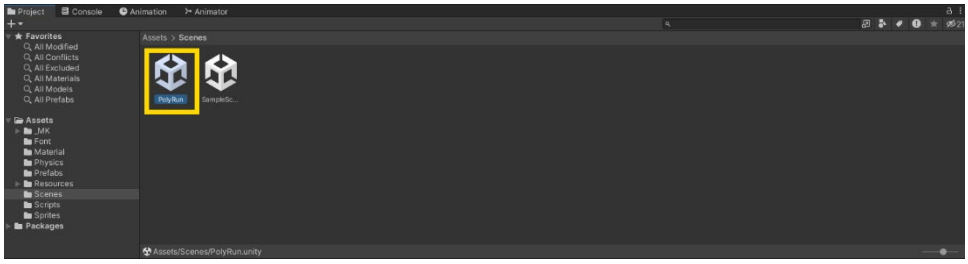


4 Go to the **Assets** menu. Select **Import Package** and click **Custom Package**. Select the **Activity 07 - PolyRun-starterpack UnityPackage** file. Once loaded, Unity will show a menu of what is inside the package.

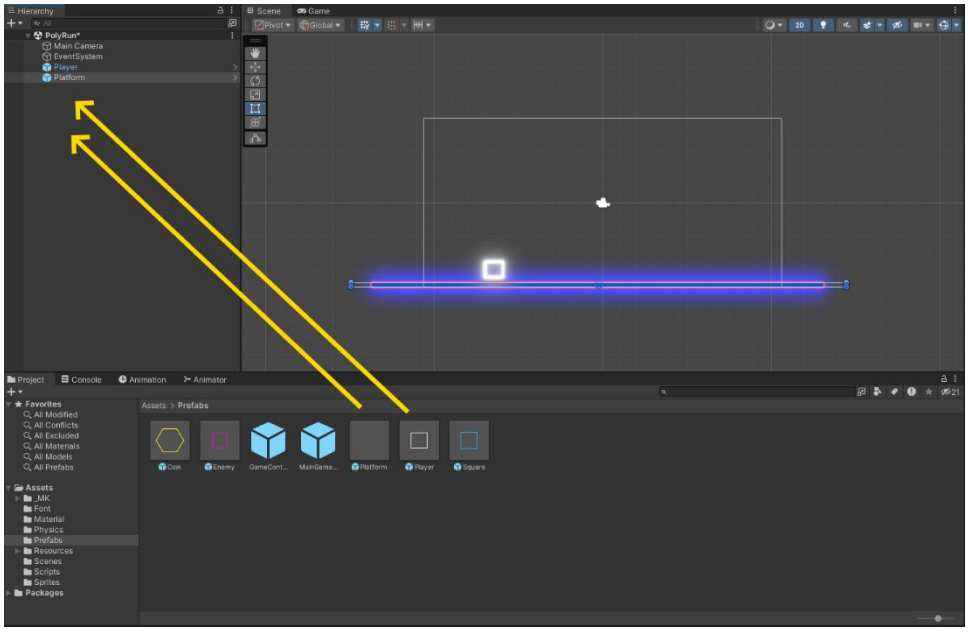


All of the material inside the Unity package should be selected. If not, click **All** to select everything, then click **Import**.

5 Even though a scene is open, there isn't anything in the Scene or Game windows. This is because we start in **SampleScene**; we want to go into the **Polyrun** scene. Go into **Scenes** and click the **Polyrun** scene.



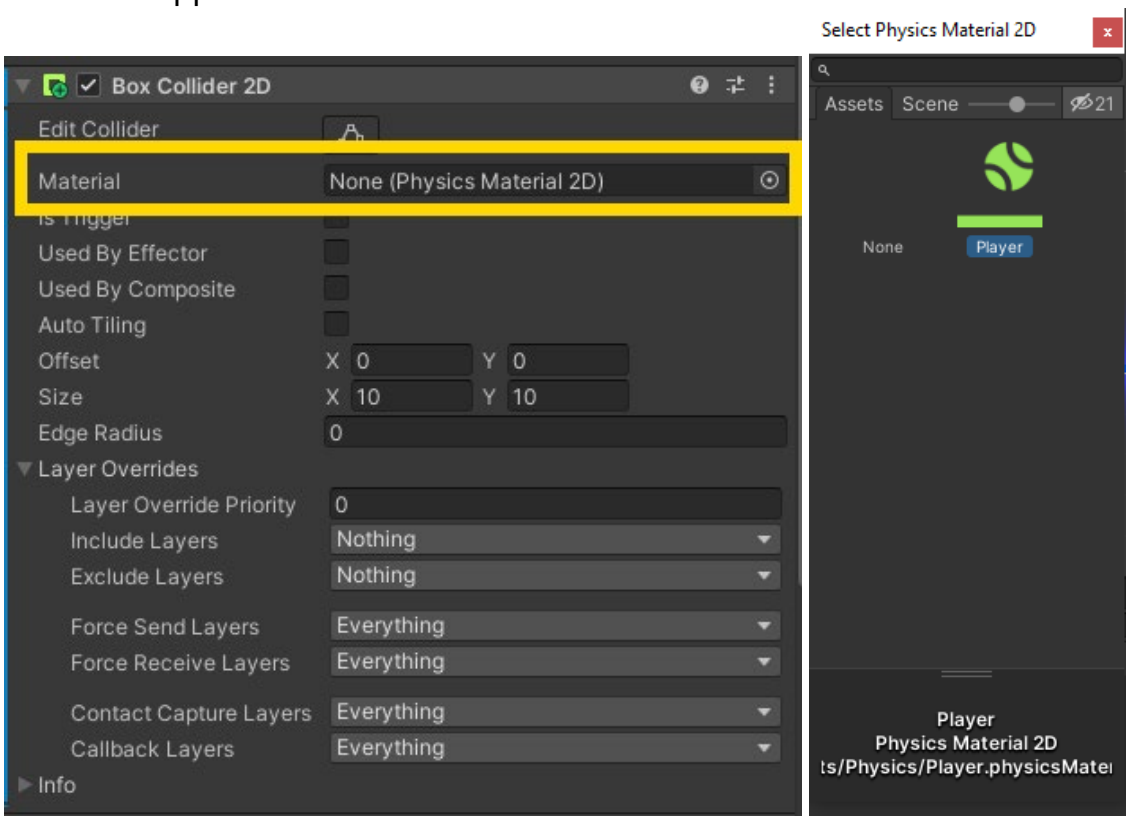
6 Go into the **Prefabs**, then drag the **Player** and **Platform** into the **Hierarchy**.



