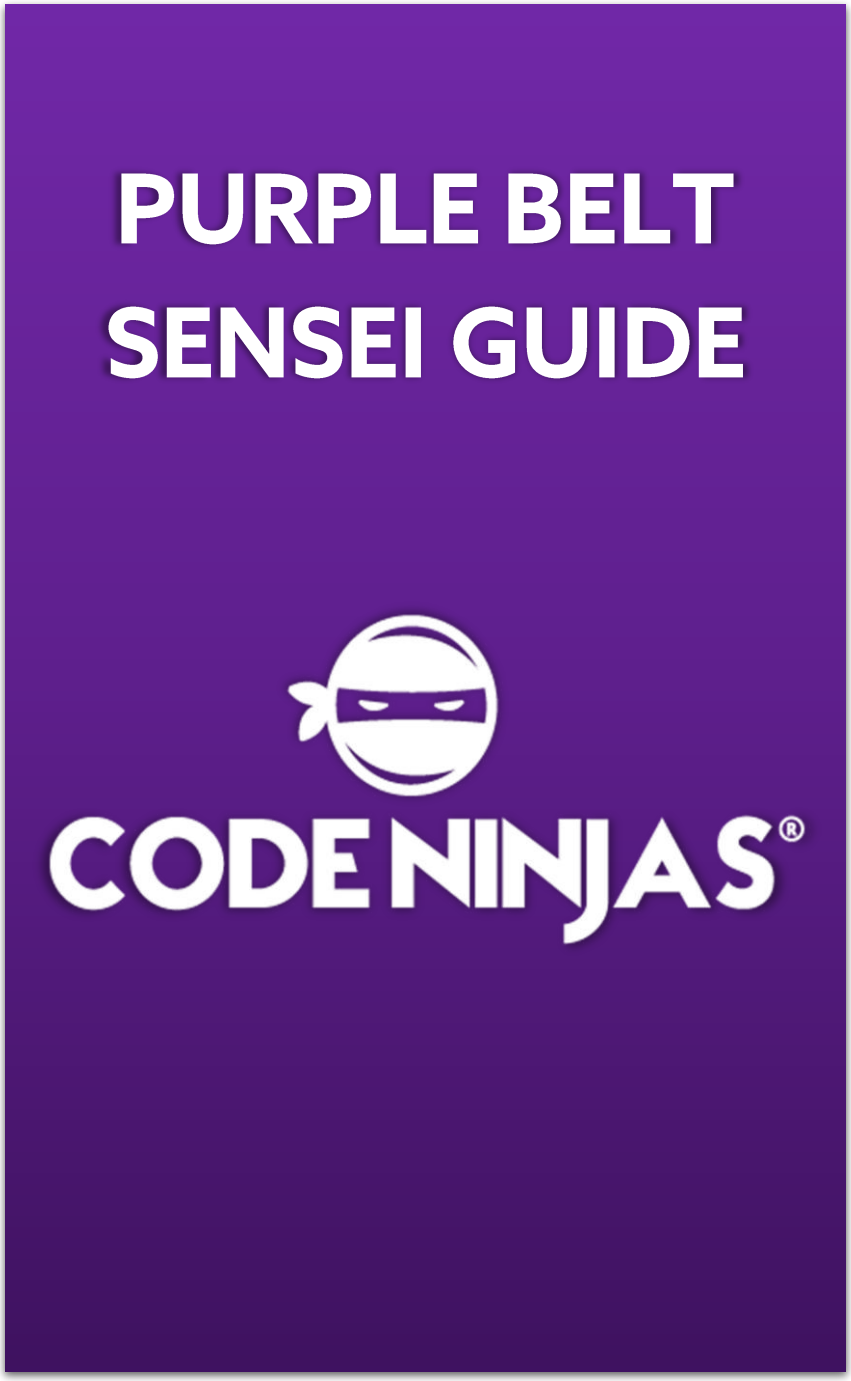
PURPLE BELT SENSEI GUIDE



Setup and Installation

There are three main components of a Unity installation. The Unity Hub manages projects and installations. The Unity game engine is used to create games. Visual Studio Community 2019 is used to code scripts.

Individual ninja accounts for Unity and Visual Studio are not required for ninjas. All project files are stored locally on the computer. You can use a flashdrive to store the projects and transfer from computer to computer. Unity Collab, Unity's built in version control system, is not a free service.

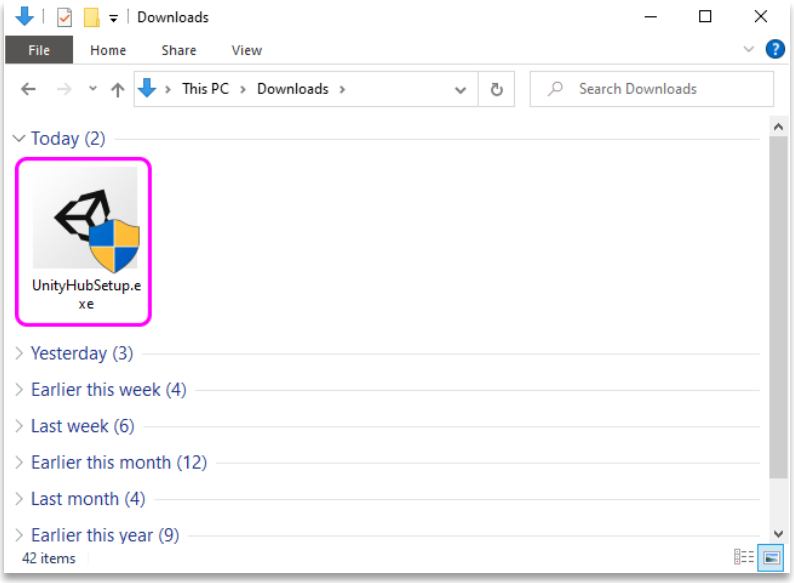
You can use GitHub and GitHub Desktop to cloud sync unlimited projects for free. Dropbox or Google Drive are easier to use, but they do not efficiently handle syncing Unity projects because each contains thousands of files.

Unity

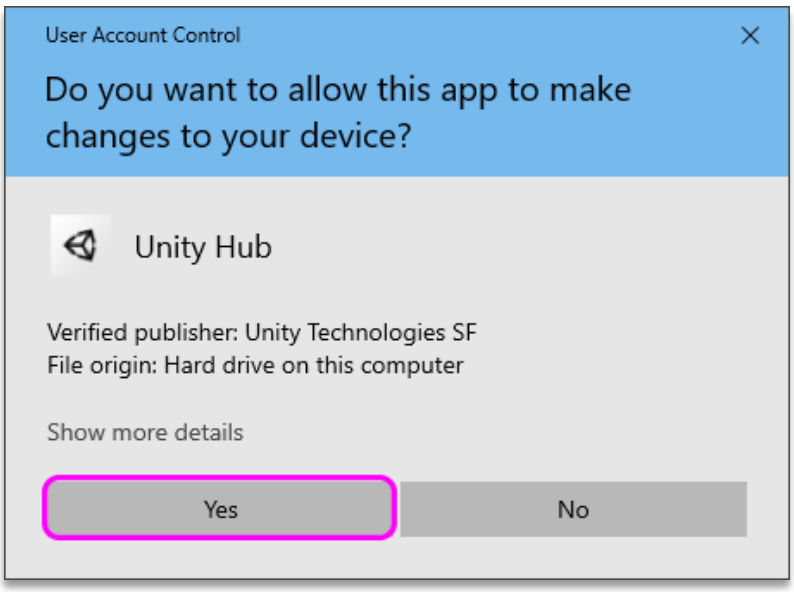
Downloading and Installing Unity Hub

Download **Unity Hub**. Do not download the beta version.
https://unity3d.com/get-unity/download

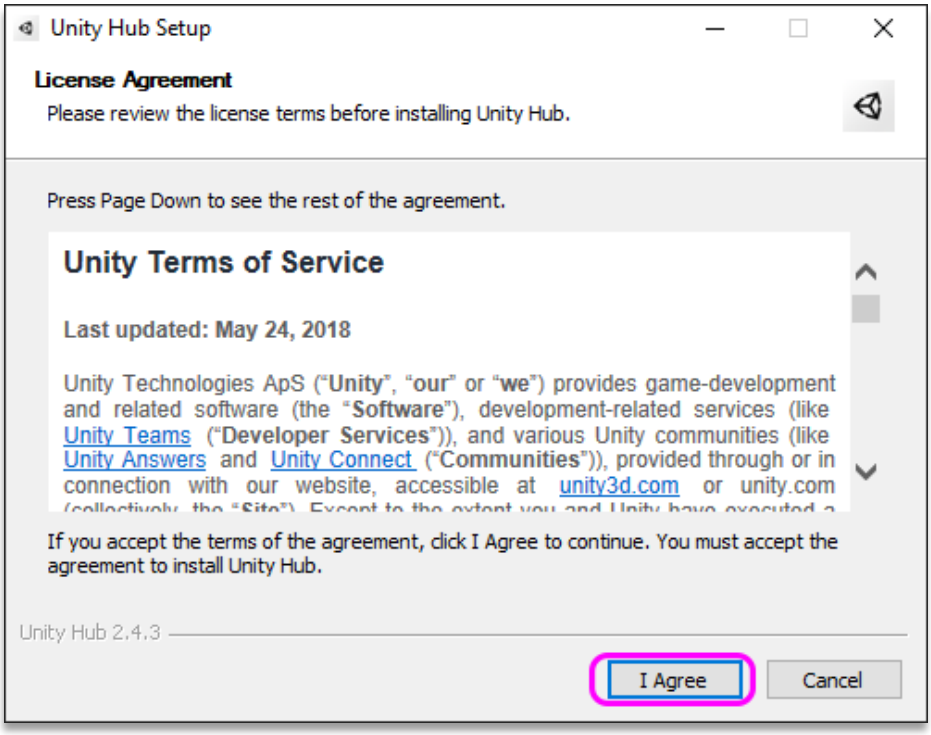
After the file downloads, find and run the executable.



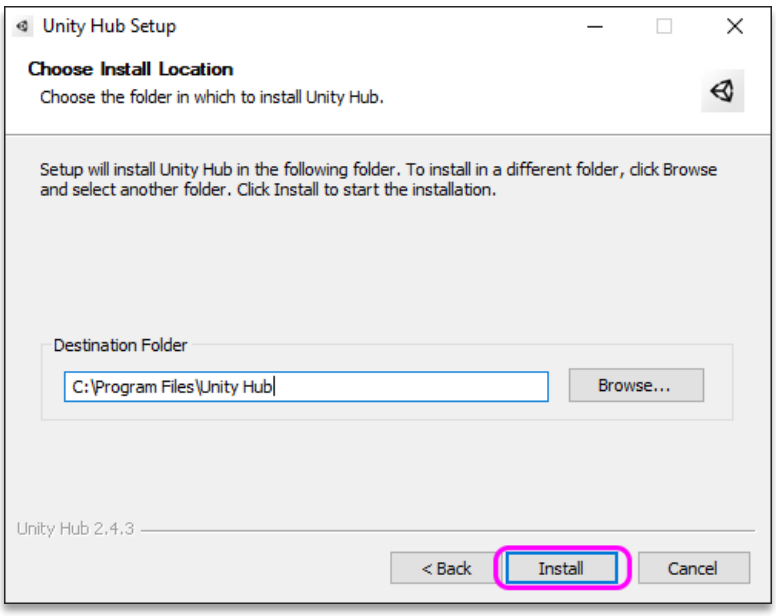
If the **Windows User Account Control** opens, click **Yes**.



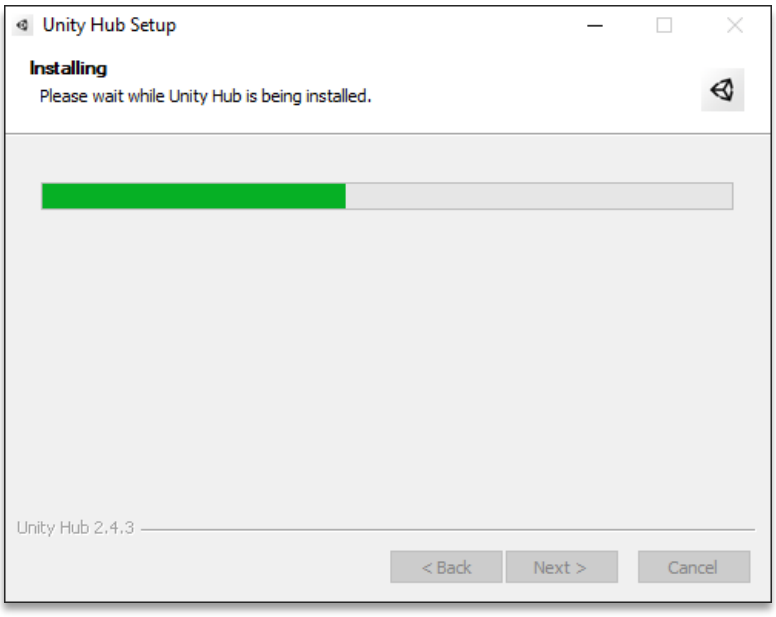
Review the License Agreement and click **I Agree**.



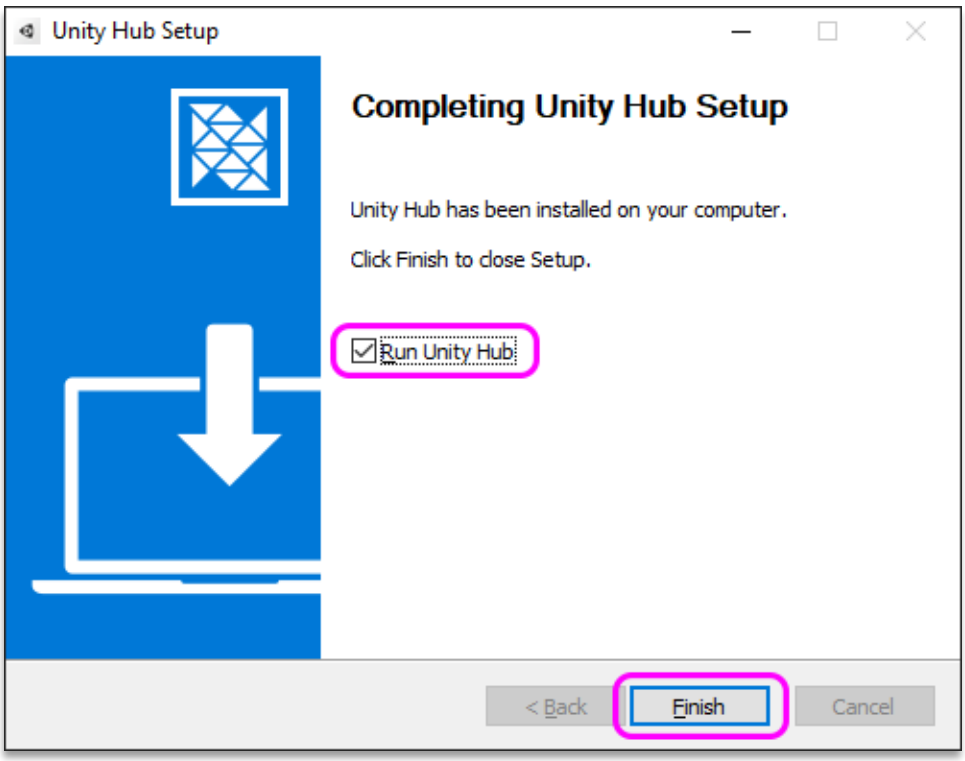
Keep the default destination folder and click **Install**.



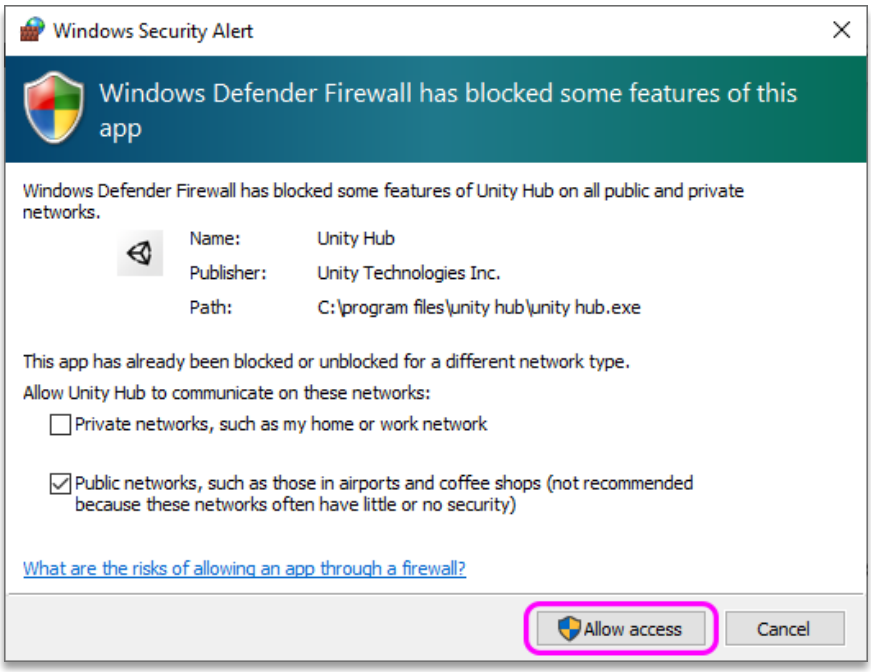
Wait for the installer to finish.



Check the box to run **Unity Hub** and click **Finish**.

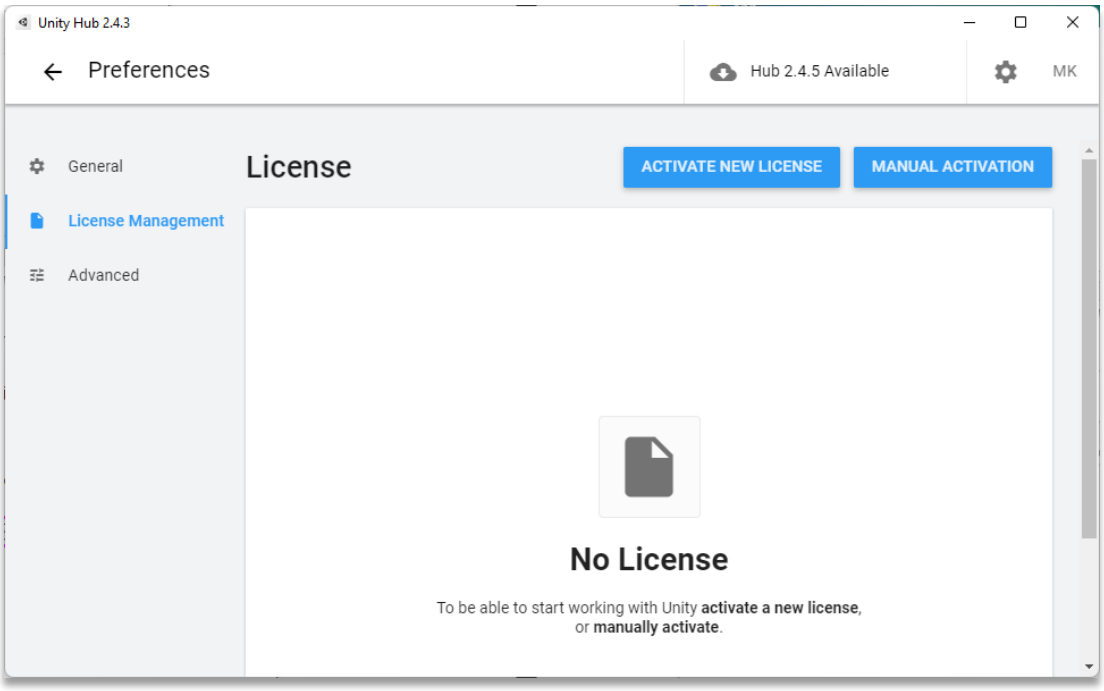


If **Windows Security Alert** opens, click **Allow access**.

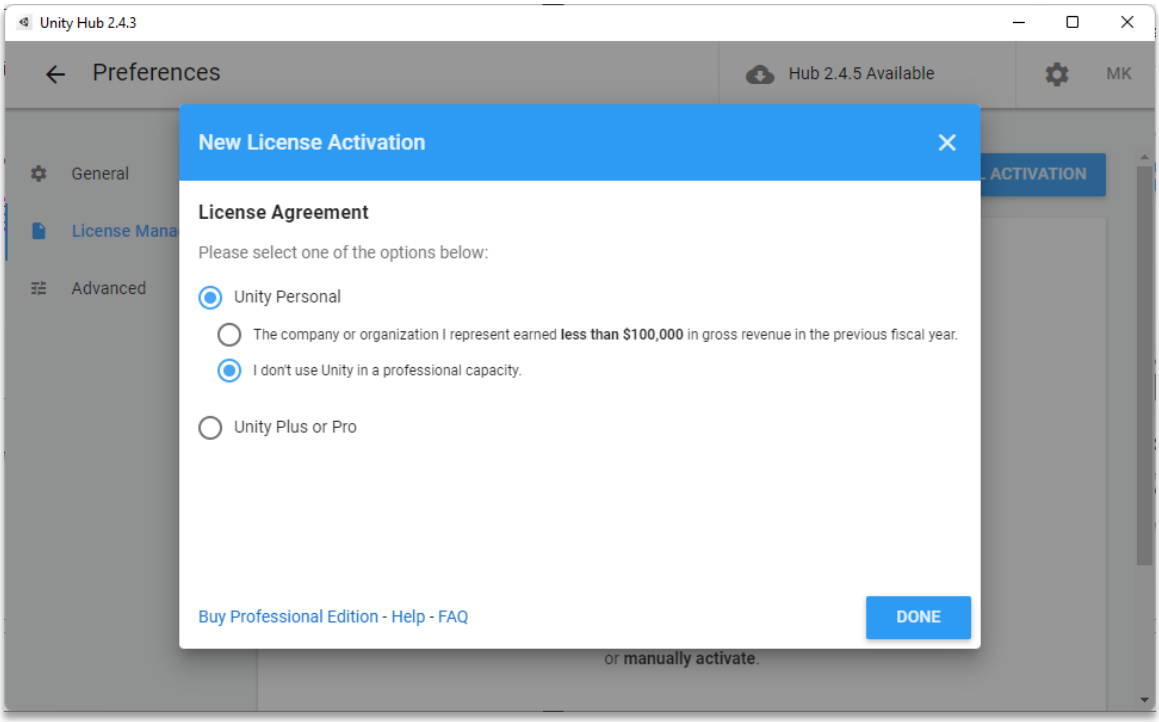


Activating Unity Licenses

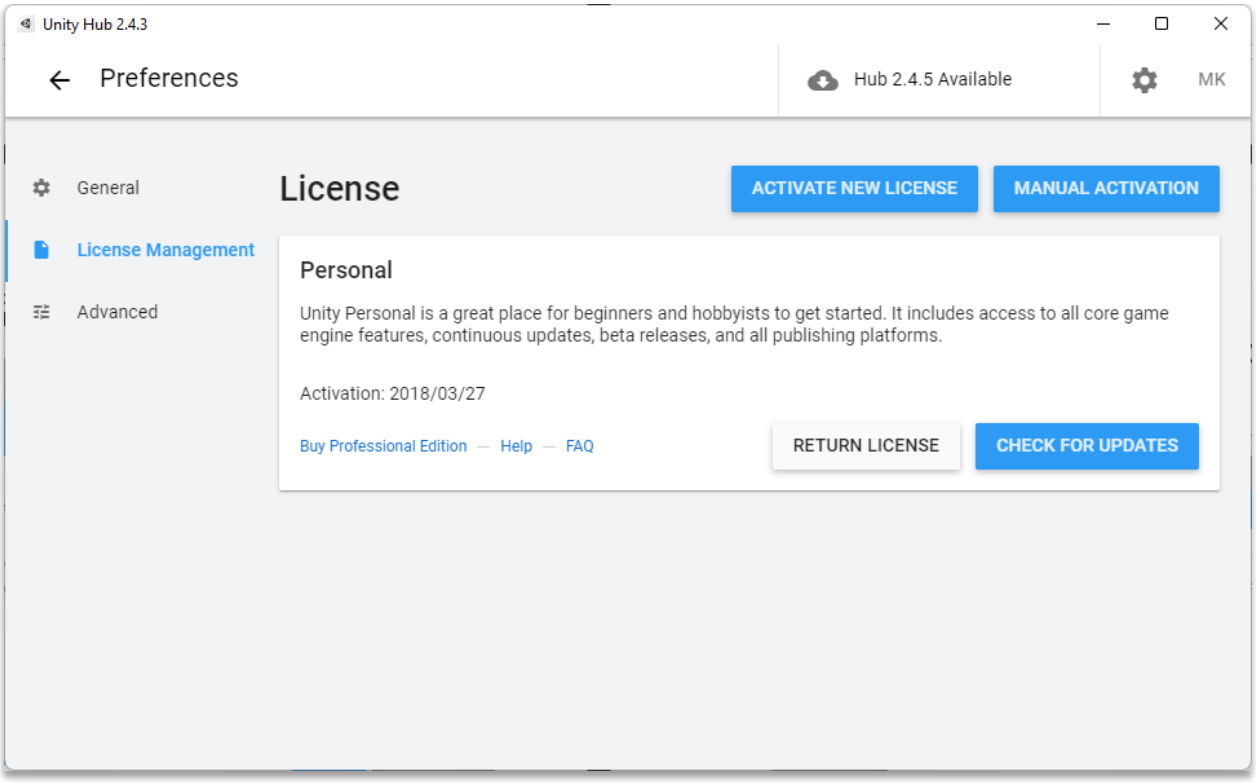
To activate a **free personal license** to use Unity, click the **settings cog**, then go to **License Management**.



Click on **Activate New License**. Select **Unity Personal** and **I don't use Unity in a professional capacity**. Click **Done**.

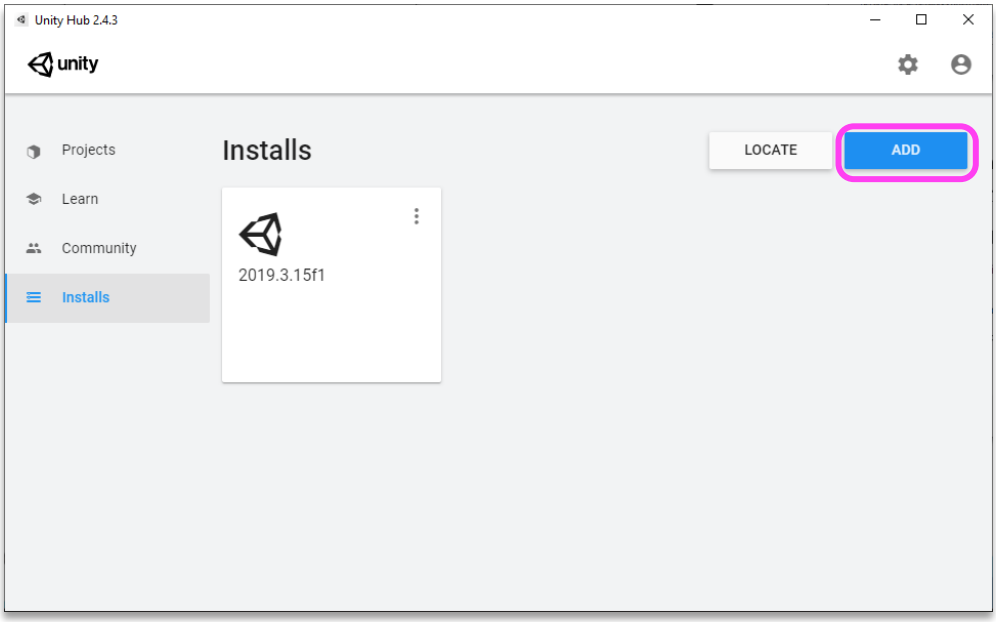


You should now see an active **Personal license**.



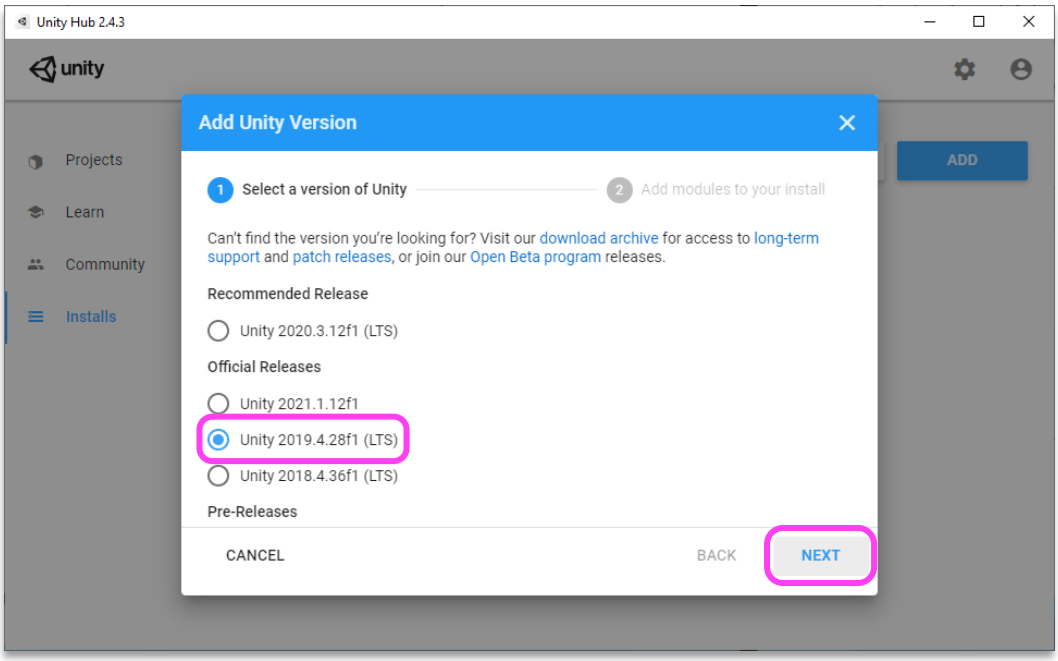
Installing Unity 2019.4 (LTS)

In **Unity Hub**, open the Installs tab and click **Add**.

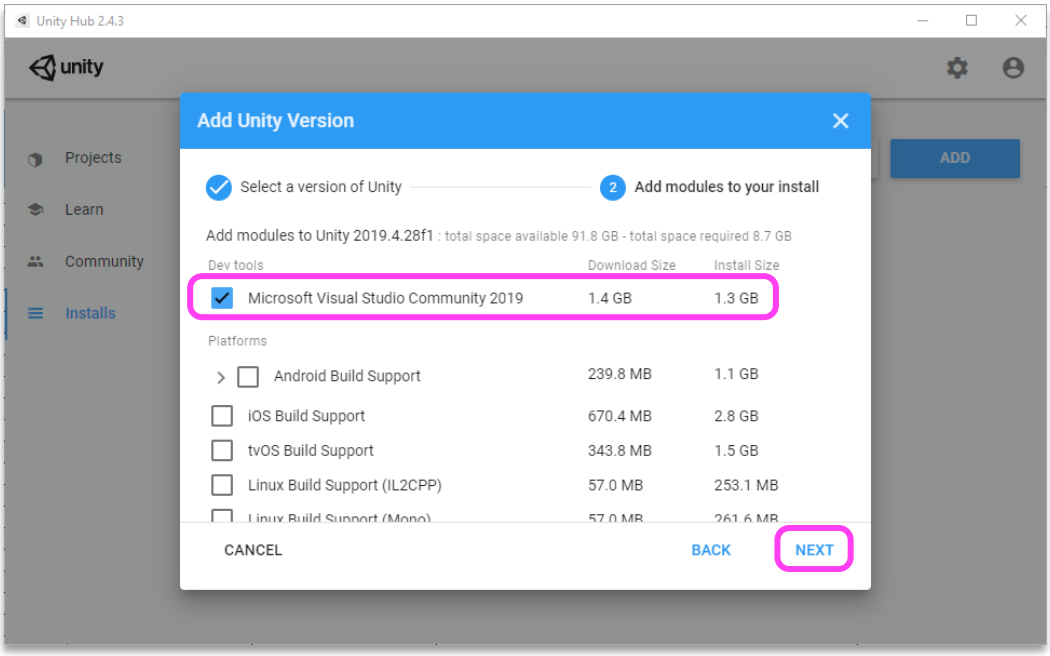


In the **Add Unity Version** window, select **Unity 2019.4 (LTS)**. Only the 2019 version has been tested with Code Ninjas curriculum. The numbers after **2019.4** may not match the picture exactly.

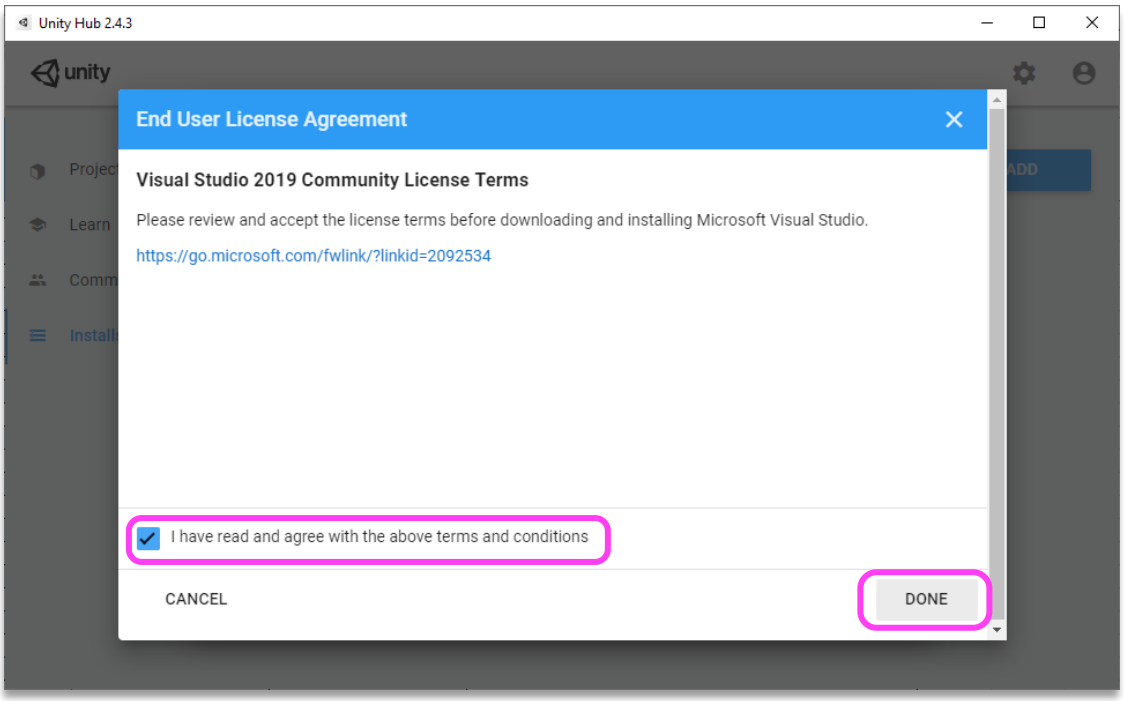
Click **Next**.



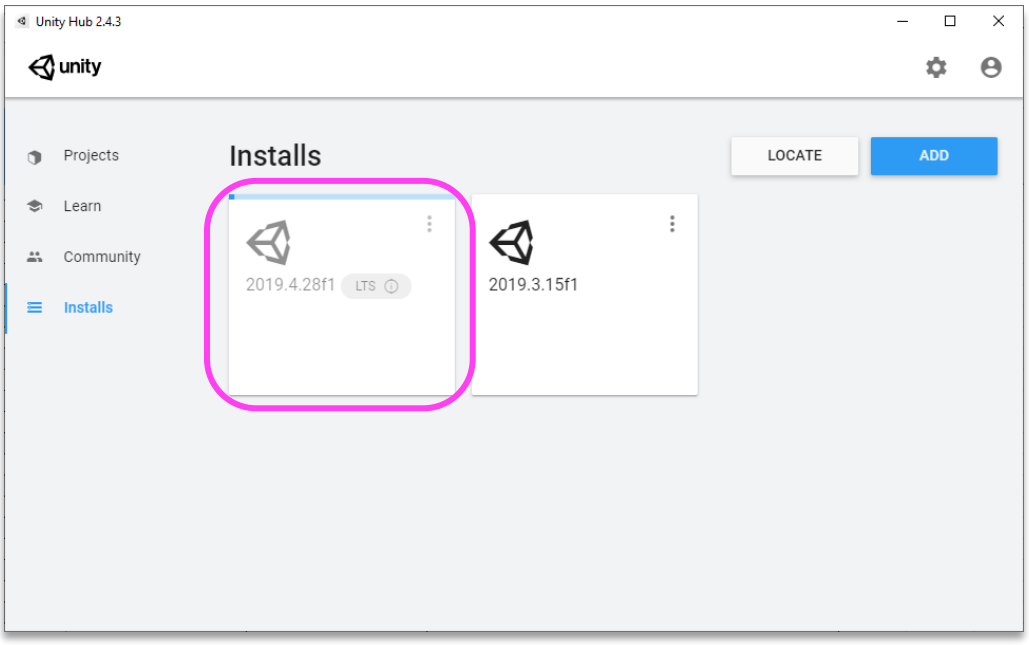
Select **Microsoft Visual Studio Community 2019** and click **Next**.



