CODE NINJAS®

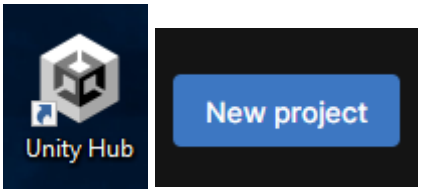
Bronze Belt Ninja Guide

Activity 03: Meany Bird

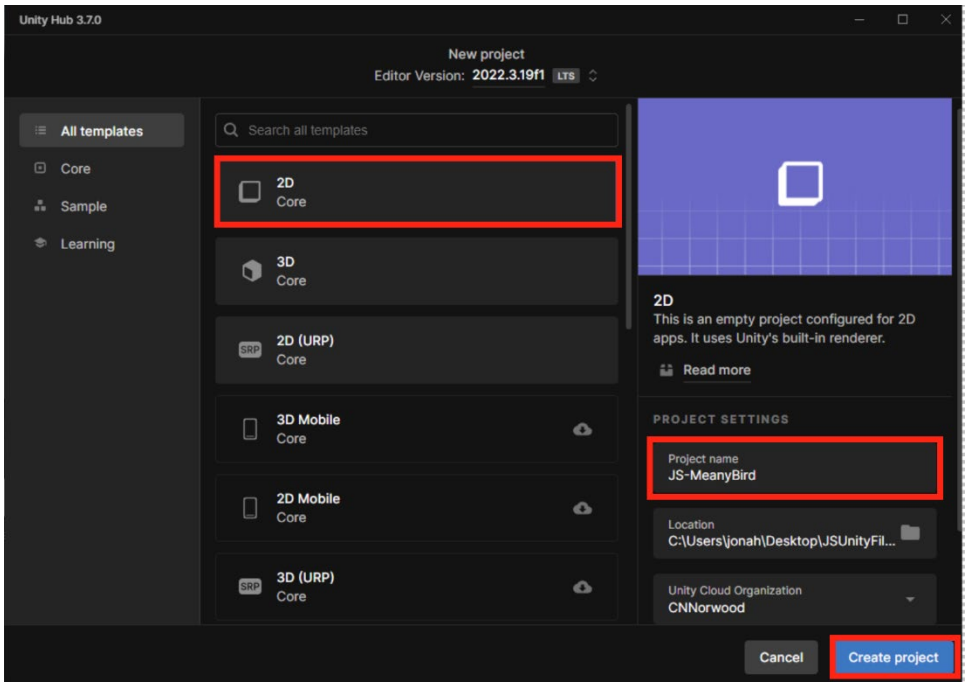
Activity 3: Meany Bird

In this lesson, you'll be learning how to create your own animations using the bird sprite we have provided for you.

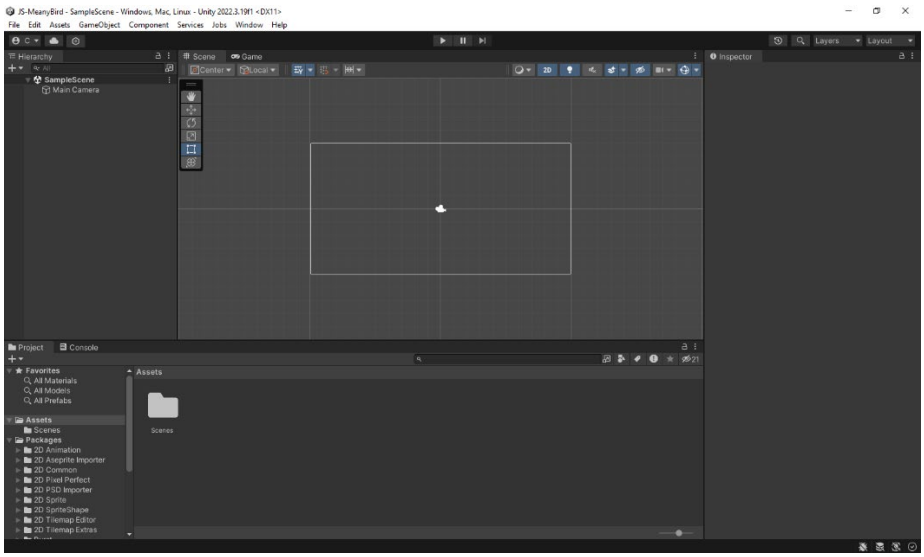
1 Open the Unity Hub application on your computer. Select the **New Project** button in the upper right-hand corner.



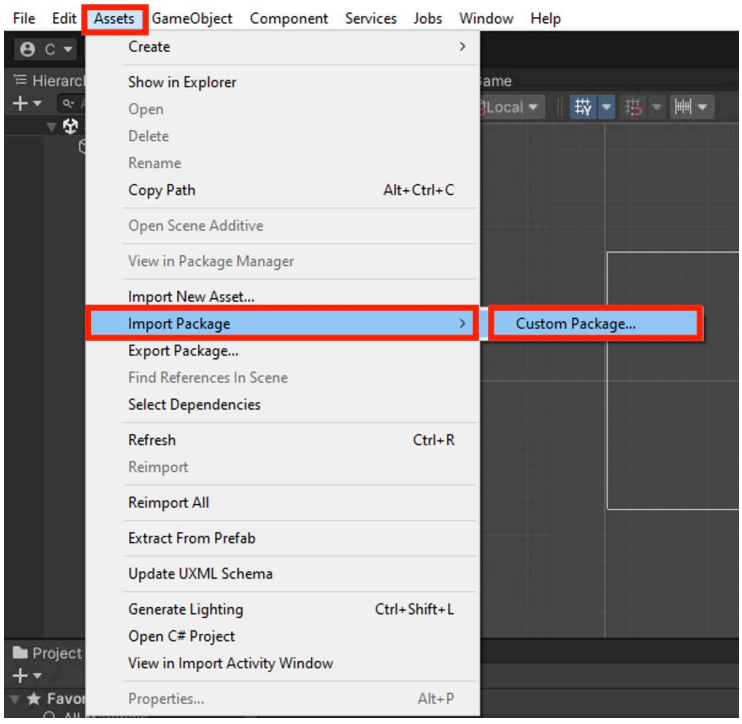
2 Ensure that the top shows that the **2022.3 LTS** version is being used. Select **2D** in the templates column. Next, type in your first name and last name initials and the name of the project. For example, John Smith would save their project as JS-MeanyBird. Select the folder location where you want to save your project. Click the **Create** button.



