CODE NINJAS®

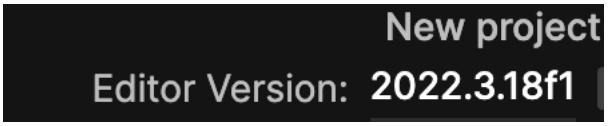
Bronze Belt Ninja Guide

Activity 05: Don't Touch the Cubes

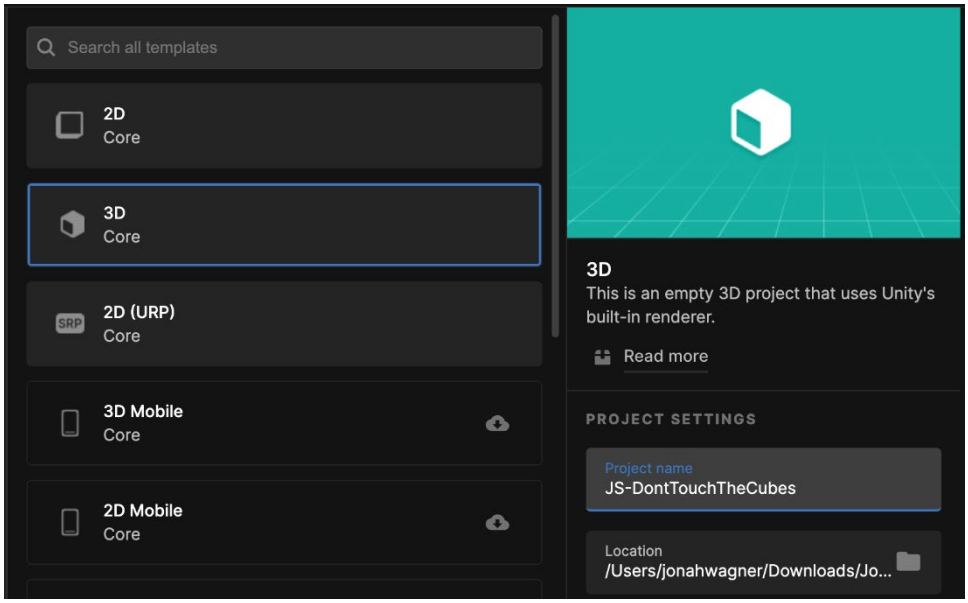
Activity 5: Don't Touch the Cubes

In this activity, you will design 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.

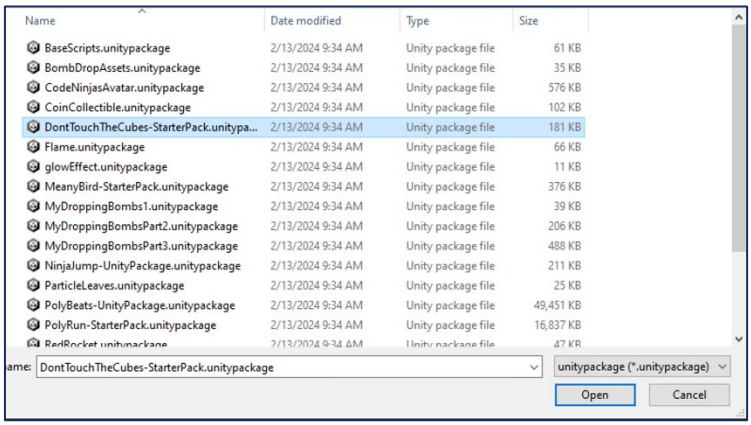


2 Select **3D** in the Templates column. Next, type in your first and last initials plus the name of the project. For example, John Smith would save their project as JS-DontTouchTheCubes. Select the folder location where you normally save your Unity games. Then select the **Create** button!

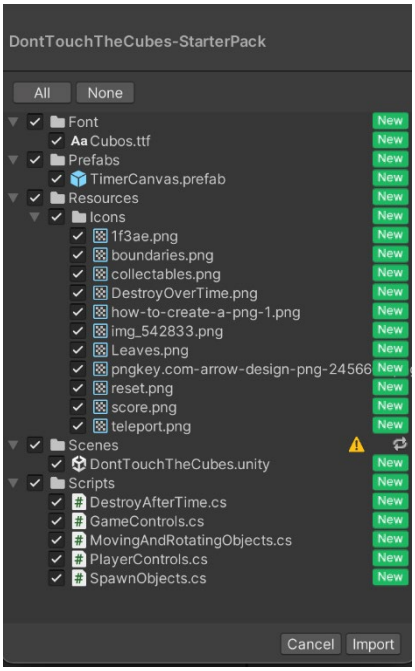


3 Once Unity loads all the necessary assets into the program you will see that a blank project exists. We're about to fix that.