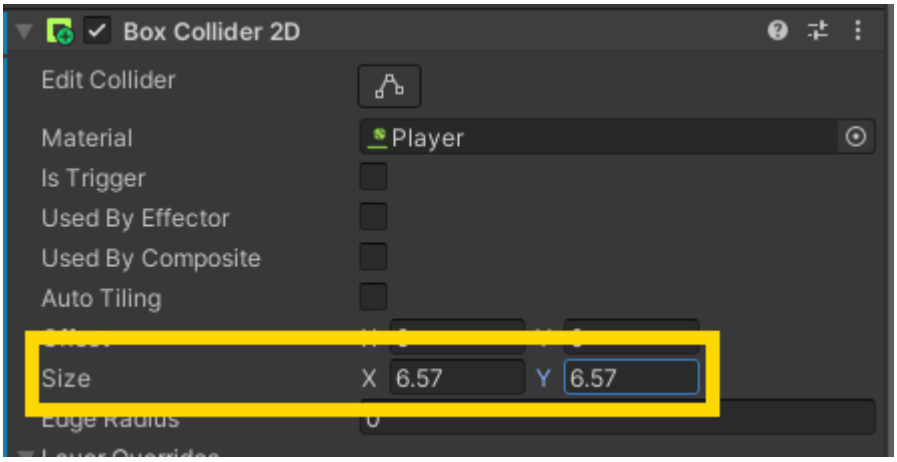
7 Next, select the **Player** GameObject. We need to add three components: a Box Collider 2D component, a Rigidbody 2D component, and... what's the third one? Can you figure it out?

If you can't figure it out, don't worry, the answer is on step 13.

8 Inside the **Box Collider 2D**, select the **Material** option and then the **Player Physics Material 2D** that appears.

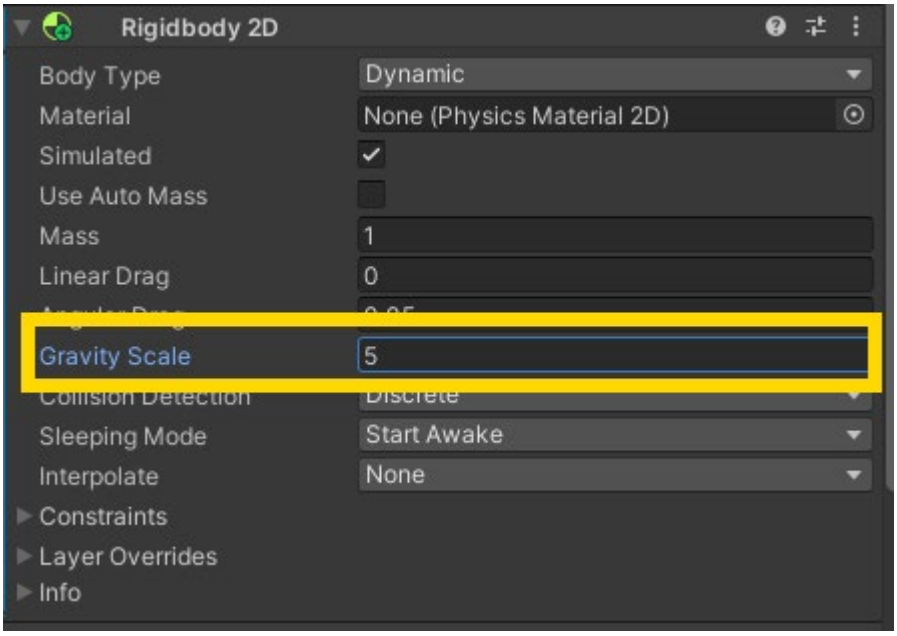


9 Change the size of the **Box Collider 2D** values. **X**: 6.57, **Y**: 6.57

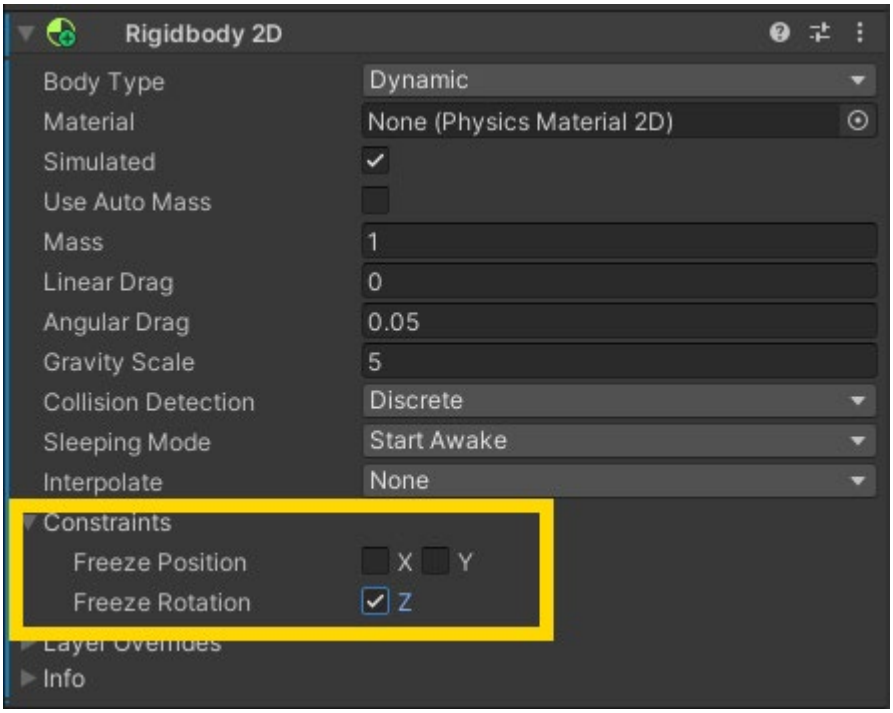


10 Next, click **Add Component**. Type ***Rigidbody2D*** in the search box and select the **Rigidbody 2D** component.

11 Change the **Gravity Scale** of the Rigidbody 2D values to **5**.

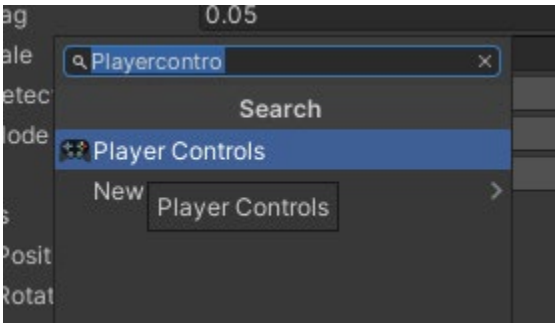


12 Click **Constraints**. Check the box next to **Freeze Rotation** for the Z-axis rotation constraints. This will stop physics from acting on its rotation, but still on its X and Y position.



13 Click **Add Component** one more time. We are going to add the final component that was talked about in step 7. We need to add the **Player Controls** script. Did you guess correctly?

Search for ***Player Controls*** and select the **Player Controls** script component.