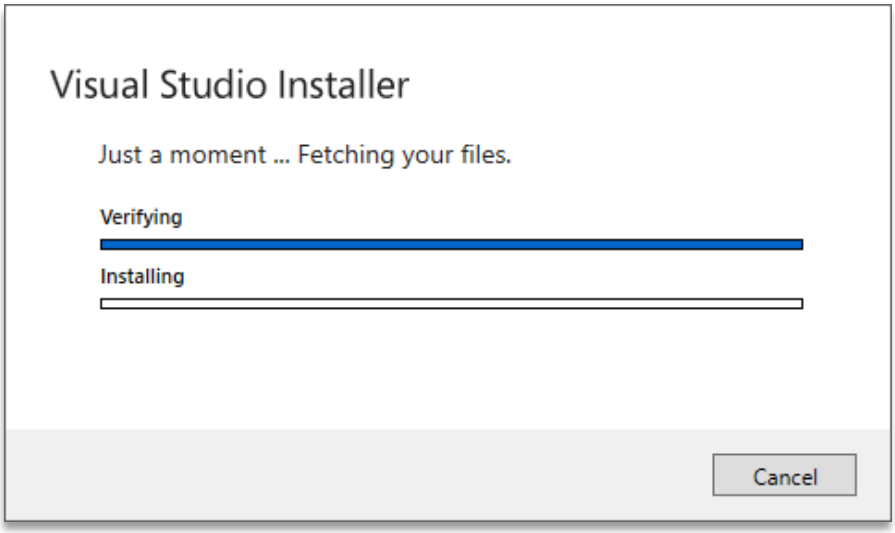
Accept the **Visual Studio** terms and conditions and click **Done**.



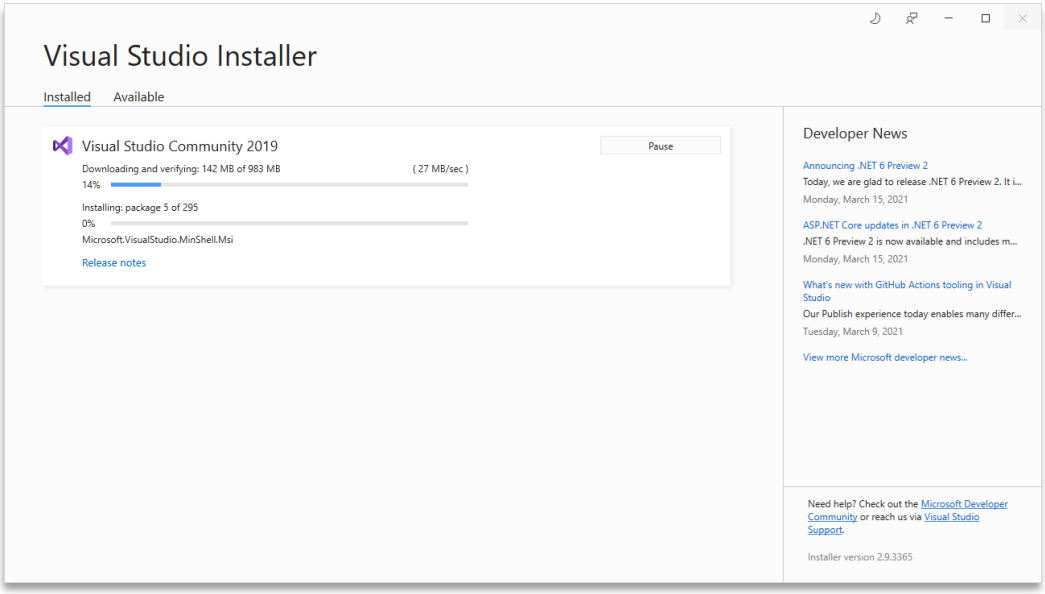
The installation will begin and might take 10 to 15 minutes depending on your Internet and computer speeds.



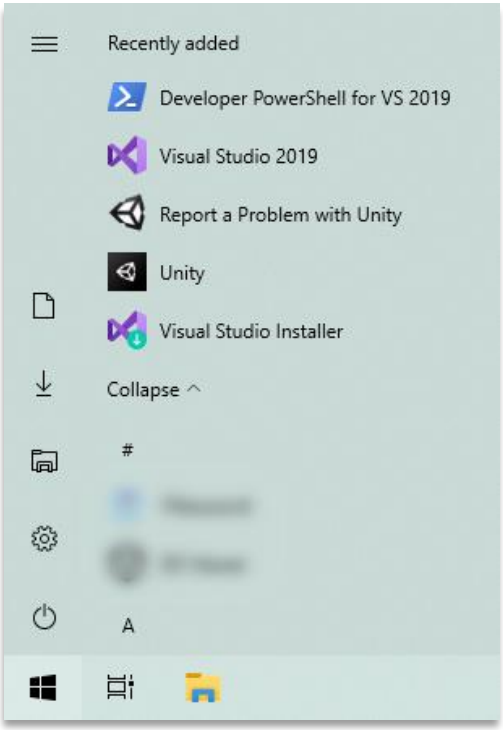
The **Visual Studio Installer** will begin if it is not already installed.



The **Visual Studio Installer** will download and install **Visual Studio Community 2019**.

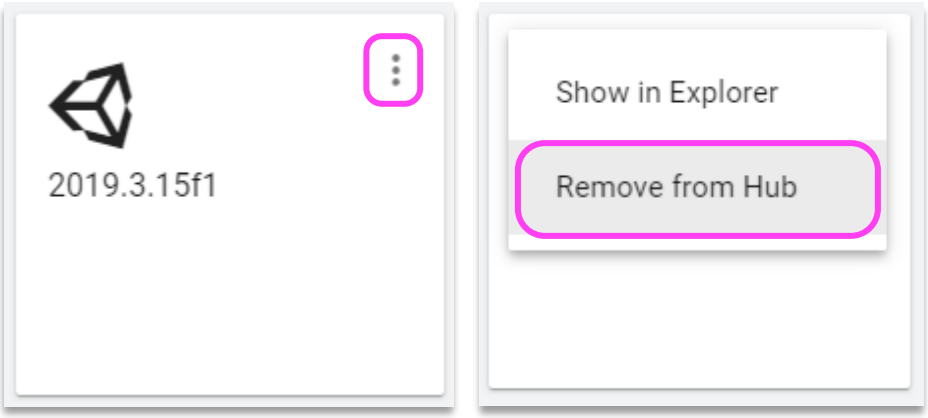


After your computer restarts, verify that the installations were successful by finding the programs in the **Start** menu.



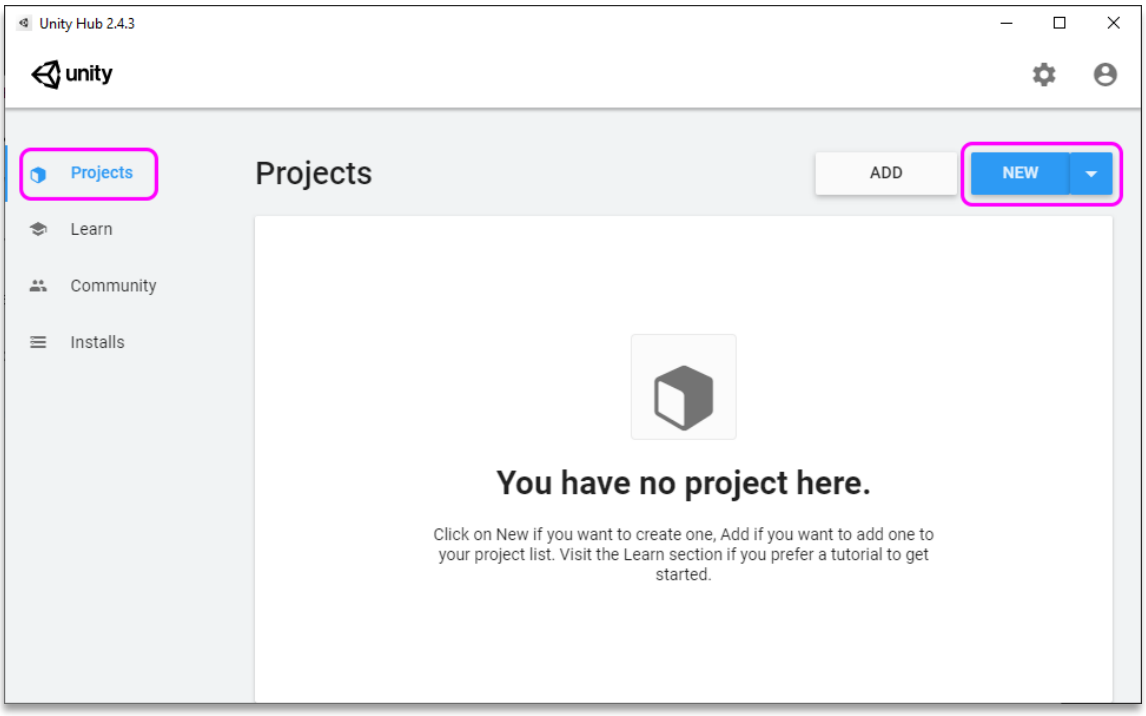
Removing Old Versions of Unity

Remove old versions of **Unity** by clicking the three dots and selecting **Remove from Hub**. Depending on how the version was installed, you might need to locate the installation's folder on your computer and delete it manually.

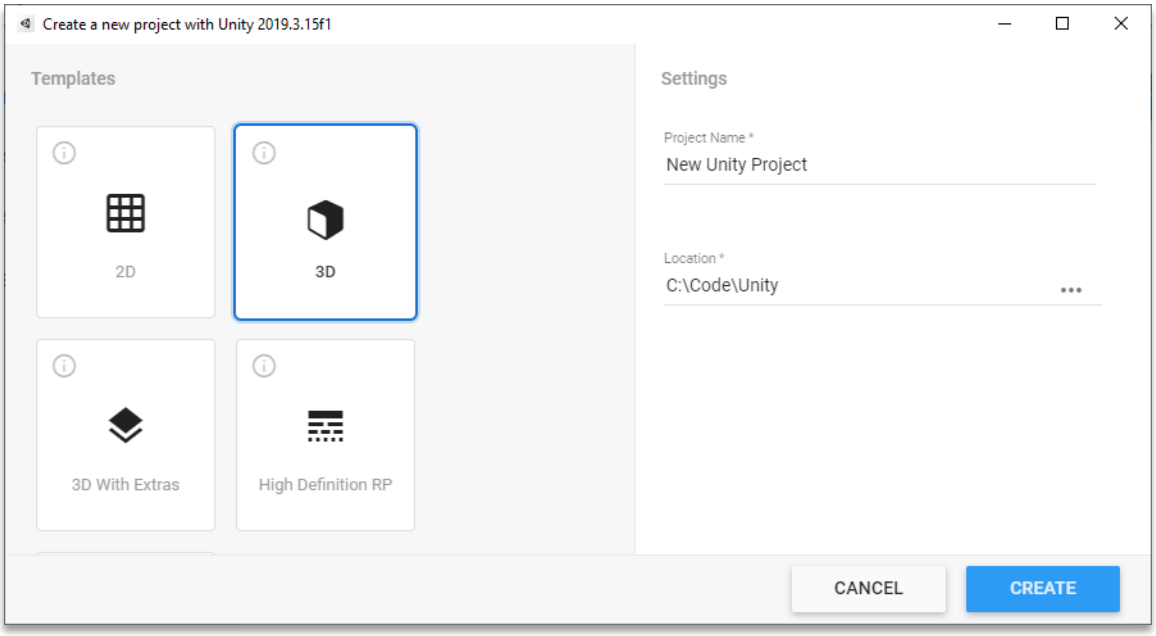


Creating a New Unity Project

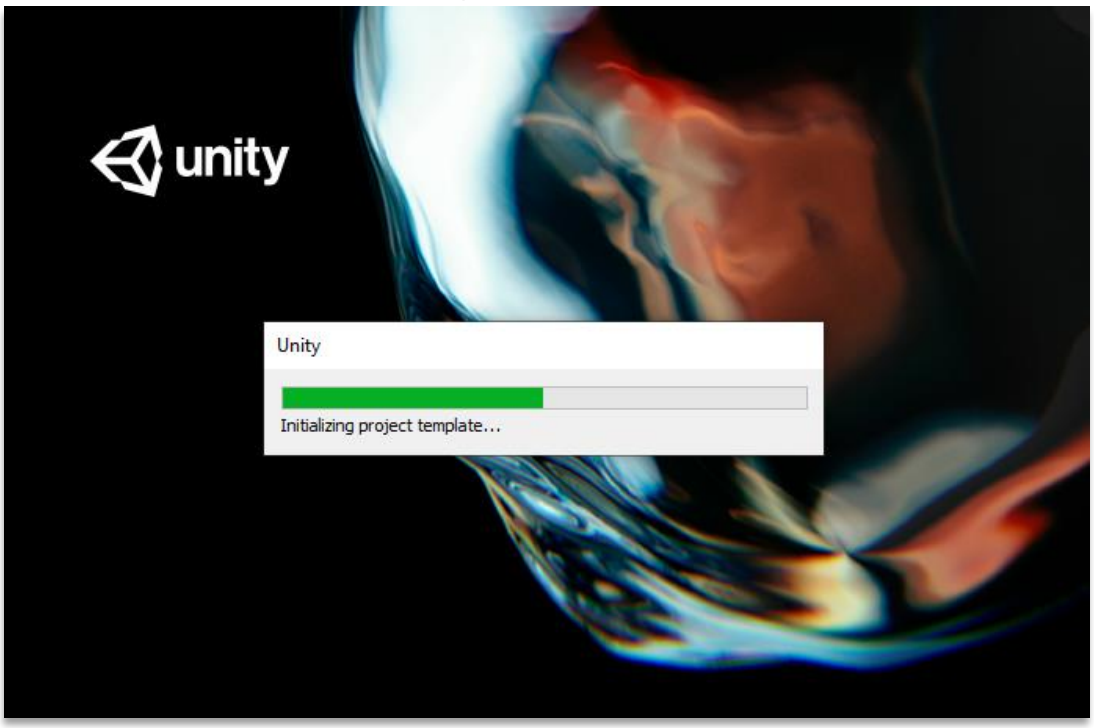
Open the **Projects** tab and click **New** to start a new **Unity project**.



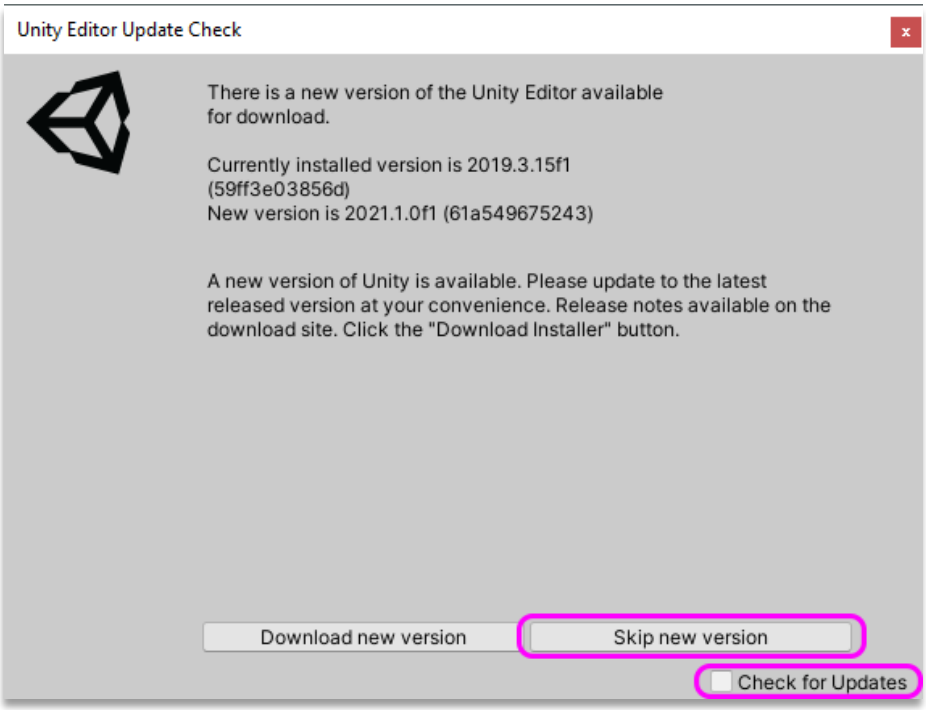
Give the project a name and specify a location that is easily accessible.



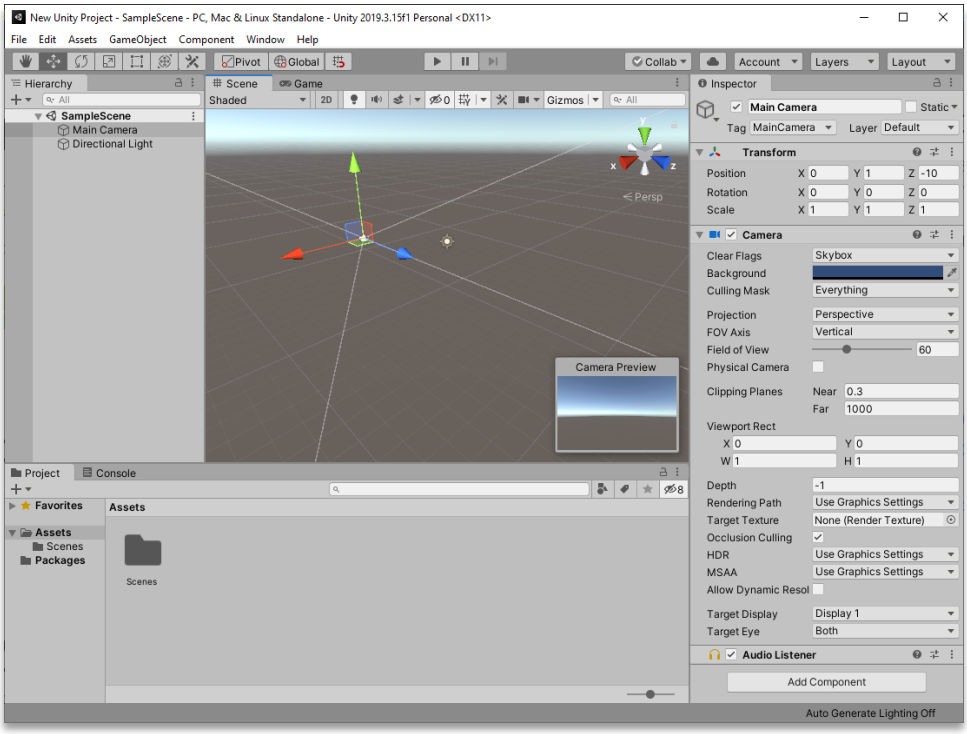
Wait for **Unity** to initialize your project.



To ensure compatibility with the curriculum, uncheck **Check for Updates** and click **Skip new version**.

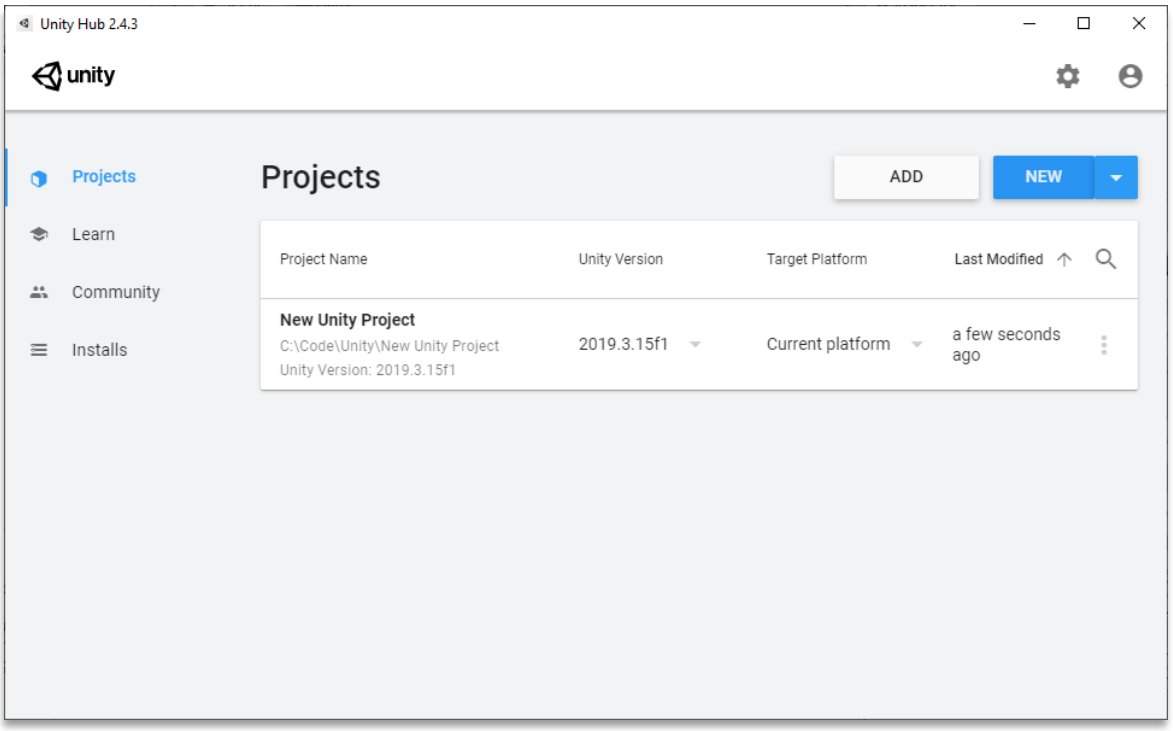


The **Unity editor** is where Ninjas will program their games.

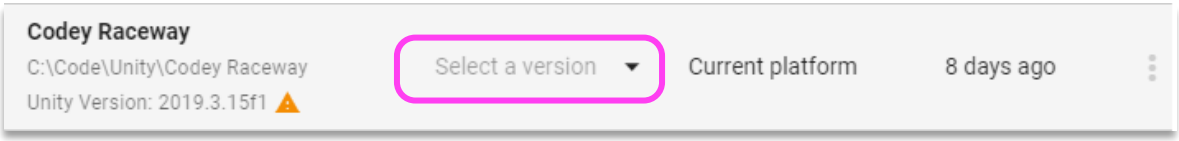


Opening an Existing Project

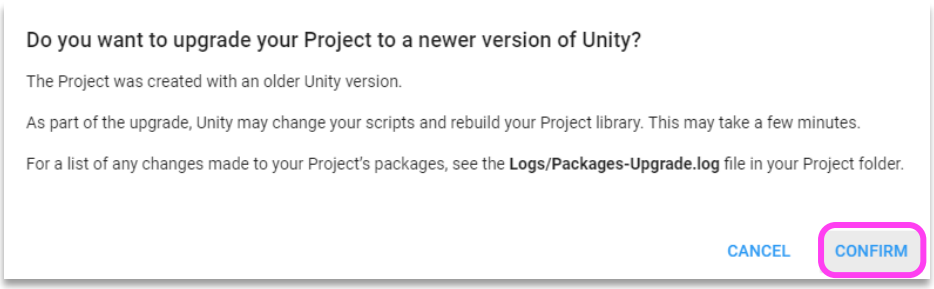
You can view a list of projects in **Unity Hub**.



Each **Project** is locked to a specific version, but you can use the drop down to change it.



When switching to a newer version, select **Confirm**.



To open the project, click the project's entry in the list.

GitHub

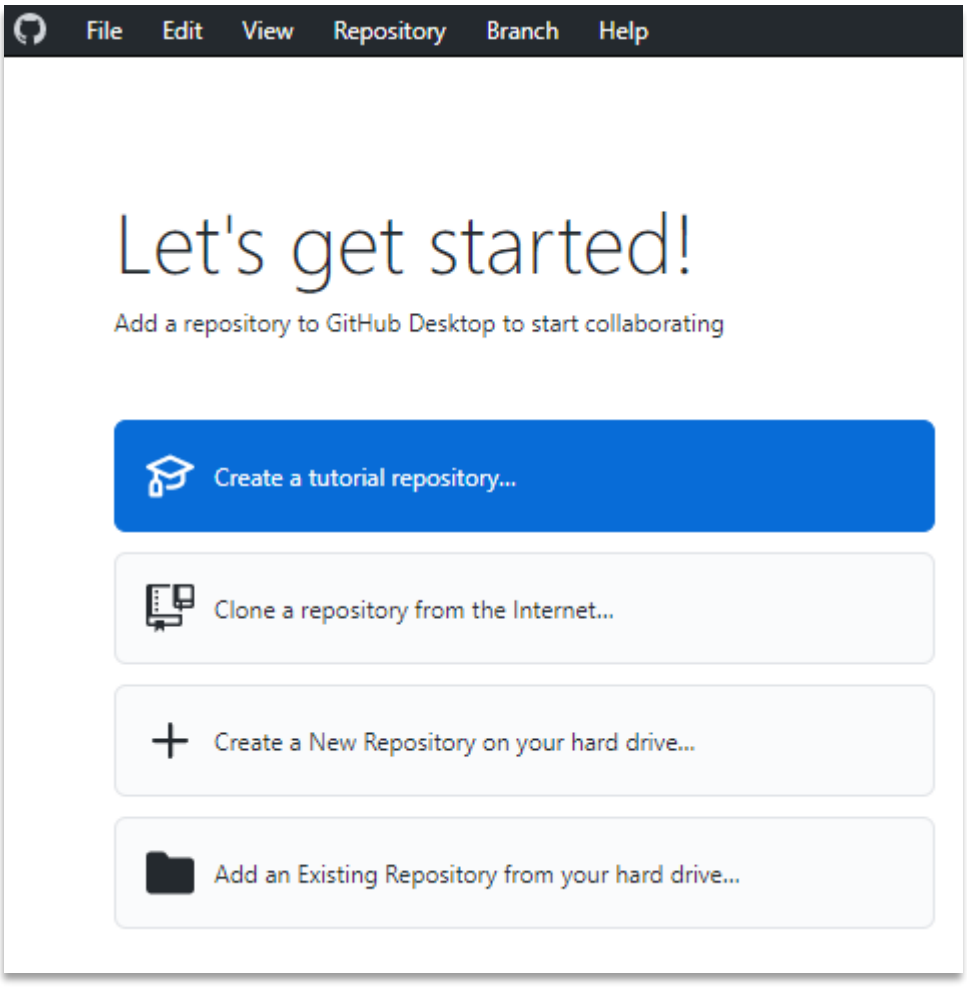
GitHub can be used to save projects to the cloud so they can be synced to more than one computer. GitHub works on a per-account basis, so you can either use one account per center or per ninja.

Download and install **GitHub Desktop** from https://desktop.github.com/.

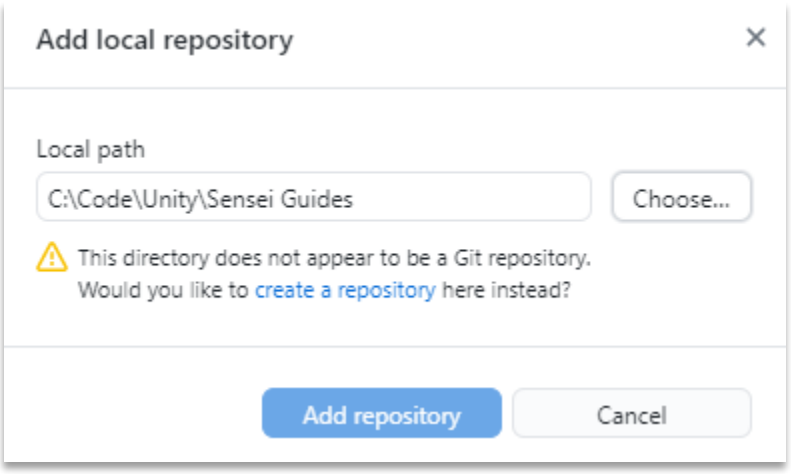
Run the program and log in.

Creating a New GitHub Repository

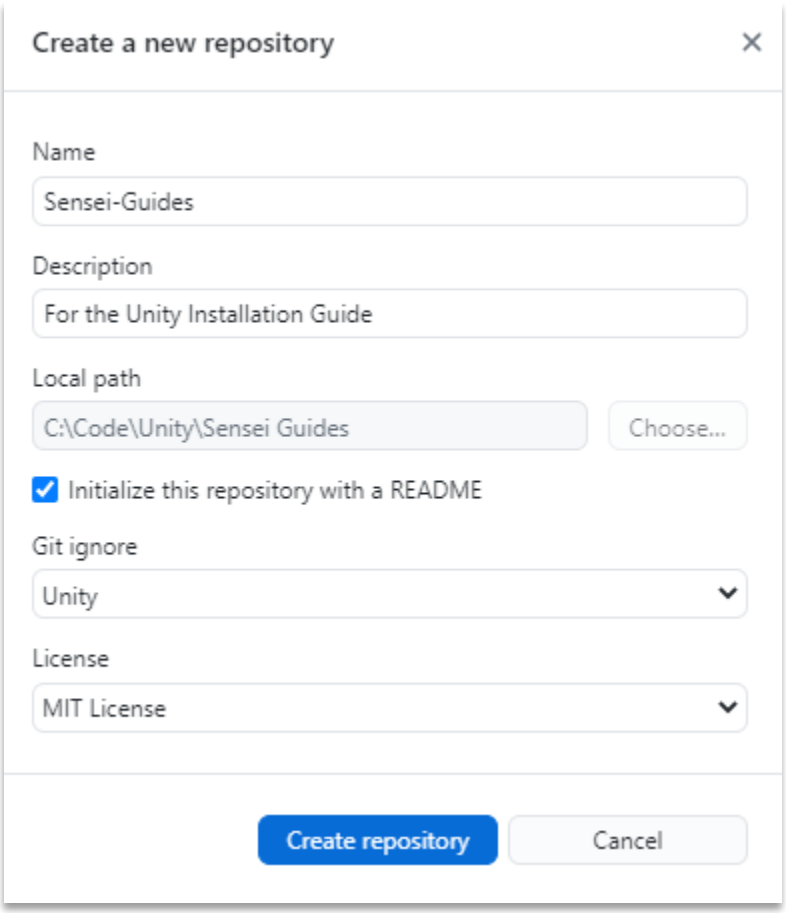
Select **Create a New Repository on your hard drive**.



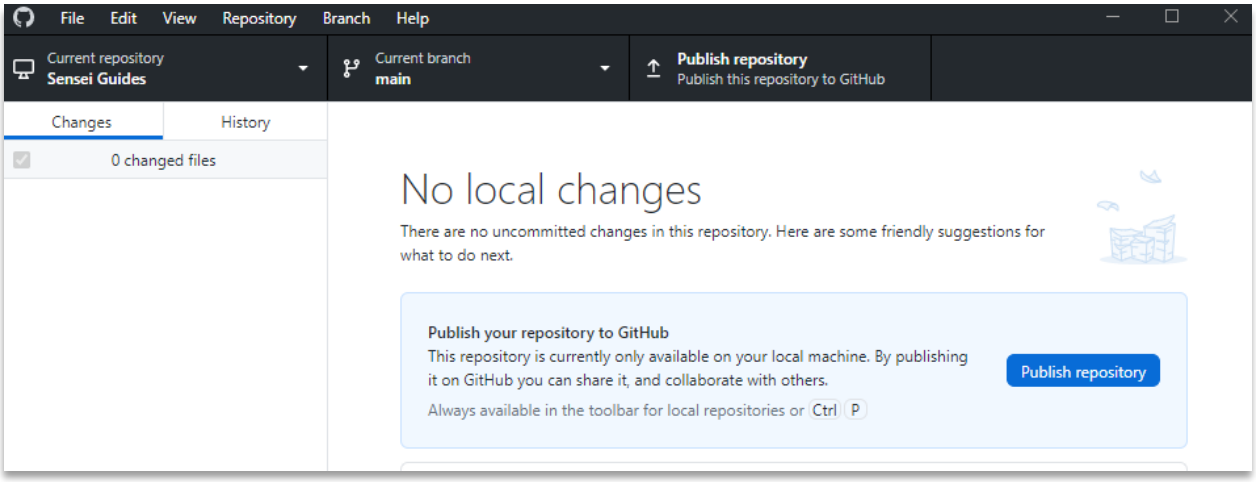
Select the existing Unity project's folder and click **create a repository**.



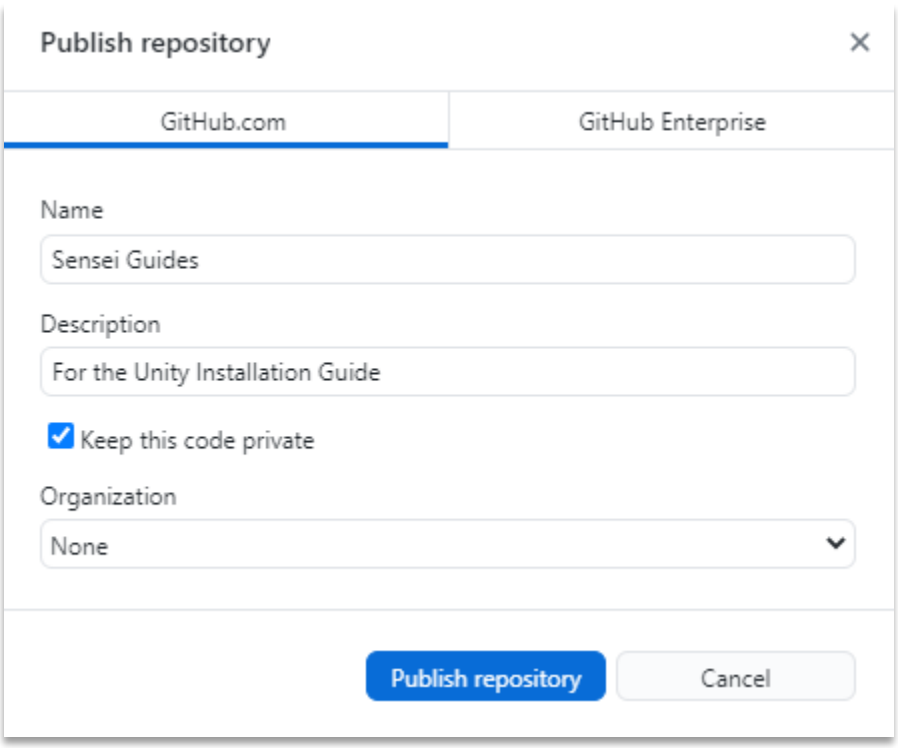
Fill out the **description** and select **Initialize this repository with a README**. Be sure to set the **Git ignore** to **Unity** and select a **license**. Click **Create repository**.



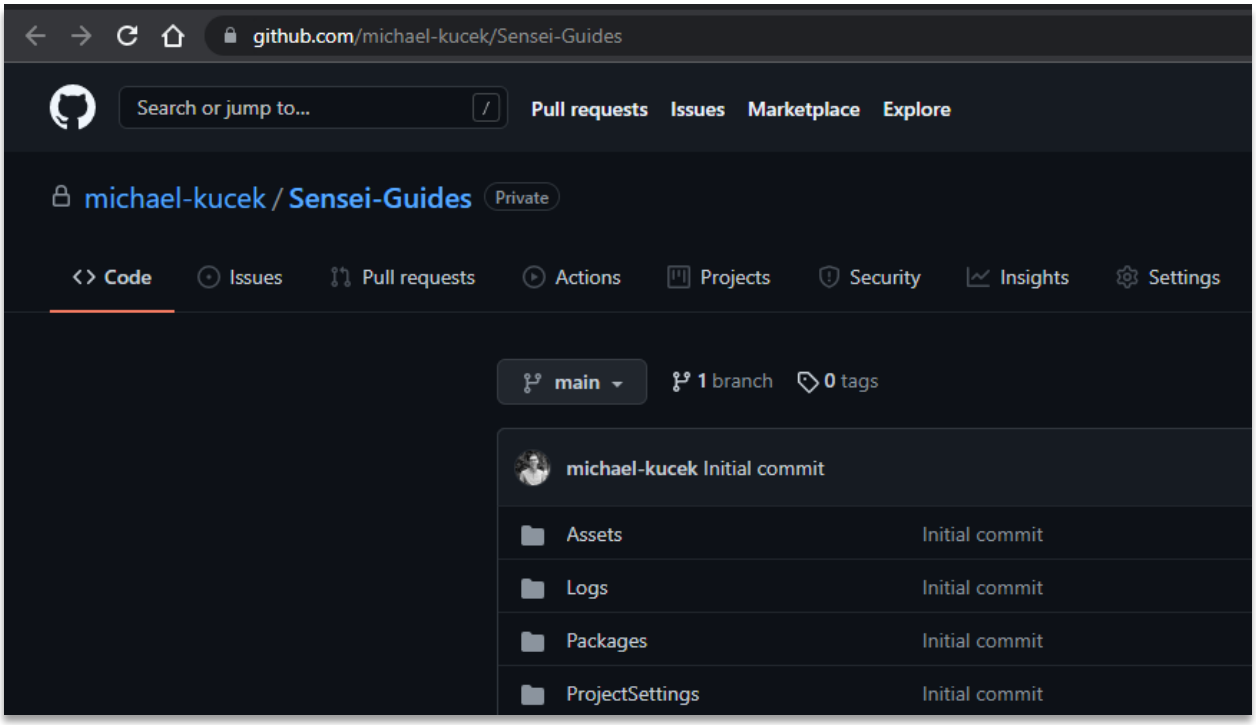
All of the files located inside the selected project folder will be scanned and added to the local git repository. Click **Publish repository** to publish the project to the GitHub account.