4 Go to the **Assets** tab at the top, select **Import Package**, and then click on **Custom Package**. By selecting **Custom Package,** we can import compressed packages into Unity. Find your Unity assets folder. Then select the **Activity 05 - DontTouchTheCubes-StarterPack.unitypackage** file. Once you select the file Unity will then show a menu of what is inside the package.

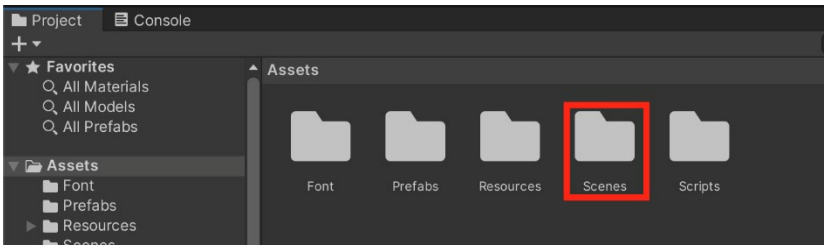


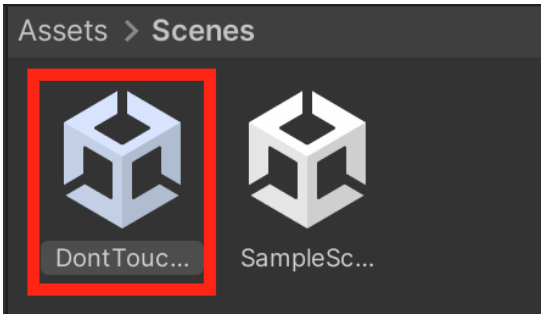
5 Since this file is built just for this game, we want to make sure all the material inside the package is selected. If all the material is not selected, click the **All** button to select everything. Click **Import** and all the material will be imported into the project.



6

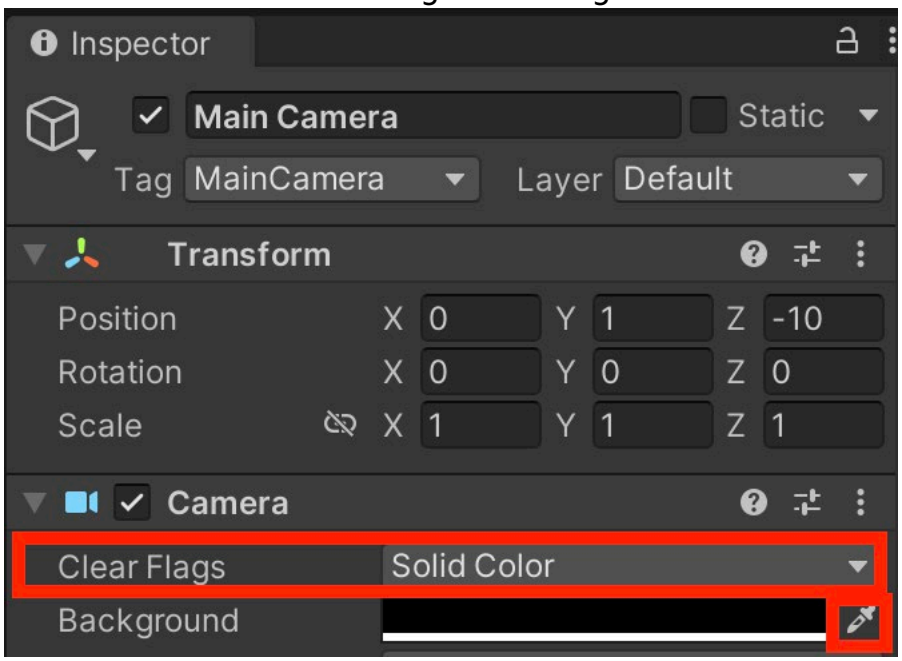
Now click **Scenes** under the **Assets** window and open the *DontTouchTheCubes* scene. Even though you've imported the scene, there is nothing in the Scene window or the Game window. However, we do have the materials that we need in our folder to start creating the game.