14 Open the **Player** script and add the following variables:

```
Unity Script | 0 references
public class PlayerControls : MonoBehaviour
{
    //Jumping power for the player object
    [Header("Default Jumping Power")]
    public float jumpPower = 6f;
    //True or false if is on the ground
    [Header("Boolean isGrounded")]
    public bool isGrounded;
    //position of the object
    float posX = 0.0f;
    //rigibody2D of the object
    Rigidbody2D rb;
```

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

```
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    //Variable rb equals to Rigidbody2D
    //component on the object
    //this script is attached to
    rb = GetComponent<Rigidbody2D>();

    //posX = starting position
    posX = transform.position.x;
}
```

16 Create the **void Update** function. Inside the function, add the following:

```
30      // Update is called once per frame
        Unity Message | 0 references
31      void Update()
32      {
33          //Only jump if we press space and if isGrounded is true
34          if (Input.GetKeyDown(KeyCode.Space) && isGrounded)
35          {
36              rb.AddForce(Vector3.up * jumpPower * rb.mass * rb.gravityScale * 20f);
37          }
38      }
```

Save the script and return to Unity.

17 If you were to go back into Unity, you'll see that the **isGrounded** checkmark doesn't change. We need to add some more code to detect if we are on the ground.

18 If you went to test the game, stop the game.

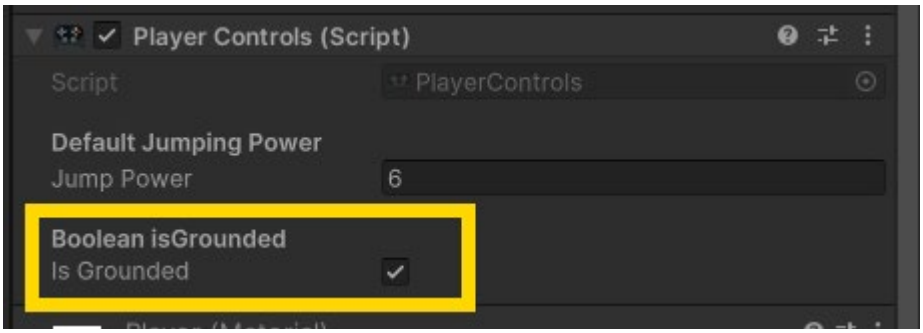
19 Inside the **Player Controls** script, add a new **void OnCollisionEnter2D** function. Add the following code inside.

```
Unity Message | 0 references
private void OnCollisionEnter2D(Collision2D collision)
{
    //If the object we collide with has the
    //tag Ground
    if(collision.gameObject.tag == "Ground")
    {
        //isGrounded is set to true
        isGrounded = true;
    }
}
```

@

Save the script and return to Unity.

20 Playtest without maximizing the scene. While the scene is playing, select the **player** object and check to see if **isGrounded** is checked.



21 After confirming that, press the spacebar to make your player object jump. Then, stop the scene.

22 Now if you make the object jump, it will continue jumping upward. To correct that, go back into the **Player Controls** script and add two more functions.

First, create a function called **void OnCollisionStay2D**. Inside that function, add the following code:

```
Unity Message | 0 references
private void OnCollisionStay2D(Collision2D collision)
{
    //If the object we collide with has the
    //tag Ground
    if (collision.gameObject.tag == "Ground")
    {
        //isGrounded is set to true
        isGrounded = true;
    }
}
```

23 The next function to create is **void OnCollisionExit2D**. Inside the function, add the following:

```
Unity Message | 0 references
private void OnCollisionExit2D(Collision2D collision)
{
    //If the object we collide with has the
    //tag Ground
    if (collision.gameObject.tag == "Ground")
    {
        //isGrounded is set to true
        isGrounded = false;
    }
}
```

24 Playtest the scene to see if the ability to jump is working properly now. If it isn't working, double check the functions you just added. Make sure you are setting **isGrounded** to **false** in the **OnCollisionExit2D** function and not the others.

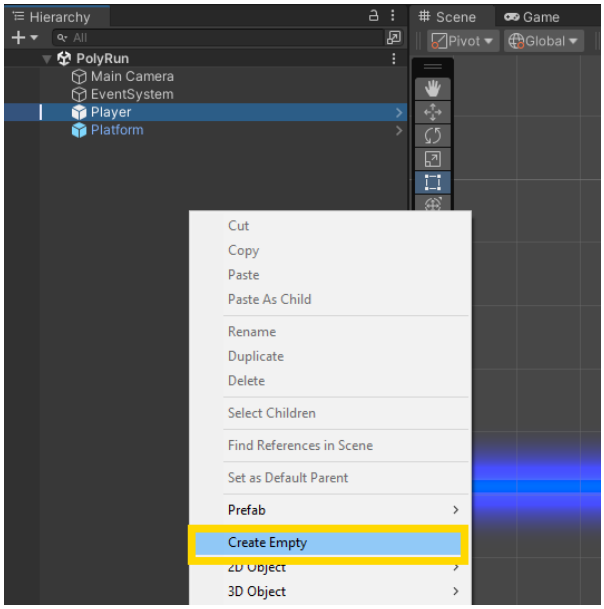
Stop the scene.

25 Save the script and return to Unity.

26 Now that we have the moving player, let's create an obstacle for the game.

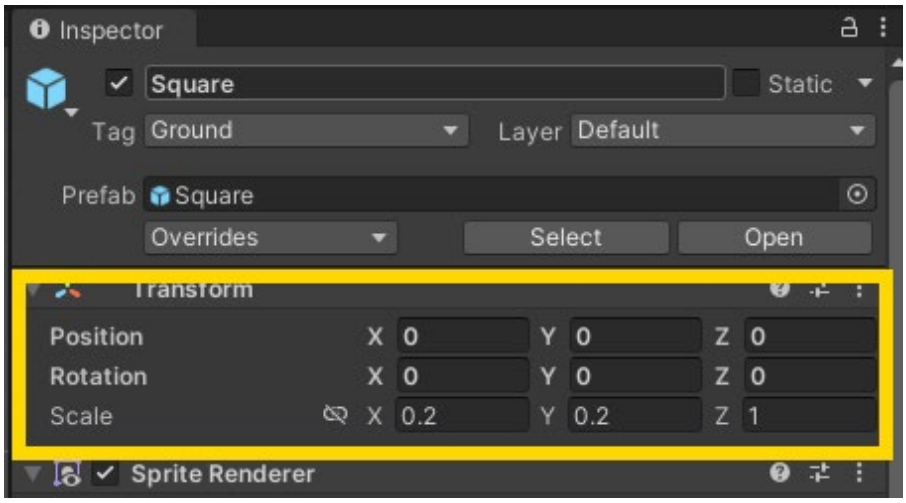
You can create any type of obstacle you want. These steps will show you how.