You can choose to keep the code private or make it public. Click **Publish repository**.

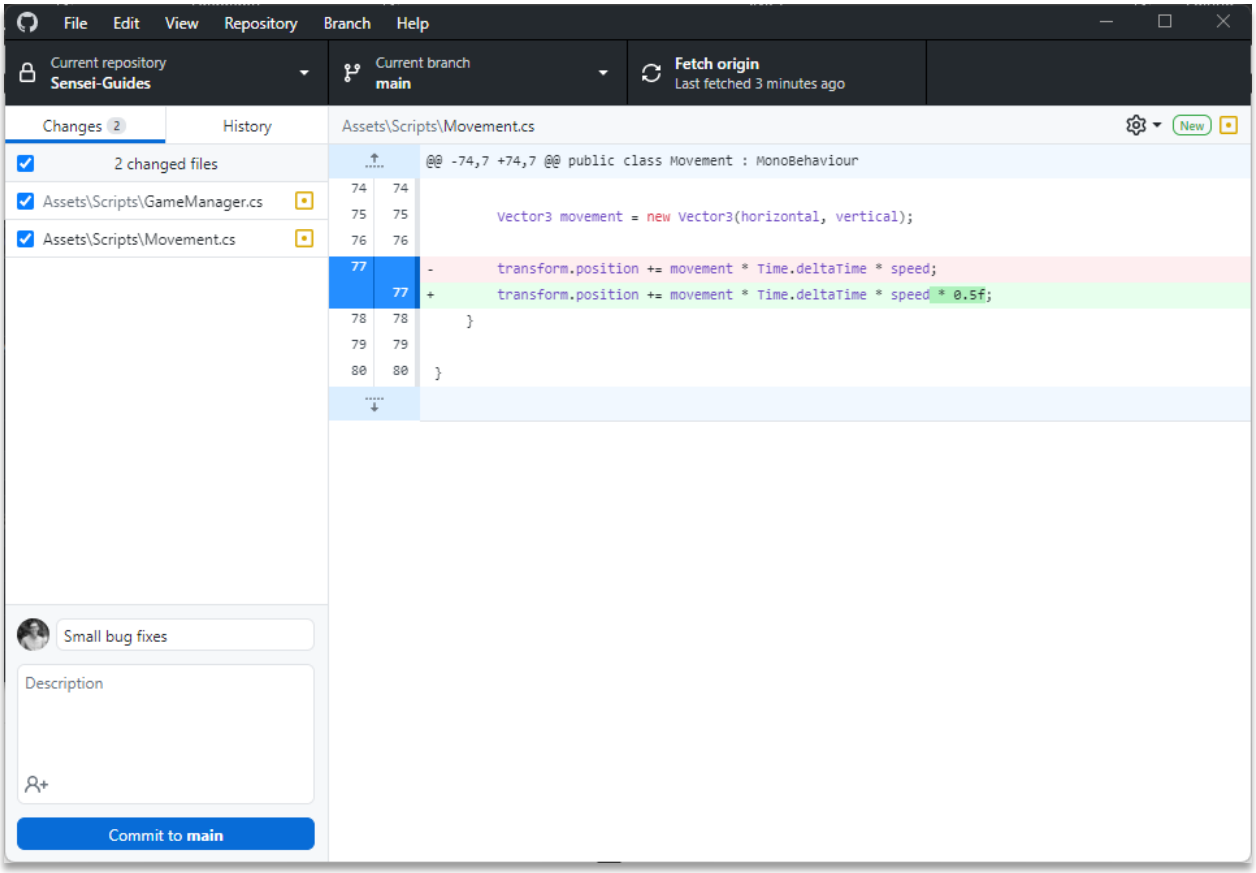


The project is now published and can be downloaded on any other computer logged in to the correct GitHub account.

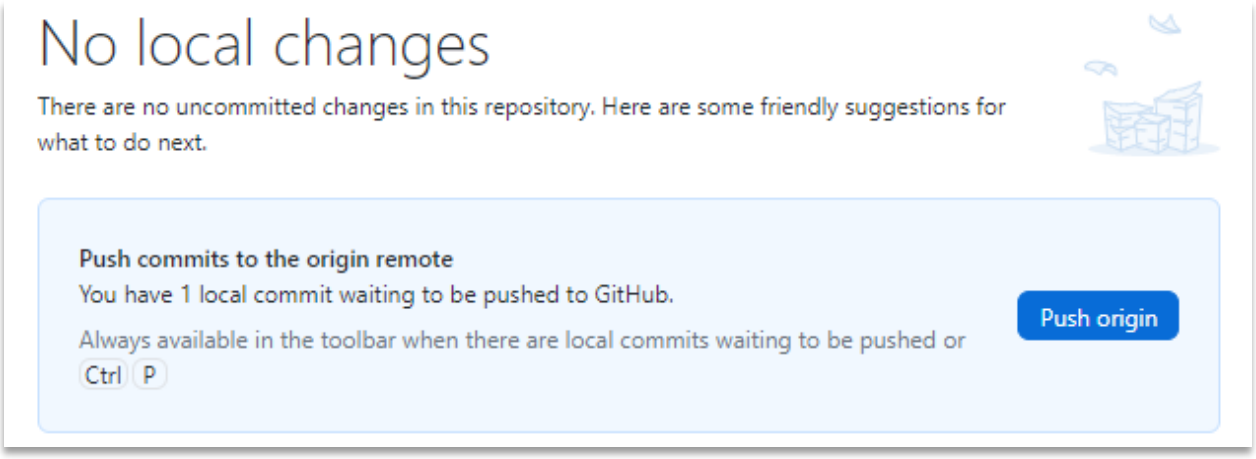


Syncing Local Changes to GitHub

Once changes are made to the project, you need to manually sync them with GitHub Desktop. With the project open, you will see a list of all the unsaved changes. Give the changes a **summary** and an optional **description**. Click **Commit to main**.



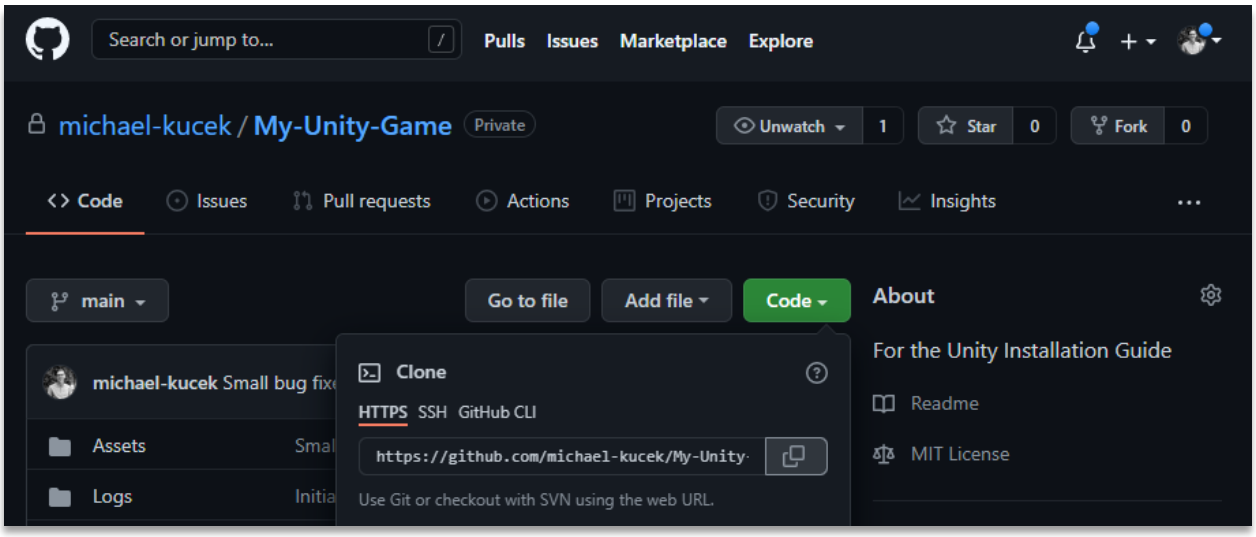
These changes are now saved to your local repository. To save them to GitHub, you need to click **Push origin**.



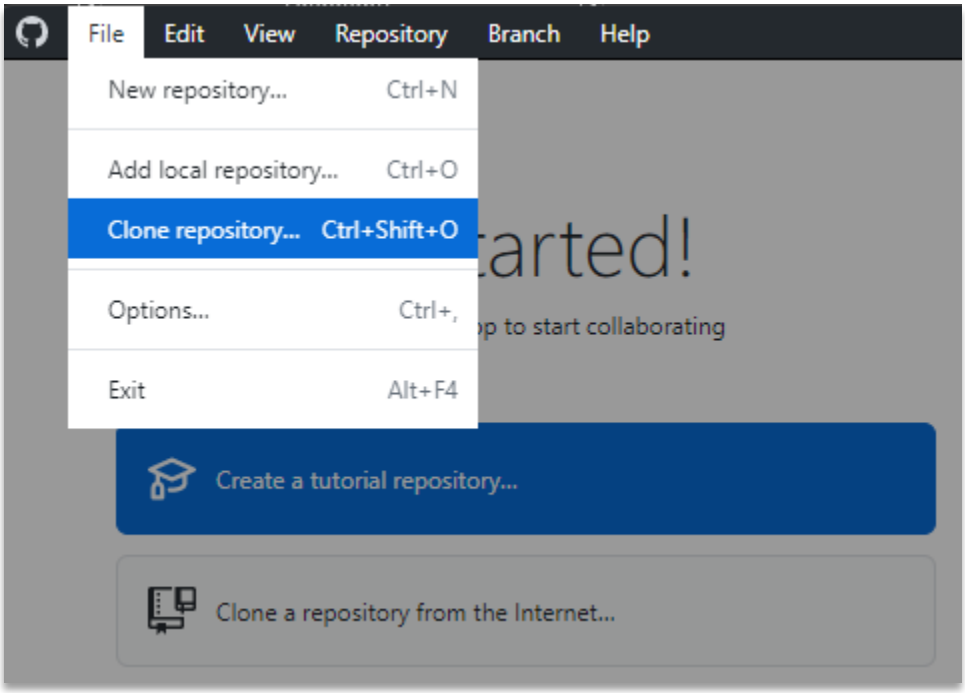
This process needs to be repeated every time changes are made.

Syncing an Existing GitHub Repository

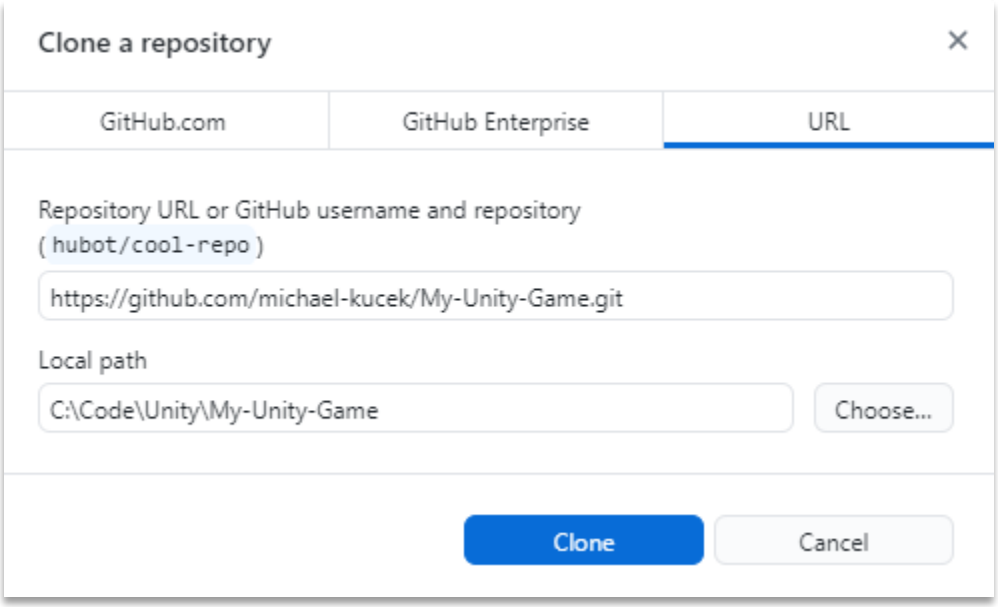
To download an existing GitHub project to a new computer, you need the project's **GitHub URL**. Navigate to the GitHub project page, click on **Code**, and then copy the URL.



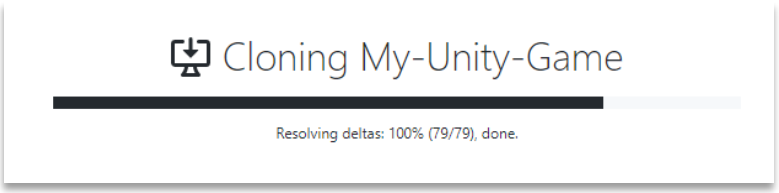
In GitHub Desktop, select **Clone a repository**.

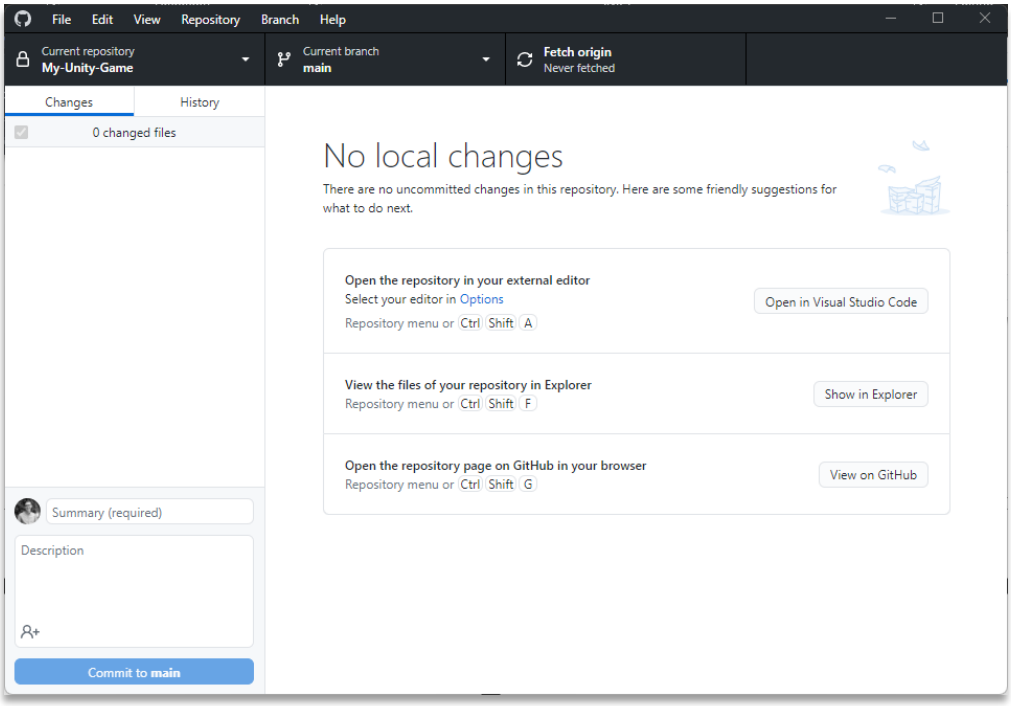


Paste in the **GitHub URL** and choose where you want to save the local repository. Click **Clone**.



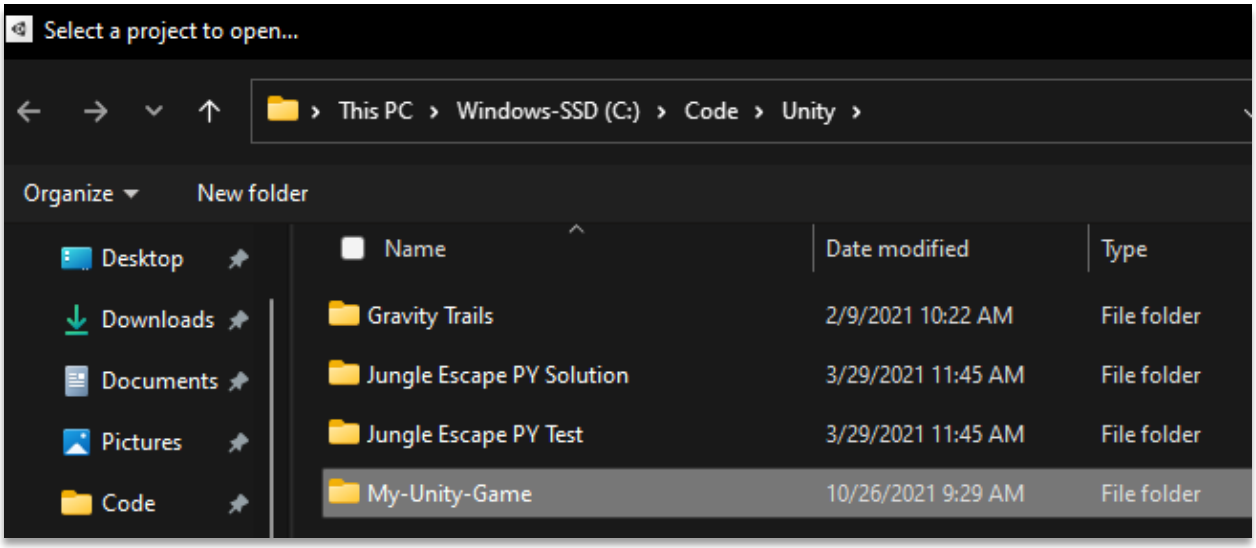
The project will be synced locally to the computer.

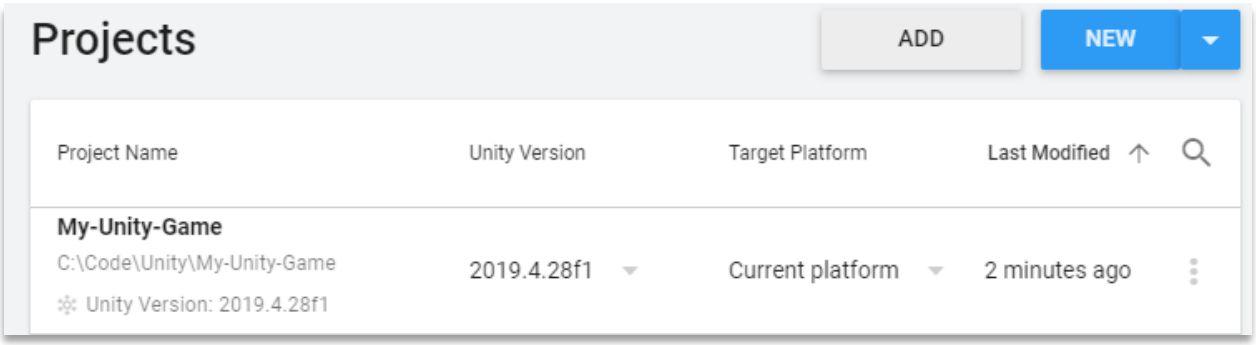




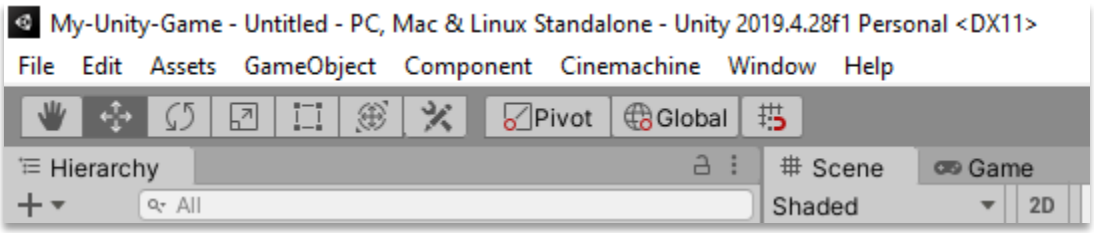
Open **Unity Hub** and click **Add**.

Find the project that was just downloaded from GitHub.



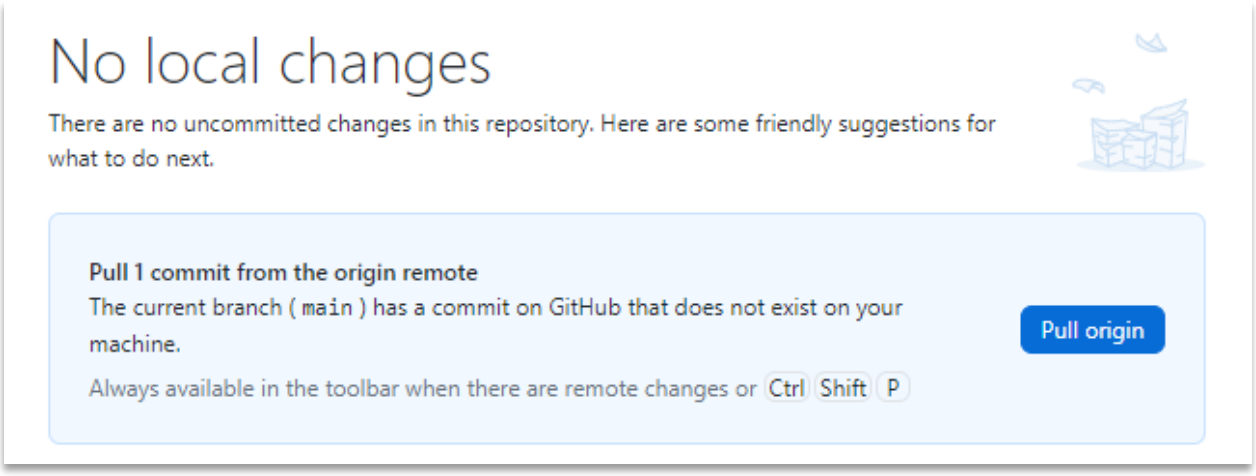


Open the project in Unity. Be sure to sync any changes.



Syncing Changes from Another Computer

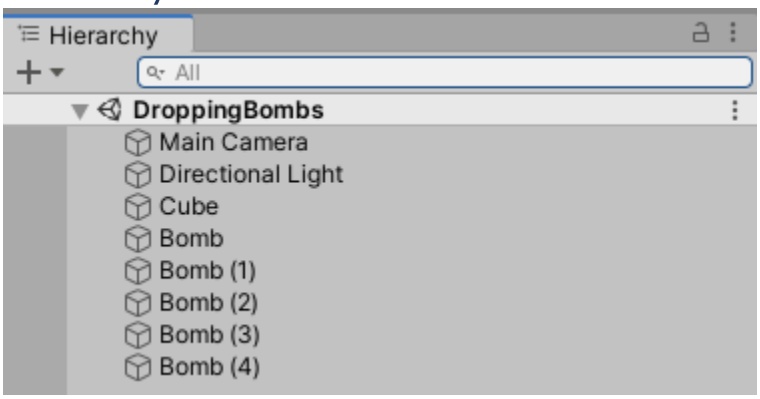
If changes to the game have been made on another computer, you must manually sync the changes using GitHub Desktop. To ensure your local copy is synced with the GitHub project, click Pull origin when GitHub detects your local version is out of date.



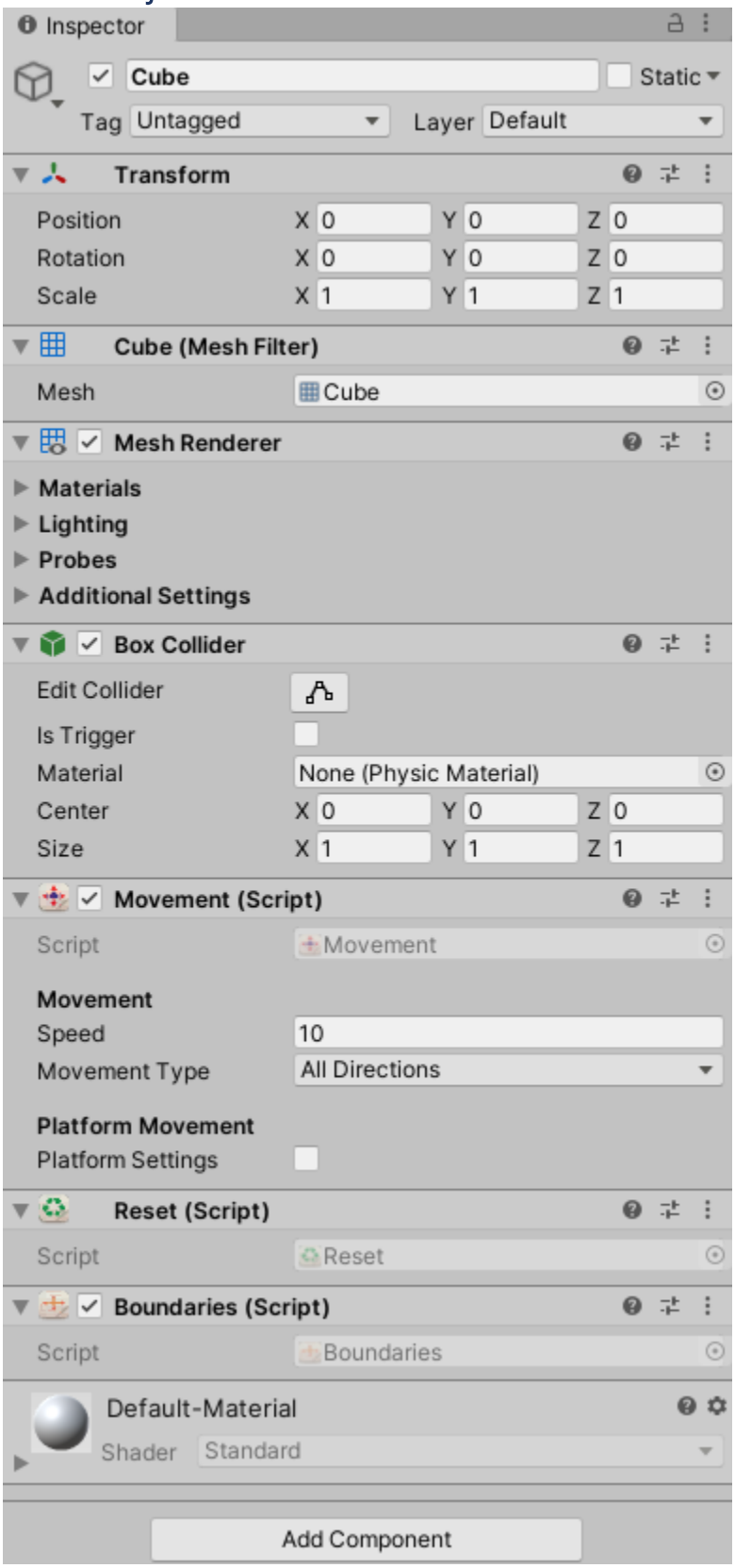
Activity Solutions

Dropping Bombs

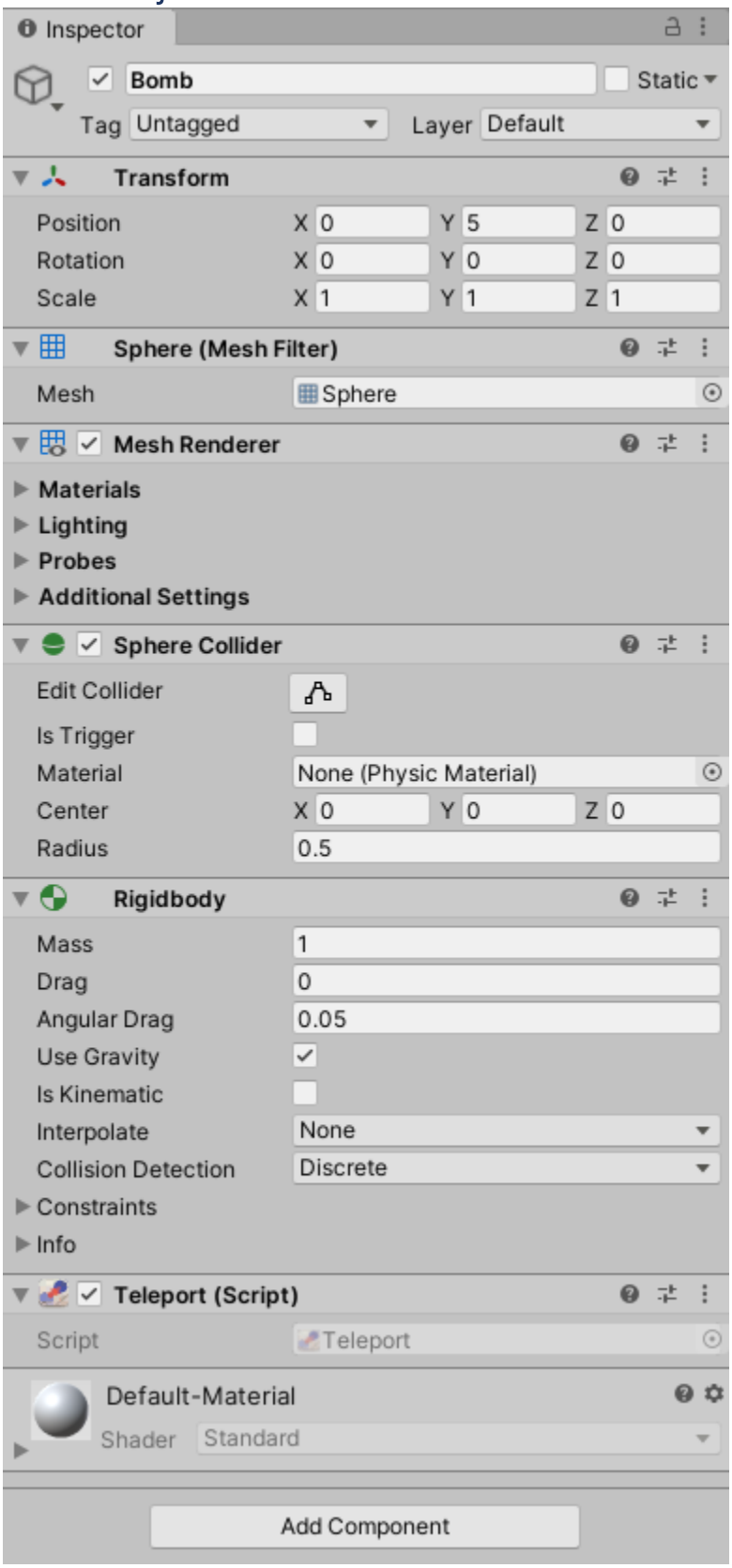
Hierarchy



Cube Object

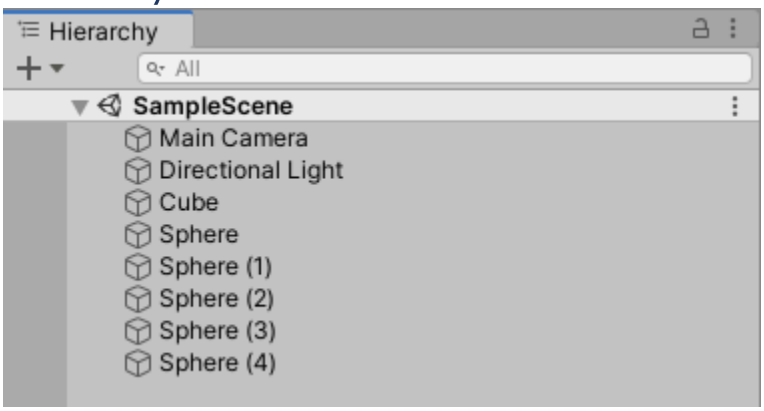


Bomb Object



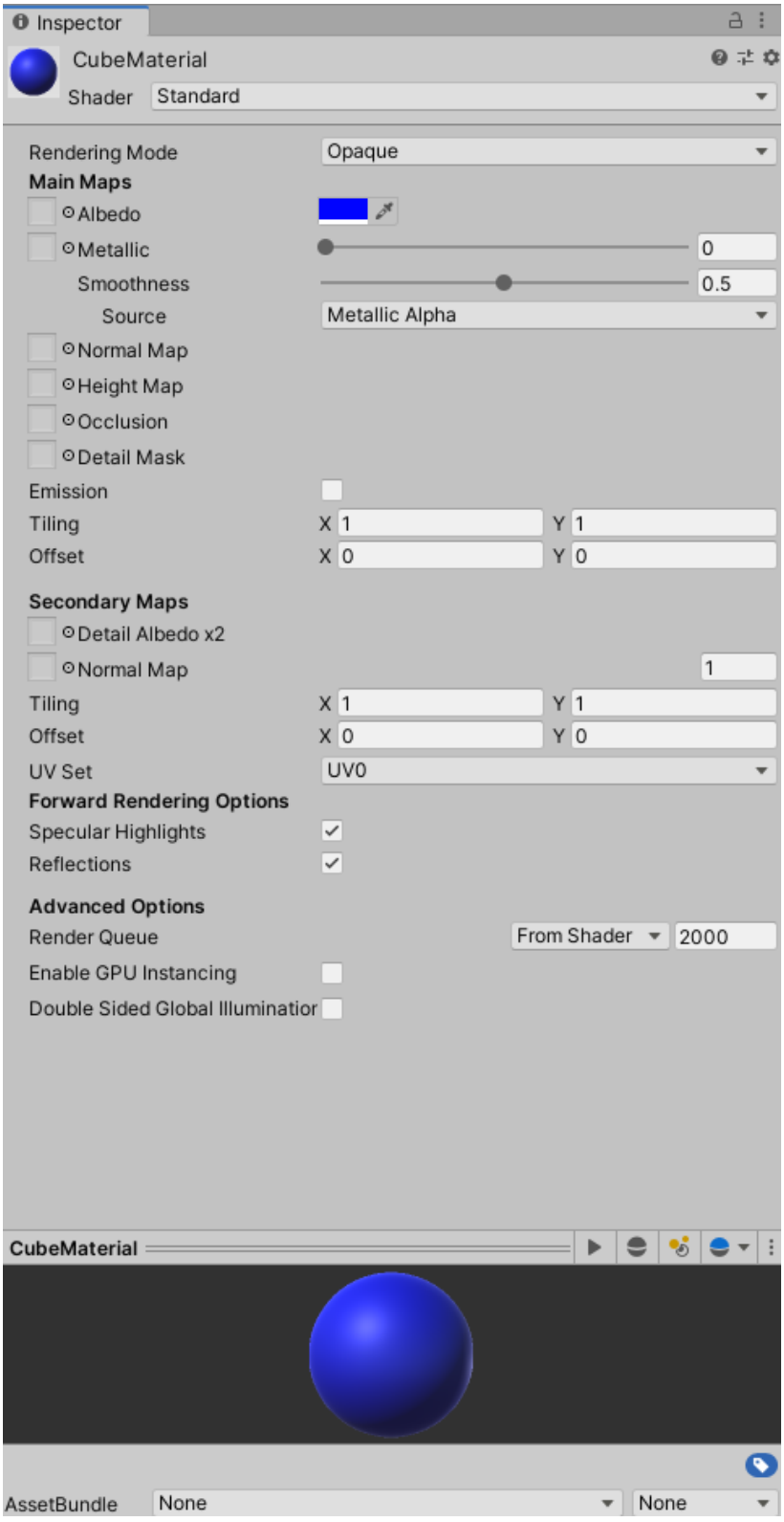
Color Drop Prove Yourself

Hierarchy



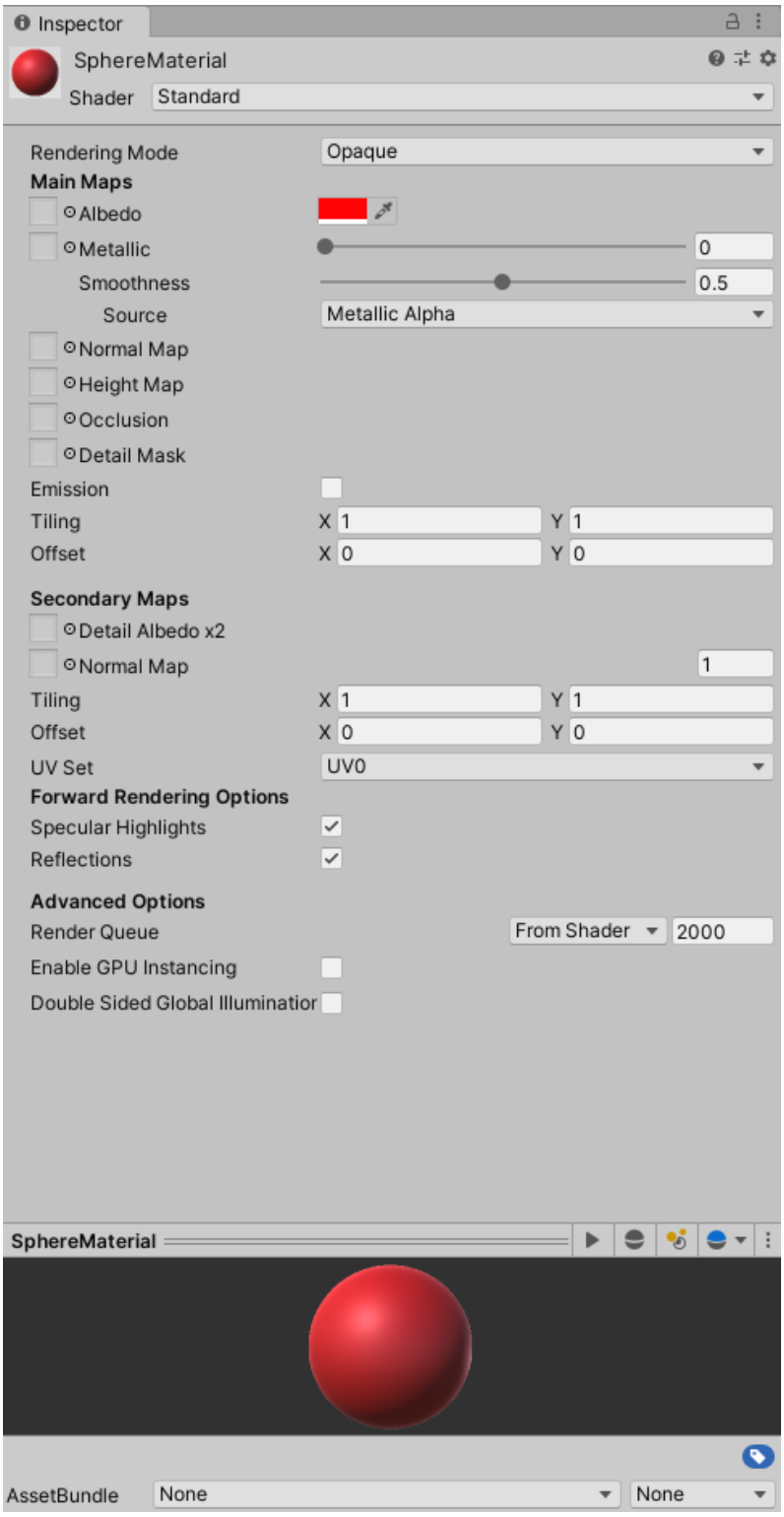
Cube Material

Ninjas can select use any color.