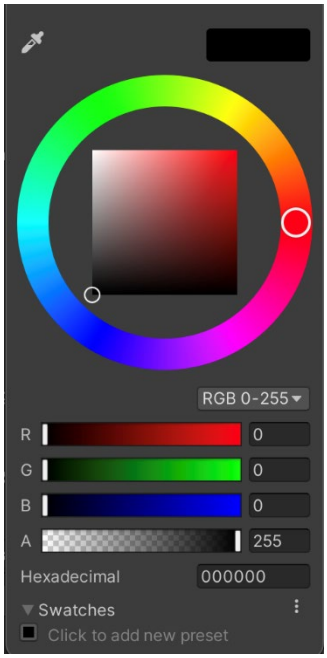




7

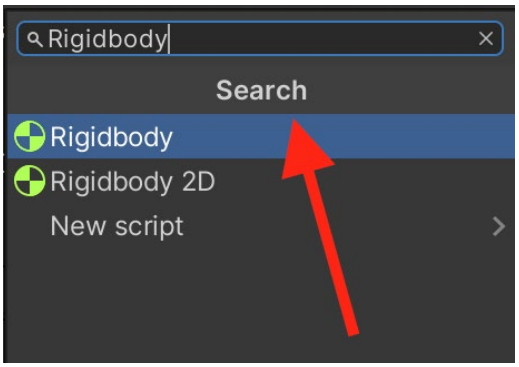
Select the **Main Camera** game object. Then, find the **Clear Flags** dropdown menu, and choose **Solid Color**. Change the background color to black.



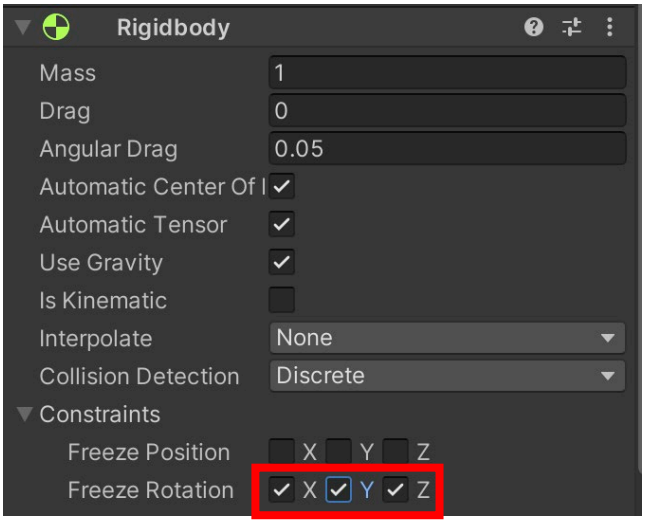


8 With the **Main Camera** game object still selected, change the rotation to **90** on the X axis.

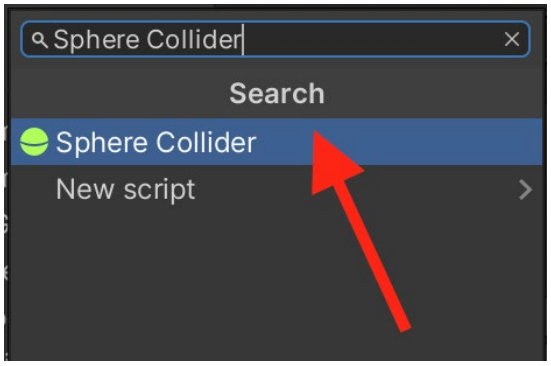
9 Select the **Main Camera** game object again and click **Add Component**. In the Search Box, type ***Rigidbody*** and then select **Rigidbody**.



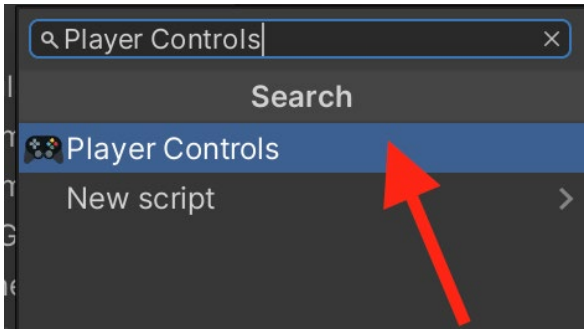
In the **Rigidbody** component, click the arrow for the **Constraints** drop-down section. Click on all boxes next to **Freeze Rotation** for all 3 axes. Then set the **Drag** to **1**.



10 Make sure you continue to have the **Main Camera** game object selected. Then select **Add Component**. In the search box, type ***Sphere Collider***, then select **Sphere Collider**.



11 Make sure the **Main Camera** is still selected, then select **Add Component**. In the search box, type ***Player Controls*** and select the **Player Controls** script.