First, create an **Empty GameObject** and call it ***ChallengeObj***. For now, we'll also change the position to something shown in the image.

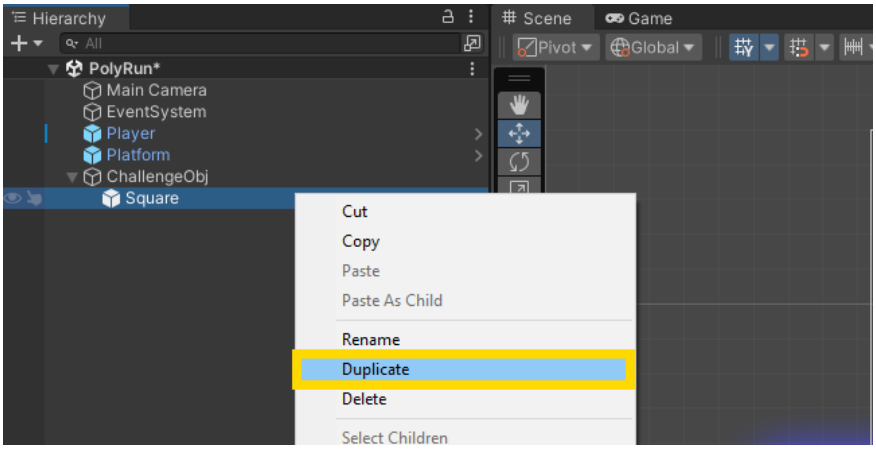


27 Go into the **Prefabs** folder and drag the **Square** to the **ChallengeObj**. It should be a child of the **ChallengeObj**.

This will give us our square that the player will need to jump over. Make sure to reset its position to **x**:0, **y**:0, and **z**: 0 (0,0,0).

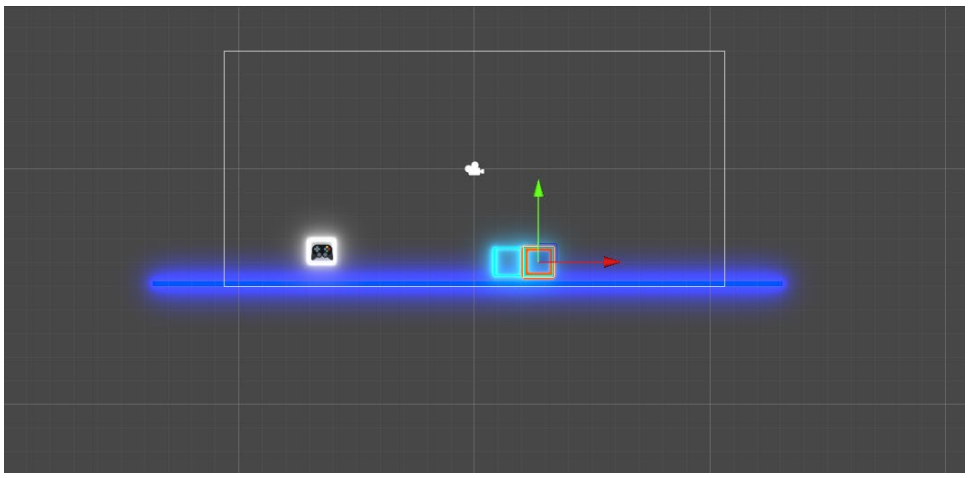


28 Next, right click the **Square** object and select duplicate.

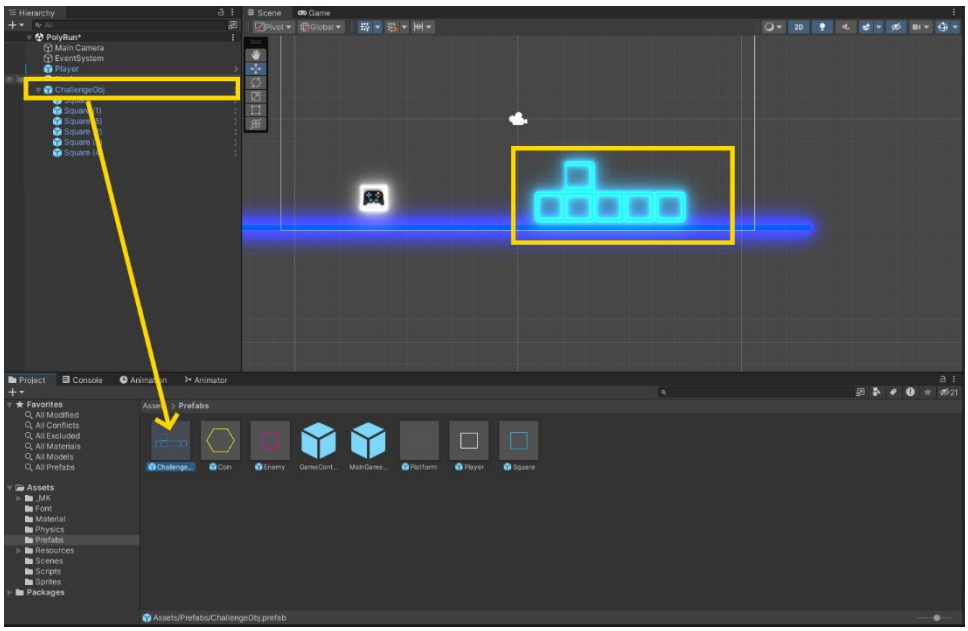


29 Move the new duplicated square to the right of the original square, as shown below.

We are going to repeat this step a few more times until we have our new object. You can make any shape, or choose one similar to the one below.

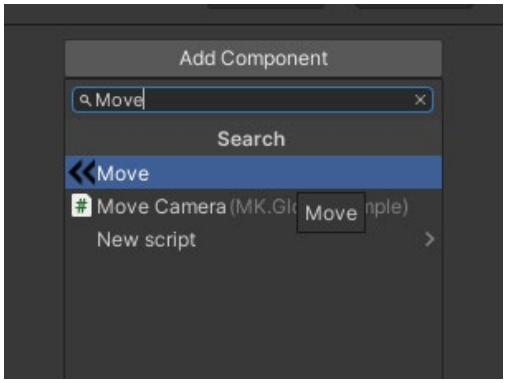


30 Once you've created your **ChallengeObj**, we need to turn it into a prefab. Creating a **Prefab** is simple. Drag the **ChallengeObj** from the **Hierarchy** to the **Prefabs** folder.



31 Next, select the prefab you just made. We are going to be editing it a bit more. You should see **ChallengeObj (Prefab Asset)** at the top, so you know you are editing the prefab.

To make the challenge object move from right to left, first click **Add Component**. Type ***Move*** in the search box and select the **Move** script component.