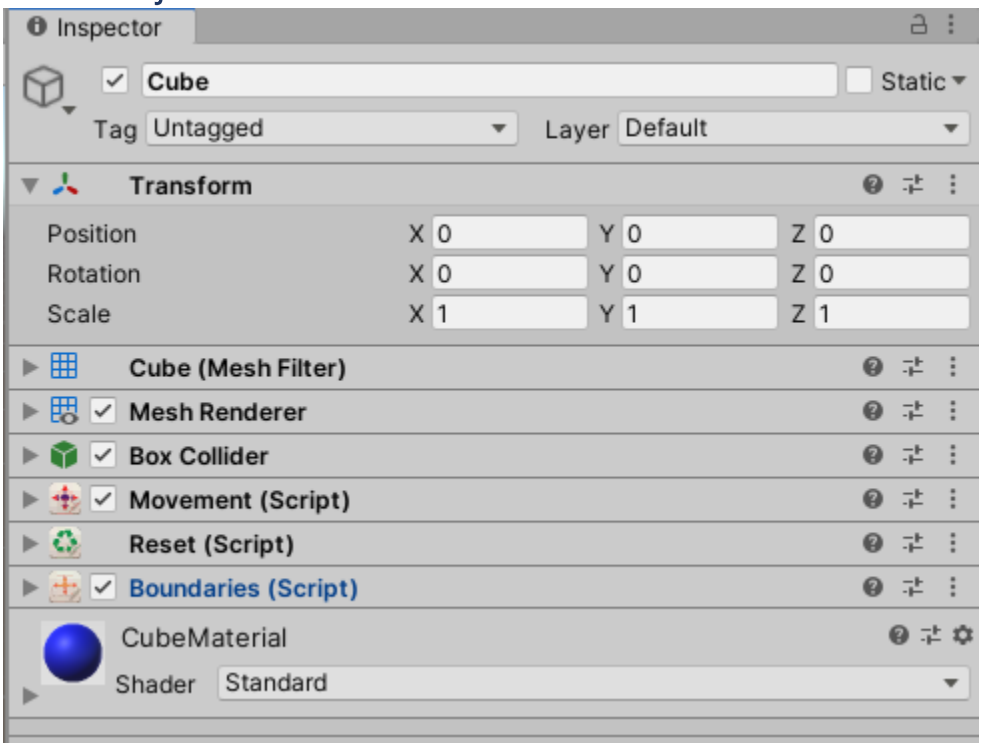


Sphere Material

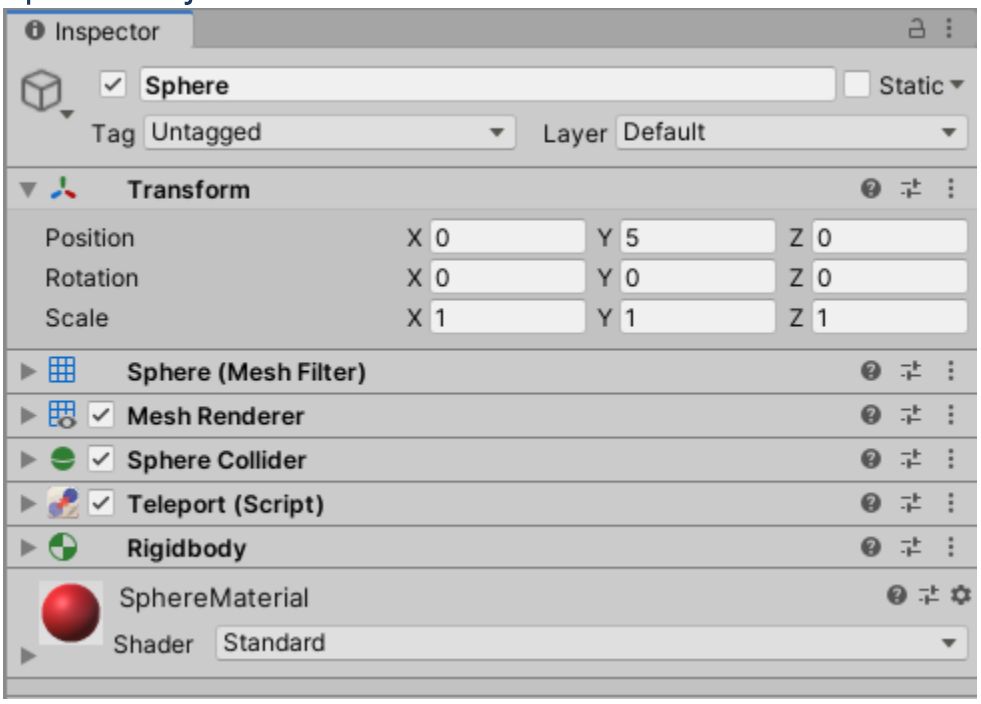
Ninjas can use any color.



Cube Object



Sphere Object

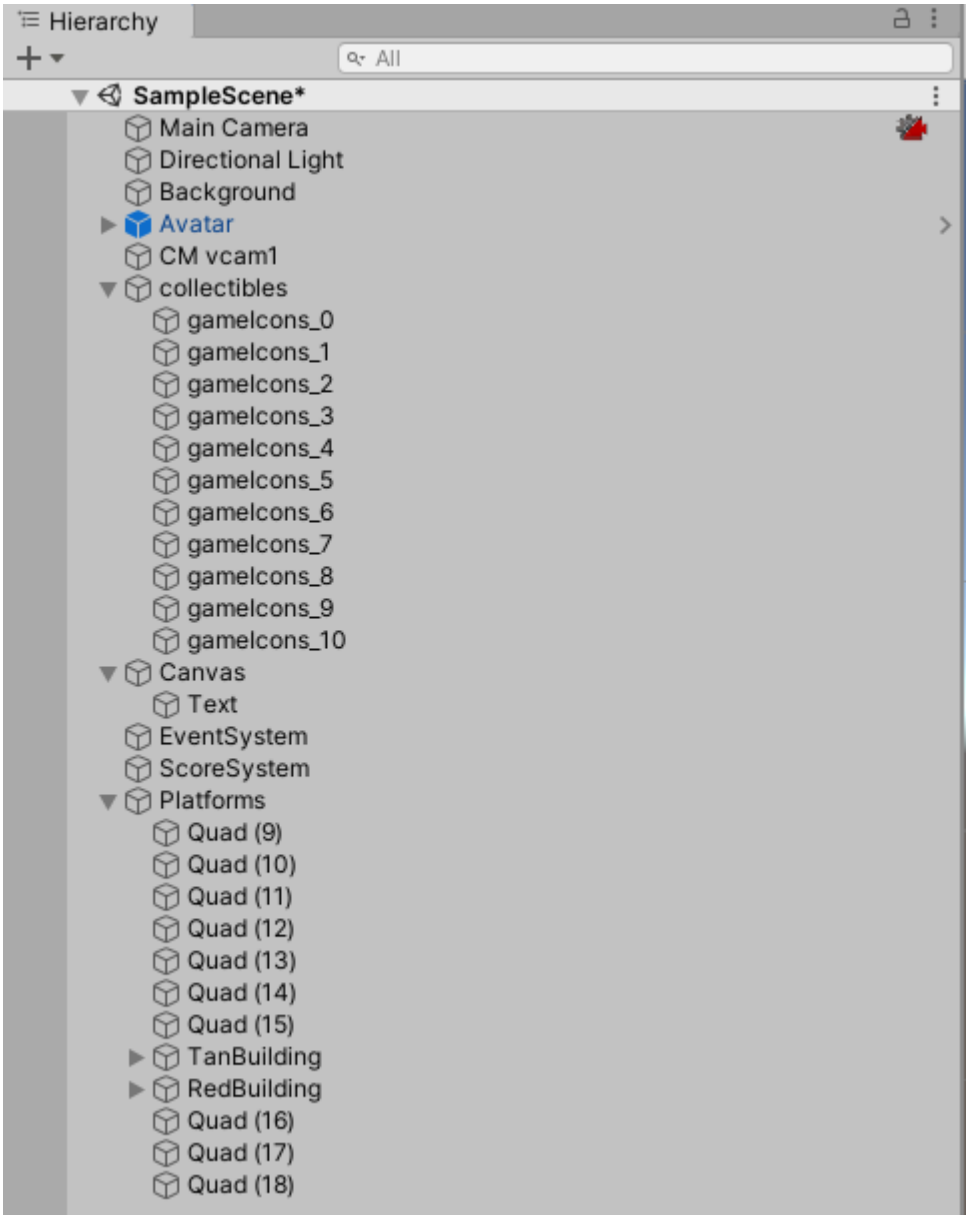


Scavenger Hunt

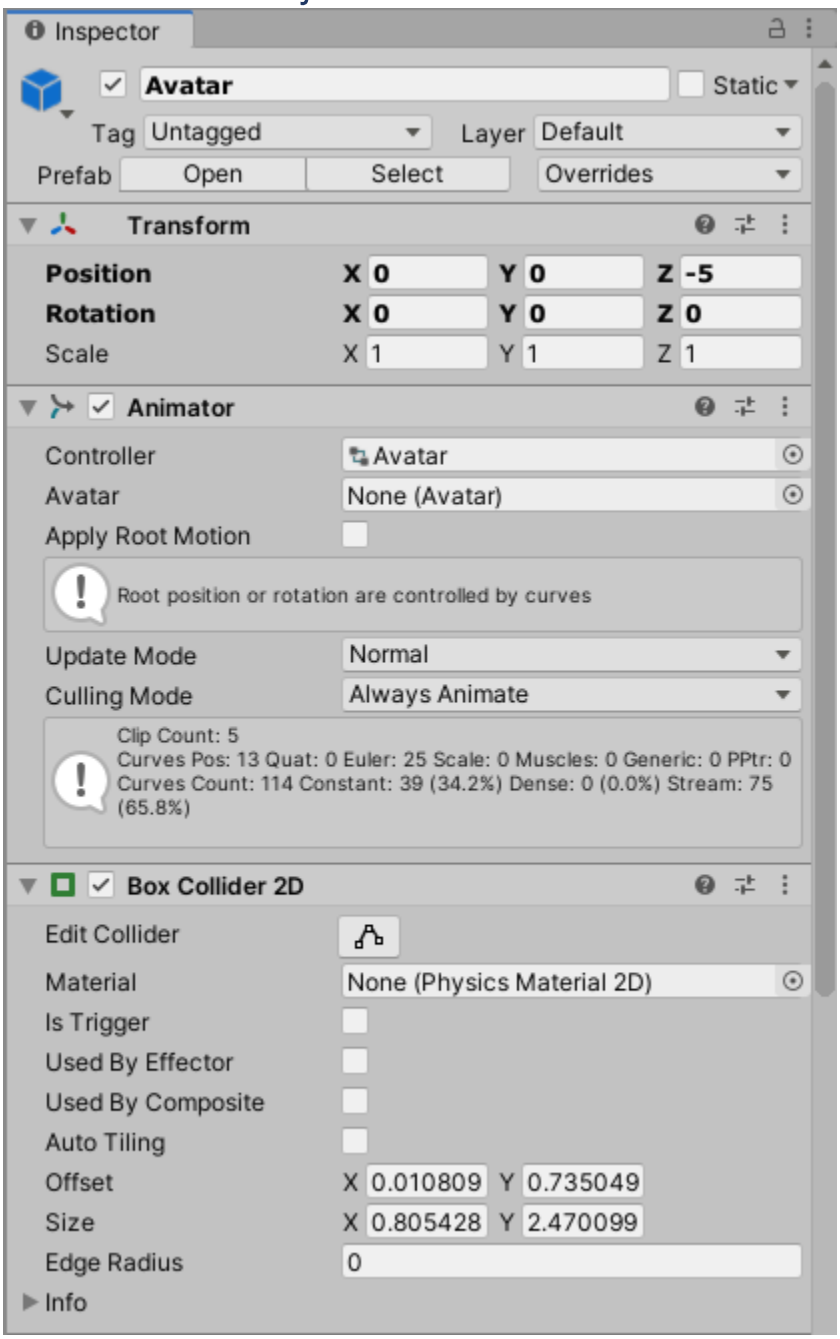
This activity uses the Cinemachine Package.

Hierarchy

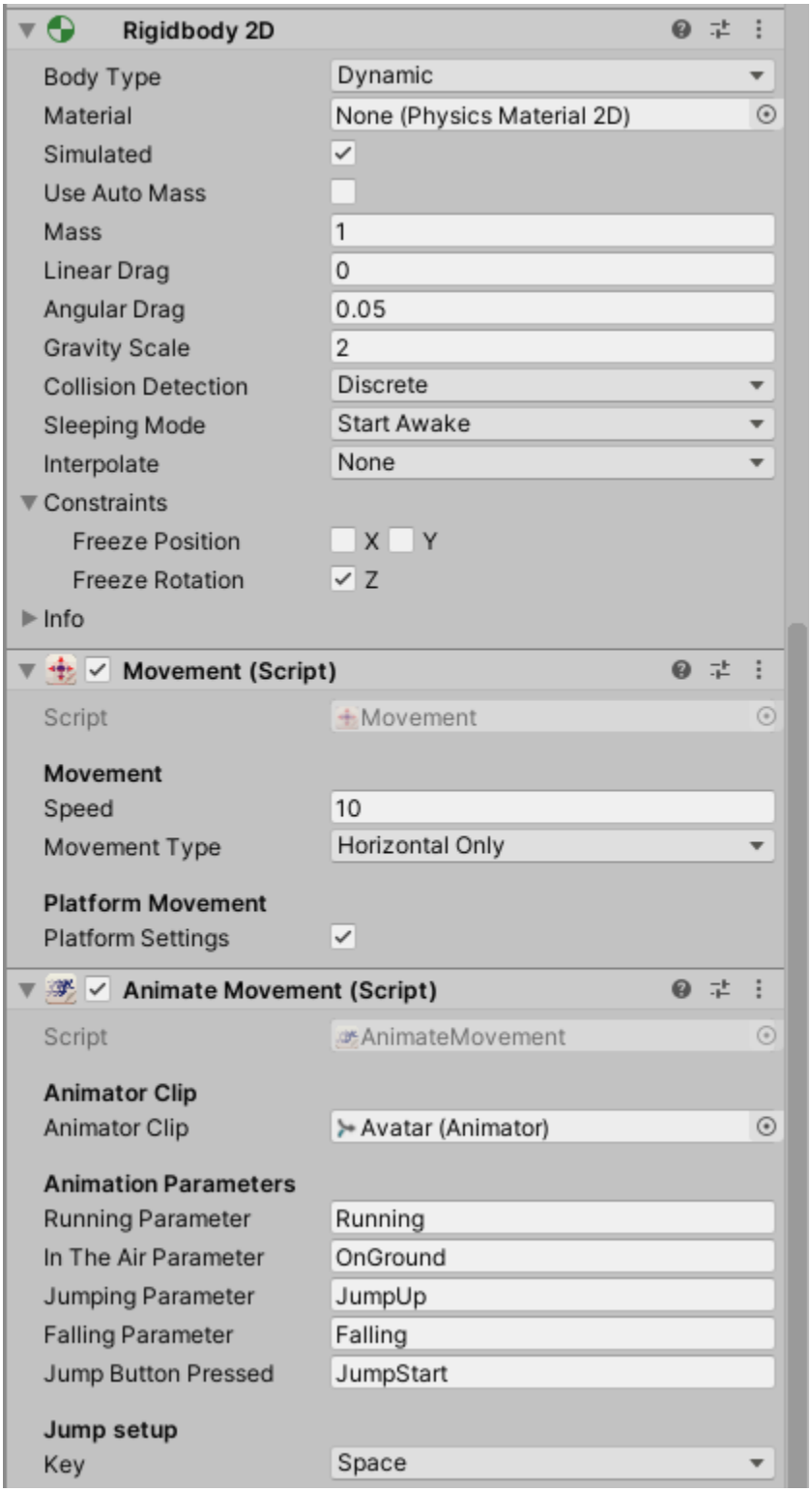
The quad objects might be named based on their location in the scene.



Avatar Prefab Object

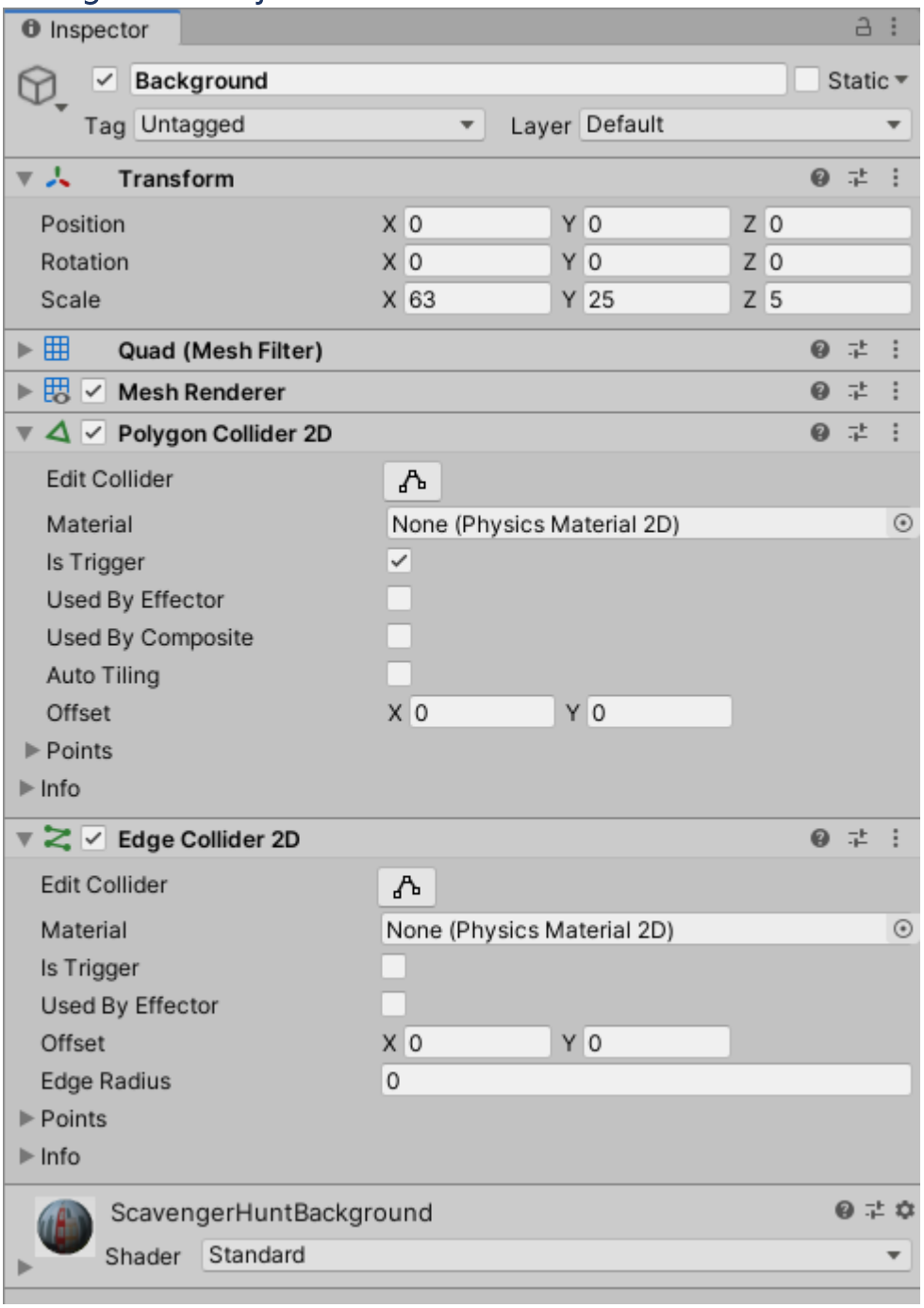


Avatar Prefab Object Continued

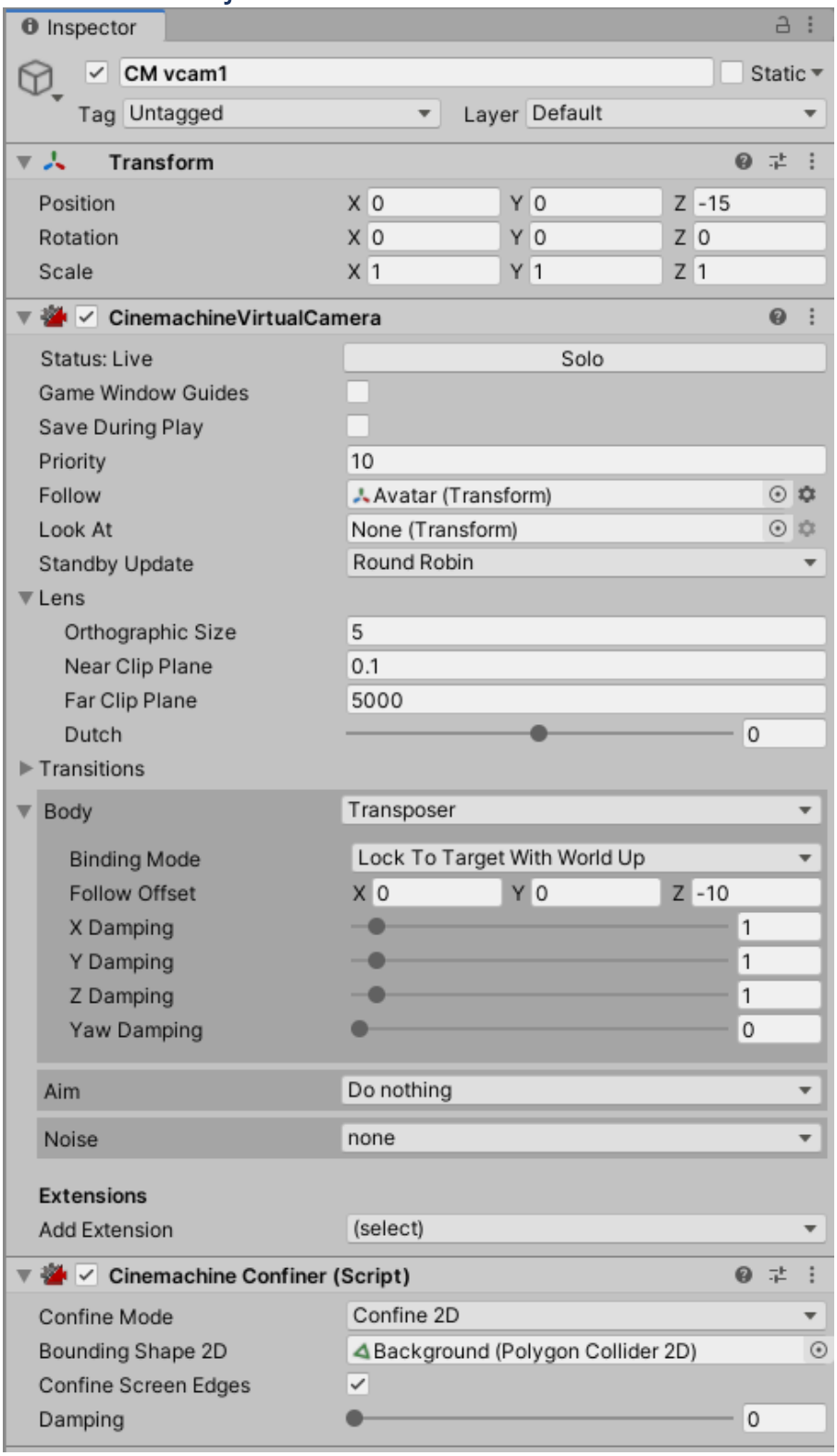


Avatar Prefab Object Continued

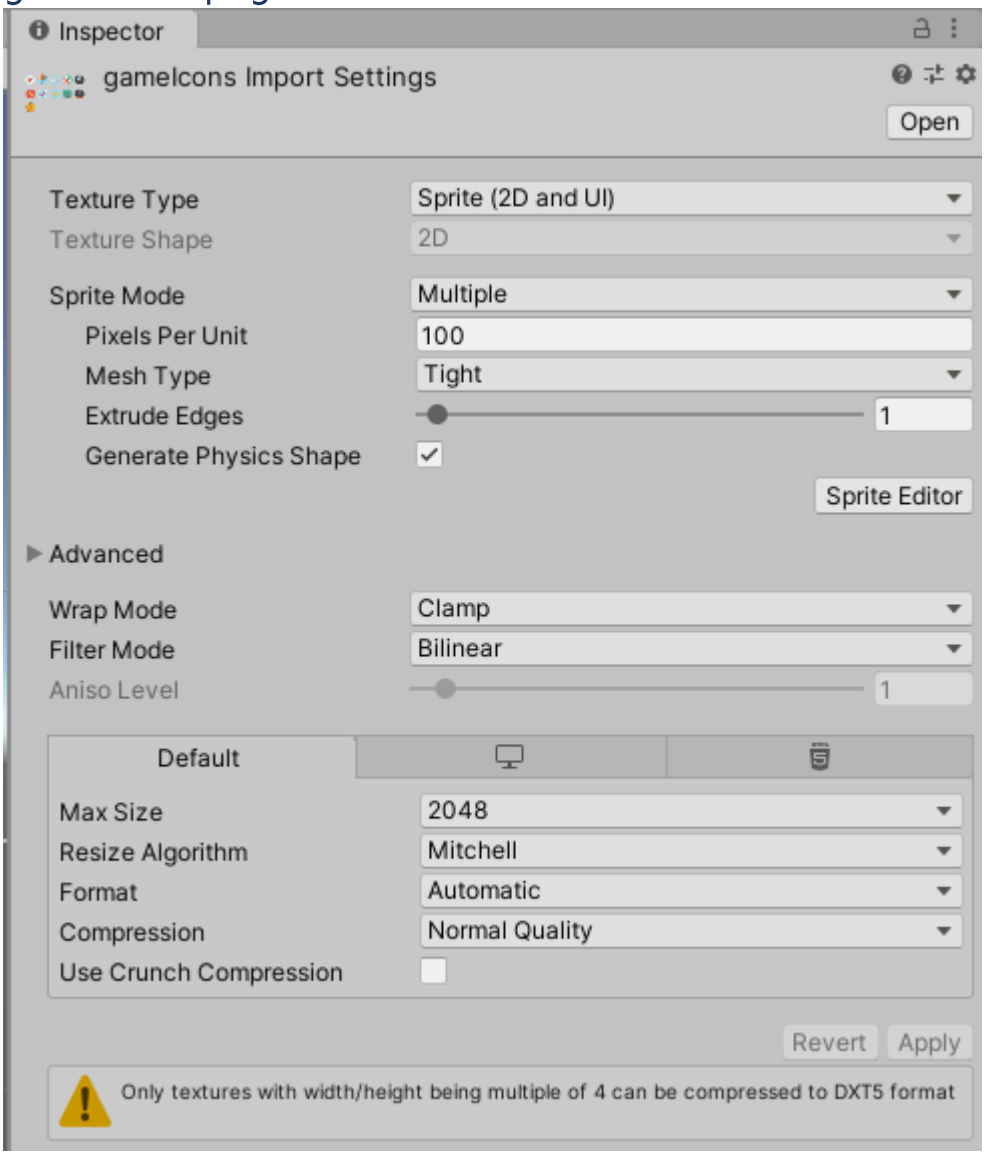
Background Object



CM vcam1 Object

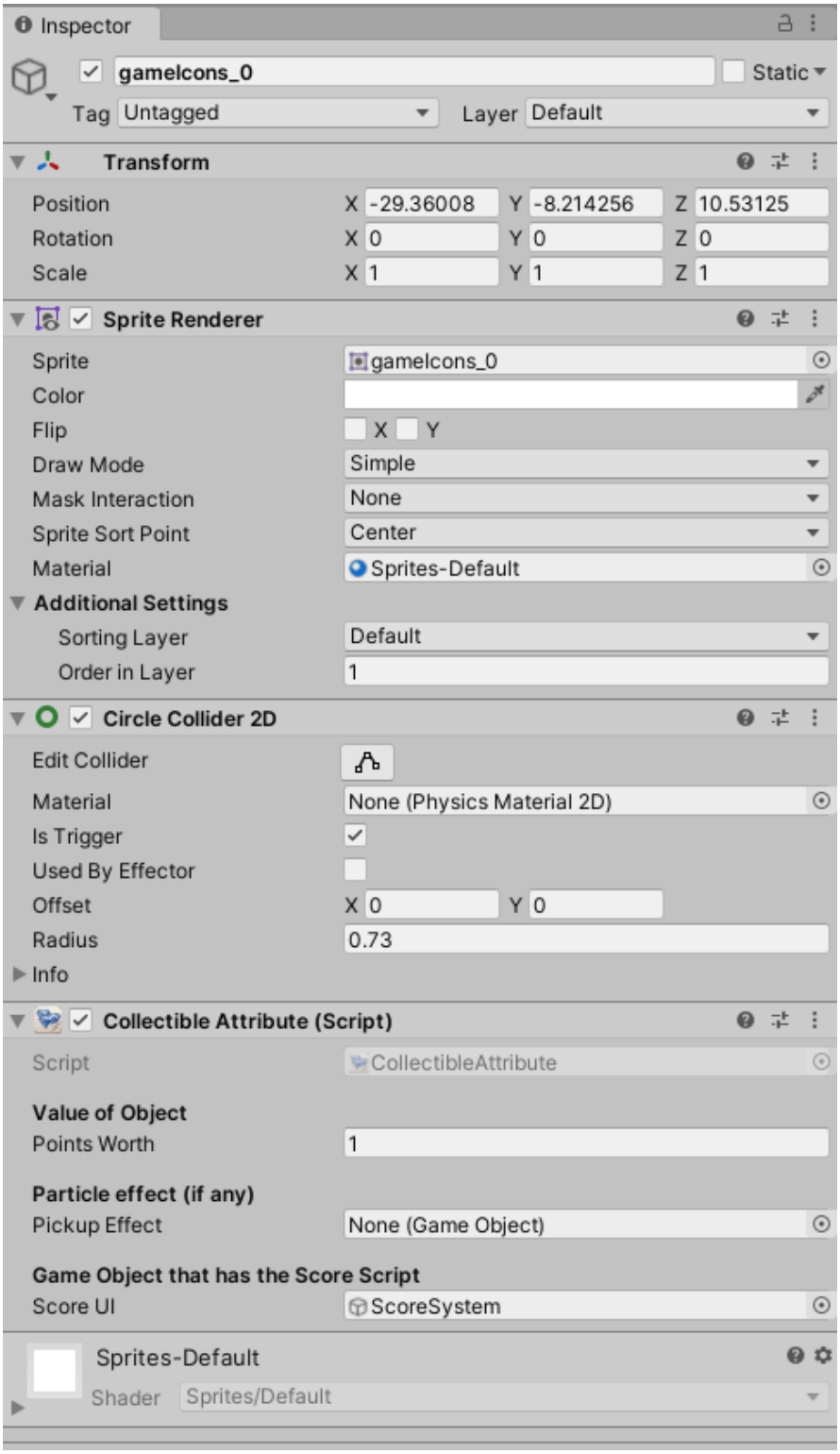


gameicons.png Asset



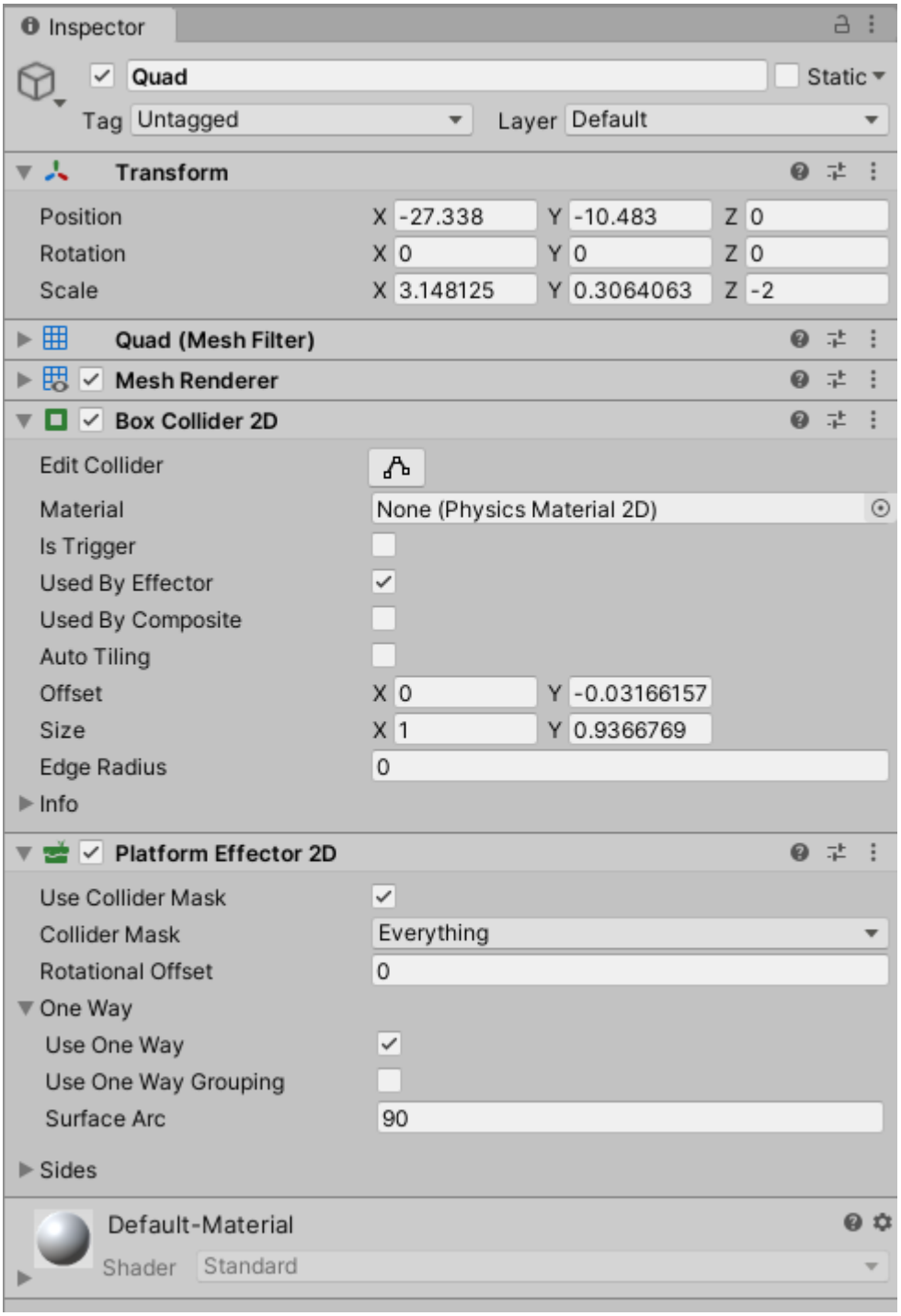
Game Icons Objects

The gamelcons objects will have a different names and positions based on their locations in the scene.