12 Open the **Player Controls** script, added to **Main Camera**, then add the following code before the **void Start** function inside the class:

```
public class PlayerControls : MonoBehaviour
{

    [Header("Rigidbody")]
    //Rigidbody component
    public Rigidbody rb;
    // Start is called before the first frame update
    void Start()
```

13

Next, within the **void Start** function, insert the code below:

```
void Start()
{
    //variable rb equals to
    //component rigidbody
    rb = GetComponent<Rigidbody>();
    //Game is at normal pace
    Time.timeScale = 1f;
}
```

14

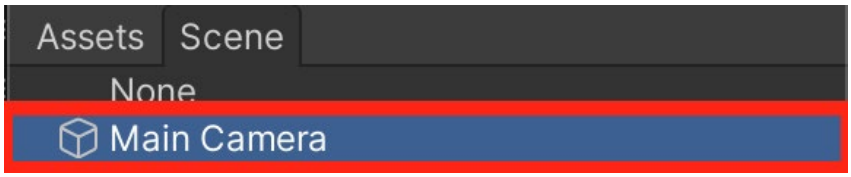
Within the **void Update** function, insert the code below:

```
void Update()
{
    // Rotating the game object on the z-axis
    // multiplied by the game frame rate by 7
    transform.rotation *= Quaternion.Euler(0, 0, 7 * Time.deltaTime);

    // Time scale of the game plus
    // physics and other fixed frame rate updates
    Time.timeScale += Time.fixedDeltaTime * 0.01f;

    // Movement and rotation of the camera
    // on the y-axis horizontally
    rb.velocity += transform.rotation * (Vector3.right * Input.GetAxisRaw("Horizontal") * 10f * Time.deltaTime);

    // Movement and rotation of the camera
    // on the y-axis vertically
    rb.velocity += transform.rotation * (Vector3.up * Input.GetAxisRaw("Vertical") * 10f * Time.deltaTime);

    // Keeping the camera position in bounds
    // Refactor the clamping into separate variables for clarity
    float clampedX = Mathf.Clamp(transform.position.x, -30f, 30f);
    float clampedZ = Mathf.Clamp(transform.position.z, -30f, 30f);

    // Apply the clamped values to the transform's position
    transform.position = new Vector3(clampedX, 0, clampedZ);
}
```

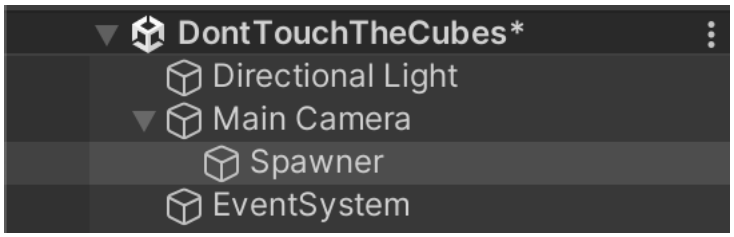
You can now save your script and return to Unity!

Clamping is a function in Unity that restricts a value within a specified range. By clamping an object's position, it helps make sure that it cannot move outside the set minimum and maximum limits.

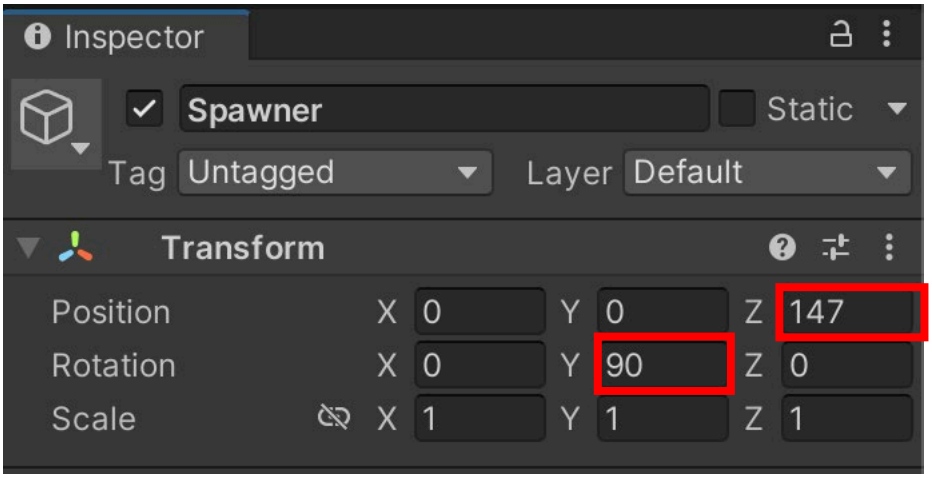
15 With the **Main Camera** selected, locate the **Player Controls** script component inside the Inspector. Find the **Rigidbody** input field, then select the properties circle button. Select the **Main Camera** in the Scene panel.



16 Now you will create a *Spawner* object, which is an object that is made for creating multiple objects from its location. First, we need to create one game object. Right click on **Main Camera** then select **Create Empty**. Rename the object to ***Spawner*** in the inspector.