32 Open the **Move** script and add the following variable:

```
Unity Script (1 asset reference) | 0 references
public class Move : MonoBehaviour
{
    [Header("Default Speed")]
    public float speed = 6;
```

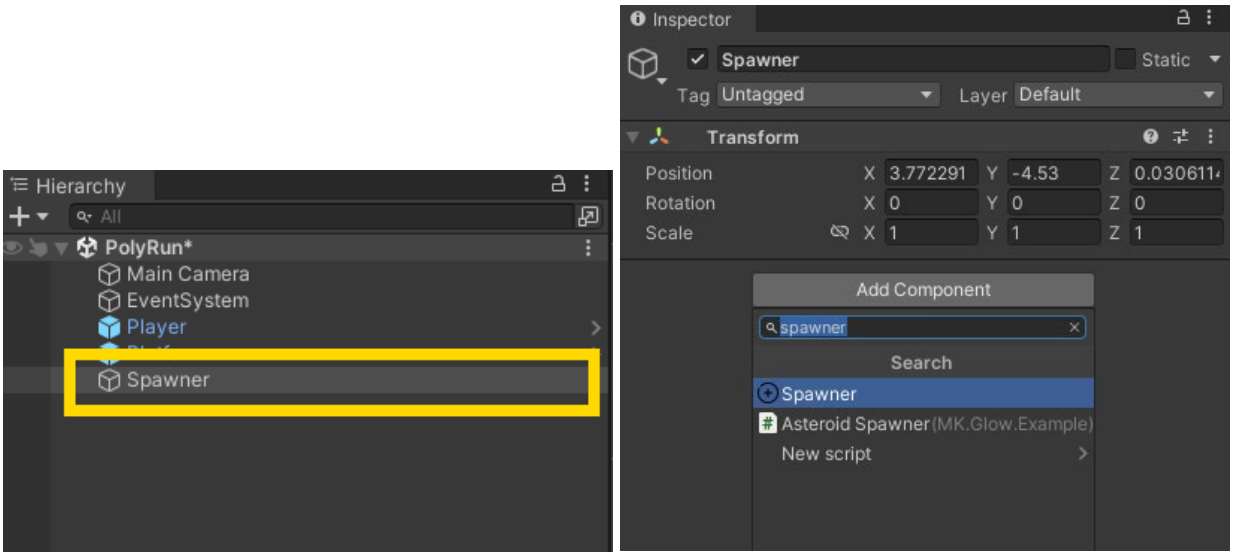
33 Inside the **void Update** function, add the following code:

```
    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        //Add vector3.left (-1,0,0) to our transform, times by speed
        transform.position += Vector3.left * speed * Time.deltaTime;
    }
```

Save the script and return to Unity.

34 You'll want to spawn the **ChallengeObj** as multiple clones. Start by creating a new **GameObject** and renaming it to ***Spawner***.

35 Select the Spawner and click **Add Component**. Type ***Spawner*** in the search box and select the **Spawner** script component.



36 Open the **Spawner** script and add the following variables:

```
Unity Script | 0 references
public class Spawner : MonoBehaviour
{
    [Header("ChallengeObj Game Object")]
    public GameObject challengeObject;
    [Header("Default Spawn Delay Time")]
    public float spawnDelay = 1f;
    [Header("Default Spawn Time")]
    public float spawnTime = 2f;
```

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

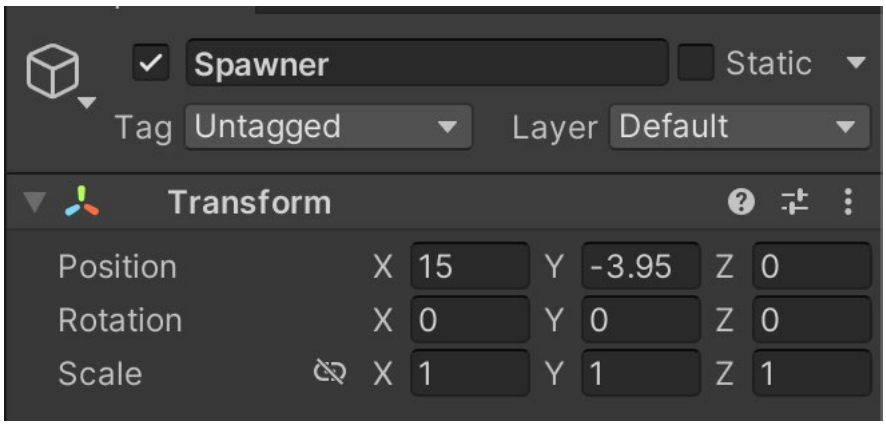
```
// Start is called before the first frame update
Unity Message | 0 references
void Start()
{
    //start the function after
    //spawnDelay (1) and Repeat the function
    //InstantiateObject every spawnTime (2)
    InvokeRepeating("InstantiateObjects", spawnDelay, spawnTime);
}
```

38 Create a new **void** function called **InstantiateObjects**. Add the following:

```
29
                0 references
30              void InstantiateObjects()
31              {
32                  Instantiate(challengeObject, transform.position, transform.rotation);
33              }
```

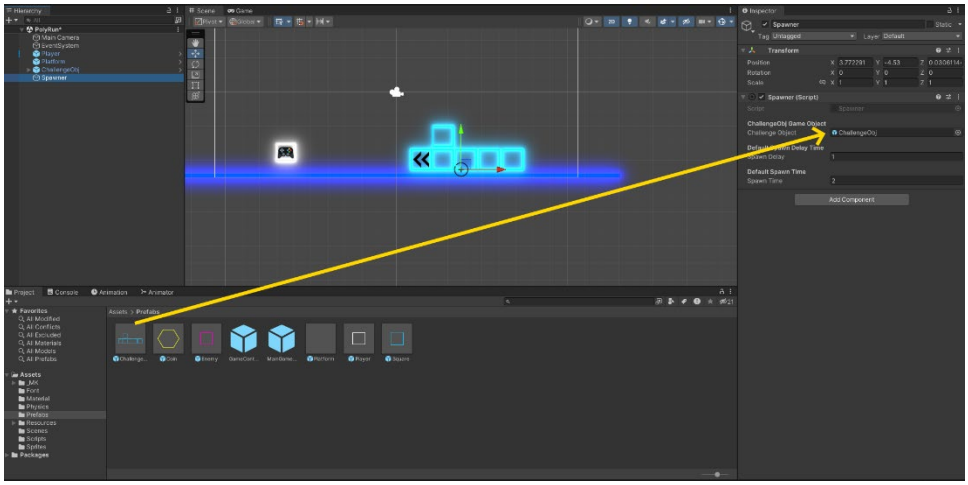
Save the script and return to Unity.