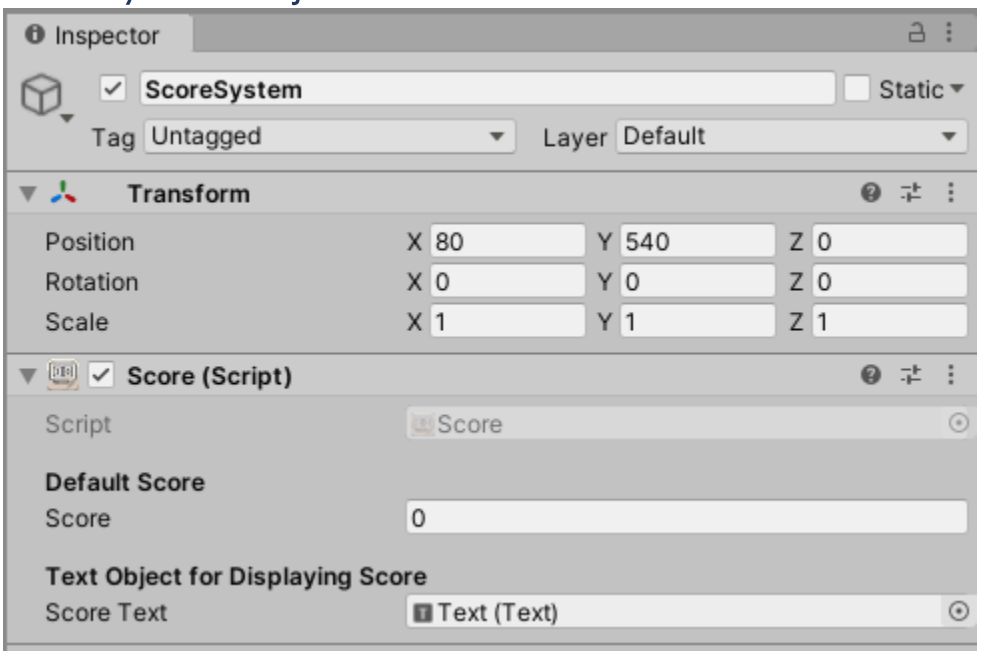


Quad objects

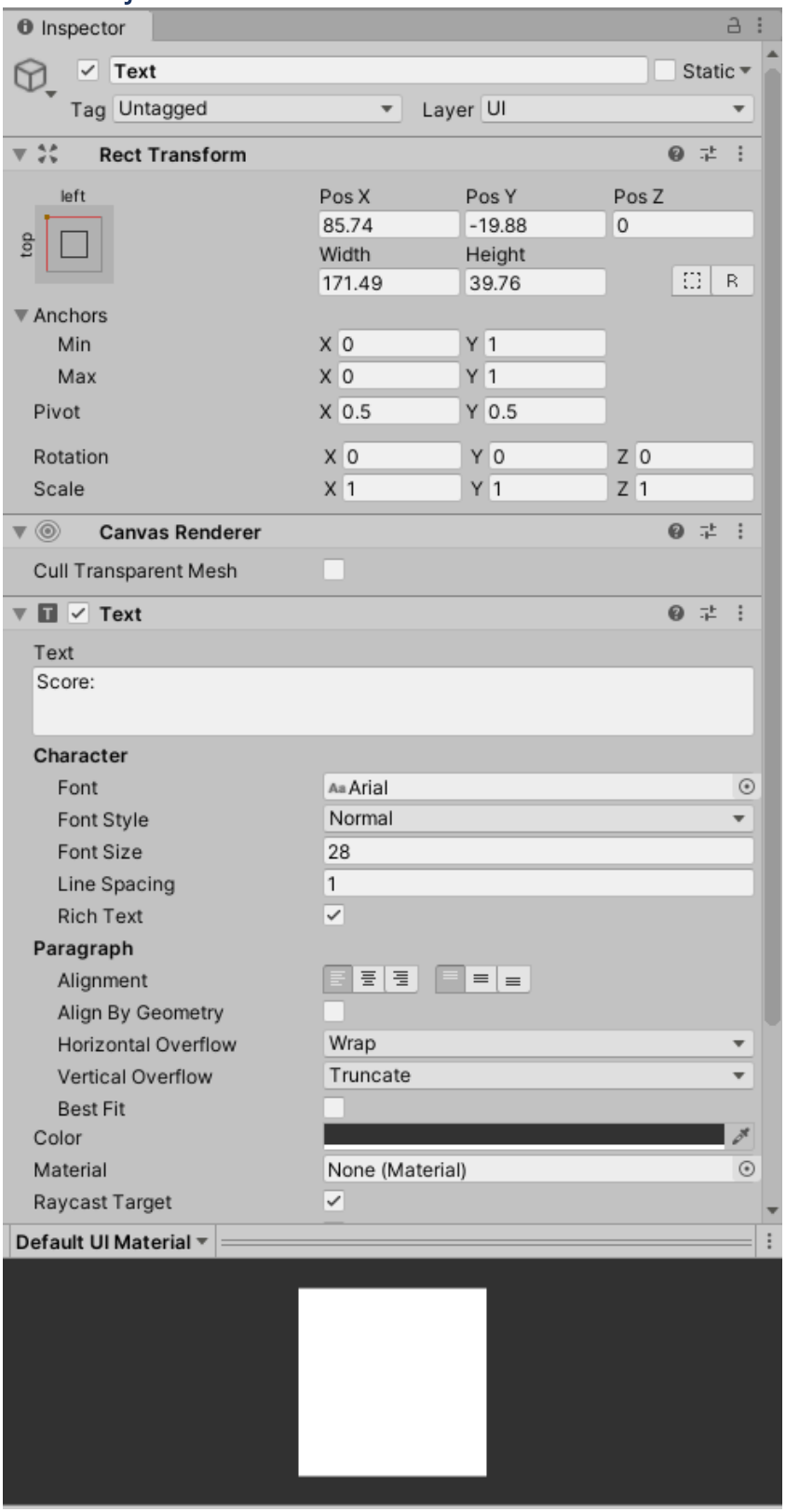
The quad objects will have a different names, positions, and scales based on their locations in the scene.



ScoreSystem Object

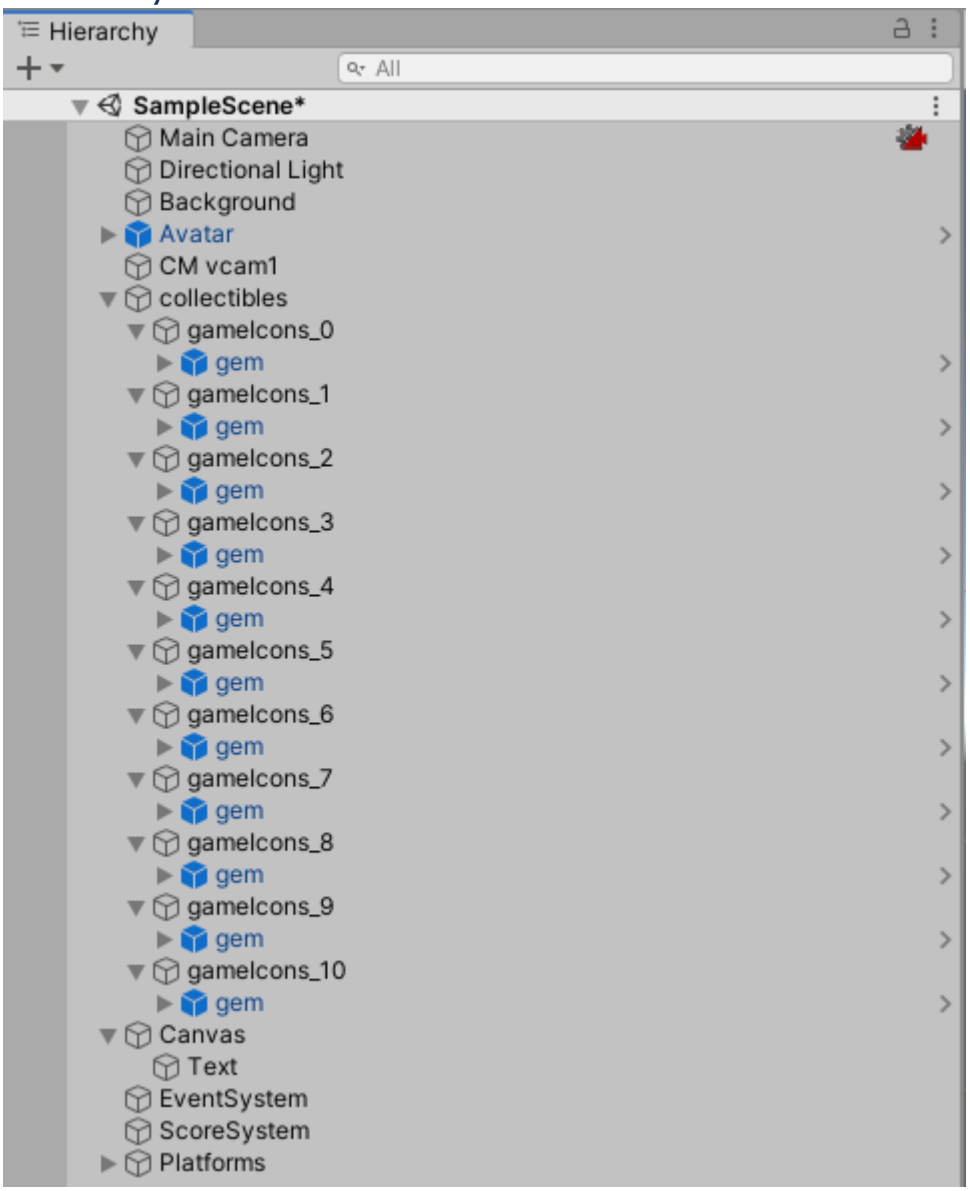


Text object

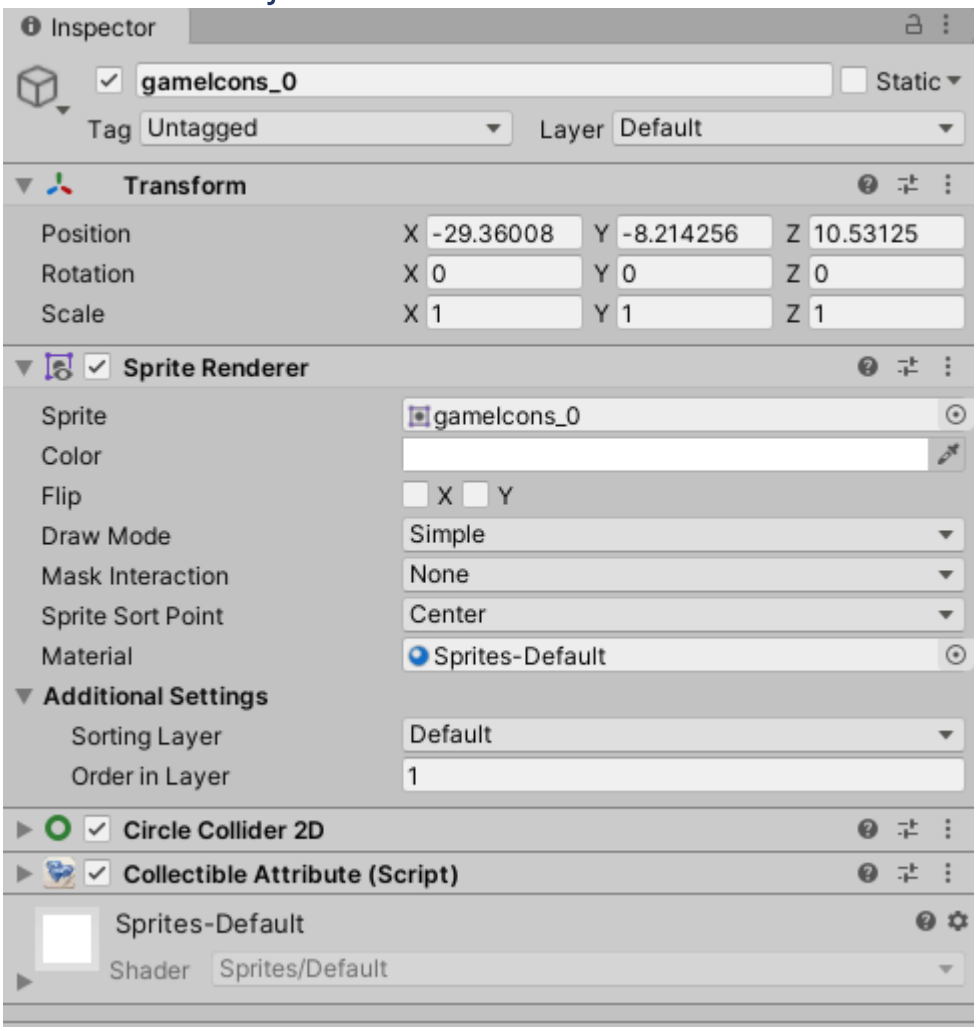


Particle Hunt Prove Yourself

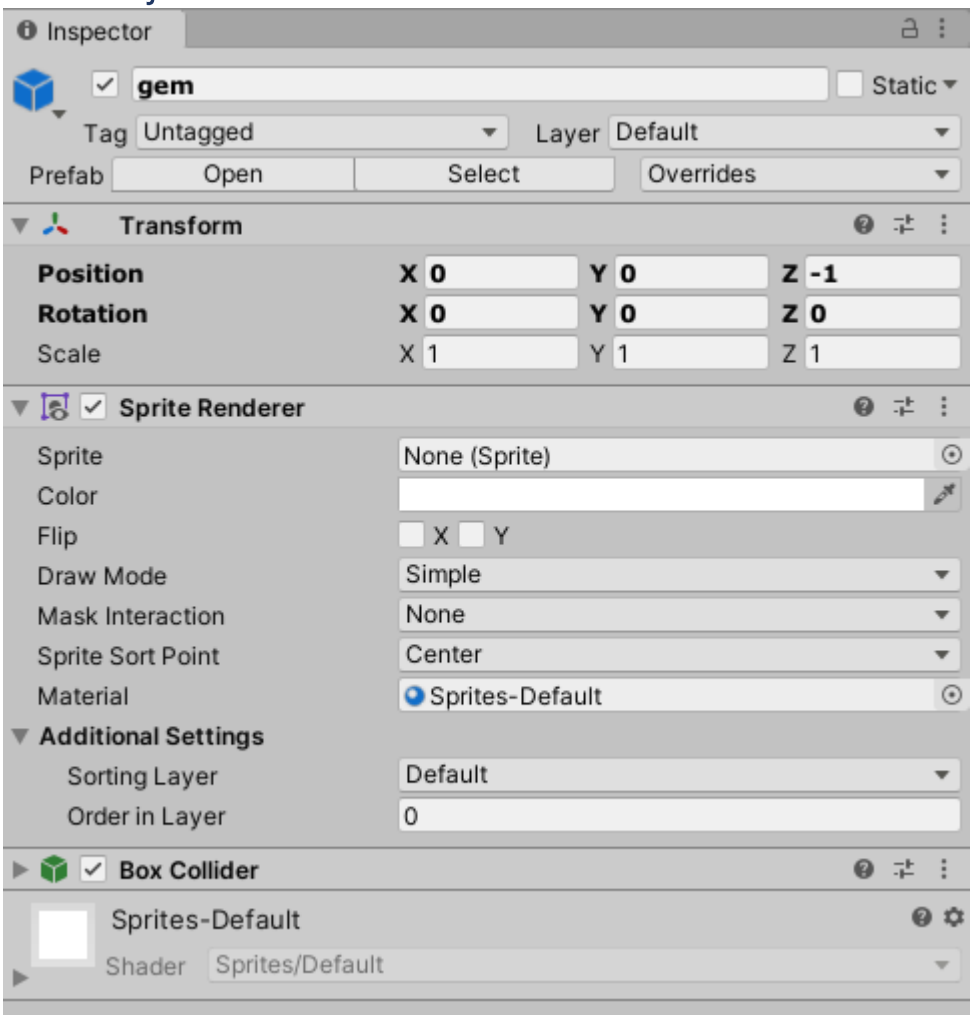
Hierarchy



Game Icons Objects

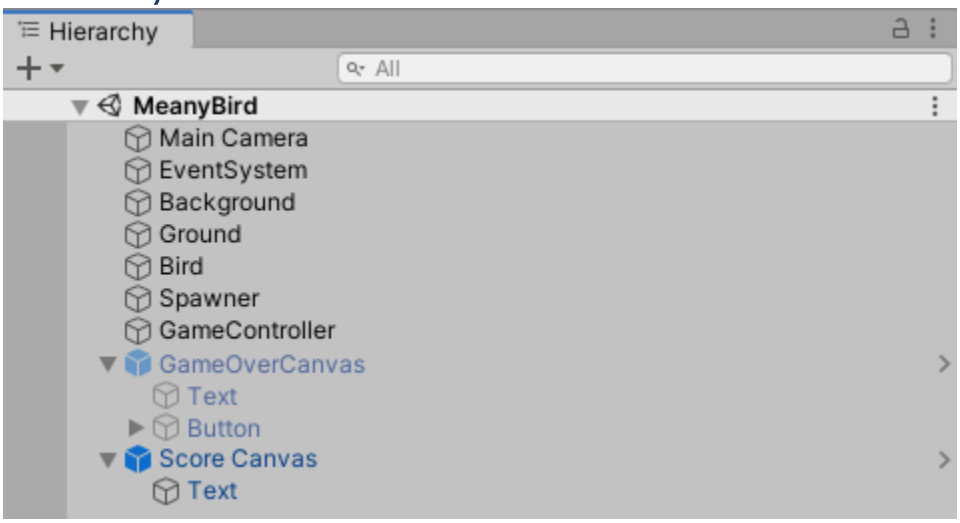


Gem Objects

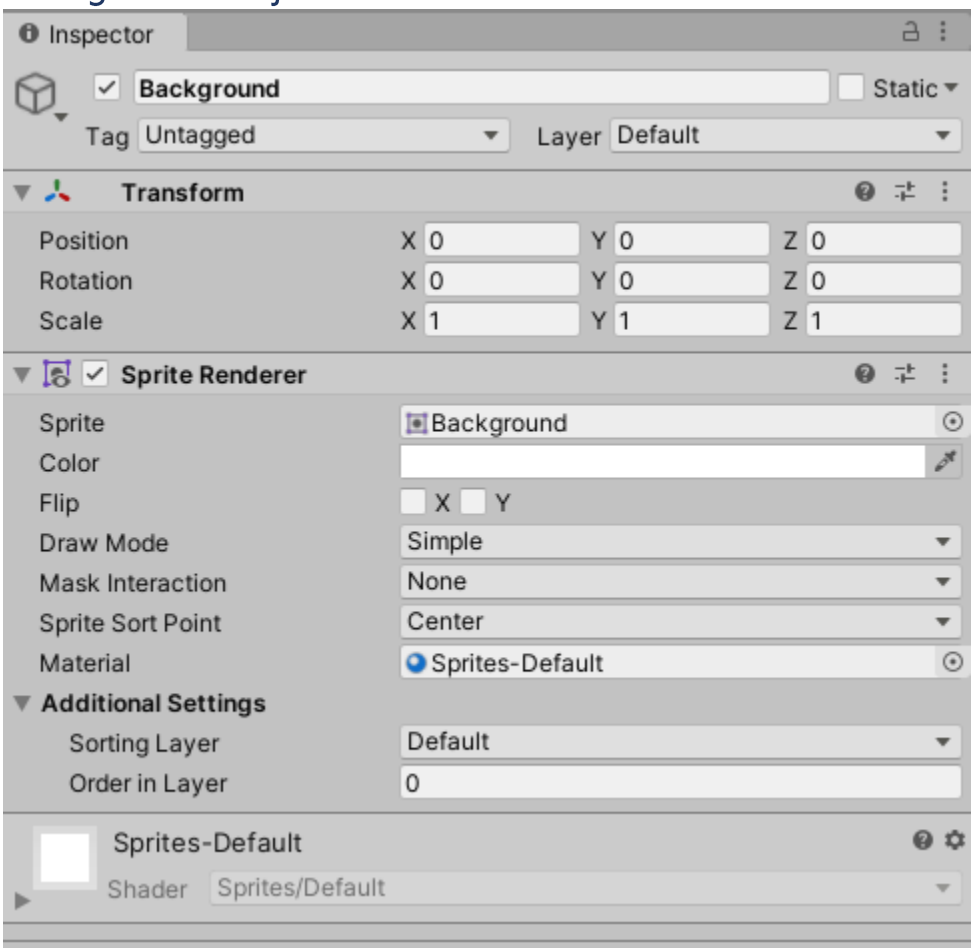


Meany Bird

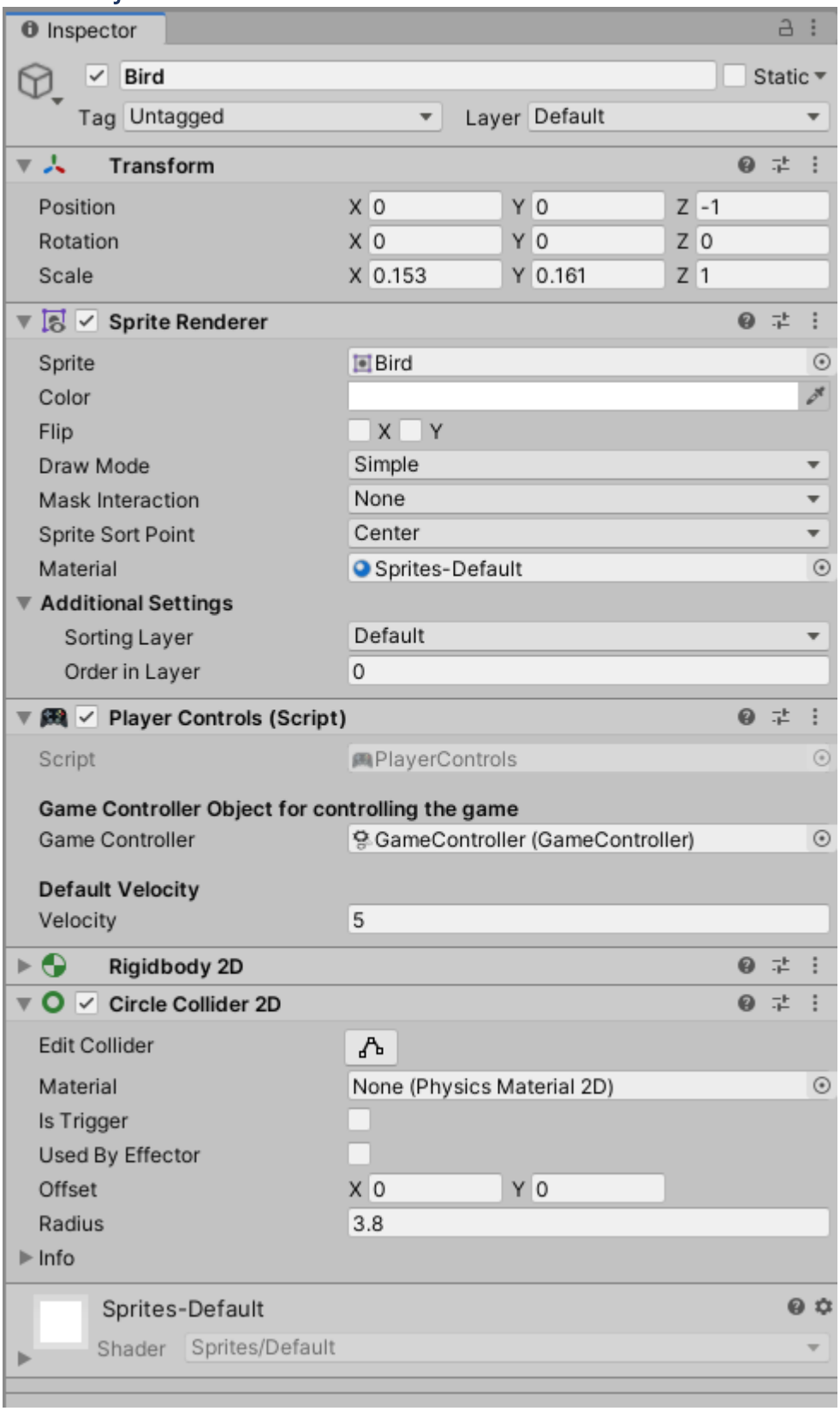
Hierarchy



Background Object



Bird Object



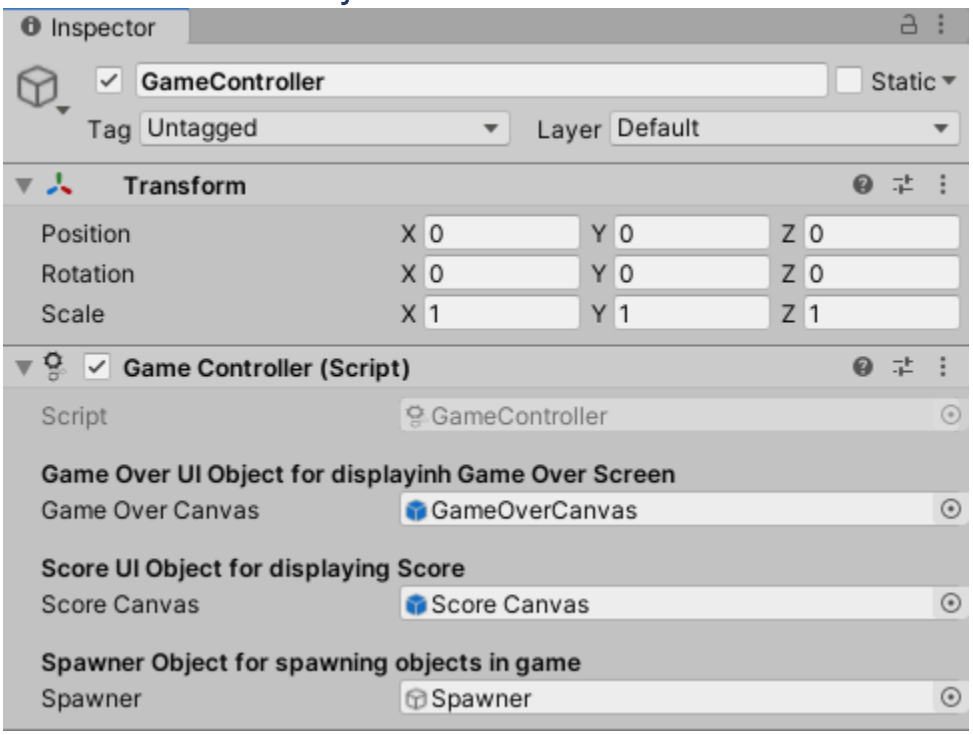
Collectibles.cs Script

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

public class Collectibles : MonoBehaviour
{
    // If an object collides with a trigger
    private void OnTriggerEnter2D(Collider2D collision)
    {
        // Add score
        Score.score++;
    }
}
```

DestroyAfterTime.cs Script

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

public class DestroyAfterTime : MonoBehaviour
{
    [Header("Default Desctrion Time")]
    public float timeToDestruction;
    // Start is called before the first frame update
    void Start()
    {
        Invoke("DestroyObject", timeToDestruction);

    }

    void DestroyObject()
    {
        Destroy(gameObject);
    }
}
```

GameController Object



GameController.cs Script

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

public class GameController : MonoBehaviour
{
    [Header("Game Over UI Object for displayinh Game Over Screen")]
    public GameObject gameOverCanvas;
    [Header("Score UI Object for displaying Score")]
    public GameObject scoreCanvas;
    [Header("Spawner Object for spawning objects in game")]
    public GameObject spawner;

    // Start is called before the first frame update
    void Start()
    {
        Time.timeScale = 1;
        scoreCanvas.SetActive(true);
        gameOverCanvas.SetActive(false);
        spawner.SetActive(true);
    }

    public void GameOver()
    {
        gameOverCanvas.SetActive(true);
        spawner.SetActive(false);
        Time.timeScale = 0;
    }

}
```

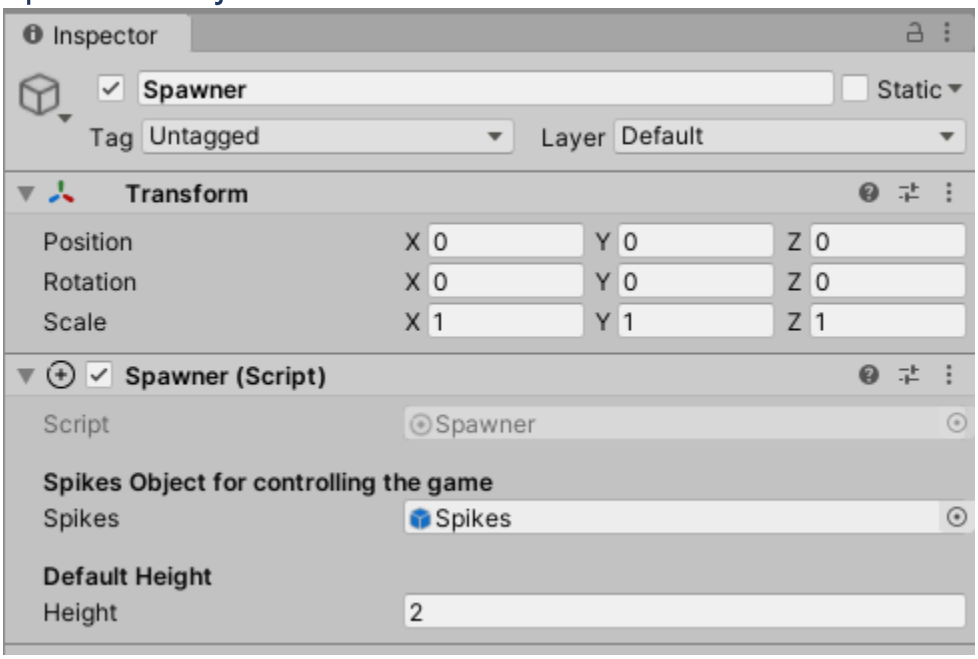
Move.cs Script

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

public class Move : MonoBehaviour
{
    [Header("Default Speed")]
    public float speed;

    // Update is called once per frame
    void Update()
    {
        transform.position += Vector3.left * speed * Time.deltaTime;
    }
}
```

PlayerControls.cs Script

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

public class PlayerControls : MonoBehaviour
{
    [Header("Game Controller Object for controlling the game")]
    public GameController gameController;
    [Header("Default Velocity")]
    public float velocity = 1;
    private Rigidbody2D rb;
    private float objectHeight;

    // Start is called before the first frame update
    void Start()
    {
        gameController = GetComponent<GameController>();
        Time.timeScale = 1;
        rb = GetComponent<Rigidbody2D>();
        objectHeight = transform.GetComponent<SpriteRenderer>().bounds.size.y / 2;

    }

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

    }
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.tag == "HighSpike"
            || collision.gameObject.tag == "LowSpike"
            || collision.gameObject.tag == "Ground") {
            GameObject.Find("GameController").GetComponent<GameController>().GameOver();
        }
    }
}
```

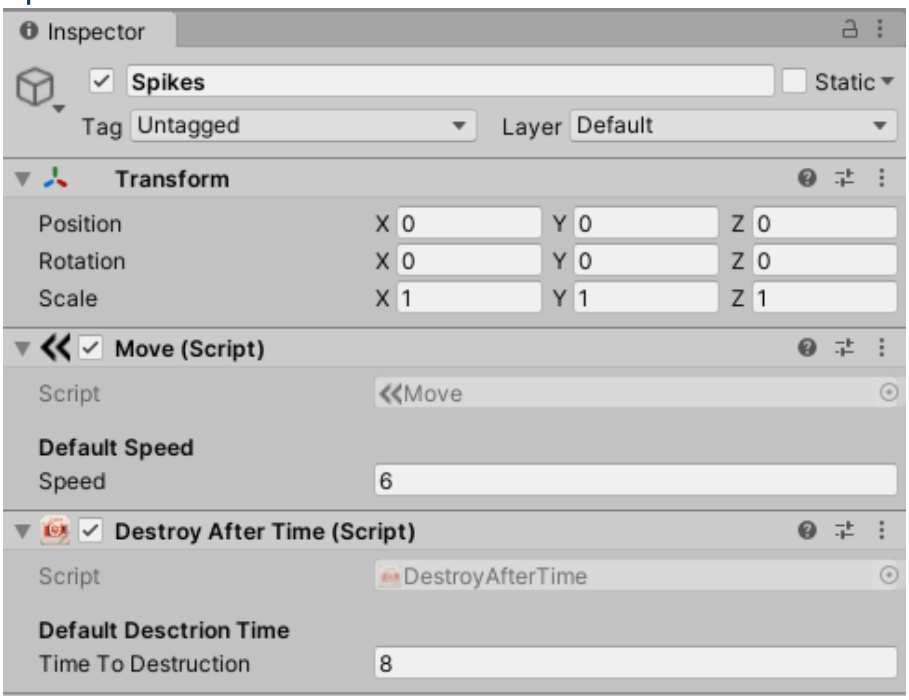
Spawner Object



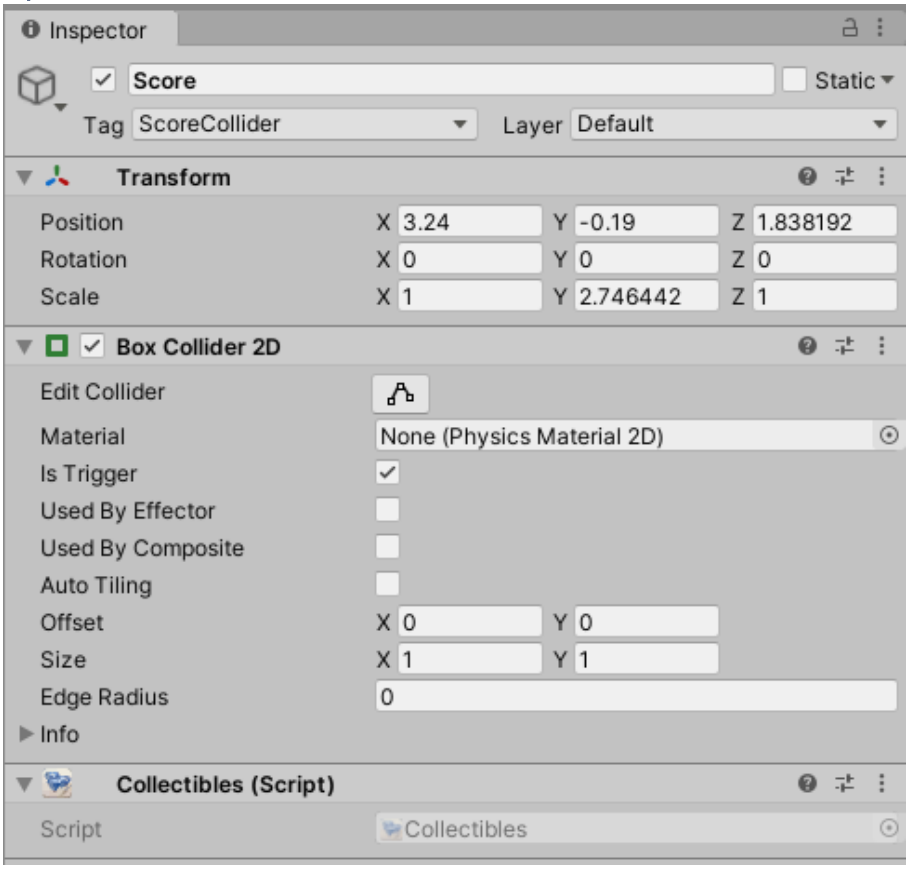
Spawner.cs Script

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

public class Spawner : MonoBehaviour
{
    [Header("Spikes Object for controlling the game")]
    public GameObject spikes;
    [Header("Default Height")]
    public float height;
    // Start is called before the first frame update
    void Start()
    {
        InvokeRepeating("InstantiateObjects", 1f, 4f);

    }

    // Update is called once per frame
    void Update()
    {
        transform.position = new Vector3(10, Random.Range(-height, height), 0);

    }

    void InstantiateObjects()
    {
        Instantiate(spikes, transform.position, transform.rotation);
    }
}
```