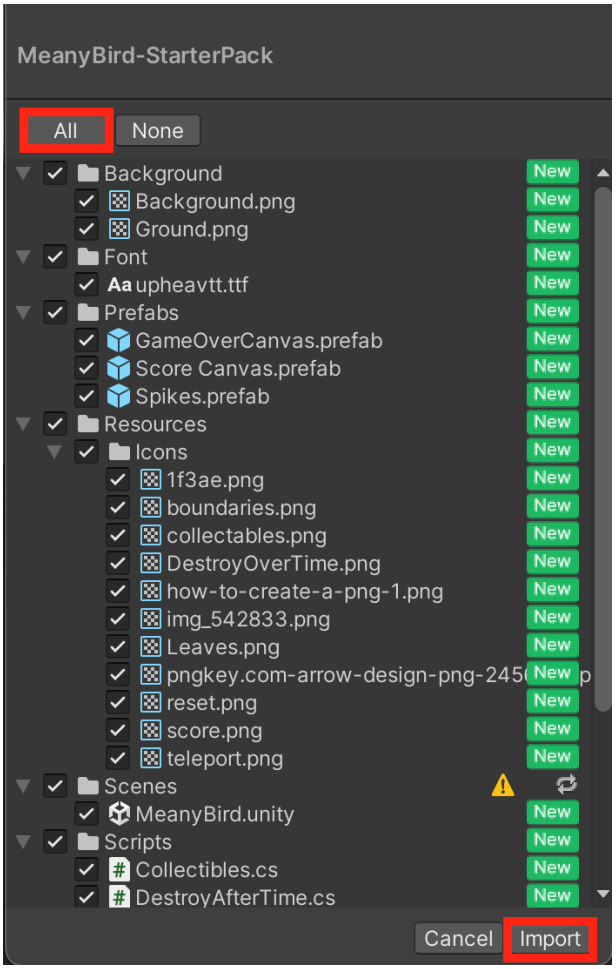
3 This is what you will see when Unity has loaded the 3 necessary assets into the program. Go to the **Assets** menu.



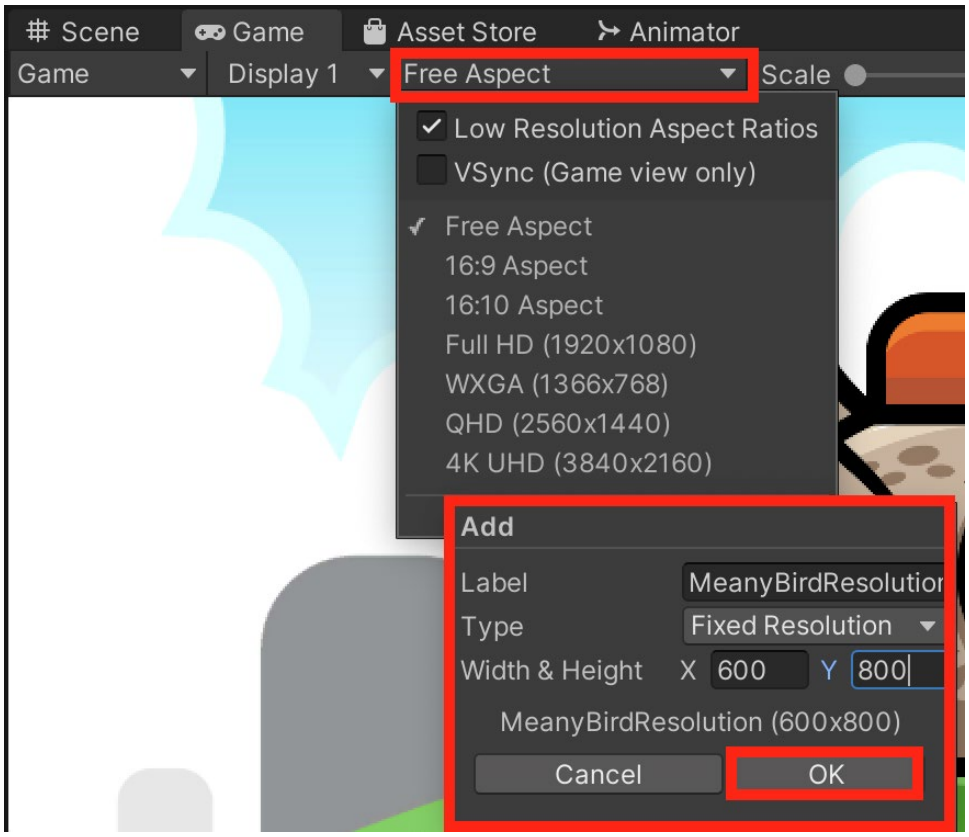
4 Select **Import Package**, then click on **Custom Package**. This allows us to import any package that has been compressed in Unity. Select the **Activity 03 - MeanyBird-StarterPack.unitypackage** file. Once imported, Unity will show a list of everything that is inside the package.



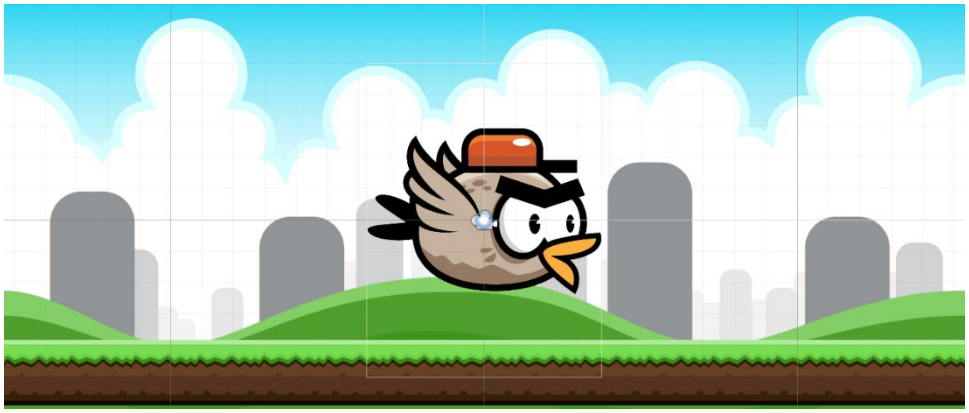
5 All the materials inside the Unity package should be selected. If they aren't, click the **All** button on the bottom-left. Click **Import** (bottom-right) and wait until all the material is imported into the project.

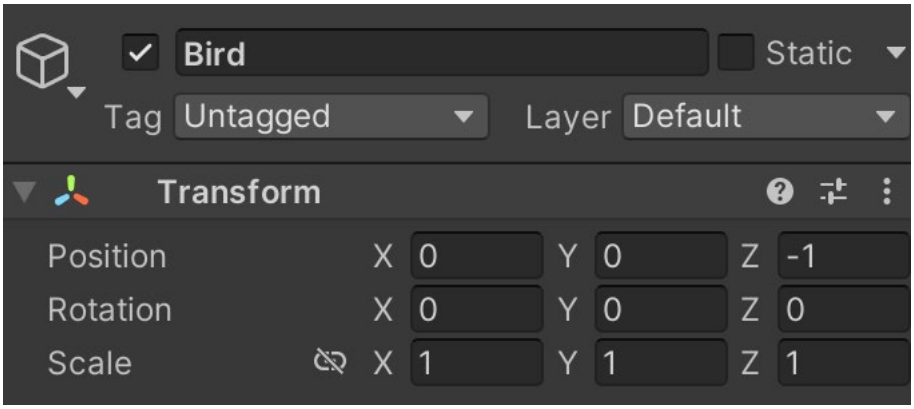


6 Navigate to the **Scenes** folder under the Assets directory (accessible via the Projects tab) and open the MeanyBird scene. In the Game window, click on Free Aspect. With this tab still open next, click on the "+" symbol at the bottom to adjust the resolution size. Give the resolution a label like MeanyBirdResolution. Then, set the Width and Height to 600x800.