39 Next with the **Spawner** still selected set the position in the **inspector** to match the values below.



Position Values: **X:15, Y: -3.95, Z:0**

40 With the **Spawner** selected, search and select the **ChallengeObj** game object from the **Assets** in the Spawner Script component or drag it from the prefabs folder into the slot.



41 Delete the original **ChallengeObj** gameobject from the hierarchy.

42 Play the game. During game play, alter the Spawn time to adjust how fast you want the objects to spawn. If you are satisfied with the current settings, change the spawn time when you stop the game.

43 Stop the game.

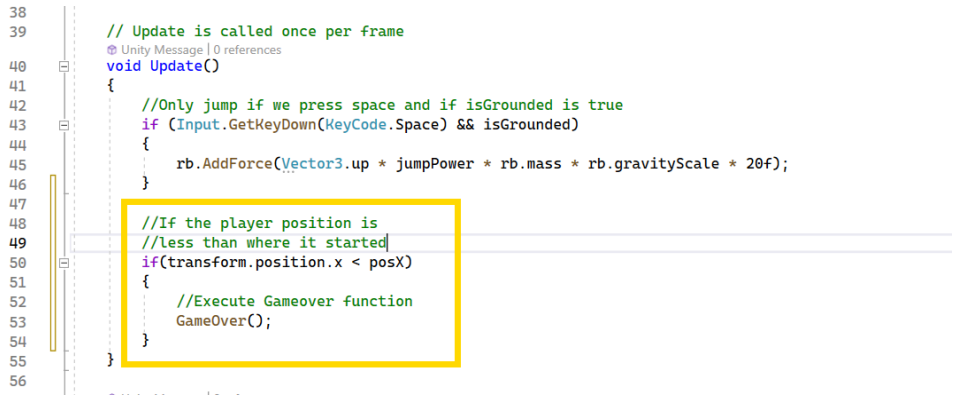
44 You may have noticed that the player object gets pushed to the side if it doesn't jump onto the platform in time. To fix this, you will need to add to both the Player and the Spawn Controller scripts.

Open the **PlayerControls** script first. Create a new **void GameOver** function, add the following:

```
17
18      // Start is called before the first frame update
        Unity Message | 0 references
19      void Start()
20      {
21          //Variable rb equals to Rigidbody2D
22          //component on the object
23          //this script is attached to
24          rb = GetComponent<Rigidbody2D>();
25
26          //posX = starting position
27          posX = transform.position.x;
28
29          Time.timeScale = 1;
30      }
31
32      //The function to call when we GameOver
        0 references
33      void GameOver()
34      {
35          Time.timeScale = 0;
36      }
37
```

We also need to reset our **TimeScale**, so add a small line **on Start** to reset it back to **1**. Otherwise, our game will stay paused!

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



```
38
39      // Update is called once per frame
        Unity Message | 0 references
40      void Update()
41      {
42          //Only jump if we press space and if isGrounded is true
43          if (Input.GetKeyDown(KeyCode.Space) && isGrounded)
44          {
45              rb.AddForce(Vector3.up * jumpPower * rb.mass * rb.gravityScale * 20f);
46          }
47
48          //If the player position is
49          //less than where it started
50          if(transform.position.x < posX)
51          {
52              //Execute Gameover function
53              GameOver();
54          }
55      }
56
```

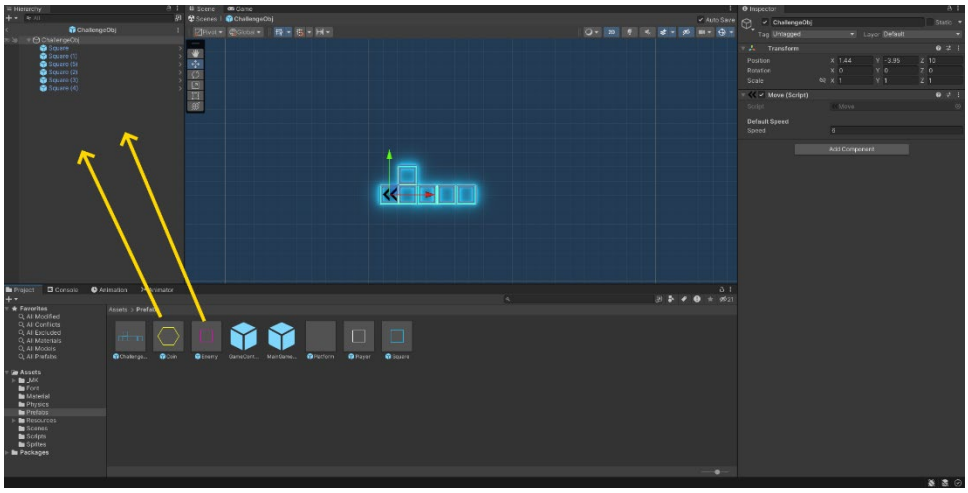
Save the script and return to Unity.

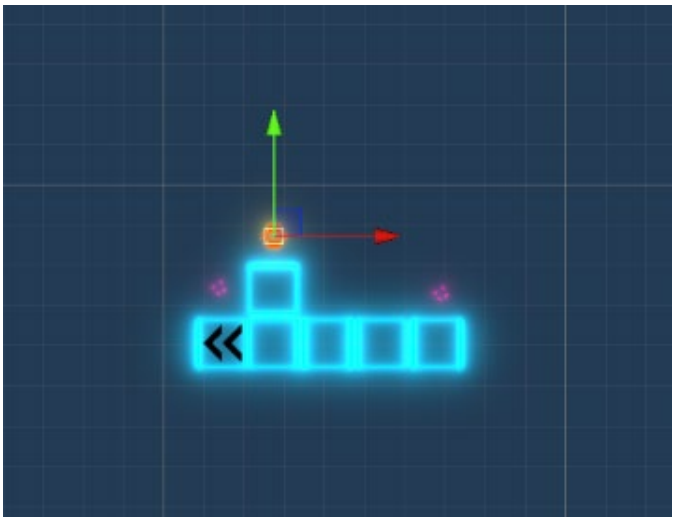
46 Play the game. Does the GameOver function work?

47 Stop the game.

48 To add more objects to the ChallengeObj, go into the **Prefabs** folder. Double-click the **ChallengeObj** to open the prefab. You'll see the Scene window turn blue as this is where you would edit the prefab.

49 In the prefab folder, drag the **coin** prefab and **2 enemy** prefabs into the scene and place the objects according to the image below.