Spikes Prefab

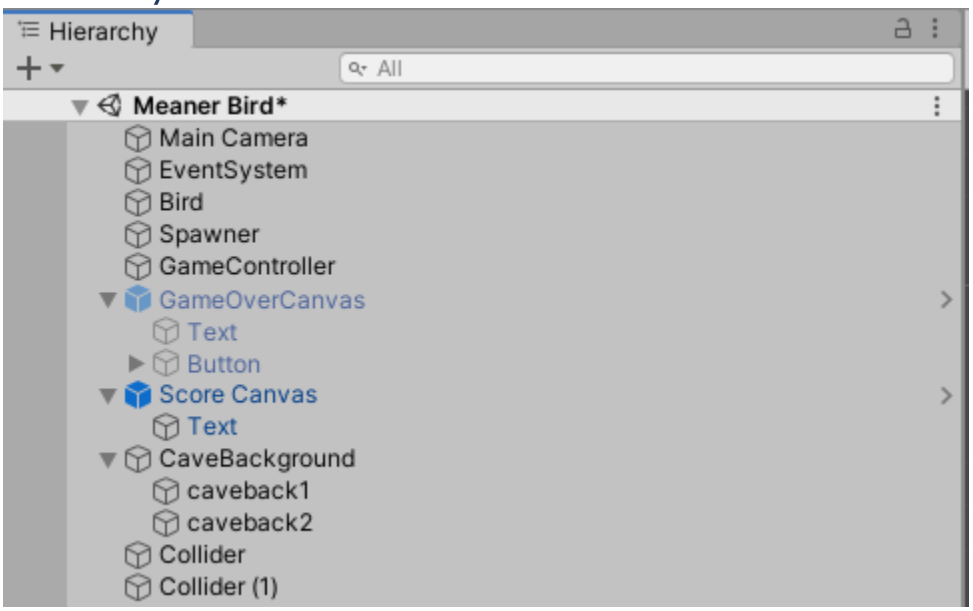


Spikes > Score Prefab

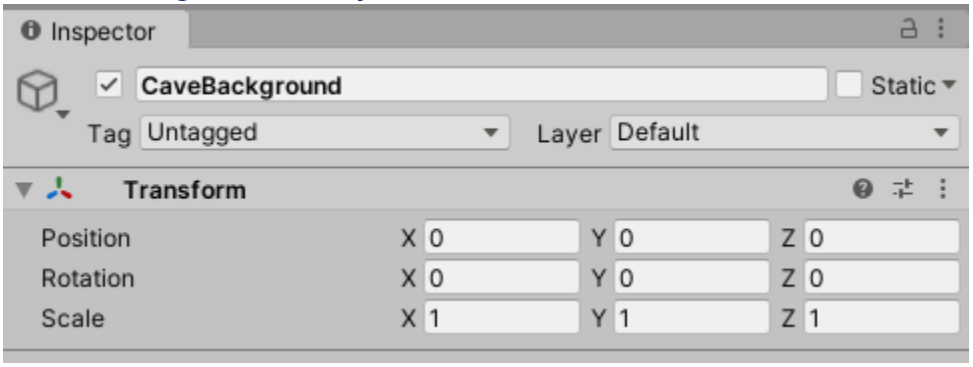


Meaner Bird Prove Yourself

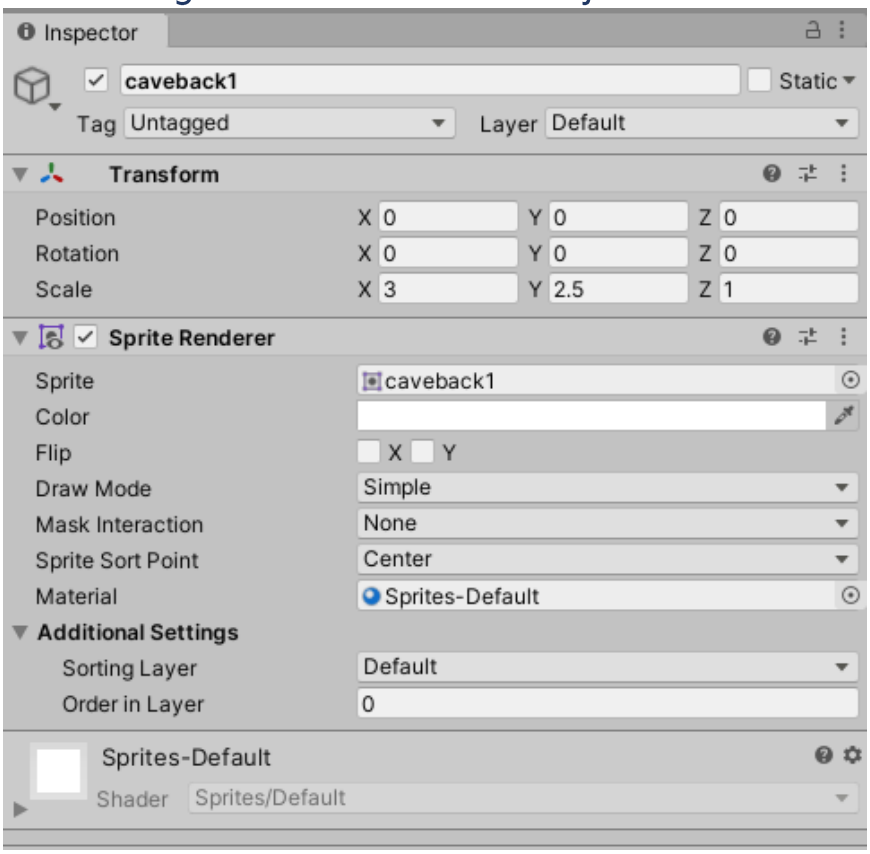
Hierarchy



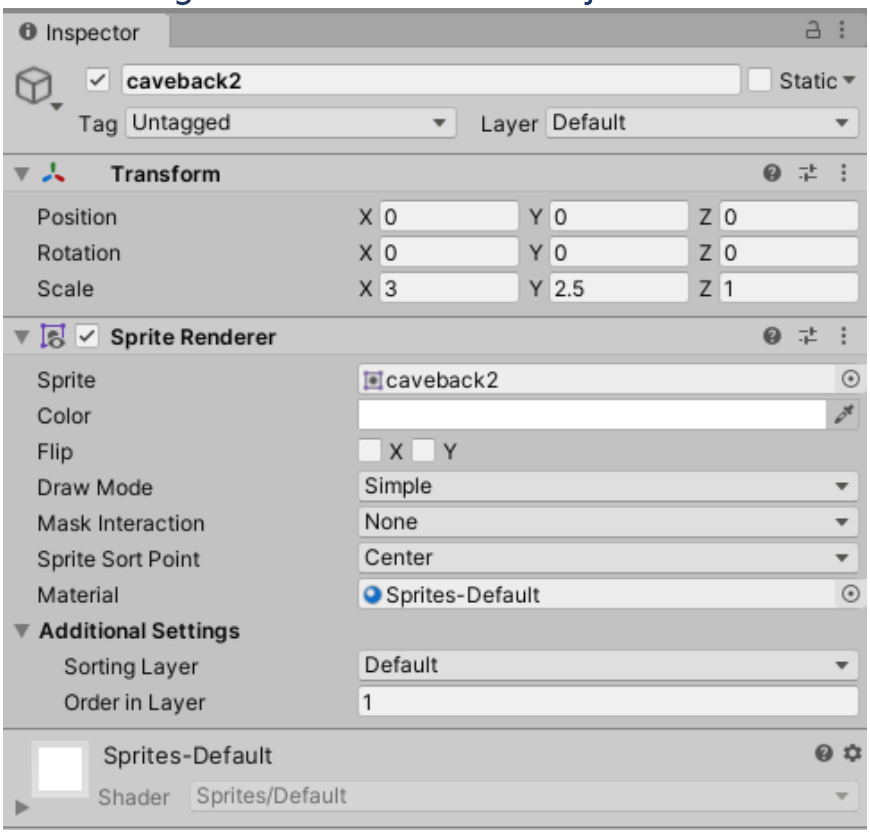
CaveBackground Object



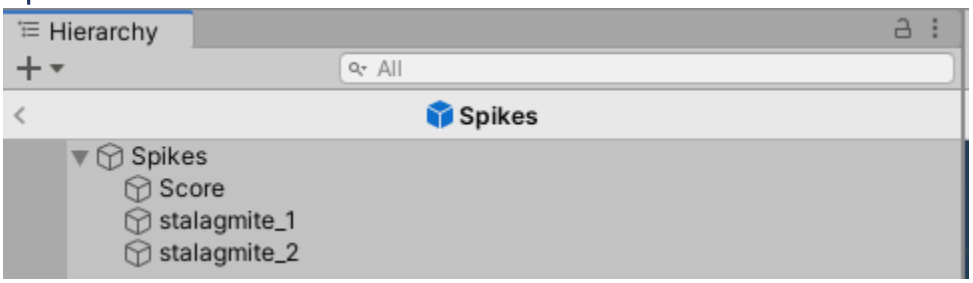
CaveBackground > caveback1 Object



CaveBackground > caveback2 Object

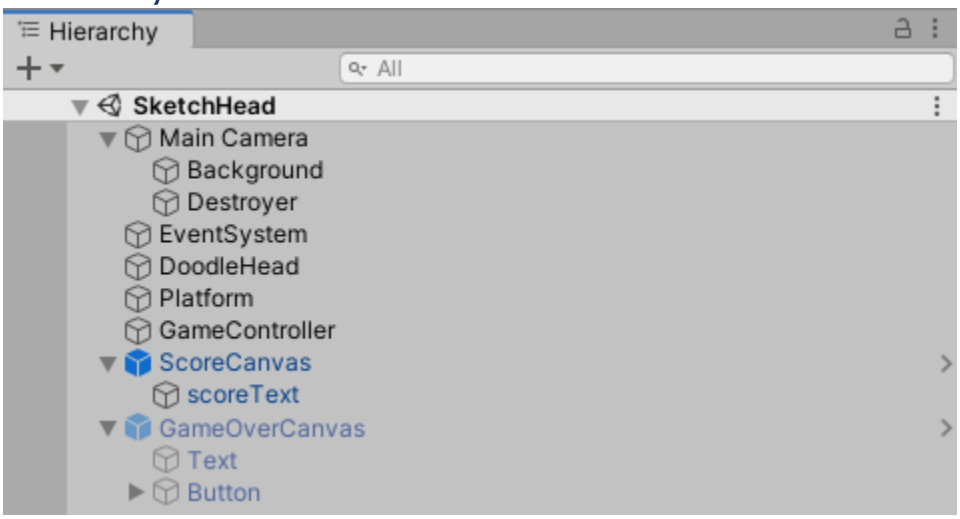


Spikes Prefab



Sketch Head

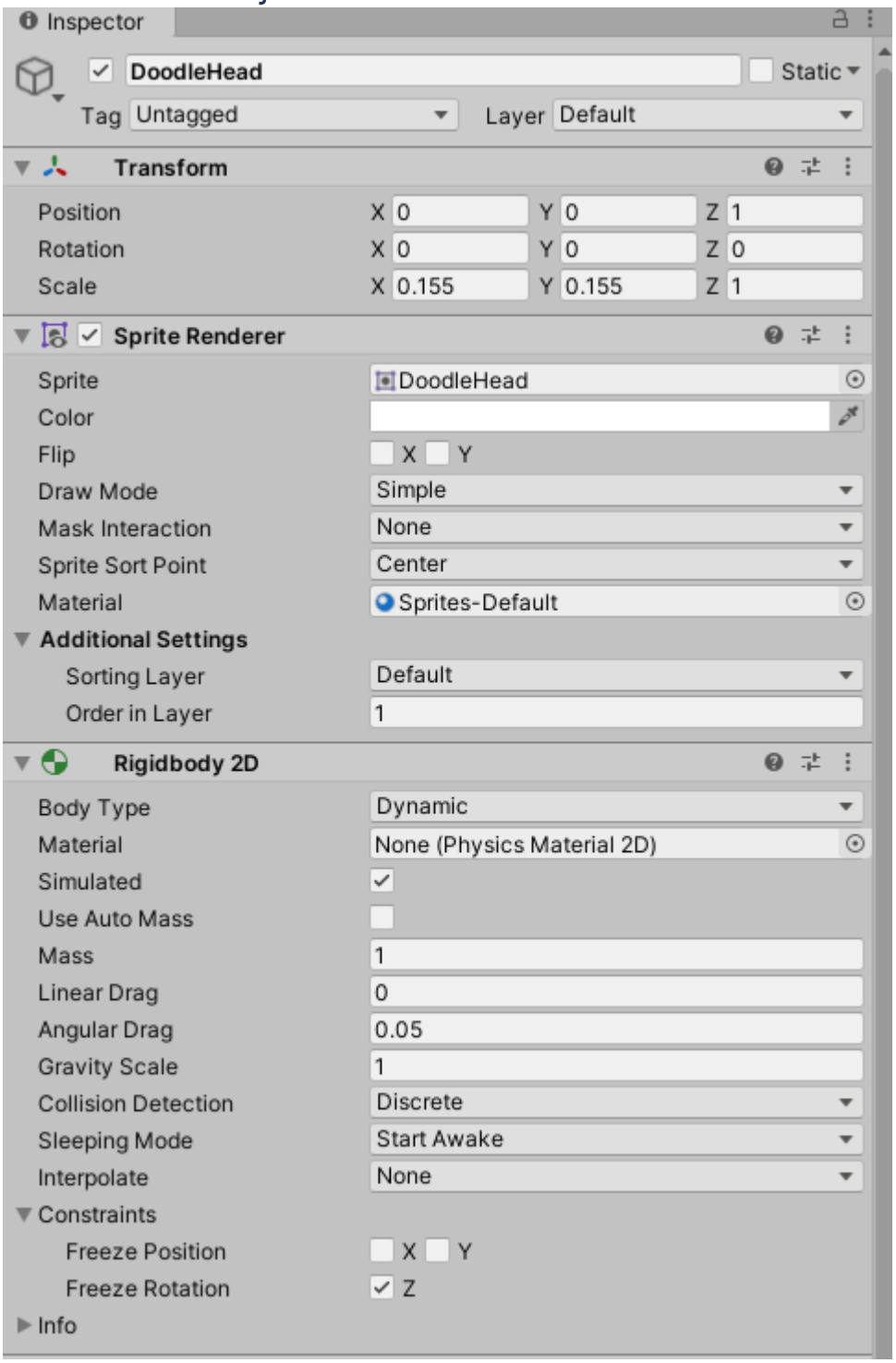
Hierarchy



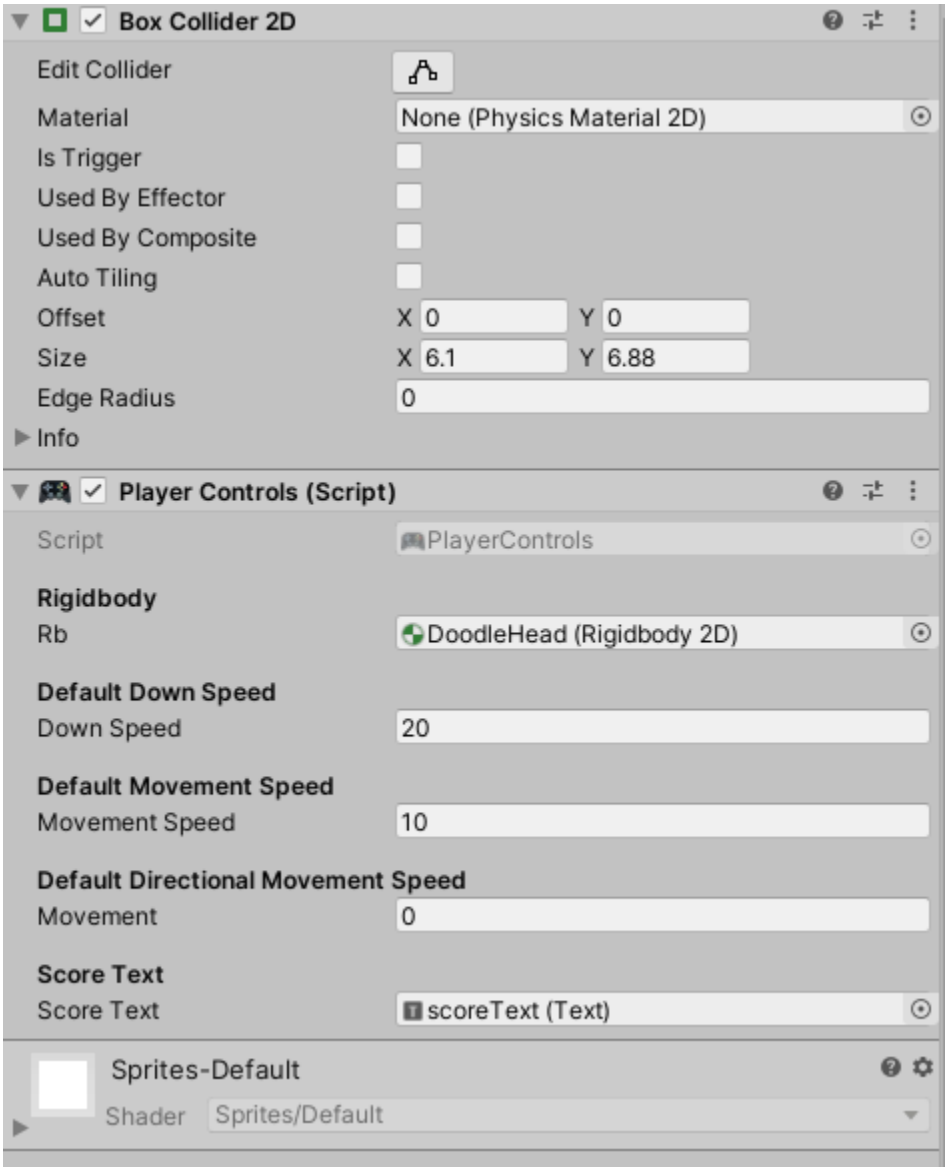
Destroyer.cs Script

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

public class Destroyer : MonoBehaviour
{
    public void OnTriggerEnter2D(Collider2D collision)
    {
        GameObject.Find("DoodleHead").SetActive(false);
        GameObject.Find("GameController").GetComponent<GameController>().GameOver();
    }
}
```

DoodleHead Object



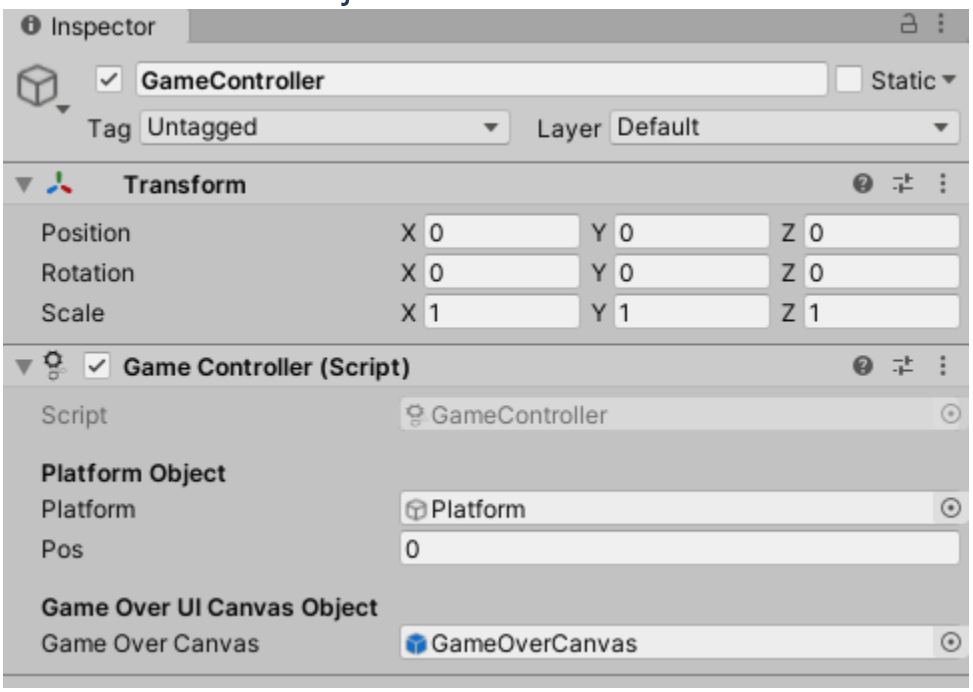
DoodleHead object continued:



CameraFollow.cs Script

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

public class CameraFollow : MonoBehaviour
{
    [Header("Target Object")]
    public Transform target;

    // Update is called once per frame
    private void Update()
    {
        if (target.position.y > transform.position.y)
        {
            transform.position = new Vector3(
                target.transform.position.x,
                target.transform.position.y,
                transform.position.z);
        }
    }
}
```

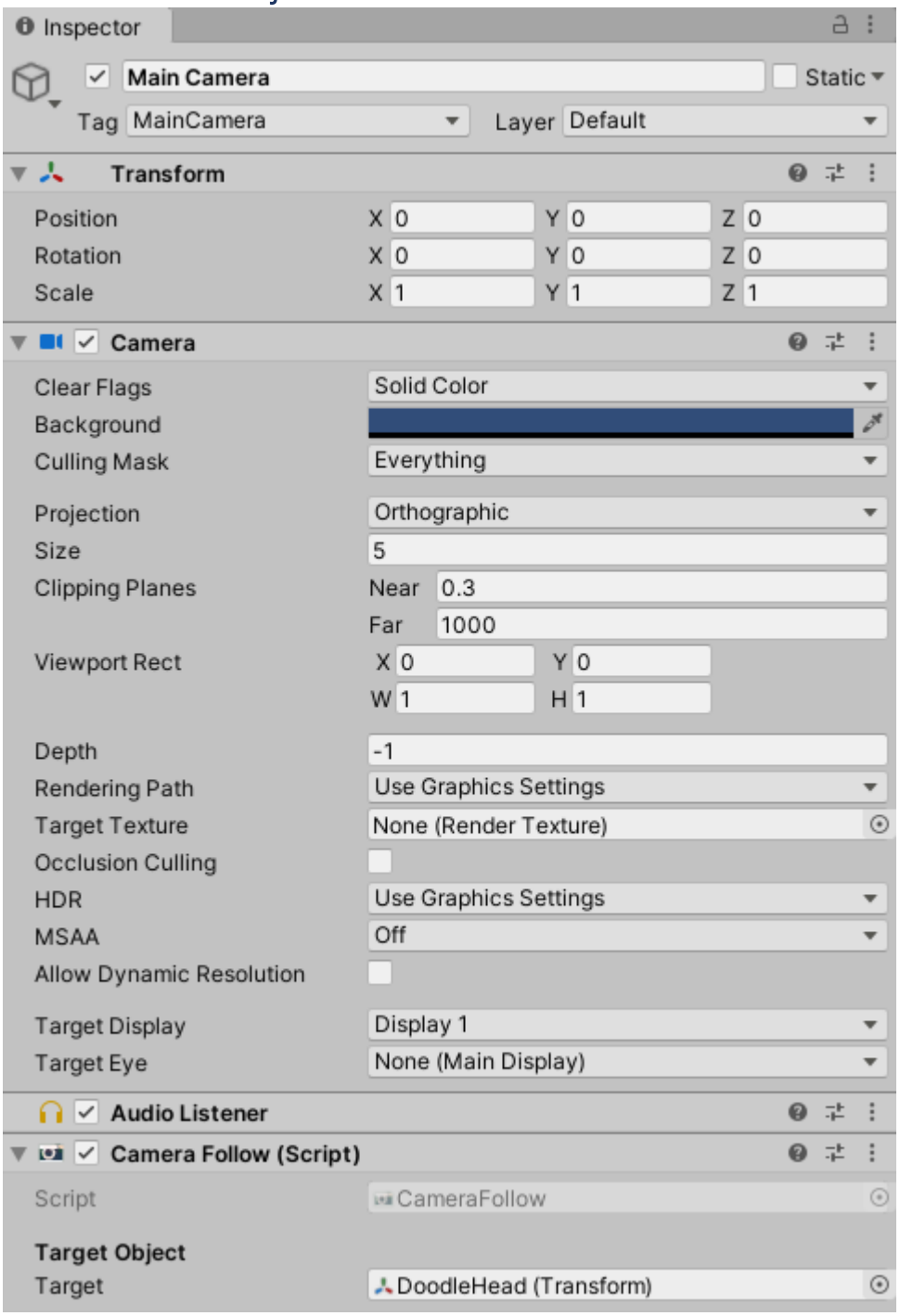
GameController Object



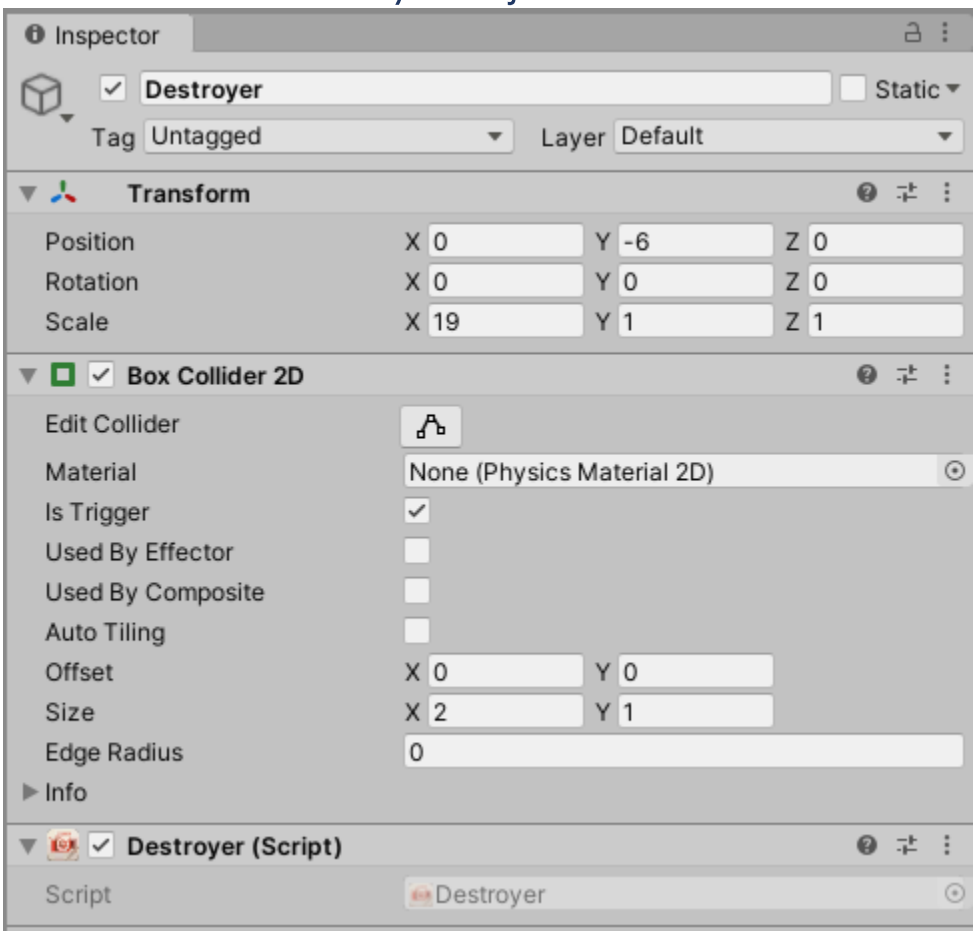
GameController.cs Script

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

public class GameController : MonoBehaviour
{
    [Header("Platform Object")]
    public GameObject platform;
    public float pos = 0;
    [Header("Game Over UI Canvas Object")]
    public GameObject gameOverCanvas;

    void Start()
    {
        for (int i = 0; i < 1000; i++)
        {
            SpawnPlatforms();
        }
    }

    void SpawnPlatforms()
    {
        Instantiate(platform, new Vector3(
            Random.value * 10 - 5f, pos, 0.5f),
            Quaternion.identity);
        pos += 5f;
    }

    public void GameOver()
    {
        gameOverCanvas.SetActive(true);
    }
}
```

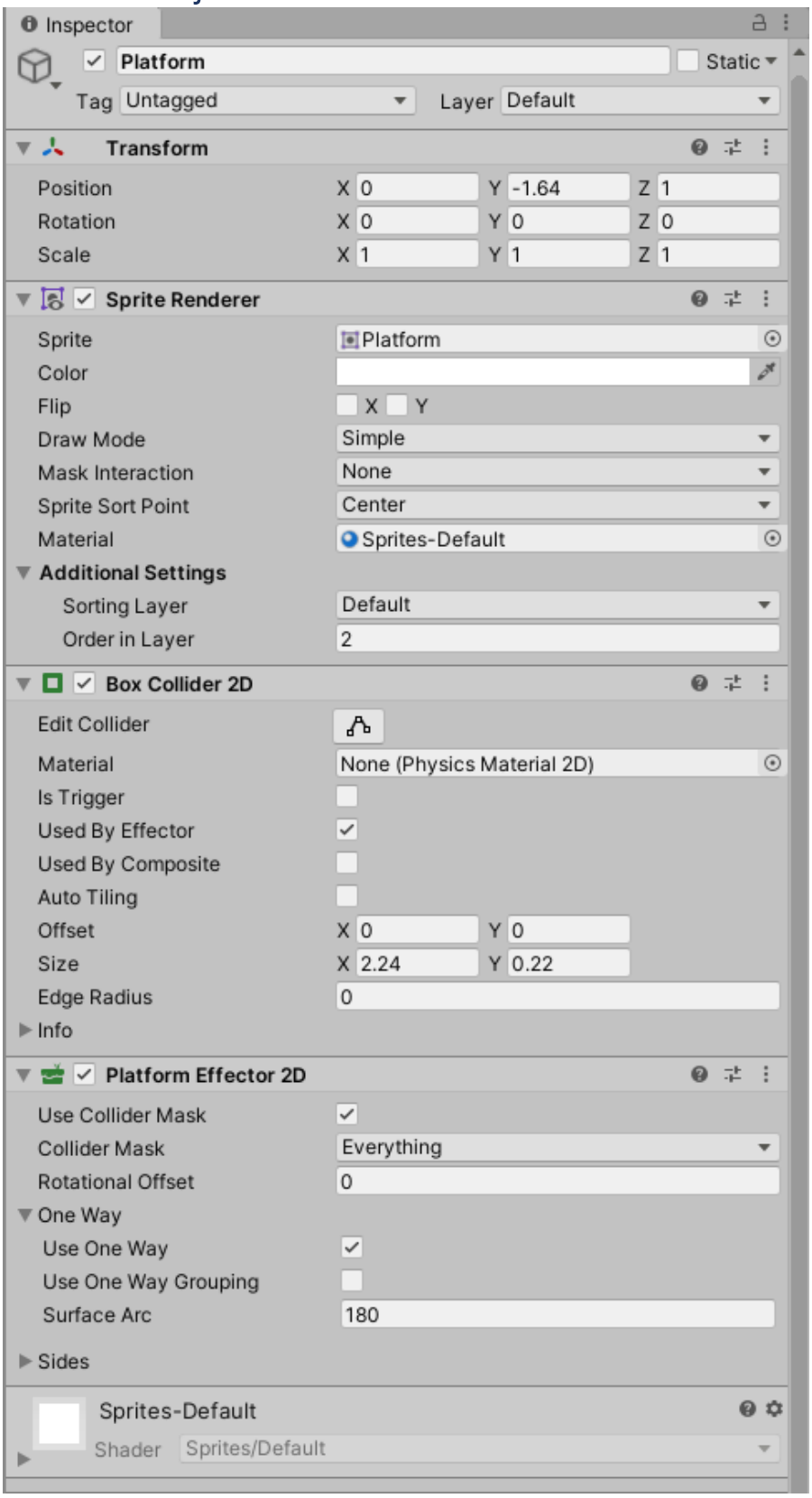
Main Camera Object



Main Camera > Destroyer Object



Platform Object

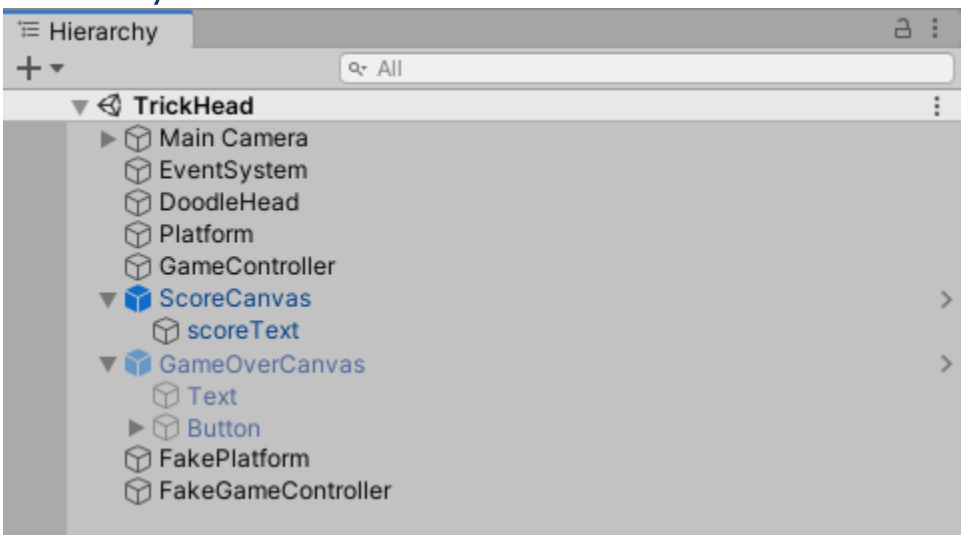


PlayerControls.cs Script

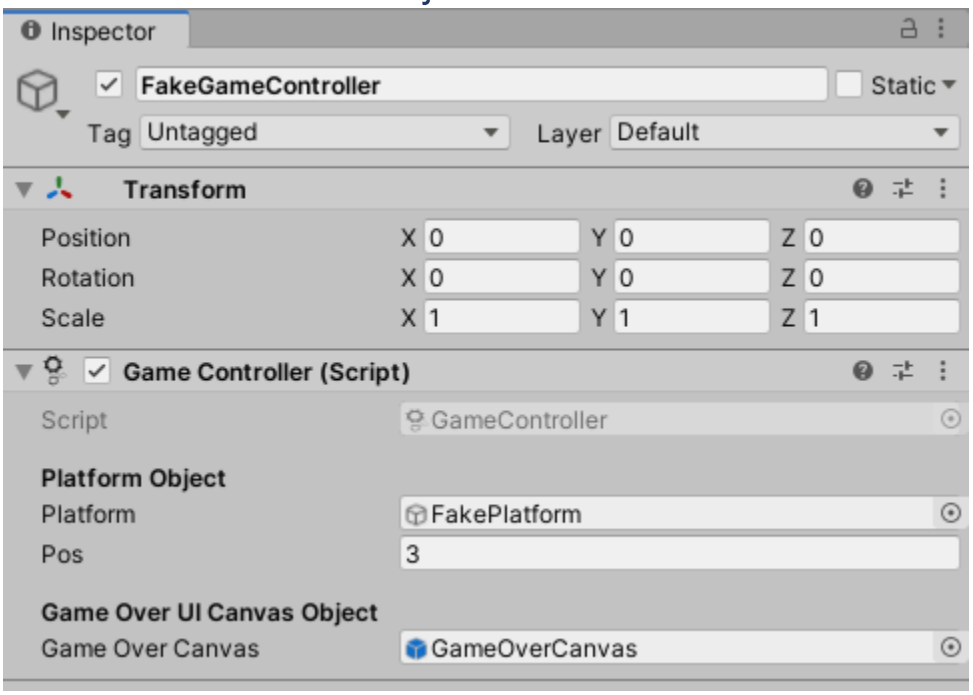
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class PlayerControls : MonoBehaviour
{
    [Header("Rigidbody")]
    public Rigidbody2D rb;
    [Header("Default Down Speed")]
    public float downSpeed = 20f;
    [Header("Default Movement Speed")]
    public float movementSpeed = 10f;
    [Header("Default Directional Movement Speed")]
    public float movement = 0f;
    [Header("Score Text")]
    public Text scoreText;
    private float topScore = 0.0f;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    void Update()
    {
        movement = Input.GetAxis("Horizontal") * movementSpeed;
        if (movement < 0)
        {
            this.GetComponent<SpriteRenderer>().flipX = false;
        }
        else
        {
            this.GetComponent<SpriteRenderer>().flipX = true;
        }
        if (rb.velocity.y > 0 && transform.position.y > topScore)
        {
            topScore = transform.position.y;
        }
        scoreText.text = "Score: " + Mathf.Round(topScore).ToString();
    }

    void FixedUpdate()
    {
        Vector2 velocity = rb.velocity;
        velocity.x = movement;
        rb.velocity = velocity;
    }

    private void OnCollisionEnter2D(Collision2D collision)
    {
        rb.velocity = new Vector3(rb.velocity.x, downSpeed, 0);
    }
}
```