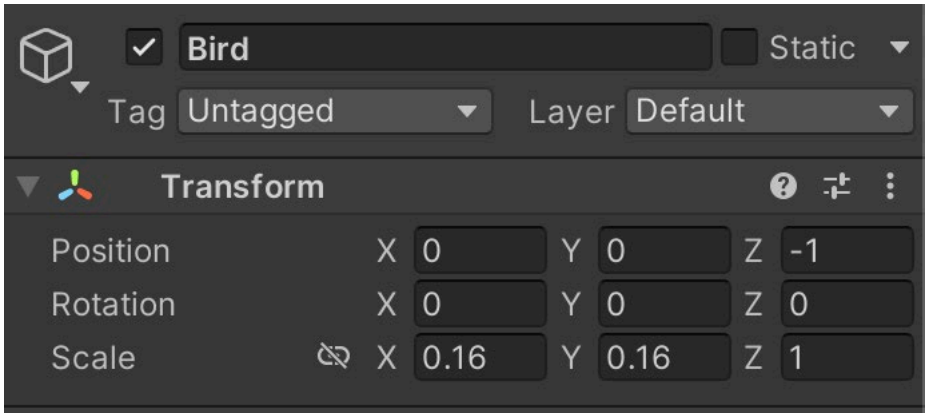
7 Now, let's add our bird sprite. Go to the **Sprites** folder and select the bird sprite. Drag it into the hierarchy and the bird should be shown in the game view. If not, change the Z values in the transform window as shown below.





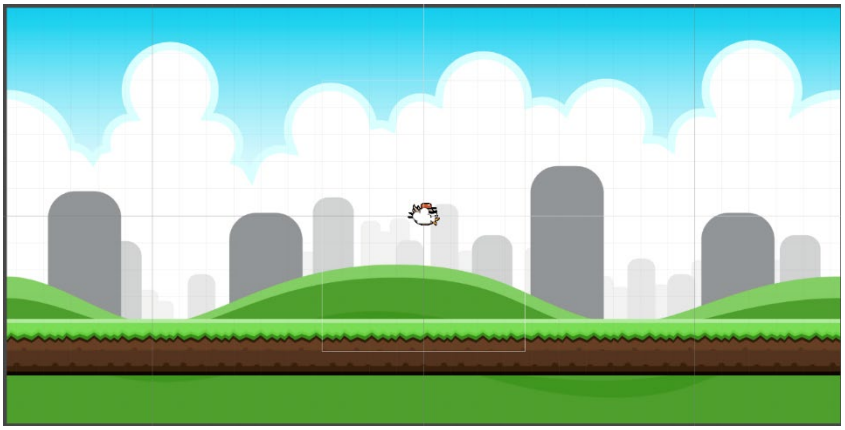
Position values: X: 0, Y: 0, Z: -1

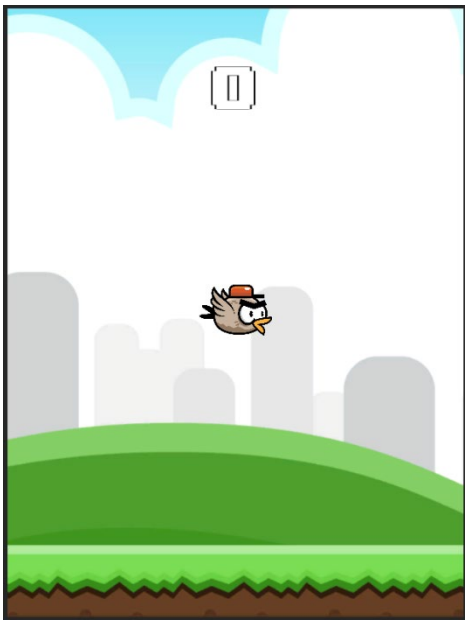
8 Now the bird is too big for the game view, so let's change its size. With the bird still selected, navigate to the **Inspector** window. Change the size of the bird using the following scale values:



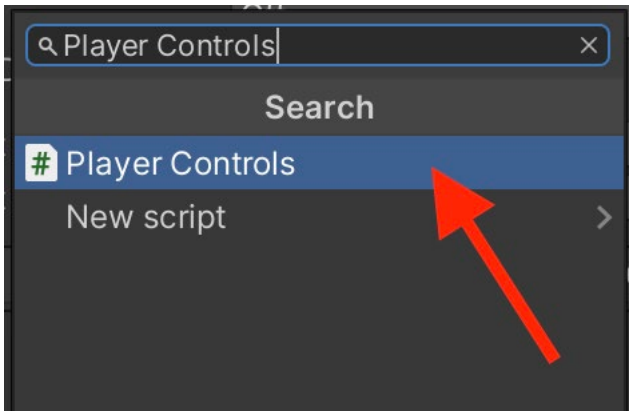
Scale values: X: 0.16, Y: 0.16, Z: 1

Your bird should look like this in both the scene and game windows:

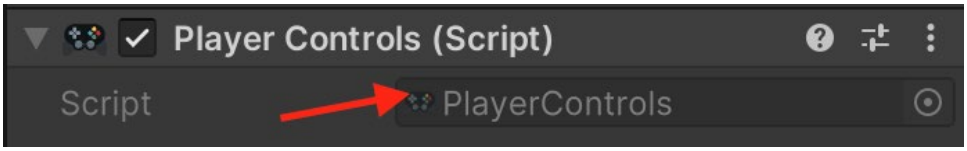




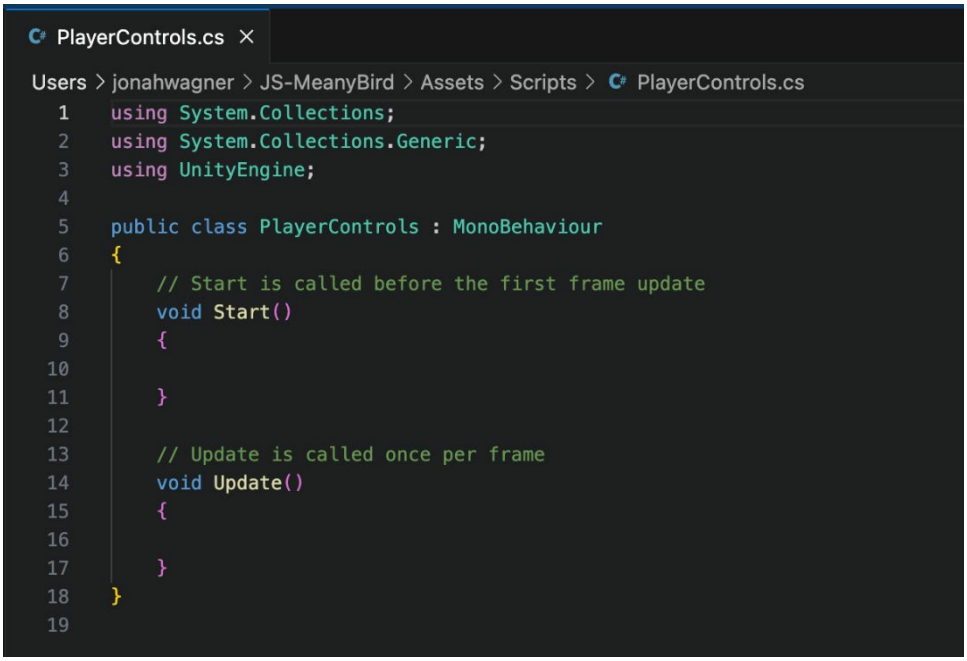
9 With the bird still selected, click the **Add Component** button in the Inspector window. Type ***Player*** and select **Player Controls**.