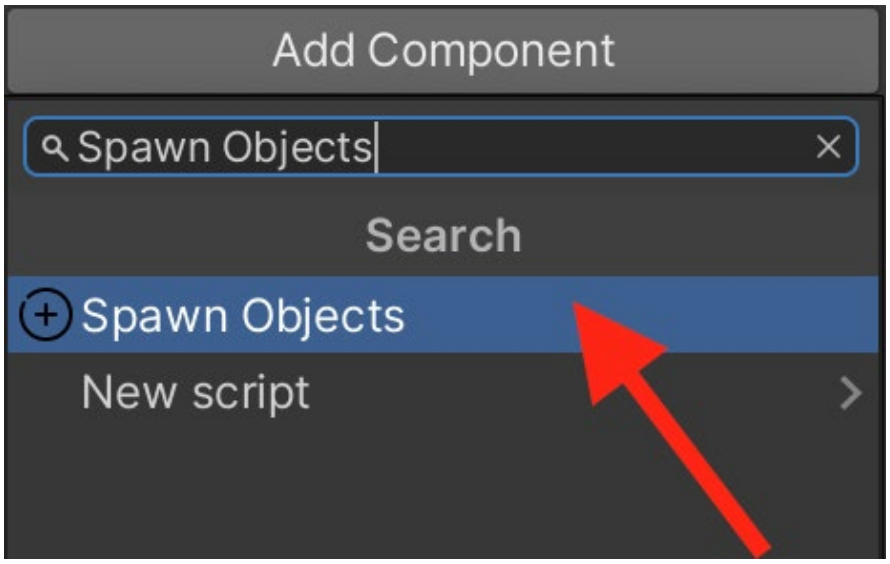


17 Change the position and rotation of the Spawner to match the values shown below.



Position values: X: 0, Y: 0, Z: 147
Rotation values: X: 0, Y: 90, Z: 0

18 With the **Spawner** still selected, click **Add Component** in the Inspector. In the search box, type ***Spawn Objects***. Then, select the **Spawn Objects** script.



Open the **Spawn objects** script and add the following code inside the class before the **void Start** function:

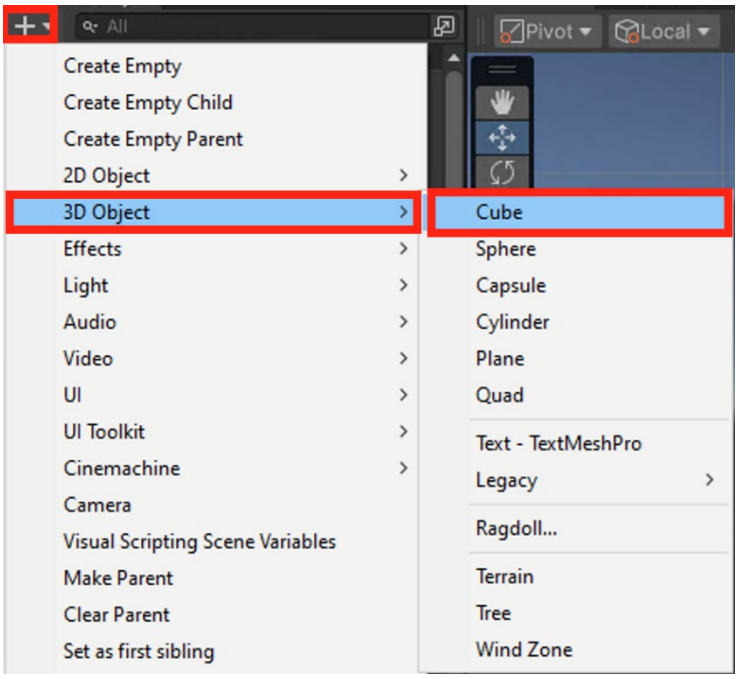
```
public class SpawnObjects : MonoBehaviour
{
    //Cube that is going to be spawned
    [Header ("Spawn Cube Object")]
    public GameObject spawnCube;
    //Difficulty of the game
    [Header ("Default Difficulty")]
    public float difficulty = 40f;
    //Time for the next cube to be spawned
    float spawn;

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

    }
```

19 Delete the **void Start** function; we don't need it here. Inside the **void Update** function, add the following code:

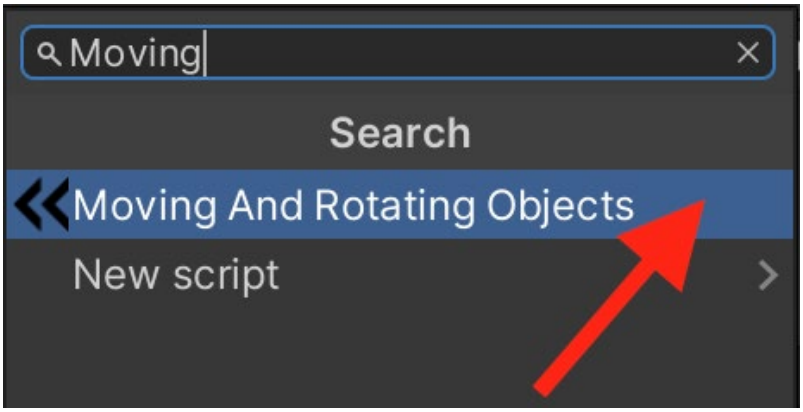
```
// Update is called once per frame
void Update()
{
    spawn += difficulty * Time.deltaTime;
    while(spawn > 0)
    {
        spawn -= 1;
        Vector3 v3Pos = transform.position + new Vector3(Random.value * 40f - 20f, 0, Random.value * 40f - 20f);
        Quaternion qRotation = Quaternion.Euler(0, Random.value * 360f, Random.value * 30f);
        GameObject createObject = Instantiate(spawnCube, v3Pos, qRotation);
    }
}
```

You can now save the script and return to Unity!

20 We now need to create cubes for the game! To do this, select the **+** button in the top left of the hierarchy. Select **3D Object** from the drop-down menu, then click **Cube**.



21 Select the **Cube** in the hierarchy, then in the Inspector, click **Add Component**. In the search box, type ***Moving*** and then select the **Moving and Rotating Objects** script.



22

Open the **Moving and Rotating Objects** script and add the following code inside the class and before the **void Start** function:

```
//Default moving speed
[Header ("Default Movement Speed")]
public float moveSpeed = 10f;
//Default Rotating Speed
[Header ("Default Rotation Speed")]
public float rotateSpeed = 50f;
```

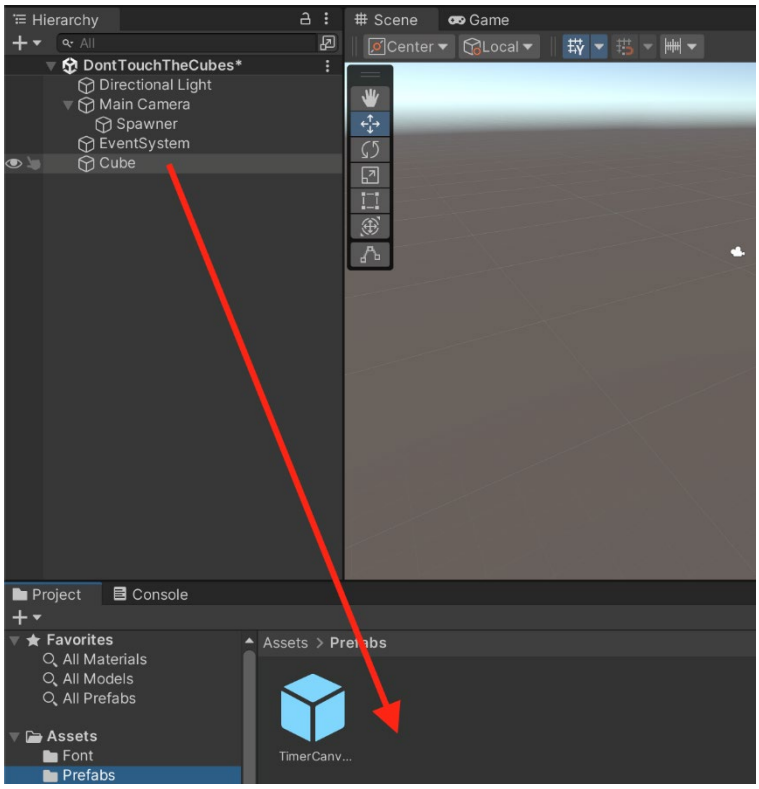
23

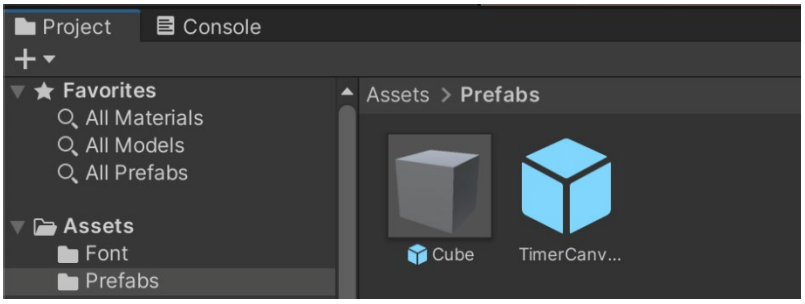
Delete the **void Start** function; we don't need it here. Inside the **void Update** function, add this code:

```
// Update is called once per frame
void Update()
{
    //object moving on the y axis based on move speed
    transform.Translate(0, moveSpeed * Time.deltaTime, 0);
    //rotate on the x axis based on rotate speed
    transform.Rotate(Vector3.up * rotateSpeed * Time.deltaTime);
}
```

You can now save the script and return to Unity!