Trick Head Prove Yourself

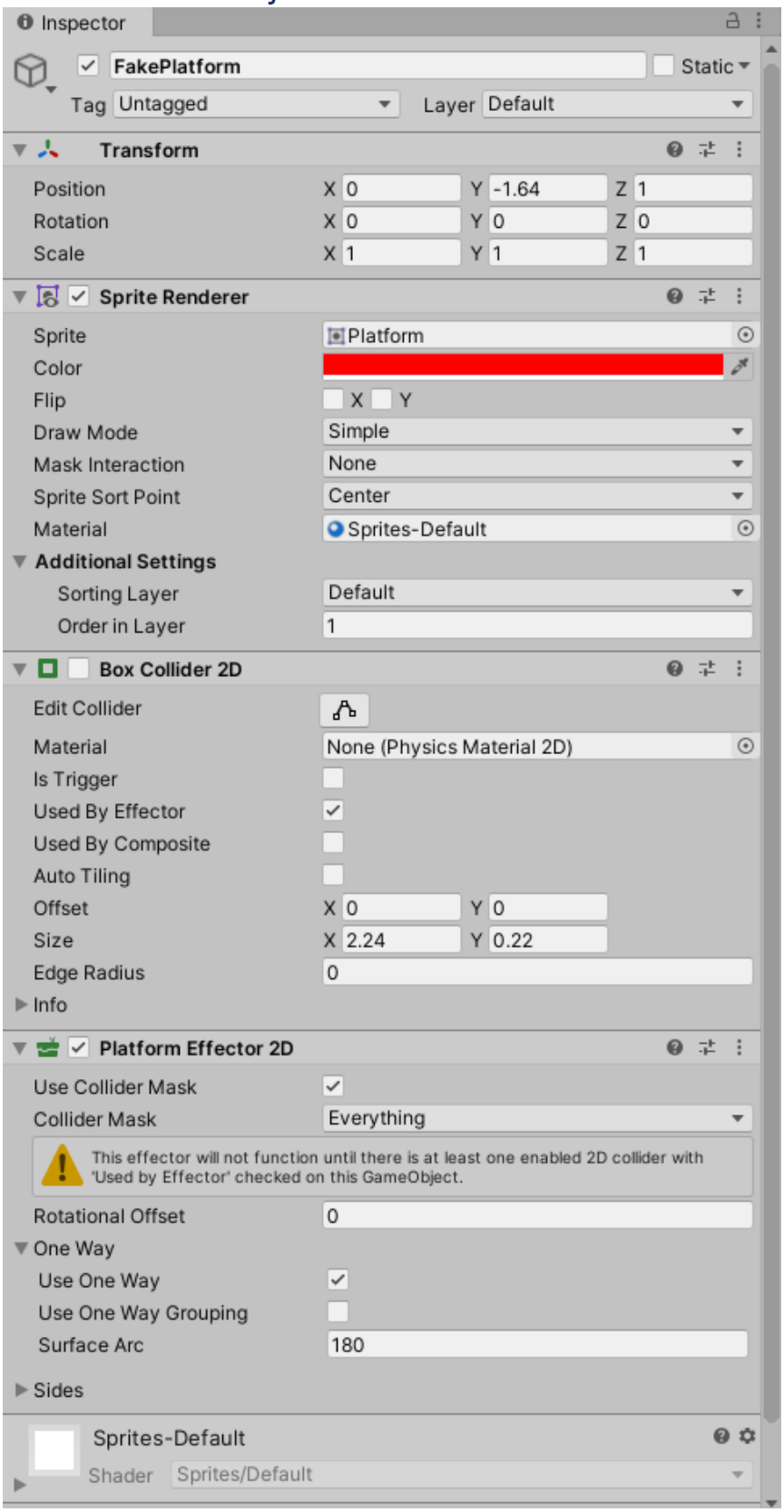
Hierarchy



FakeGameController Object

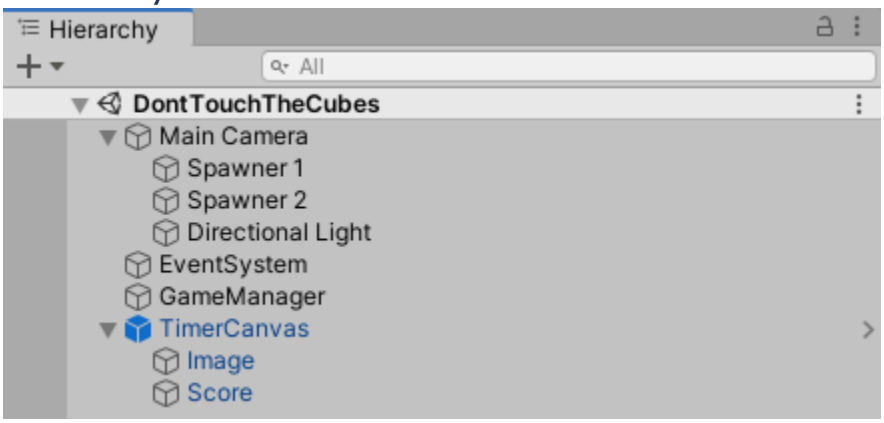


FakePlatform Object



Don't Touch the Cubes

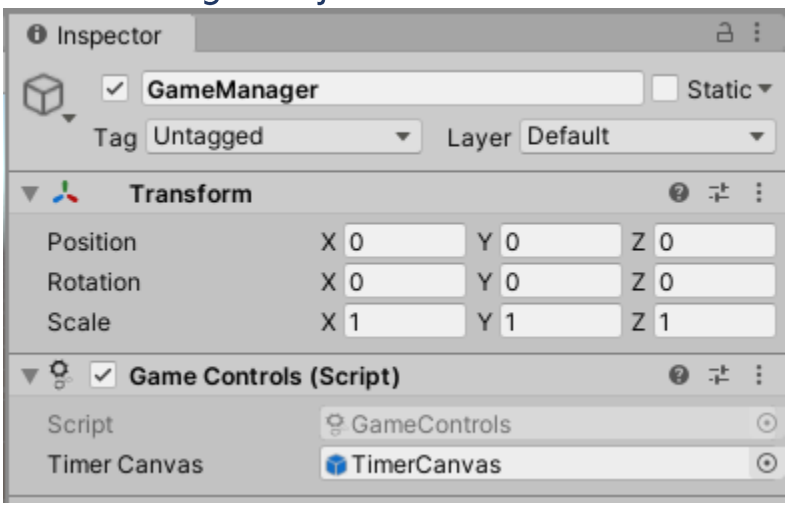
Hierarchy



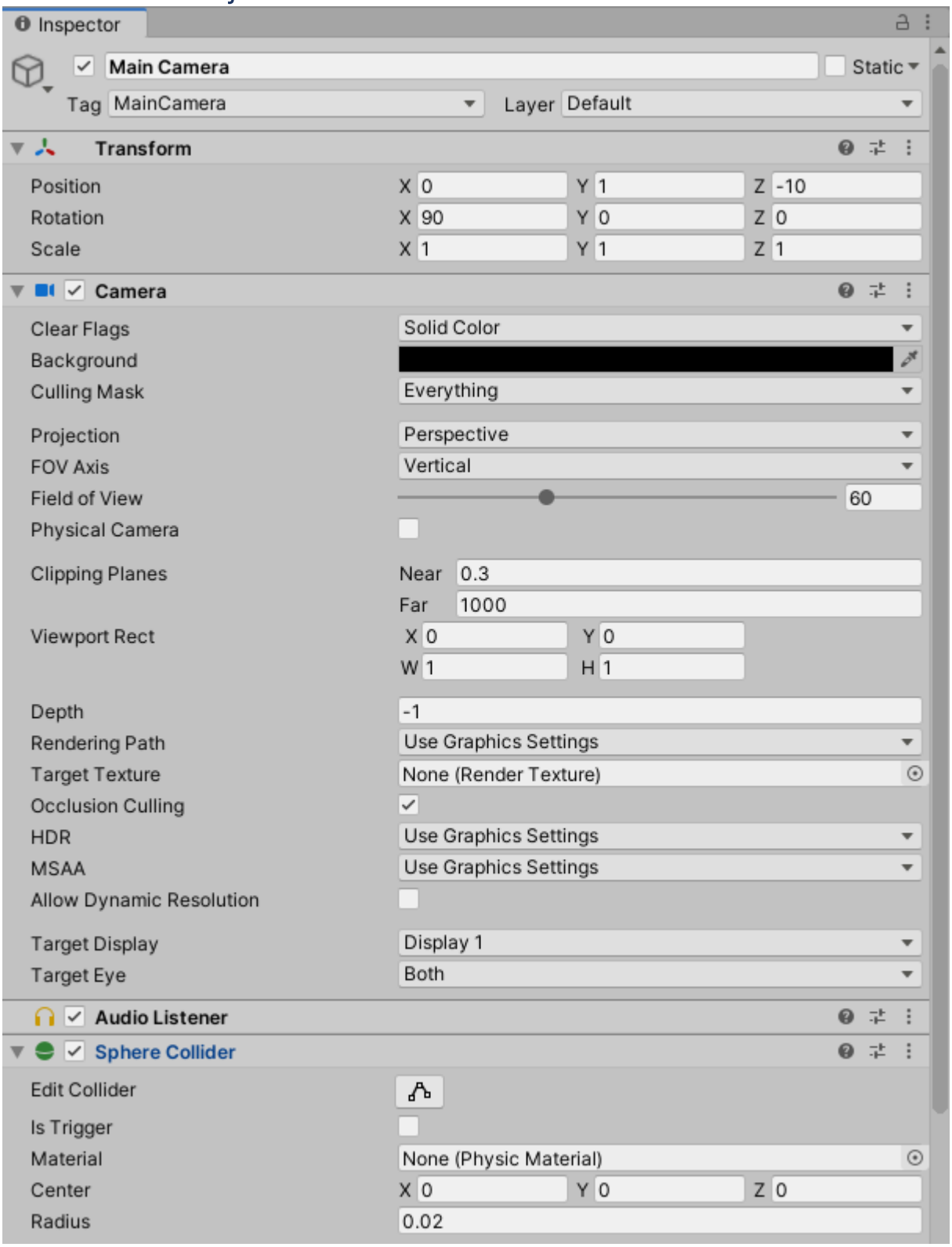
GameControls.cs Script

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

public class GameControls : MonoBehaviour
{
    public GameObject timerCanvas;
    private Text timerText;
    private int timerCount;

    void Start()
    {
        timerCanvas.SetActive(true);
        StartCoroutine(CountTime());
        Time.timeScale = 1f;
        timerText = GameObject.Find("Score").GetComponent<Text>();
    }

    IEnumerator CountTime()
    {
        // add one point every second
        yield return new WaitForSeconds(1f);
        timerCount++;
        timerText.text = "Score: " + timerCount;
        StartCoroutine(CountTime());
    }
}
```

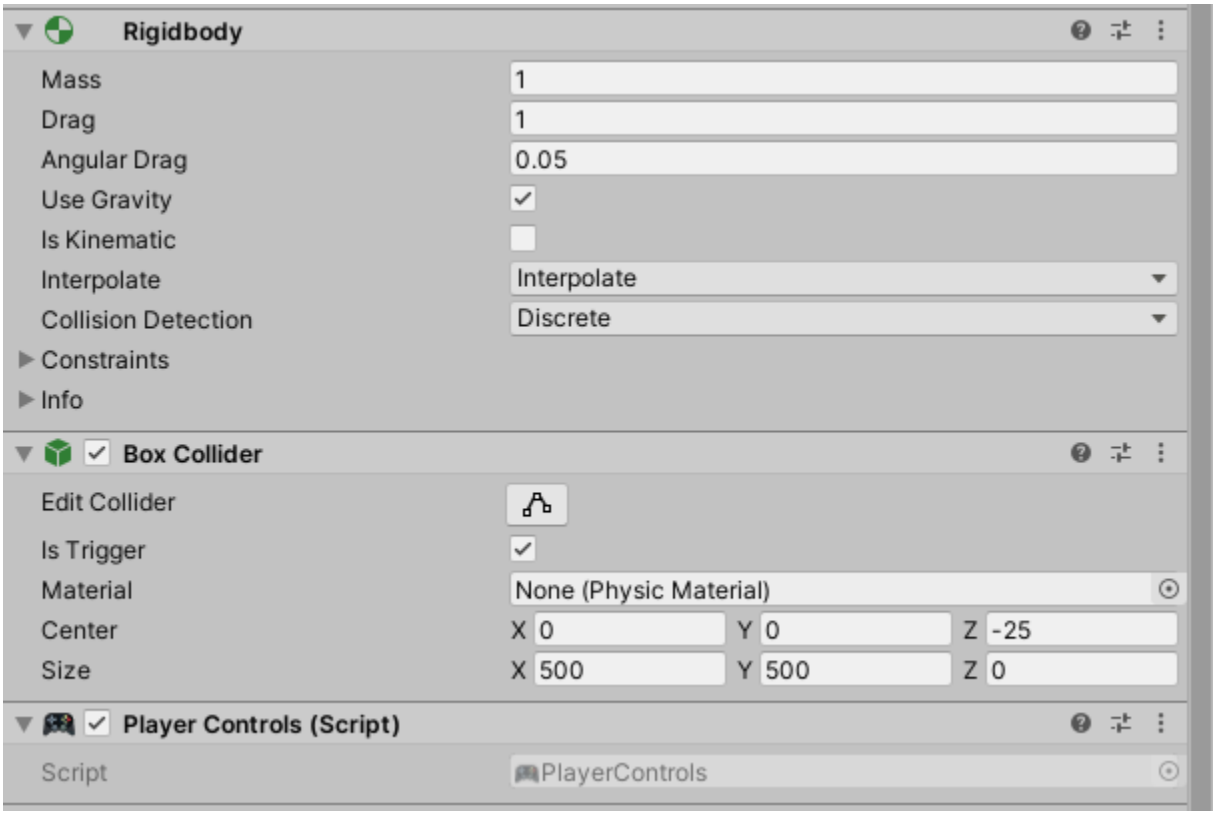
GameManager Object



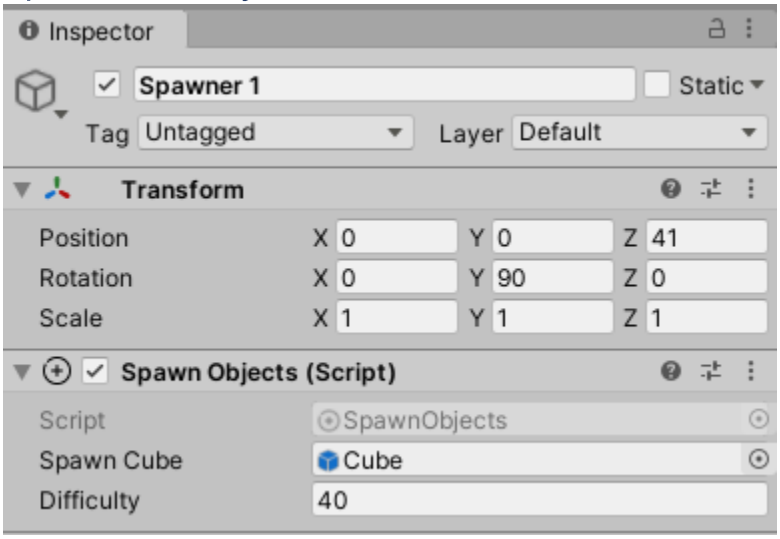
Main Camera Object



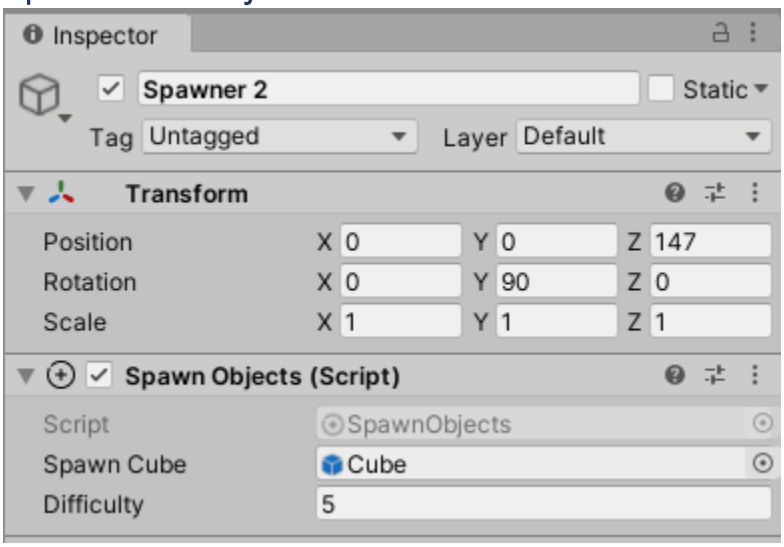
Main Camera Object Continued



Spawner 1 Object



Spawner 2 Object

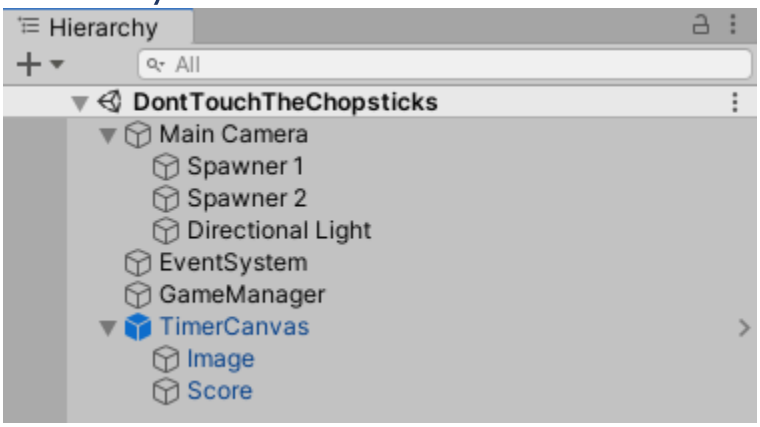


SpawnObjects.cs Script

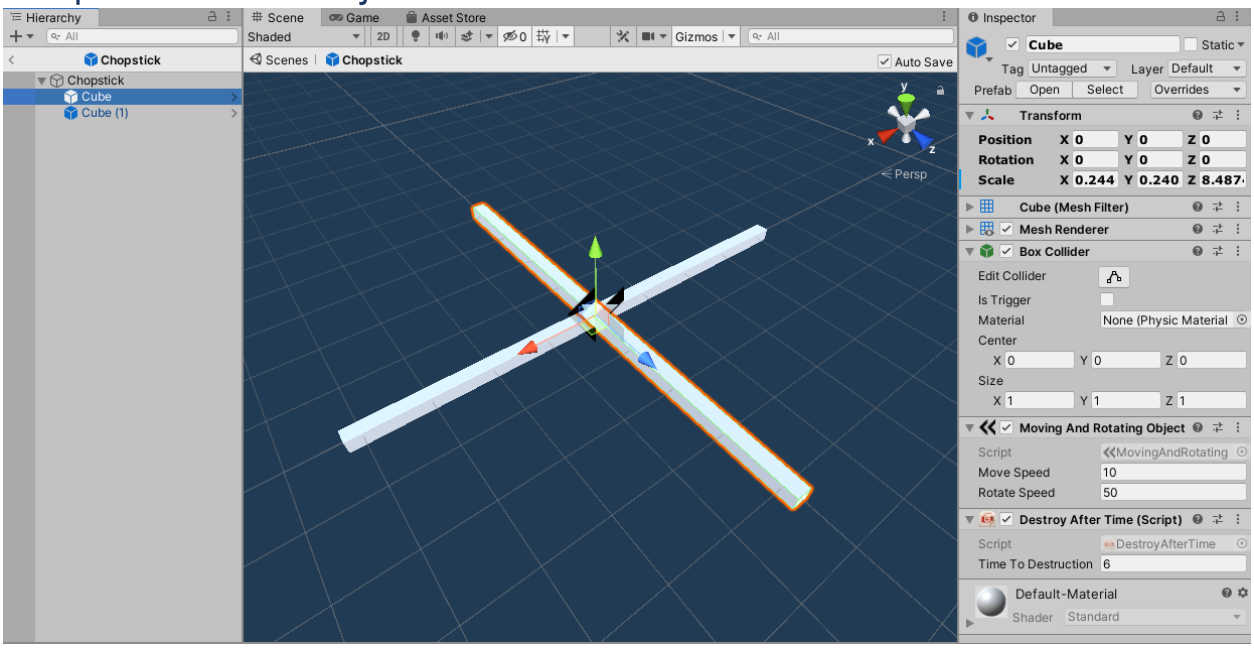
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SpawnObjects : MonoBehaviour
{
    public GameObject spawnCube;
    public float difficulty = 40f;
    // time delay between spawns
    float spawn;

    void Update()
    {
        // The next cube to be spawn will
        // be based on the difficulty and game speed
        spawn = difficulty * Time.deltaTime;

        // the difficulty of the game is based on
        // the speed of the game times 4
        difficulty = Time.deltaTime * 4f;

        // the while loop will spawn cubes
        while (spawn > 0)
        {
            // subtract the spawn time by one
            spawn -= 1;

            // give the cubes random x and z positions
            Vector3 v3Pos = transform.position + new Vector3(
                Random.value * 40f - 20f,
                0,
                Random.value * 40f - 20f
                );

            // give the cubes random x and z rotations
            Quaternion qRotation = Quaternion.Euler(
                0,
                Random.value * 360f,
                Random.value * 30f
                );

            // create the object
            GameObject createObject = Instantiate(spawnCube, v3Pos, qRotation);
        }
    }
}
```

Don't Touch the Chopsticks Prove Yourself

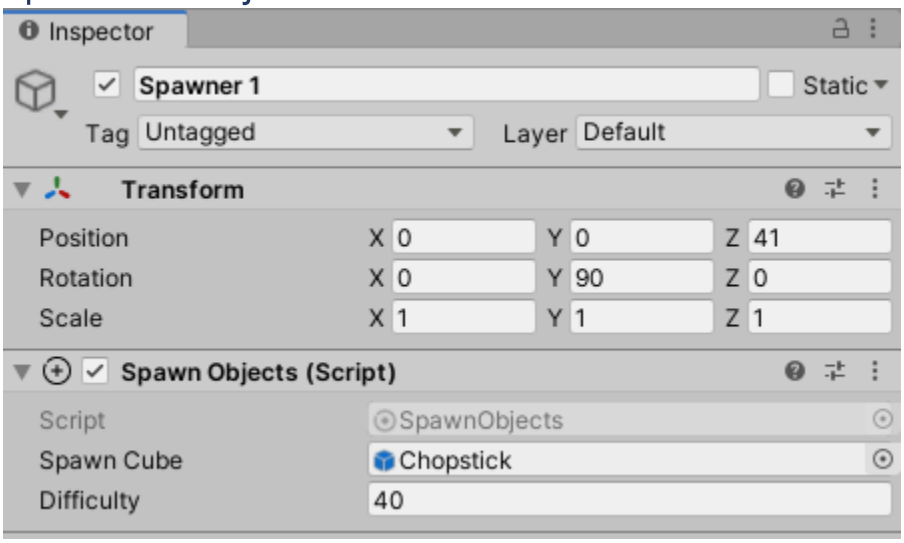
Hierarchy



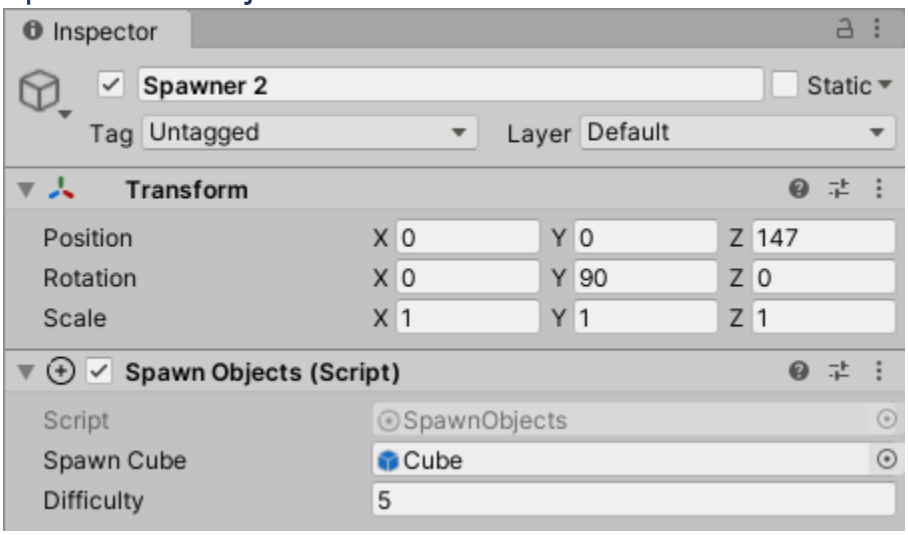
Chopstick Prefab Object



Spawner 1 Object

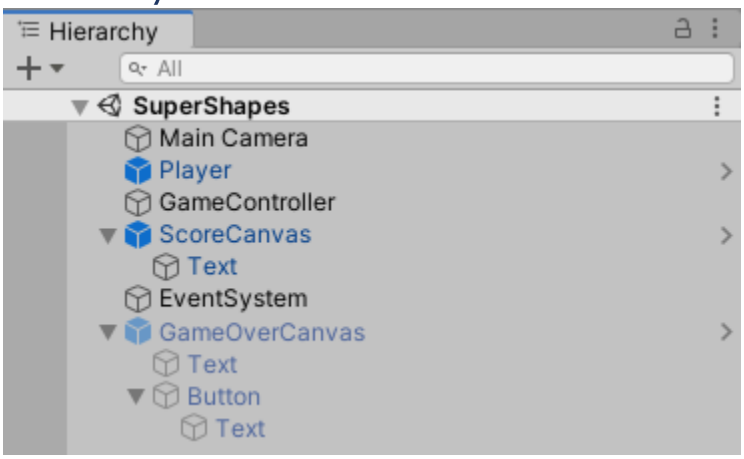


Spawner 2 Object

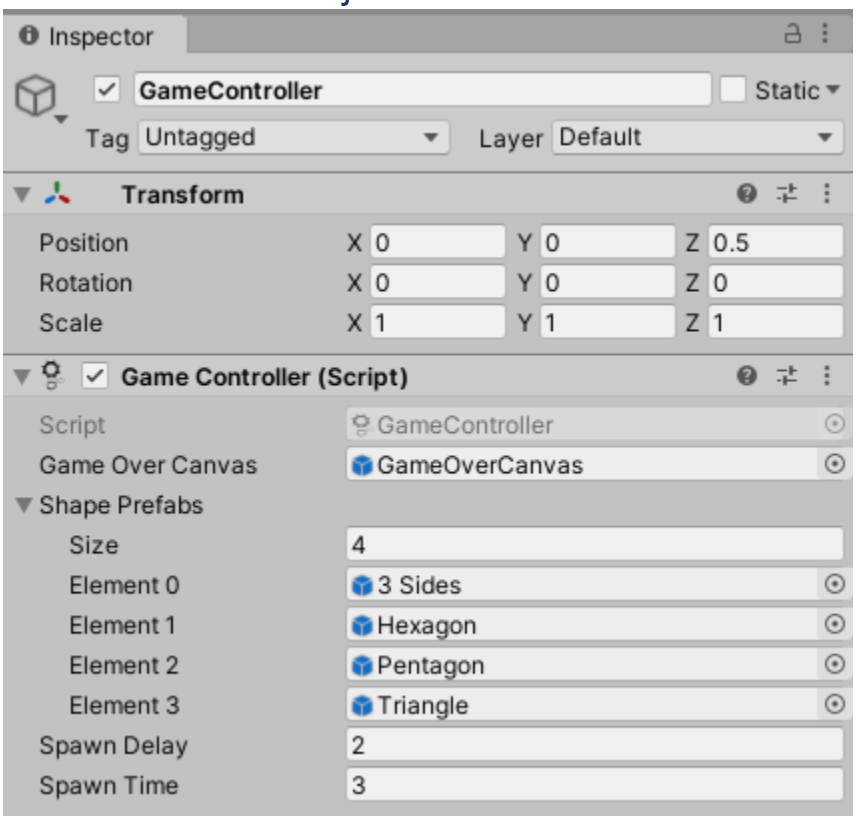


Super Shapes

Hierarchy



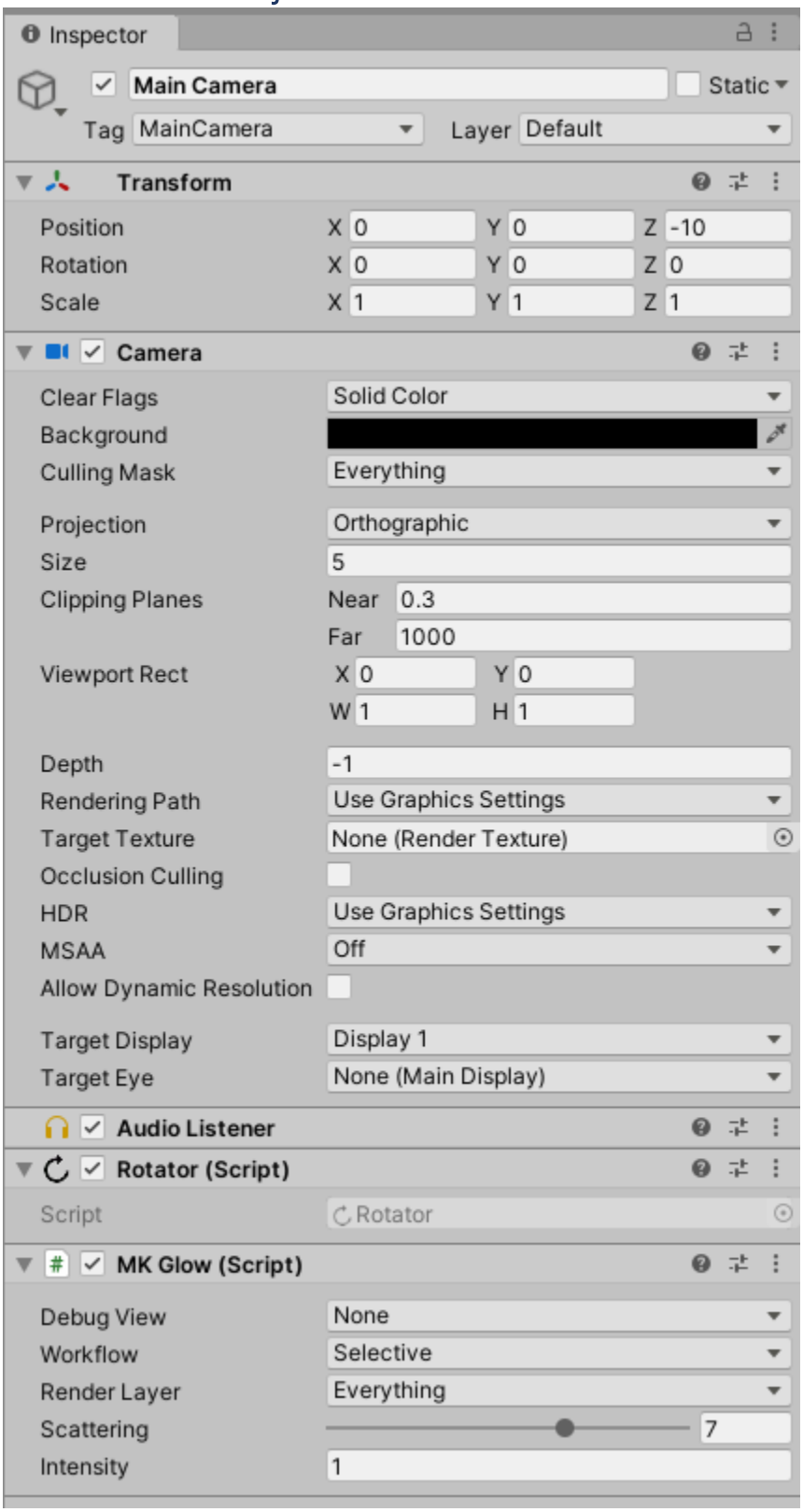
GameController Object



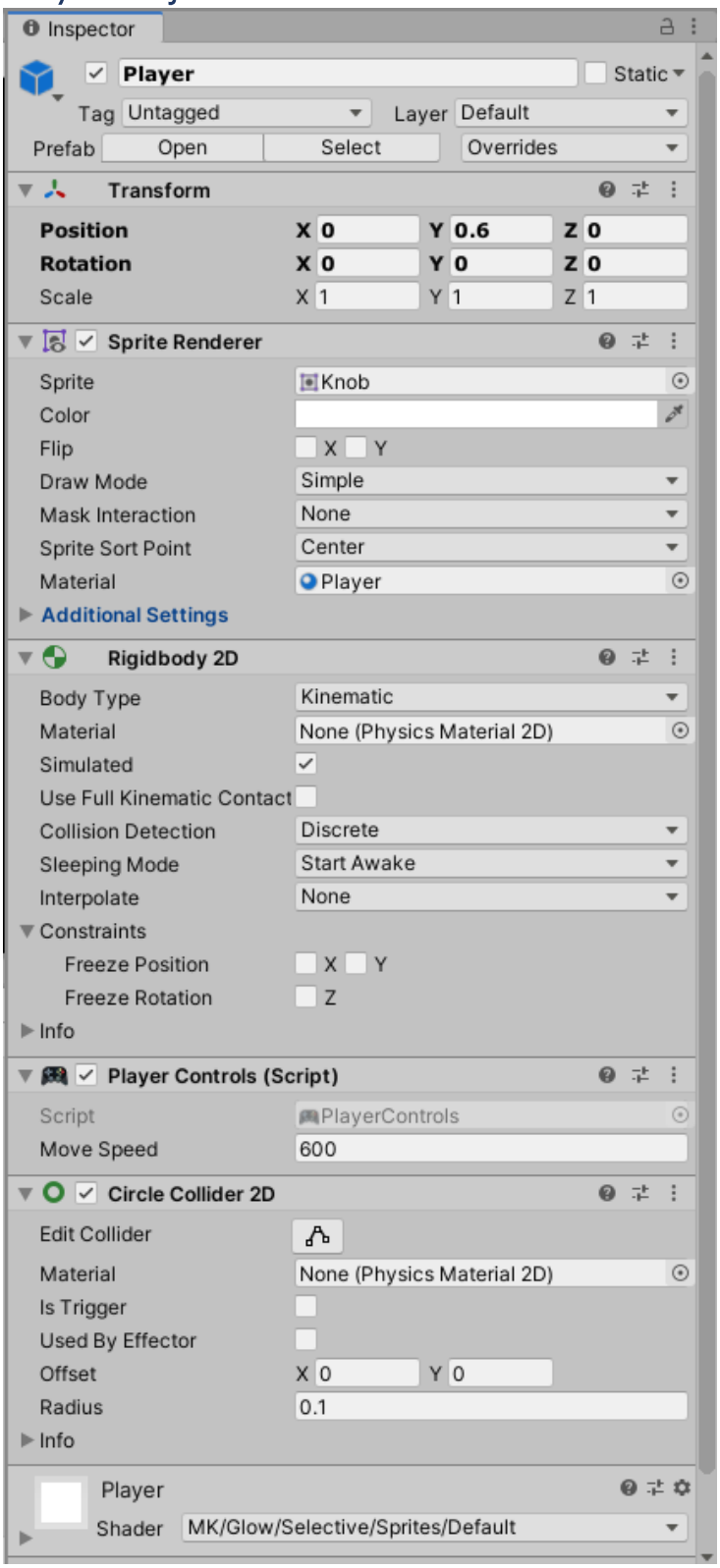
GameController.cs Script

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

public class GameController : MonoBehaviour
{
    public GameObject gameOverCanvas;
    public GameObject[] shapePrefabs;
    public float spawnDelay = 1f;
    public float spawnTime = 2f;

    void Start()
    {
        InvokeRepeating("Spawn", spawnDelay, spawnTime);
    }

    void Spawn()
    {
        int randomInt = Random.Range(0, shapePrefabs.Length);
        // Spawn a new random shape
        Instantiate(shapePrefabs[randomInt], Vector3.zero, Quaternion.identity);
    }

    public void GameOver()
    {
        Time.timeScale = 0;
        gameOverCanvas.SetActive(true);
        CancelInvoke("Spawn");
    }

    public void Restart()
    {
        SceneManager.LoadScene(0);
    }
}
```

Main Camera Object



Player Object

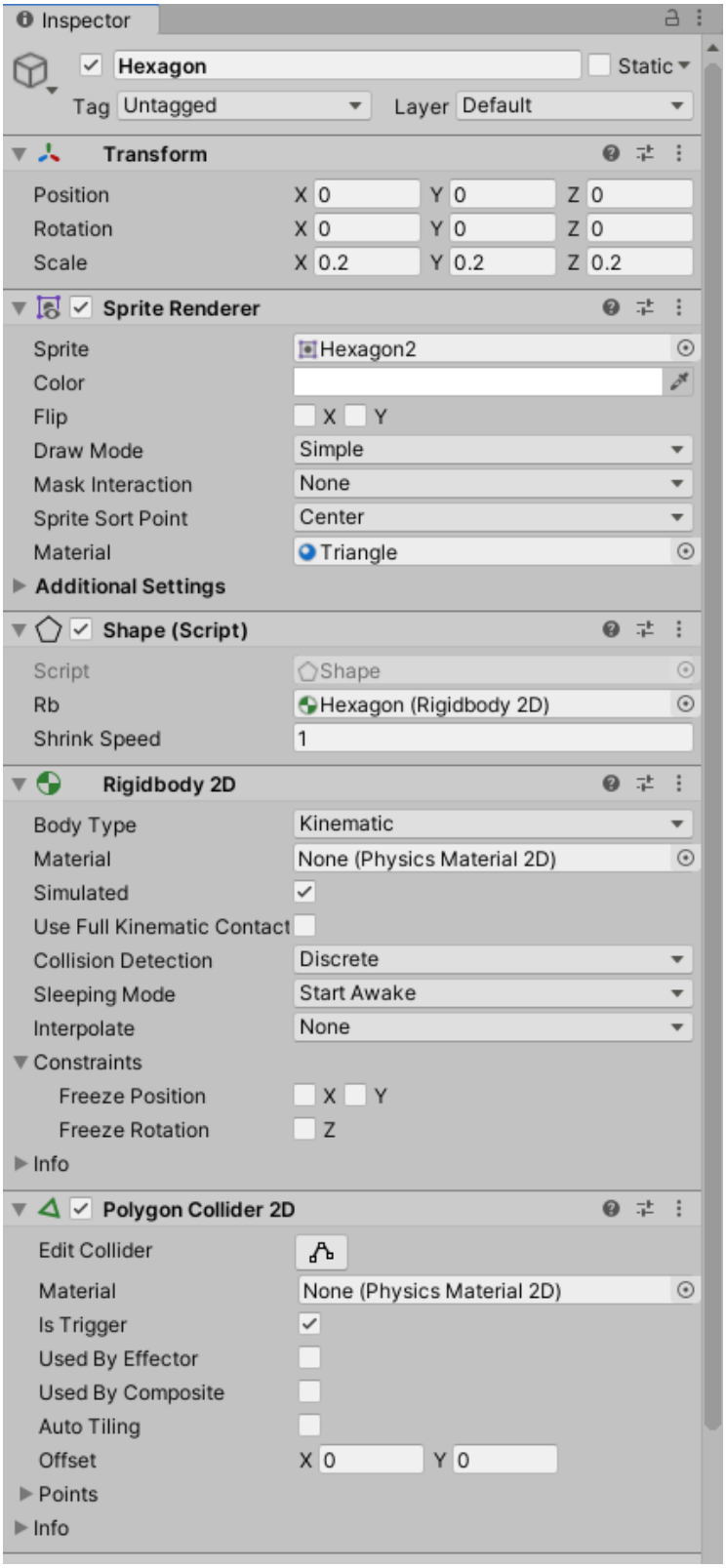


PlayerControls.cs Script

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

public class PlayerControls : MonoBehaviour
{
    public float moveSpeed = 600f;
    float movement = 0f;

    void Start()
    {
        Time.timeScale = 1;
    }

    void Update()
    {
        // is the player pressing left or right?
        movement = Input.GetAxisRaw("Horizontal");
    }

    private void FixedUpdate()
    {
        // rotate the player based on the value of movement
        transform.RotateAround(
            Vector3.zero,
            Vector3.forward,
            movement * Time.fixedDeltaTime * -moveSpeed);
    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        GameObject.Find("GameController").GetComponent<GameController>().GameOver();
    }
}
```

Shape Objects

Each shape will have a different name and Rb property in the Shape script component.