10 Now, double-click on the script to open Visual Studio. Note: If Visual Studio opens and there is nothing showing, double-click the script again.



This is what your script should look like when Visual Studio is opened:



Pro Tip: Scripting

- The "**using**" section at the top of the script imports additional tools and resources, allowing the program to access predefined functionalities.
- A **class** is a blueprint or template for creating objects in programming. It defines the properties (attributes) and behaviors (methods) of objects.
- A **public class** is accessible from other parts of the program, meaning its properties and methods can be accessed and modified from anywhere.
- A **private class** restricts access to its properties and methods only within the class itself, keeping them hidden from other parts of the program.
- A **header** in programming typically refers to the introductory section of a file or script, containing essential information like the name of the file, author details, creation date, and a brief description of its purpose.
- A **public variable** is a data container accessible and modified from any part of the program. It is like a public resource that anyone can use.
- A **private variable**, on the other hand, can only be accessed and modified within the class it is declared in. It is like a secret resource that is only available to certain parts of the program.
- A **float** is a data type used in programming to represent numbers with decimal points. It is used for storing values such as distances, measurements, or percentages.
- **Void** can be used as the return type of a method (or a local function)

11 Now, let's start coding our bird. The code needs to get placed under **public class playerControls: monobehaviour**. This code defines variables. Add this code:

```
public class PlayerControls : MonoBehaviour
{
    //Game manager object
    [Header("Game Controller Object for controlling the game")]
    public GameController gameController;
    [Header("Velocity")]
    public float velocity = 1;  // Corrected the header to "Velocity"
    //Physics for the bird
    private Rigidbody2D rb;
    //Height of the bird object on the y axis
    private float objectHeight;
```

Next, type inside the **void Start** function:

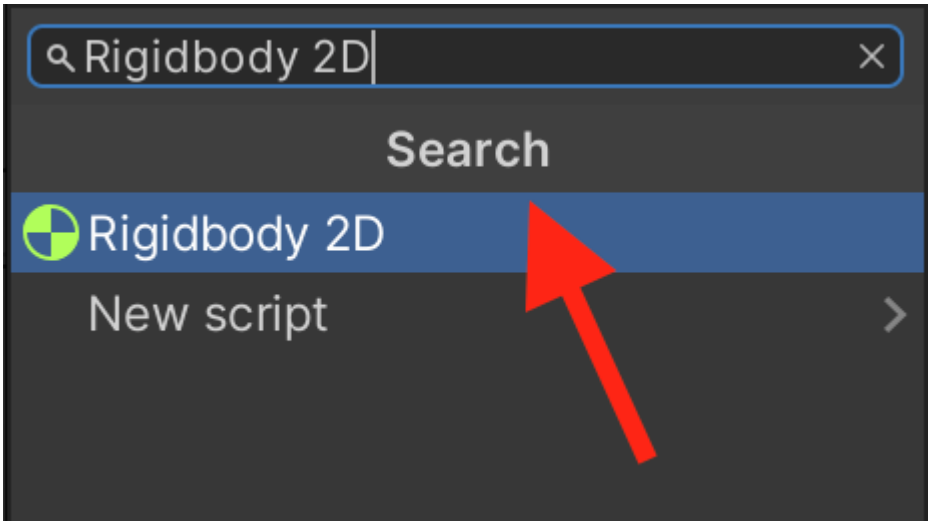
```
// Start is called before the first frame update
void Start()
{
    //Game Controller component
    gameController = GetComponent<GameController>();
    //Speed for the game is at a playing state
    Time.timeScale = 1;
    rb = GetComponent<Rigidbody2D>();
    //Object Height equals the size of the height of the sprite
    objectHeight = transform.GetComponent<SpriteRenderer>().bounds.size.y / 2;
}
```

Now, type inside the **void Update** function:

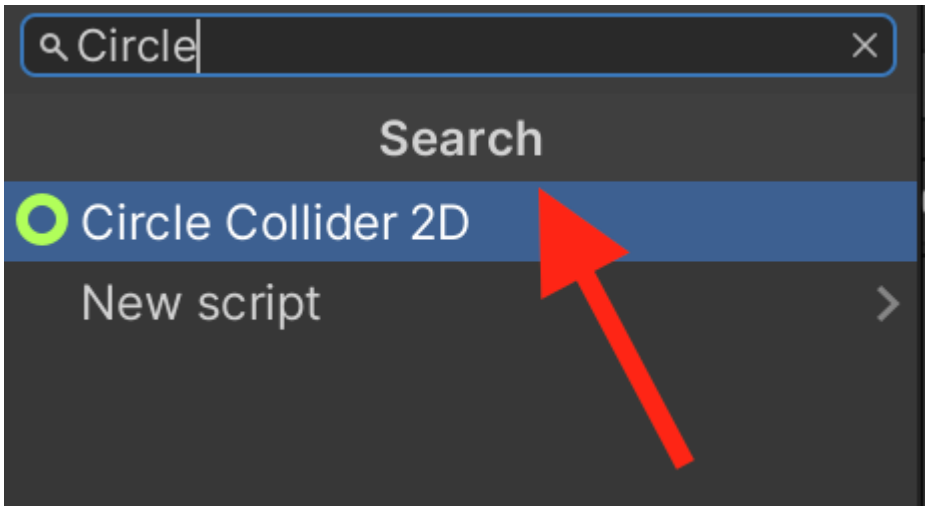
```
// Update is called once per frame
void Update()
{
    //If the left mouse button is clicked
    if (Input.GetMouseButtonDown(0))
    {
        //The bird will float up on the Y axis
        //and float back down on Y axis
        rb.velocity = Vector2.up * velocity;
    }
}
```

Now, save the script by pressing **Crtl + S** and return to Unity!

12 We are almost ready to playtest the game, but we must do a few more things first! Select the **Bird** GameObject in the **Hierarchy** panel, then click **add component** in the **Inspector** menu. In the input field, type ***Rigidbody*** **2D** and then select **Rigidbody 2D**.



13 Click **Add Component** once more. Then type ***Circle Collider 2D*** in the input field**.** Select **Circle Collider 2D.**



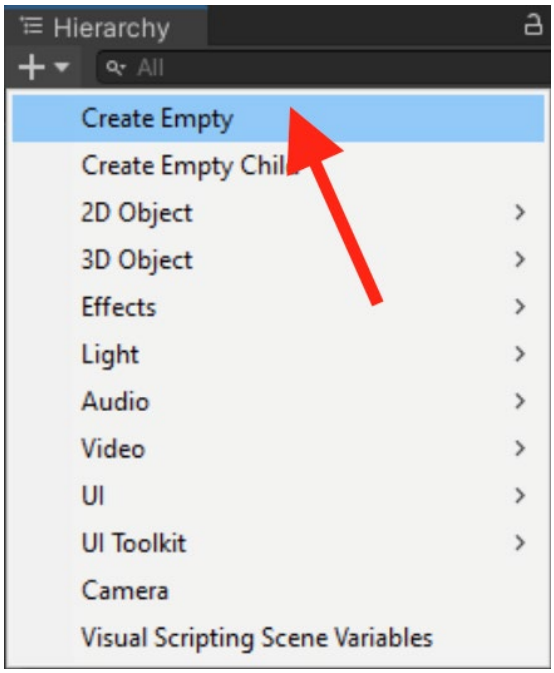
14 Finally, change the velocity under player controls to 5 in the inspector so the bird can float around. Make sure to change from **Play Focused** or **Play Unfocused** to **Play Maximized** before you click the Play button.

```
public class PlayerControls : MonoBehaviour
{
    //Game manager object
    [Header("Game Controller Object for controlling the game")]
    public GameController gameController;
    [Header("Default Score")]
    public float velocity = 5;
    //Physics for the bird
    private Rigidbody2D rb;
    //height of the bird object on the y axis
    private float objectHeight;

    // Start is called before the first frame update
    void Start()
    {

    }
```