50 Click the arrow next to the prefab name to go back to the regular scene. The prefab will save automatically.

51 Go back to the **PlayerControls** script. Inside the **void OnCollisionEnter2D** function, add the following:

```
Unity Message | 0 references
private void OnCollisionEnter2D(Collision2D collision)
{
    //If the object we collide with has the
    //tag Ground
    if(collision.gameObject.tag == "Ground")
    {
        //isGrounded is set to true
        isGrounded = true;
    }

    if(collision.gameObject.tag == "Enemy")
    {
        GameOver();
    }
}
```

52 Create a new **void OnTriggerEnter2D** function and add the following:

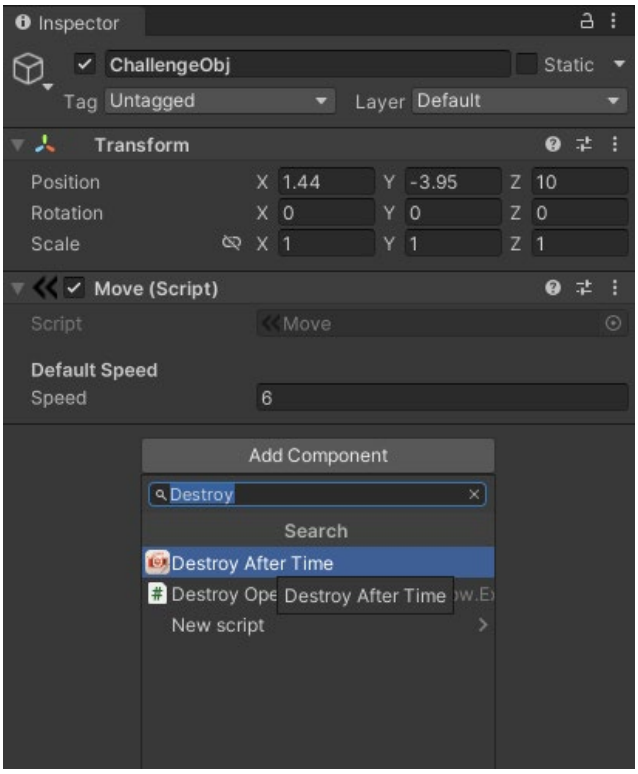
```
Unity Message | 0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    if(collision.tag == "Coin")
    {
        Destroy(collision.gameObject);
    }
}
```

Save the script and return to Unity.

53 Once back in Unity, play the scene. Collect the coins and try not to touch the enemies!

54 Stop the game.

55 At this point, you'll want to destroy the challenge objects that are still active off screen. You can destroy these objects by using a Destroyer script. Select the **ChallengeObj** in the prefab folder, select **Add Component**. Type ***Destroy*** in the search box and select the **Destroy After Time** script.



56 Open the **Destroyer** script and add the following variable:

```
Unity Script (1 asset reference) | 0 references
public class DestroyAfterTime : MonoBehaviour
{
    [Header("Destruction Time")]
    public float timeToDestruction = 6;
```

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

```
10  // Start is called before the first frame update
    Unity Message | 0 references
11  void Start()
12  {
13      Invoke("DestroyObject", timeToDestruction);
14  }
15
```

58 Create a new **void DestroyObject** function and add this to the script:

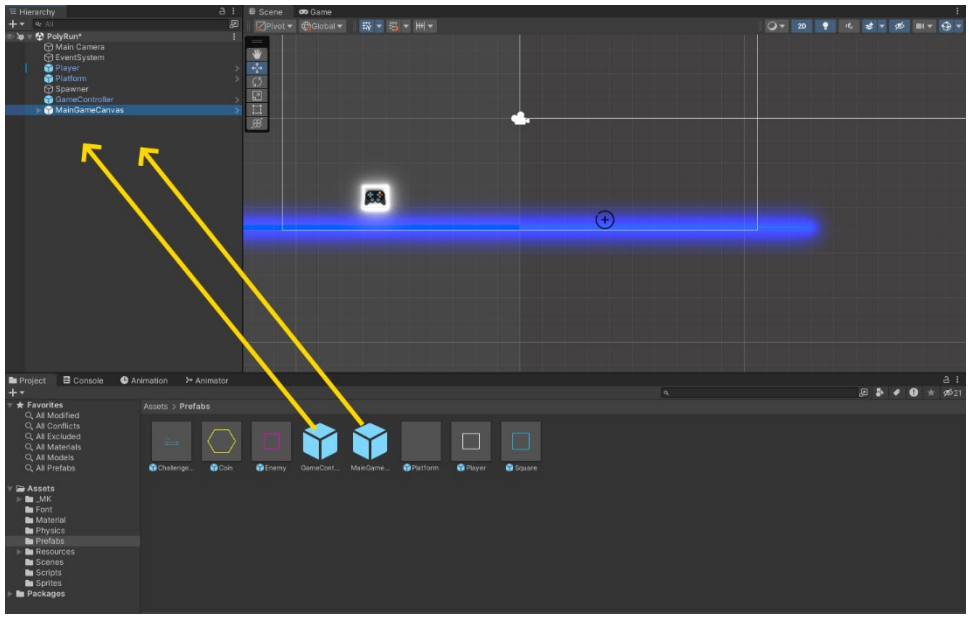
```
15
    0 references
16  void DestroyObject()
17  {
18      Destroy(gameObject);
19  }
```

Save the script and return to Unity.

59 Play the game. Look at the Hierarchy, you should see the objects being destroyed.

60 Stop the game.

61 Like the other games, it's time to add scoring and a Game Over panel. Drag the **GameController** and **MainGameCanvas** to the Hierarchy.



62 Now, open the **PlayerControls** script. Inside the **OnTriggerEnter2D** function, add the following:

```
94
        Unity Message | 0 references
95      private void OnTriggerEnter2D(Collider2D collision)
96      {
97          if(collision.tag == "Coin")
98          {
99              //Find the game controller and add to our score
100             GameObject.Find("GameController").GetComponent<GameController>().IncrementScore();
101
102             Destroy(collision.gameObject);
103         }
104     }
105
```

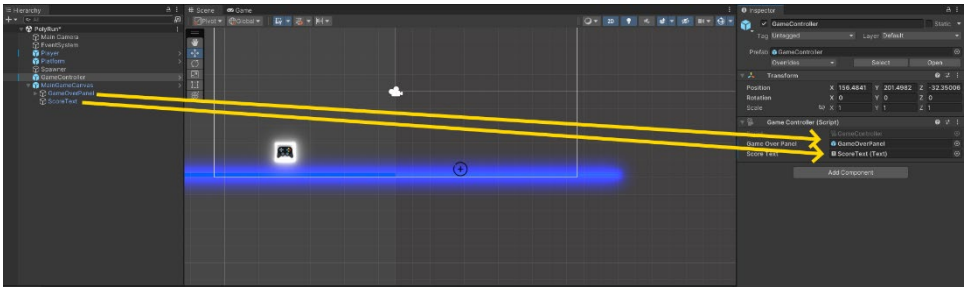
63 Inside the **void Game Over** function, delete the old **GameOver** function call. Then, add the following:

```
32      //The function to call when we GameOver
        2 references
33      void GameOver()
34      {
35          GameObject.Find("GameController").GetComponent<GameController>().GameOver();
36
37          //GameOver()
38      }
39
```

Save the script and return to Unity.