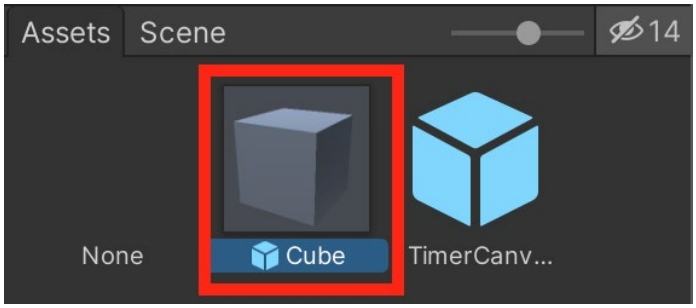
24 In the Hierarchy keep the **Cube** selected, then select the **Prefabs** folder in the Project menu and drag the **Cube** into the folder.





You can delete the **Cube** from the Hierarchy as you won't need it to show up in the scene anymore!

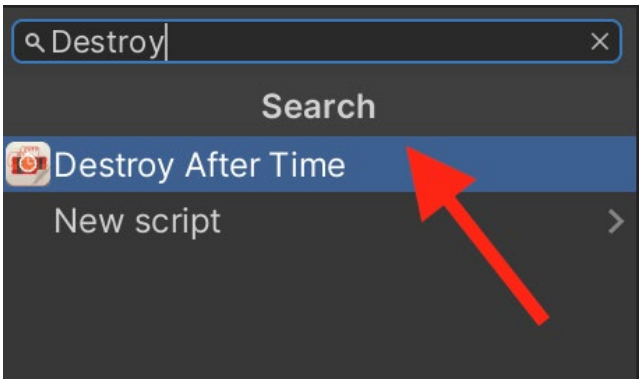
25 Select the Spawner. In the Inspector, find the **Spawn Objects** script component. Add cubes to the **Spawn Cube** input field by clicking on the Properties button and selecting the **Cube** prefab in the **Assets** window.



26 Play the game! You should see that the cubes will begin to spawn and move toward the screen.

27 Stop the game.

28 We are moving downwards but the cubes are still active off the screen. To fix this we need to destroy these cubes by using a **Destroyer** script. Select the **Cube** in the **Prefab** folder then click **Add Component**. Type ***Destroy*** in the search box and select the **Destroy After Time** script.



29 Open the **Destroy After Time** script and add the following code inside the class and before the **void Start** function:

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

    }
```

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

```
// Start is called before the first frame update
void Start()
{
//Execute function based on timeToDestruction
Invoke("DestroyObject", timeToDestruction);
}
```

Create a new **void DestroyObject** function by adding this code:

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

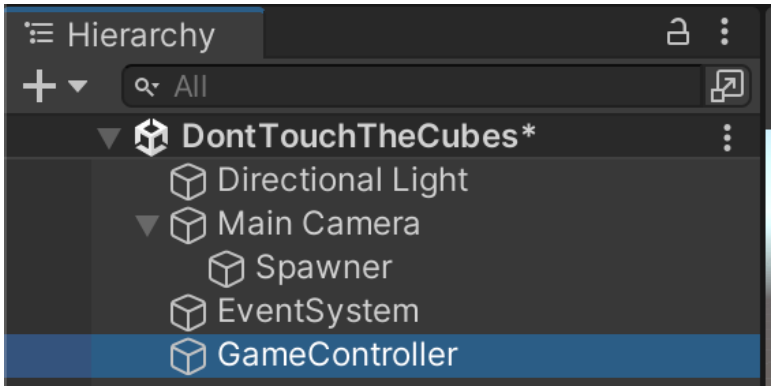
You can delete the **void Update** function as we don't need it here. Now save the script and return to Unity!

31 While you still have the Cube selected, in the inspector find the **Destroy After Time** script component. Change the **Time to Destruction** to **12** seconds.

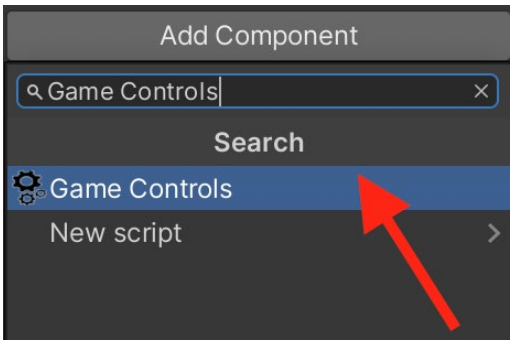
32 Play the game but this time, don't maximize it. You should see the cubes being destroyed by looking at the hierarchy.

33 Stop the game.

34 In the hierarchy, add a new empty **GameObject** to the game. Rename it **GameController**.



35 With the **GameController** GameObject selected, click **Add Component**. In the search box, type **Game Controls**, then add the **Game Controls** script.



36 Open the **Game Controls** script and add the following code inside the class and before the **void Start** function:

```
//Timer text object
private Text timerText;
//Timer counter for adding score
private int timerCount;
```

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

```
void Start()
{
    //Game is at a playing state
    Time.timeScale = 1f;
}
```

Before we save the script, you can delete the **void Update** function in this script as we do not need it here. You can now save the script and return to Unity!

38 We are almost ready to test out the game! First, we need to add a new function to our player controls in order to have the game reset if we touch a collider. Open the **Player Controls** script in the **Main Camera** object.

39 Create a new **void OnCollisionEnter** function, then add the following:

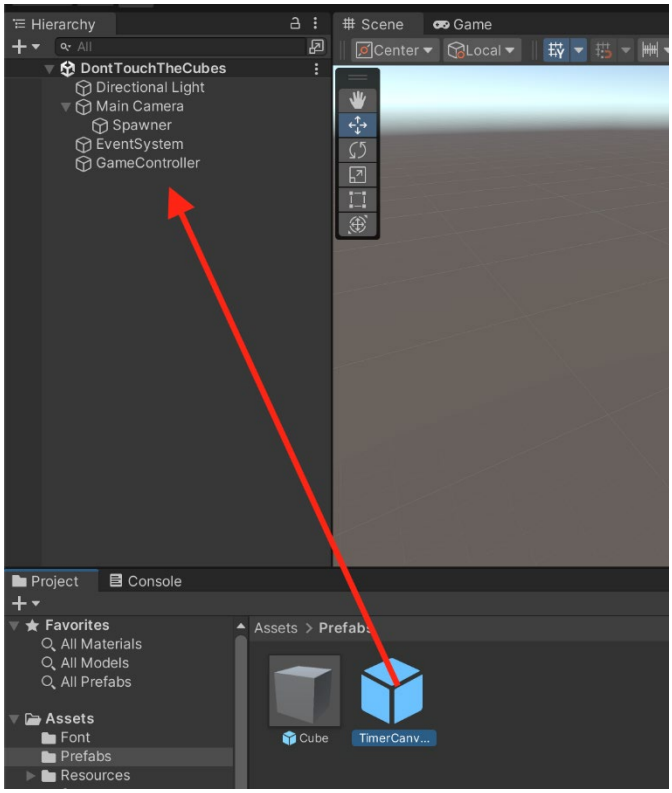
```
void OnCollisionEnter(Collision collision)
{
    SceneManager.LoadScene(0);
}
```

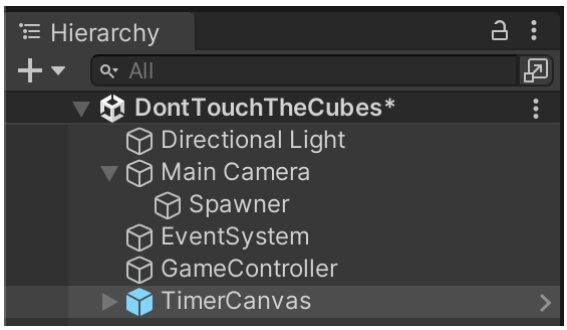
You can now save the script and return to Unity!

40 Play the game! Notice that if you touch the cubes, the game will reset.

41 Stop the game.

42 Let's make the game even more fun by adding scoring to the game! Go to the **Prefabs** folder and drag **TimerCanvas** to the **Hierarchy**.





43 In the hierarchy, select the **Game Controller**. In the inspector, open the **Game Controls** script to add scoring.

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

```
void Start()
    {
        //Game is at a playing state
        Time.timeScale = 1f;
        //Executing a coroutine
        StartCoroutine(CountTime());
        //Timer text equals finding
        //the score object and using
        //the text component
        timerText = GameObject.Find("Score").GetComponent<Text>();
    }
```

Under the **void Start** function, create a new **IEnumerator CountTime** function inside the class and add this code:

```
    IEnumerator CountTime()
{
    //After 1 second
    //1 point is added to the score
    //and will repeat the function
    yield return new WaitForSeconds(1f);
    timerCount++;
    timerText.text = "Score: " + timerCount;
    StartCoroutine(CountTime());
}
```

You can now save the script and return to Unity!

45 Play your completed game! How high of a score are you able to get?

46 Stop the game. **Save**, **Export**, and **Submit** your game.