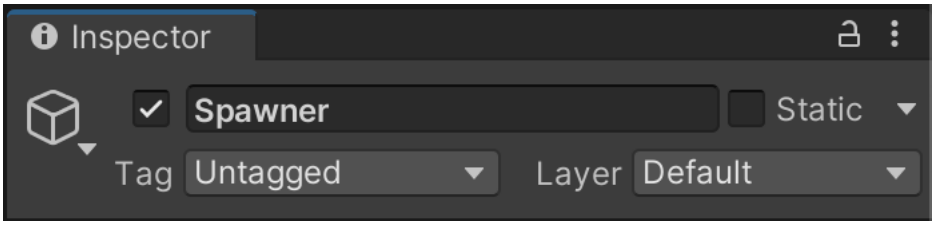
Now, let's playtest your game! Continue clicking on the game screen to make the bird fly.

15 Stop the game by selecting the play button again.

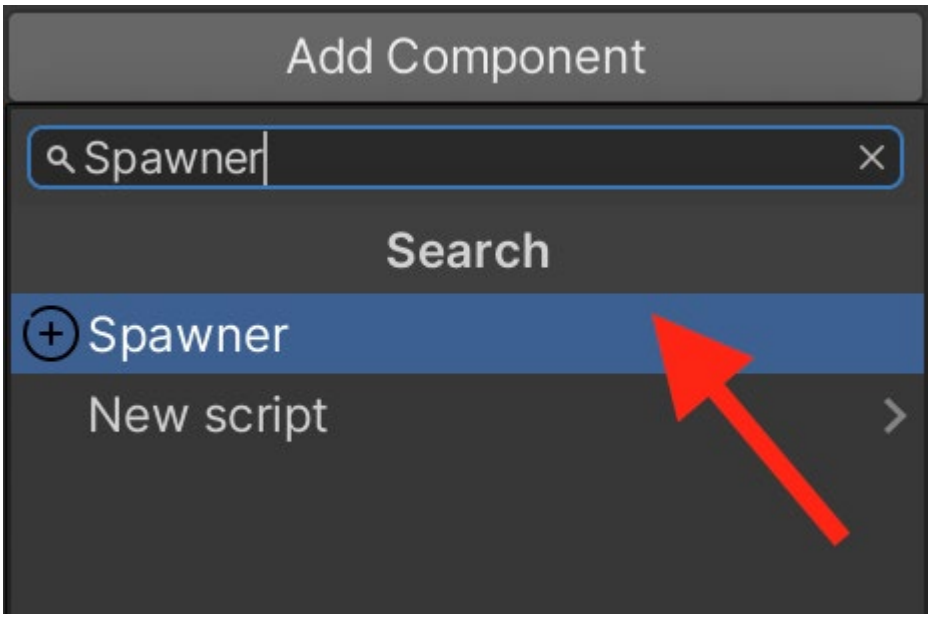
16 With our bird set up, let's add spikes into the game. In the **Hierarchy** panel, click the **+** button, then select **Create Empty**.



In the Inspector menu, rename the empty game object, "Spawner".



17 In the **Inspector** window for the Spawner, select **Add Component**. In the search box, type ***Spawner*** and select the **Spawner** script component.



18 Open the Spawner script. Add the following to the script inside the **class** before the void start function:

```
public class Spawner : MonoBehaviour
{
    //Object that we will attach to the script for spawning object
    [Header("Spikes Object for controlling the game")]
    public GameObject spikes;
    //Height position of the spikes
    [Header("Default Height")]
    public float height;
    // Start is called before the first frame update
    void Start()
    {

    }
```

Next, type inside the **void Start** function:

```
// Start is called before the first frame update
void Start()
{
    //Start function repeating every 4 seconds
    InvokeRepeating("InstantiateObjects", 1f, 4f);
}
```

This will help us set up the timer to decide when the next object will appear.

Next, type inside the **void Update** function:

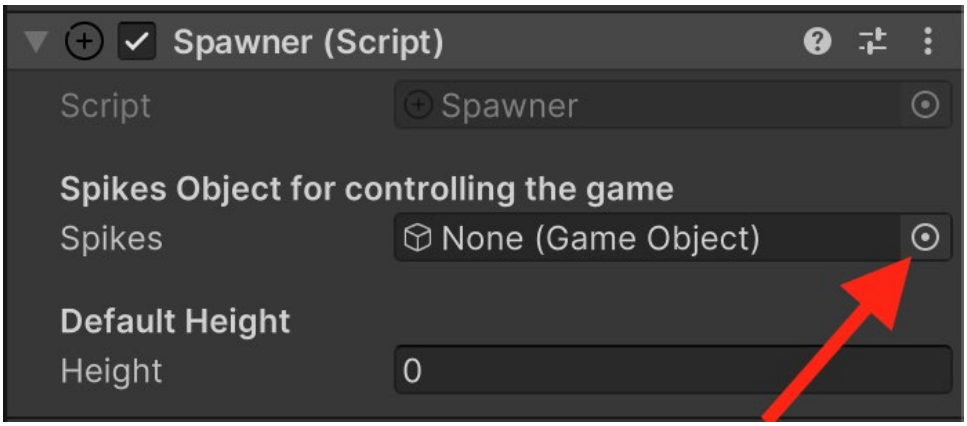
```
void Update()
{
    //Position for the gameobjects
    spikes.transform.position = new Vector3(5, Random.Range(-height, height), 0);
}
```

Create a new void function called ***InstantiateObjects*** inside the class and just after the **void Update** function. Add this code inside:

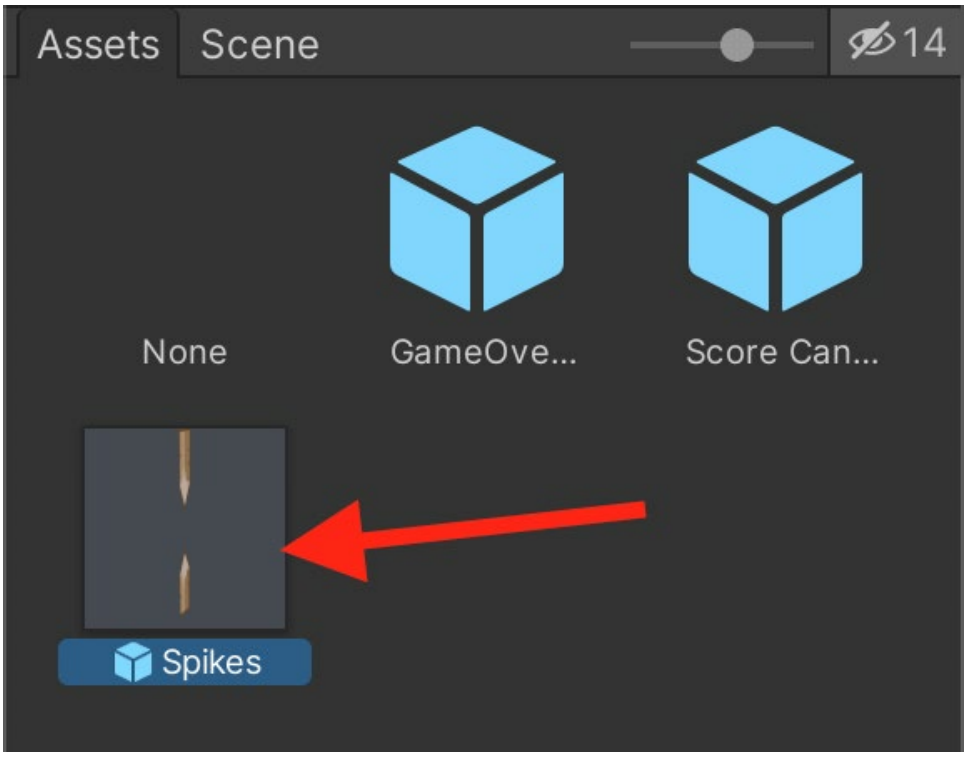
```
//InstantiateObjects Function
void InstantiateObjects()
{
    //Spawn object by position and rotation
    Instantiate(spikes, spikes.transform.position, spikes.transform.rotation);
}
```

Save the script (**Ctrl + S**) and return to Unity!

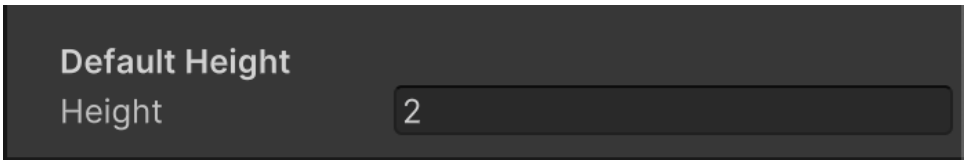
19 Using our updated script, look in the **Inspector** window. The Spawner has a **GameObject** input field. Click the properties button next to the input field as shown below.



20 Select **Assets** on the top left of the window that appeared. You should see a menu like the one below. Then select the **Spikes** object.



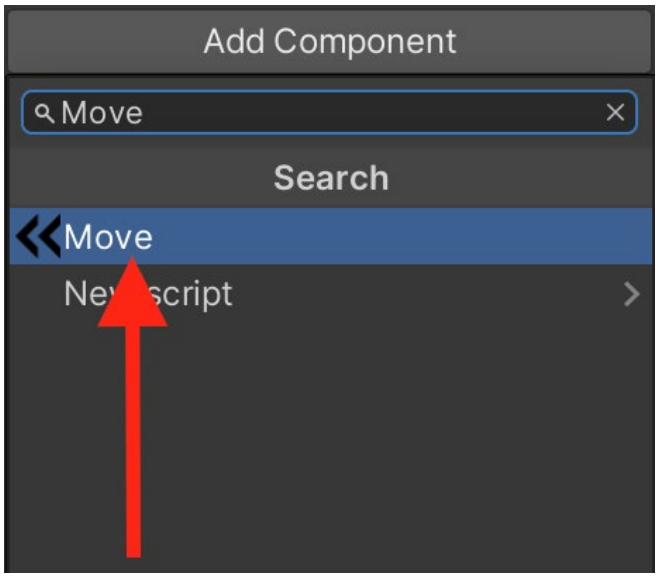
21 In the **Inspector**, navigate to the **Spawner** component and adjust the height value from **0** to **2**.



22 Now, let's go ahead and play the game. While playing, look at the Hierarchy. You should notice that the objects are being generated but aren't moving across the screen.

23 Stop the game.

24 Switch back to the scene tab. Go into the **Prefabs** folder and double-click the **Spikes** prefab. Make sure the **Spikes** parent object is selected and shown in the **inspector** menu. Then, click **Add Component** in this **inspector** menu and type **Move** in the search box. Select the **Move** script component.



Open the Move script and add the following:

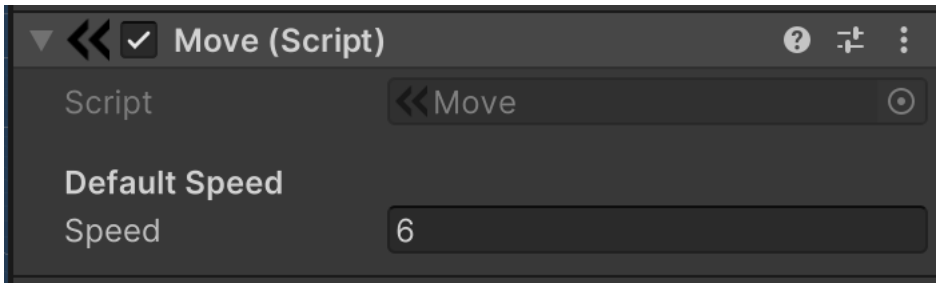
```
public class Move : MonoBehaviour
{
    [Header("Default Speed")]
    //Speed for the movement
    public float speed;
    // Start is called before the first frame update
    void Start()
    {

    }
```

Next, delete the **void Start** function and type inside the **void Update** function:

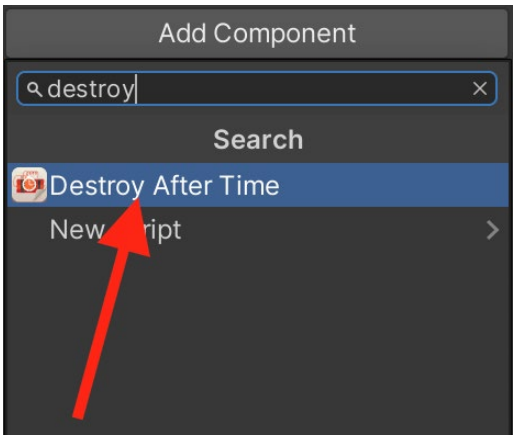
```
public class Move : MonoBehaviour
{
    [Header("Default Speed")]
    //Speed for the movement
    public float speed;

    // Update is called once per frame
    void Update()
    {
        //Transform the object to move left
        //according to the axis and speed
        transform.position += Vector3.left * speed * Time.deltaTime;
    }
}
```

Save the script (**Ctrl + S**) and return to Unity!

25 With the **Spikes** parent object still selected in the inspector, change the speed value from **0** to **6**.



26 Now, with the **Spikes** parent object still selected, click **Add Component**. In the search box, type **Destroy** and select the **Destroy After Time** script.



Open the **Destroy After Time** script and add the following:

```
public class DestroyAfterTime : MonoBehaviour
{
    //After this time, the object will be destroyed
    [Header("Default Destruction Time")]
    public float timeToDestruction;
    // Start is called before the first frame update
    void Start()
    {
```

Next, type inside the **void Start** function:

```
void Start()
{
    //Execute DestroyObject function based on timeToDestruction
    Invoke("DestroyObject", timeToDestruction);
}
```

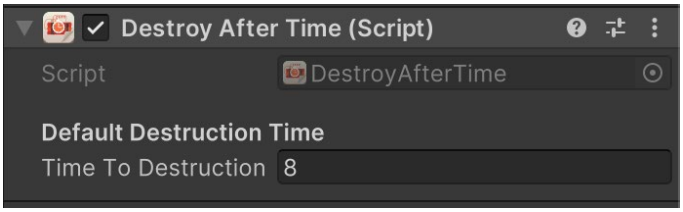
Invoke allows you to specify the time when a function is run.