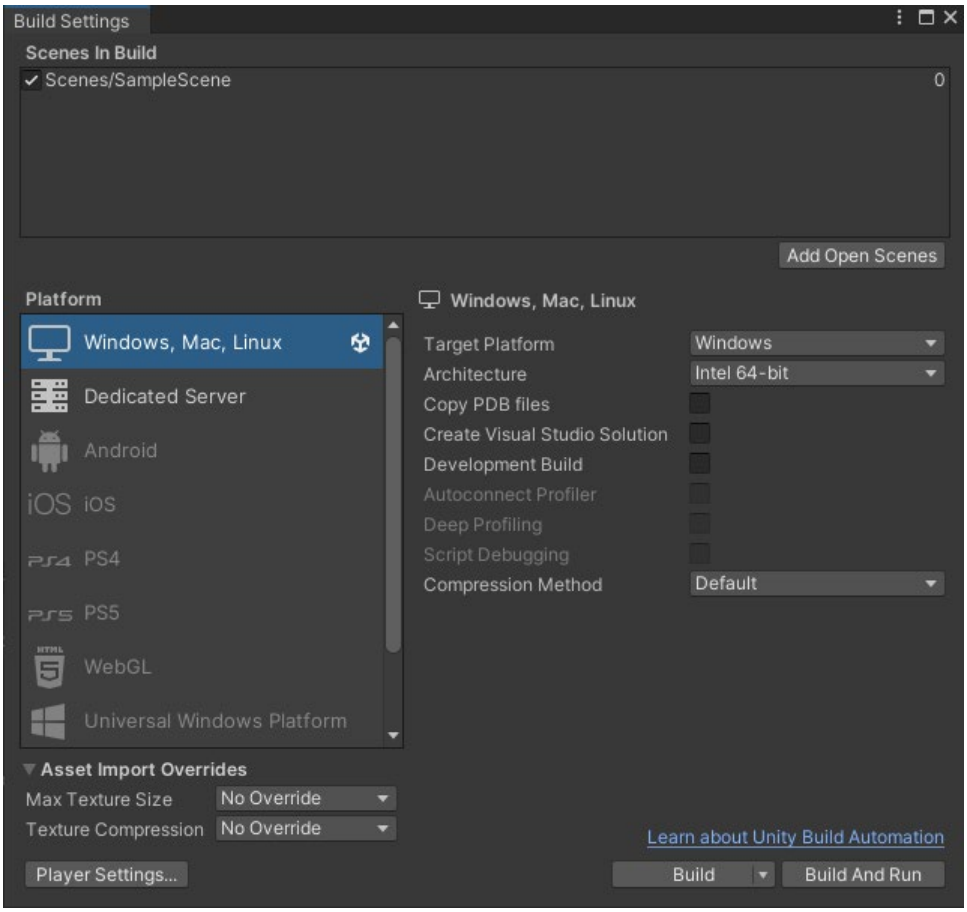
64 Select the **GameController** object. In the script components, in the **GameOverPanel** input field, search and select the **GameOverPanel** object in your Hierarchy. (You can also drag it into the slot from the Hierarchy).

In the same script component, we need to do the same for the **Score Text** input field, search and select **ScoreText** from the scene panel. (Or drag from the Hierarchy).

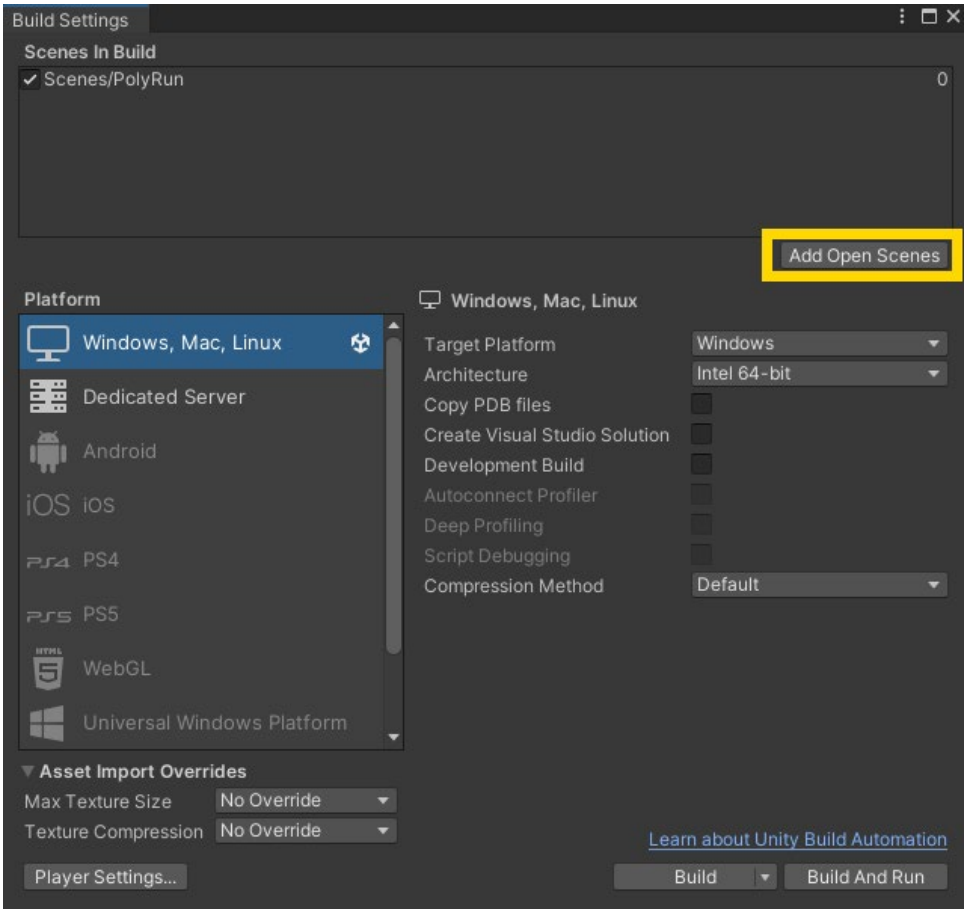


65 Play the scene. You'll see that if the player touches the coin the score counter shown above will add 1 to the score. If the player moves from its position on the X-axis or touches the enemy, the **GameOverCanvas** will appear to show the final score. There will also be a button to restart the game.

66 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.



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



68 Play your game and see what you get. Now is the time to change the settings. Maybe you need a more powerful jump? Or maybe the spawning is too quick?

When you are ready, **save** your game and **export** it. Then, **Submit** your game.