Create a new void function called **DestroyObject**, then add this:

```
//This function will destroy this object
void DestroyObject()
{
    //Destroy Object
    Destroy(gameObject);
}
```

You can delete the **void Update** function as this script does not require it in this context. Once finished, save the script and return to Unity!

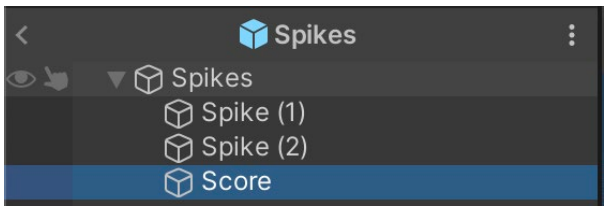
27 Now, in the **Destroy After Time** component, change the value for Time to Destruction from **0** to **8**.



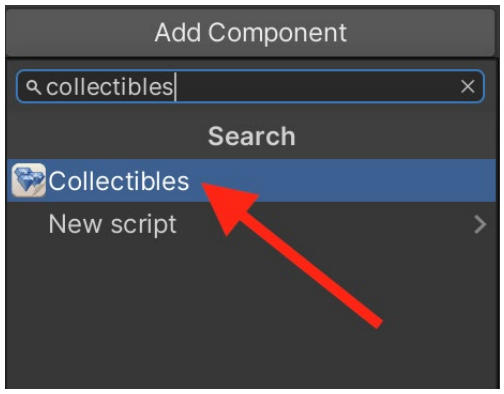
28 Now, play the game!

29 Stop the game.

30 Double-click the **Spikes** prefab and this time select the **Score** child object.



31 Click **Add Component**. Type ***Collectibles*** in the search box and select the **Collectibles** script.

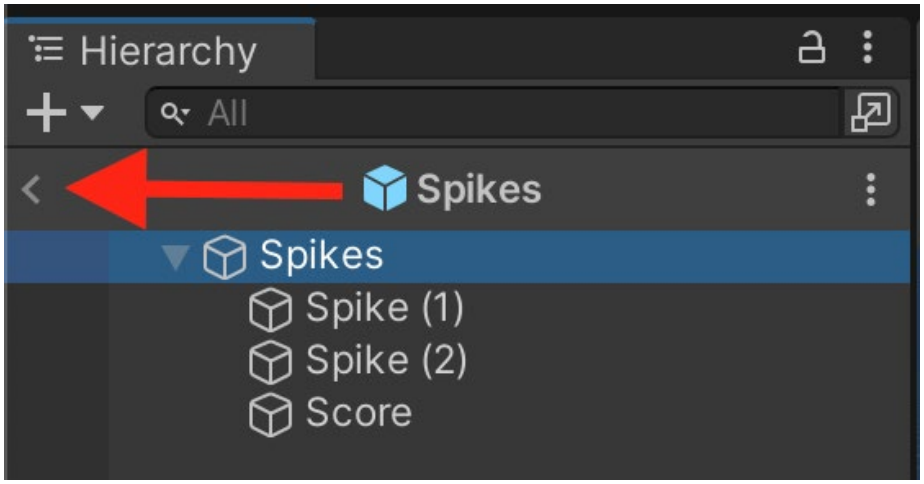


Open the **Collectibles** script. Delete both **void Start** and **Update** functions. Inside the class, create a new void function called **OnTriggerEnter2D**, add this:

```
public class Collectibles : MonoBehaviour
{
    // Start is called before the first frame update
   //If an object collides with a trigger
    private void OnTriggerEnter2D(Collider2D collision)
        {
            //Add score
            Score.score++;
        }
}
```

Save the script and return to Unity!

32 In the Hierarchy panel, click the arrow shown blown to return to Scene mode.



33 Now, play the game. Try to aim between the spikes to add points to the score.

34 Stop the game.

35 Before moving on, let's add a function for if the player touches the two spikes or the ground, then the game is over. Reopen the **Player Controls** script.

Create a new void function inside the class below the void Update function called **void OnCollisionEnter2D**, afterwards add this:

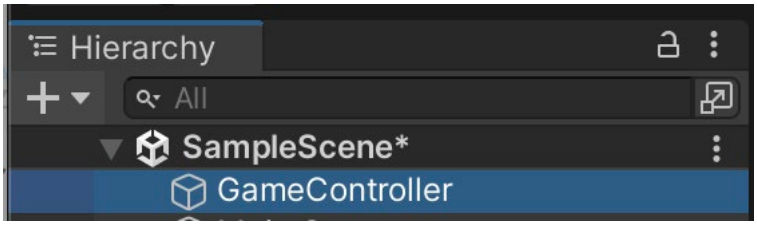
```
    // Update is called once per frame
    void Update()
    {
        if (Input.GetMouseButtonDown (0))
        {
            rb.velocity = Vector2.up * velocity;
        }
    }

    //Function where the player collides with an object
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if(collision.gameObject.tag == "HighSpike" || collision.gameObject.tag == "LowSpike" || collision.gameObject.tag == "Ground")
        {
            //Game is at a stopping state
            Time.timeScale = 0;
            }
    }
}
```

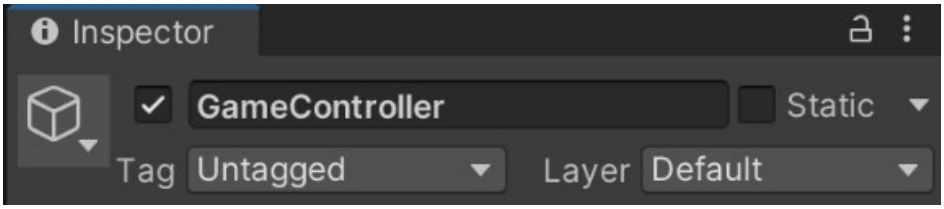
Save the script and return to Unity!

36 Now, try playing the game. You will notice that if the bird touches the ground or spikes, then the game will be stopped.

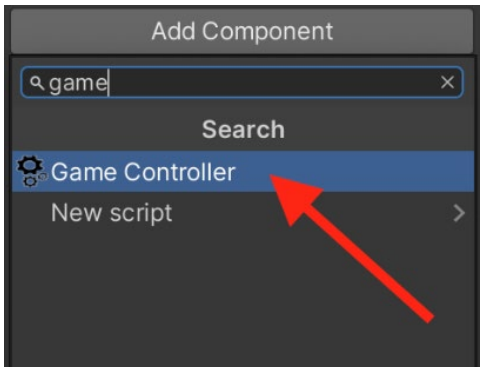
37 Stop the game.

38 Let's add a new game object. Go to **Hierarchy** and click the **+** button. Select "**Create Empty**". Then, rename the new **GameObject** to ***GameController***.





39 With the **GameController** object selected, click **Add Component**. Type **Game** in the search box and select the **Game Controller** script component.



Open the **Game Controller** script. Add the following to the script inside the class before the void Start function:

```
public class GameController : MonoBehaviour
{
    [Header("References")]
    //Game Over Canvas that is used for the game
    public GameObject gameOverCanvas;

    //Score Canvas that is used for the game
    public GameObject scoreCanvas;

    //Spawner object that is used for the game
    public GameObject spawner;

    // Start is called before the first frame update
    void Start()
    {

    }
}
```

Next, type inside the **void Start** function:

```
void Start()
{
//Speed for the game is at a playing state
Time.timeScale = 1;
//Score is visible
scoreCanvas.SetActive(true);
//Game Over UI is invisible
gameOverCanvas.SetActive(false);
//The spawner is shown in the game
spawner.SetActive(true);
}
```

Create a new void function called **void GameOver** inside the class and after the void Start function. Afterwards add this:

```
void Start()
{
        //Speed for the game is at a playing state
        Time.timeScale = 1;
        //Score in visible
        scoreCanvas.SetActive(true);
        //Game Over UI is invisible
        gameOverCanvas.SetActive(false);
        //The spawner is shown in the game
        spawner.SetActive(true);
}
public void GameOver()
{
    //Game Over UI is visible
    gameOverCanvas.SetActive(true);
    //The spawner is now invisible in the game
    spawner.SetActive(false);
    //The speed for the game is now at a stopping state
    Time.timeScale = 0;
}
```

The **void Update** function can be deleted as it is not needed in this context. Once you do that save your script and return to Unity!

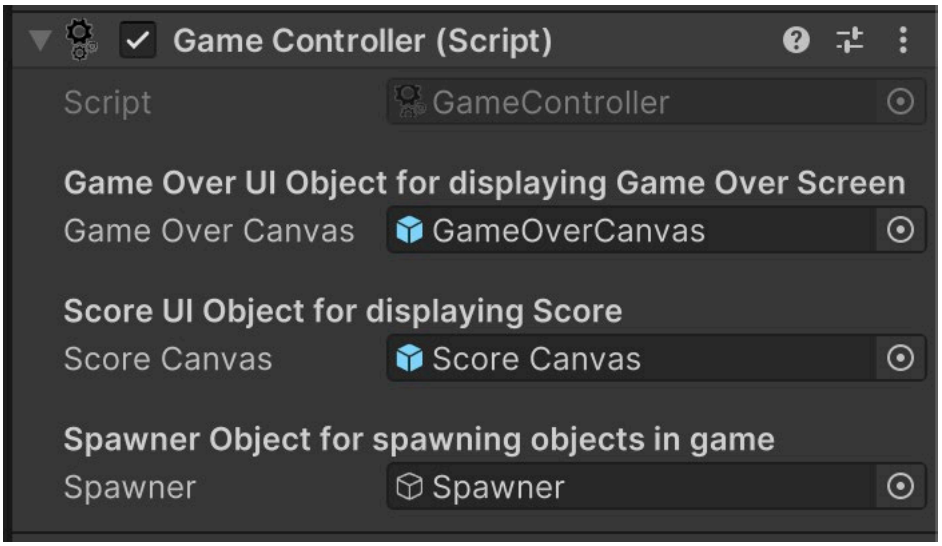
40 Before moving on, let's update the code in the player controls for the **Game Over Canvas** to appear. Select the Bird game object in the hierarchy panel then open the Bird's **Player Controls** Script.

Inside the **void OnCollisionEnter2D** method, replace the code inside the if statement with the code below:

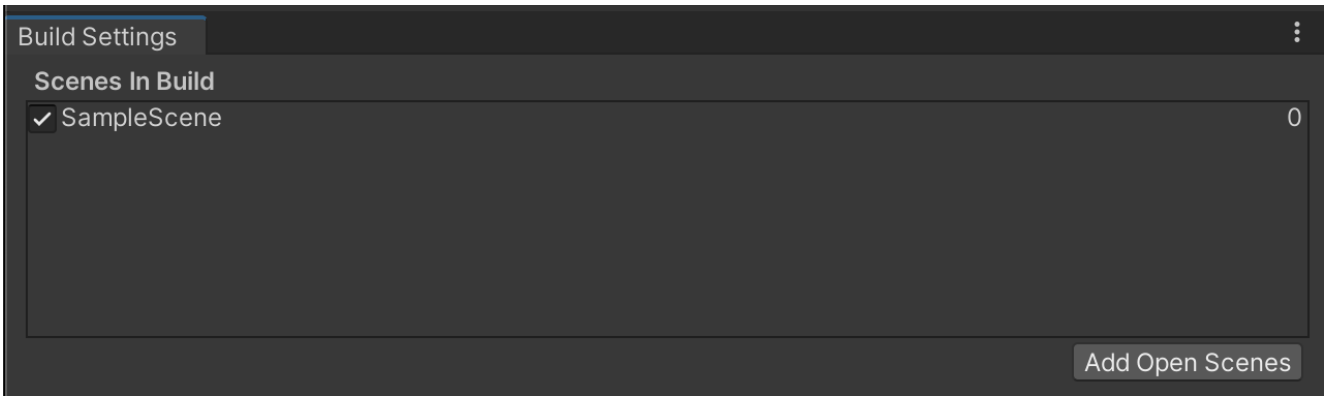
```
//Game Over function is called from the game manager
GameObject.Find("GameController").GetComponent<GameController>().GameOver();
```

Save the script and return to Unity!

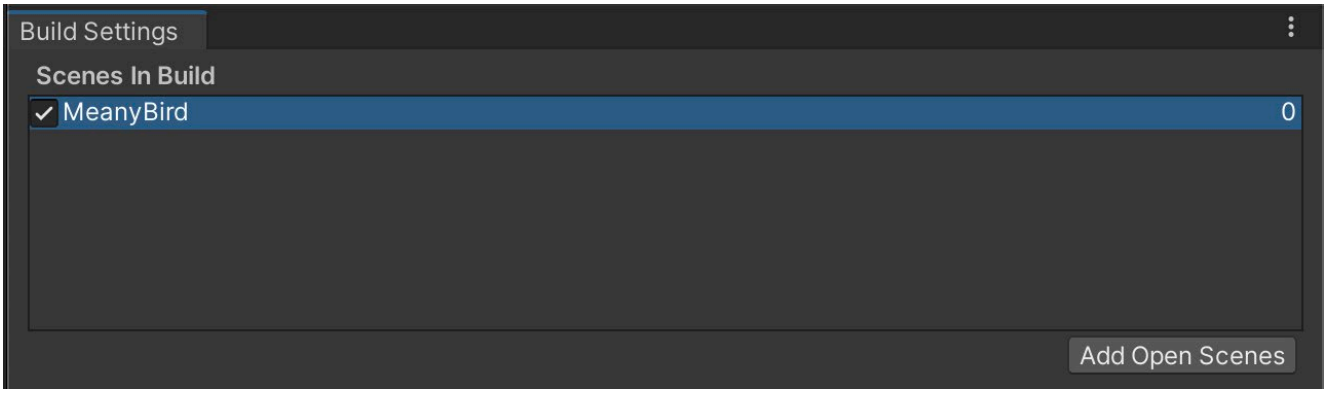
41 For the Game Controller script component, let's select our objects. Make sure the checkbox next to the Game Controller (Script) is activated, then within the Scene menu, select the **GameOverCanvas**, **Score Canvas**, and the **Spawner**.



42 To get the game over 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.



43 Once deleted, you want to click the **Add Open Scenes** button. This will allow the game to restart.



44 Play the game! Make sure that it looks like everything's working.

45 Stop the game. Save your game and export it. Submit your game.