Shape.cs Script

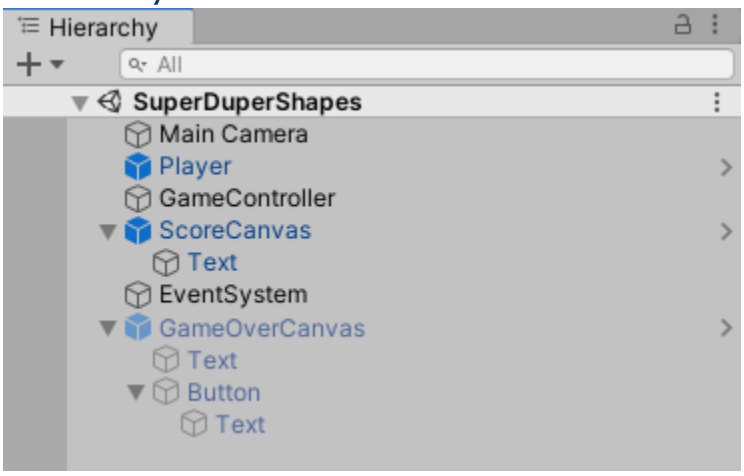
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Shape : MonoBehaviour
{
    public Rigidbody2D rb;
    public float shrinkSpeed = 3f;

    void Start()
    {
        // start the shape with a random rotation
        rb.rotation = Random.Range(0f, 360f);
        // start the shape with a large scale
        transform.localScale = Vector3.one * 10f;
    }

    void Update()
    {
        // shrink the shape every frame
        transform.localScale -= Vector3.one * shrinkSpeed * Time.deltaTime;
        // destroy the shape if it gets really small
        if (transform.localScale.x <= .05f)
        {
            Destroy(gameObject);
            Score.score++;
        }
    }
}
```

Rotator.cs Script

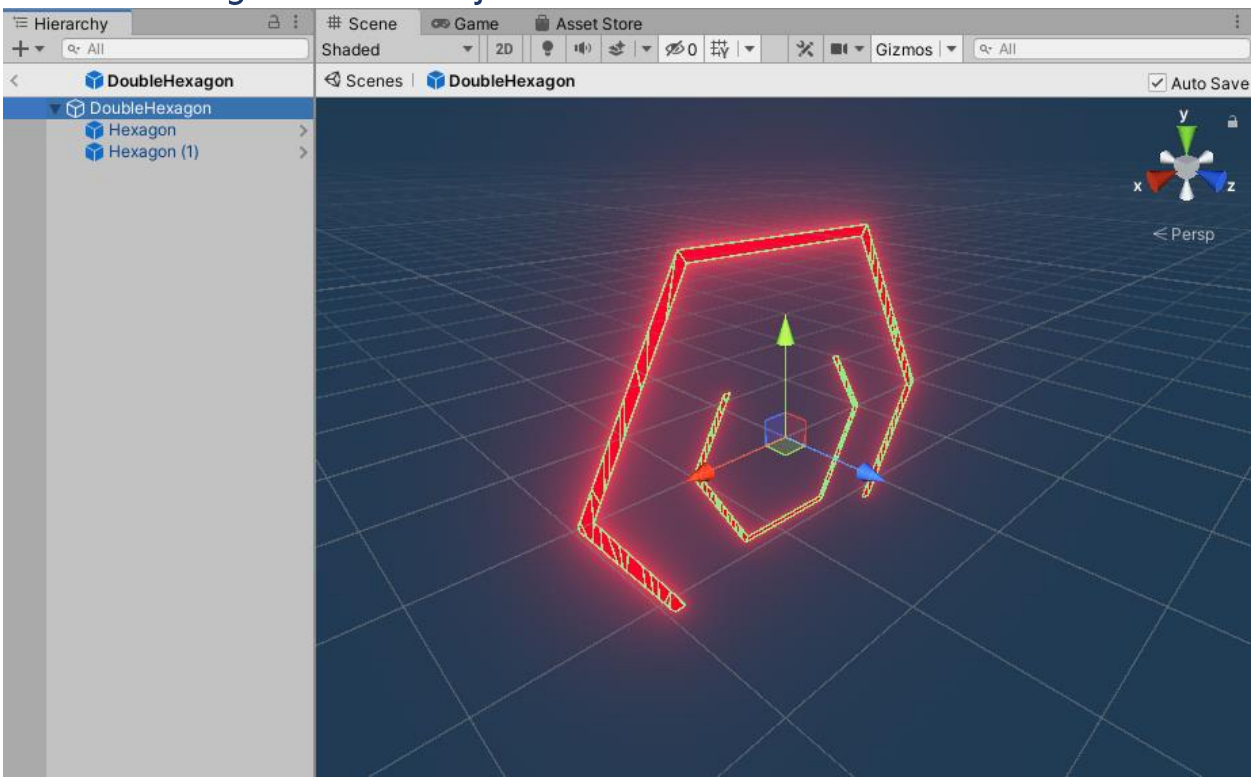
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Rotator : MonoBehaviour
{
    void Update()
    {
        transform.Rotate(Vector3.forward, Time.deltaTime * 30f);
    }
}
```

Super Duper Shapes Prove Yourself

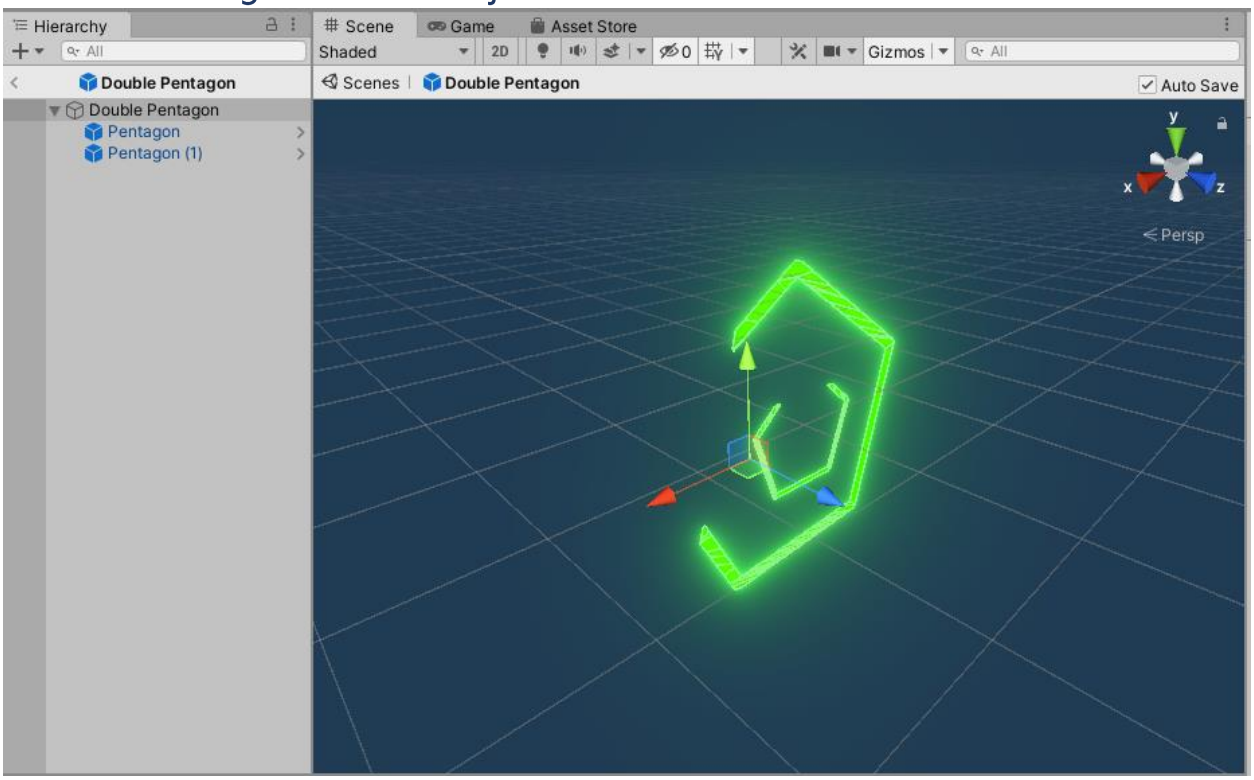
Hierarchy



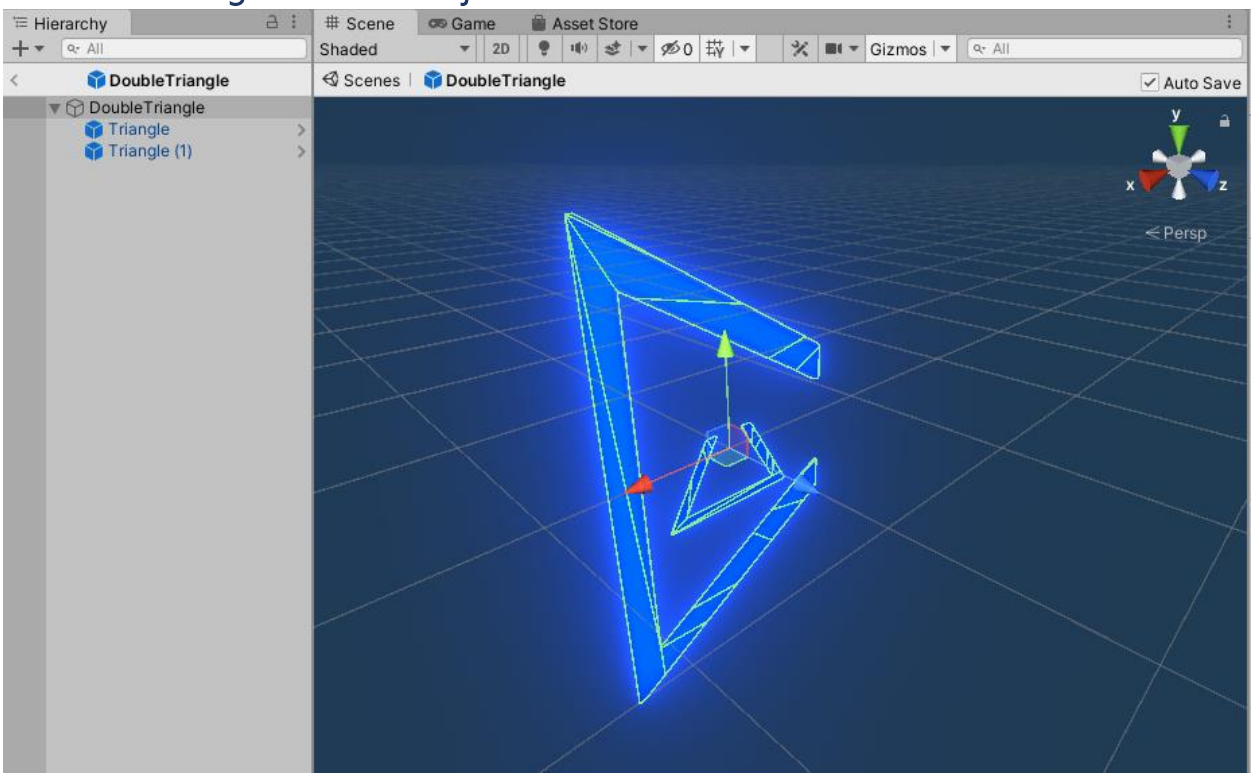
Double Hexagon Prefab Object



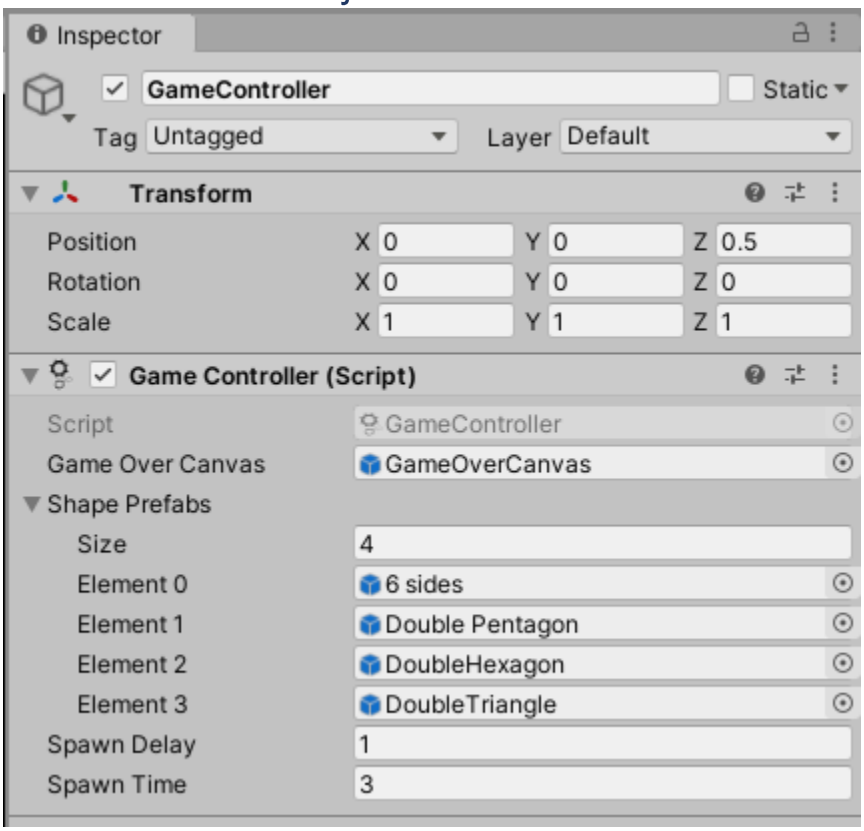
Double Pentagon Prefab Object



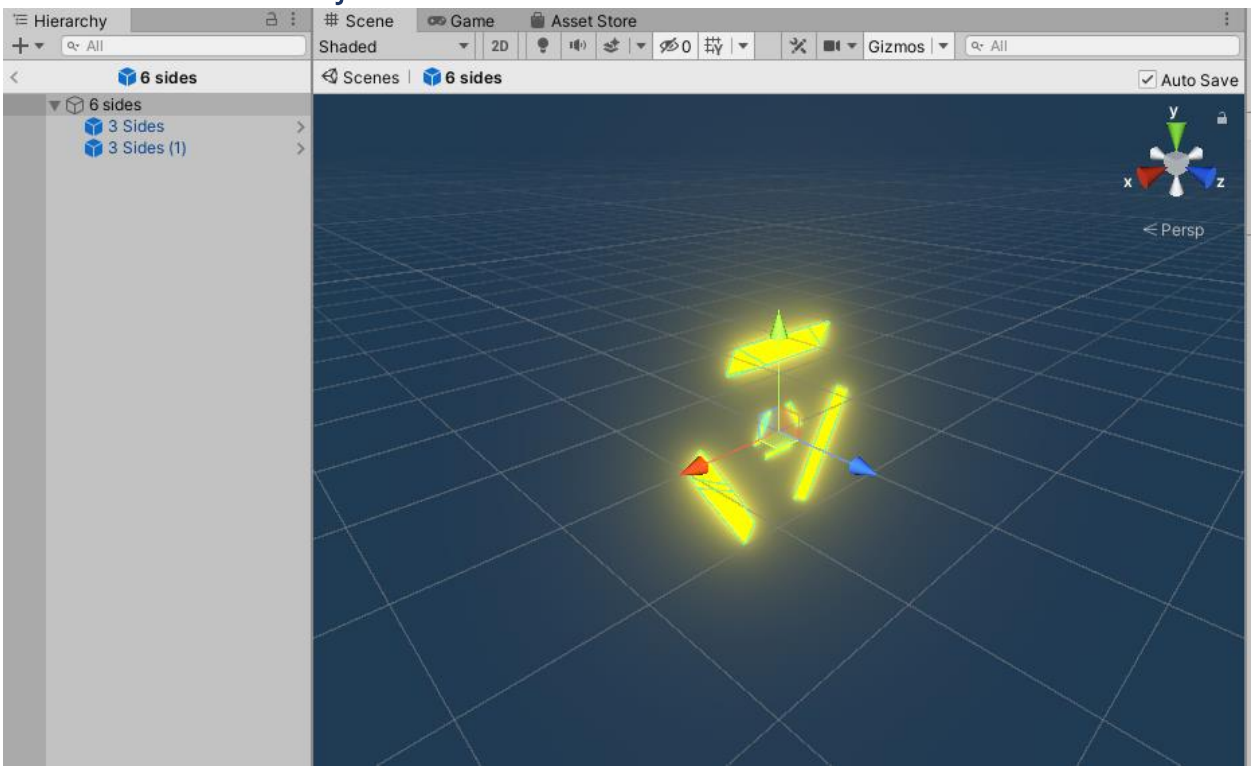
Double Triangle Prefab Object



GameController Object

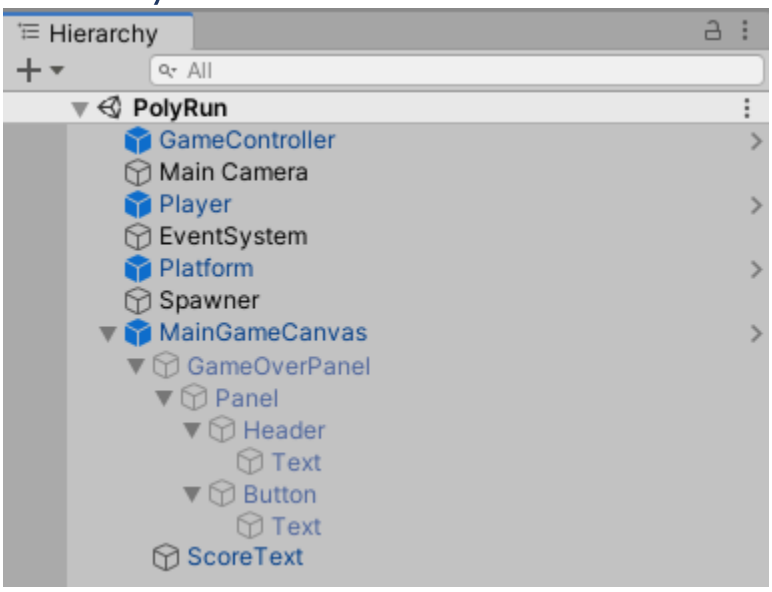


Six Sides Prefab Object

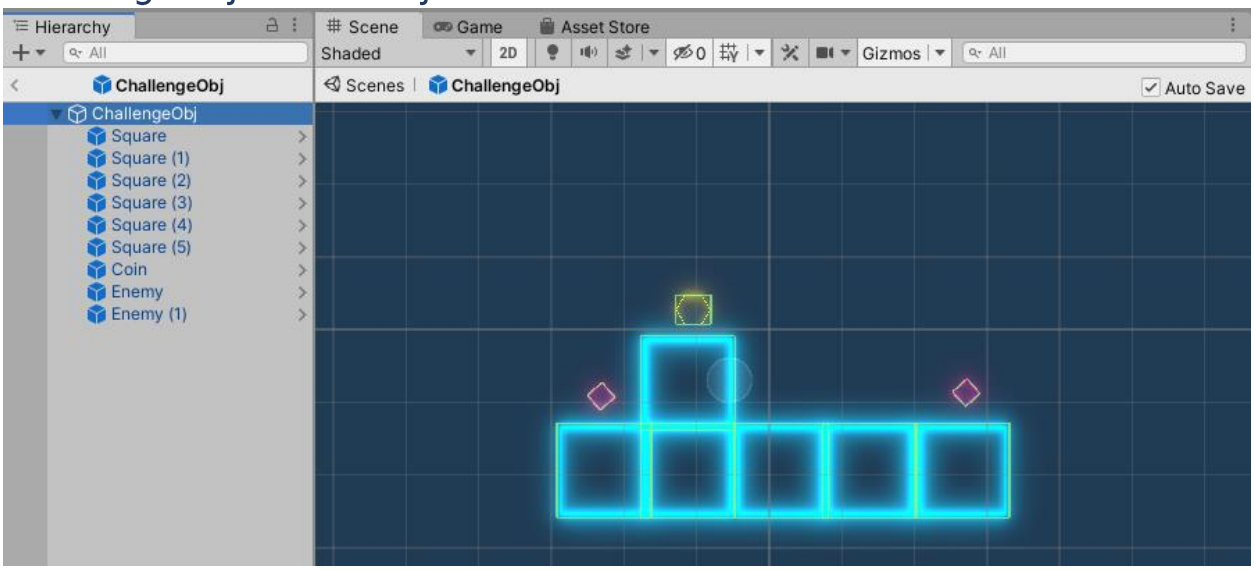


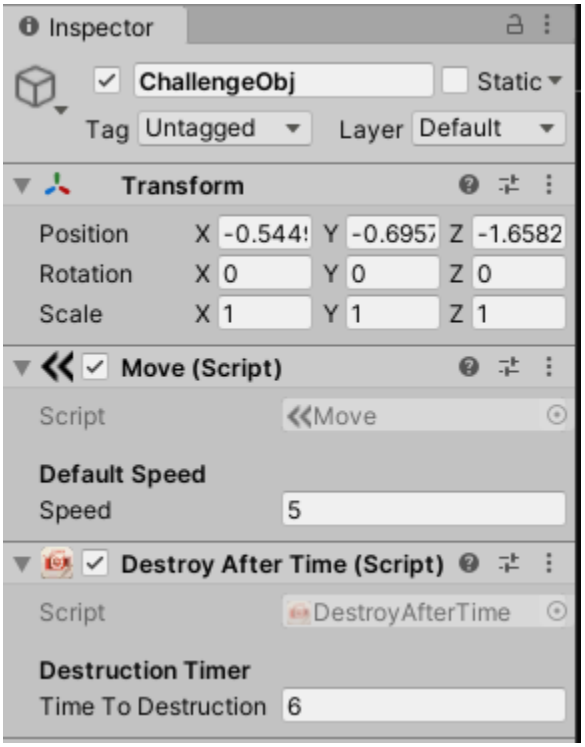
Polyrun

Hierarchy



ChallengeObj Prefab Object

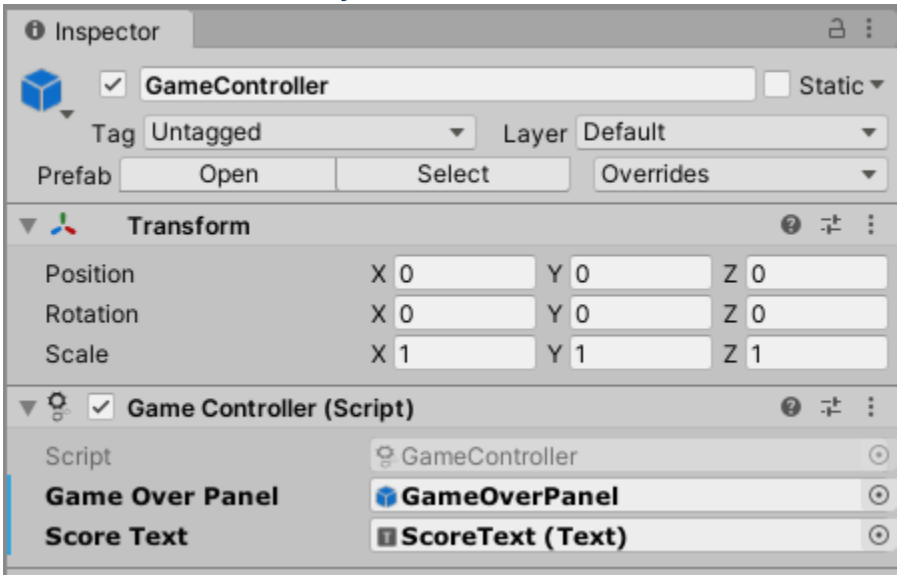




DestroyAfterTime.cs Script

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

public class DestroyAfterTime : MonoBehaviour
{
    [Header("Destruction Timer")]
    public float timeToDestruction;

    void Start()
    {
        Invoke("DestroyObject", timeToDestruction);
    }

    void DestroyObject()
    {
        Destroy(gameObject);
    }
}
```

GameController Object



GameController.cs Script

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

public class GameController : MonoBehaviour
{
    public GameObject gameOverPanel;
    public Text scoreText;
    int score = 0;

    void Start()
    {
        Time.timeScale = 1;
    }

    public void GameOver()
    {
        Time.timeScale = 0;
        gameOverPanel.SetActive(true);
    }

    public void IncrementScore()
    {
        score++;
        scoreText.text = score.ToString();
    }
}
```

Move.cs Script

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

public class Move : MonoBehaviour
{
    [Header("Default Speed")]
    public float speed;

    void Update()
    {
        transform.position += Vector3.left * speed * Time.deltaTime;
    }
}
```

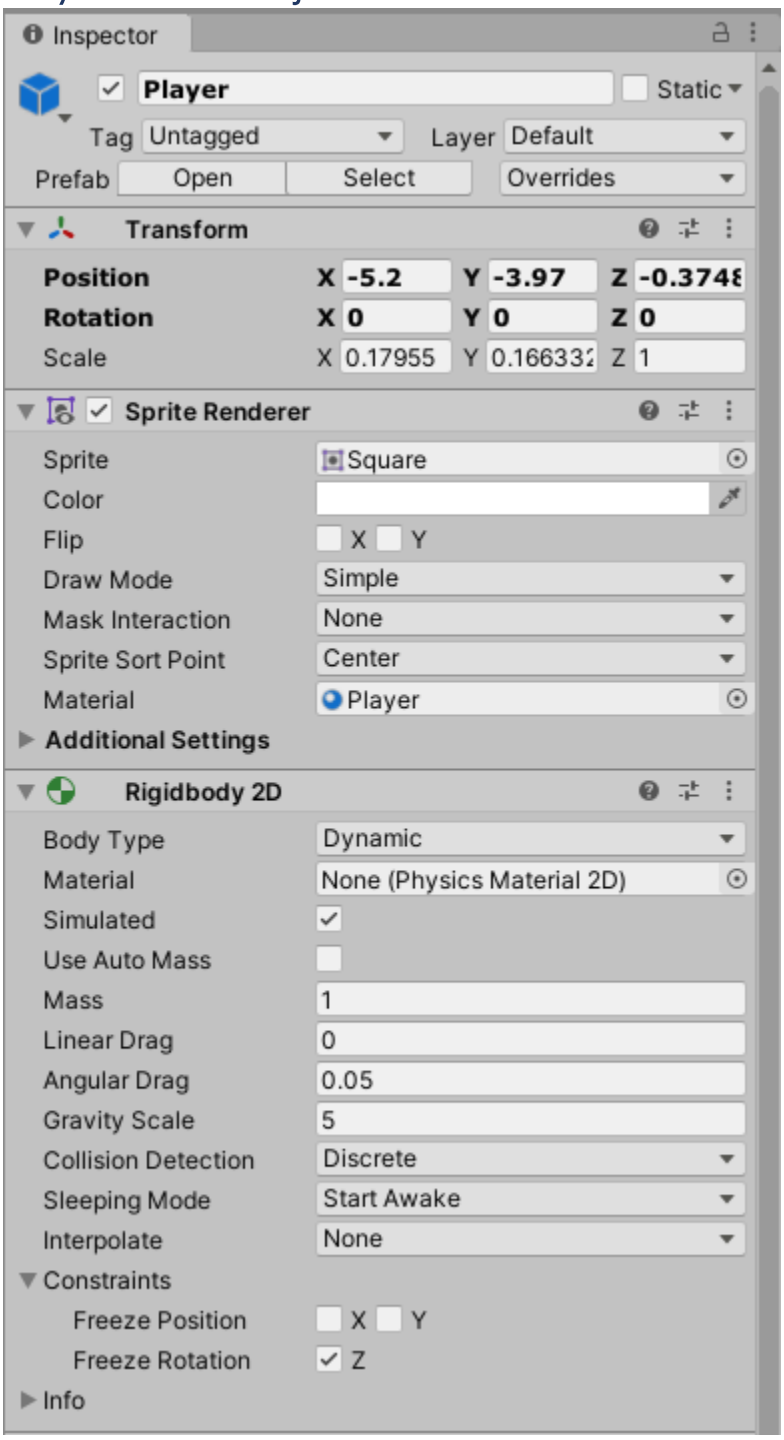
PlayerControls.cs Script

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

public class PlayerControls : MonoBehaviour
{
    [Header("Default Jumping Power")]
    public float jumpPower = 6.5f;
    [Header("Boolean isGrounded")]
    public bool isGrounded = false;
    float posX = 0.0f;
    Rigidbody2D rb;

    void Start()
    {
        rb = transform.GetComponent<Rigidbody2D>();
        posX = transform.position.x;
    }

    void FixedUpdate()
    {
        if (Input.GetKey(KeyCode.Space) && isGrounded)
        {
            rb.AddForce(Vector3.up * (jumpPower * rb.mass * rb.gravityScale * 20.0f));
        }
        if (transform.position.x < posX)
        {
            GameOver();
        }
    }

    void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.collider.tag == "Ground")
        {
            isGrounded = true;
        }
        if (collision.collider.tag == "Enemy")
        {
            GameOver();
        }
    }

    void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.tag == "Coin")
        {
            GameObject.Find("GameController")
                      .GetComponent<GameController>().IncrementScore();
            Destroy(collision.gameObject);
        }
    }
```

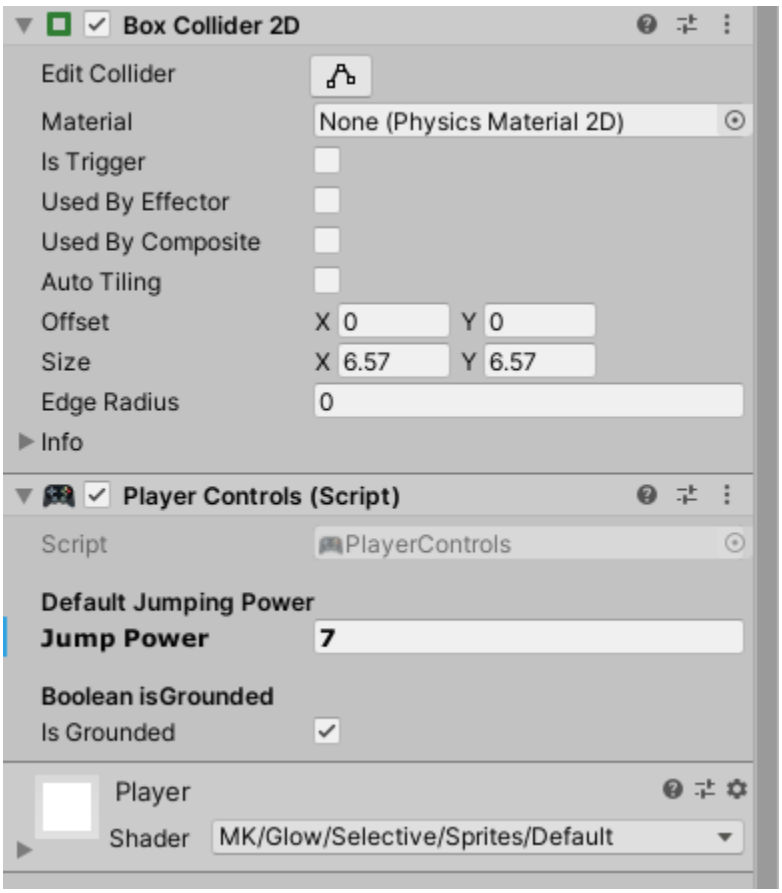
PlayerControls.cs Script Continued

```
    void OnCollisionStay2D(Collision2D collision)
    {
        if (collision.collider.tag == "Ground")
        {
            isGrounded = true;
        }
    }

    void OnCollisionExit2D(Collision2D collision)
    {
        if (collision.collider.tag == "Ground")
        {
            isGrounded = false;
        }
    }

    void GameOver()
    {
        GameObject.Find("GameController").GetComponent<GameController>().GameOver();
    }
}
```

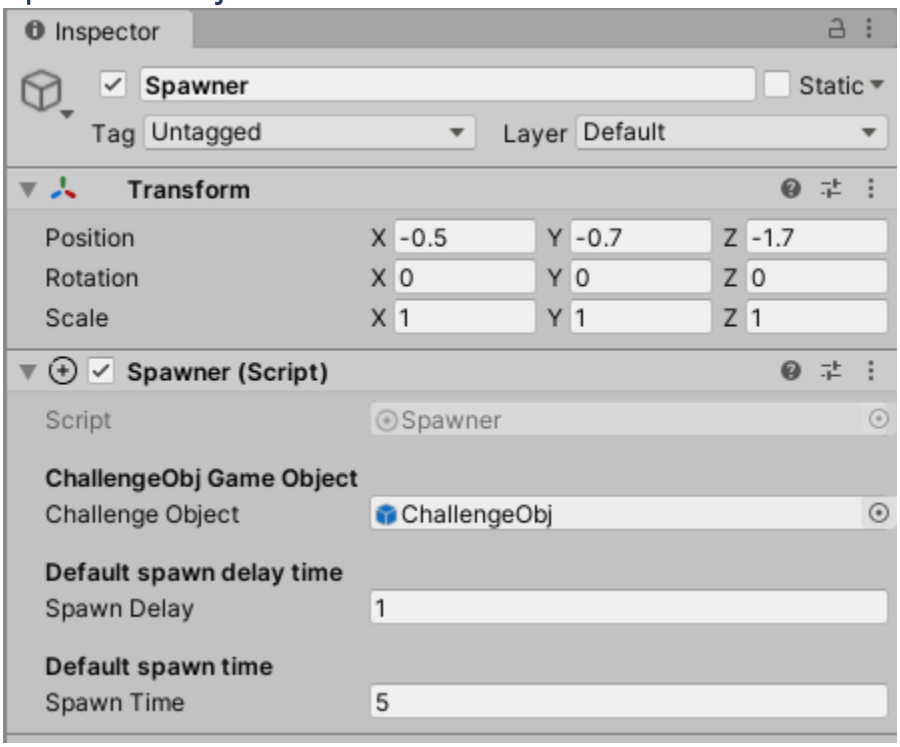
Player Prefab Object



Player Prefab Object Continued

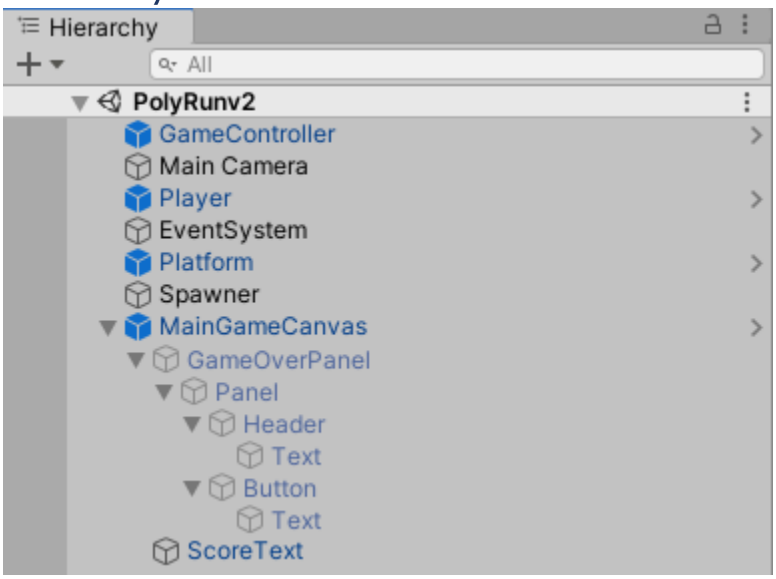


Spawner Object

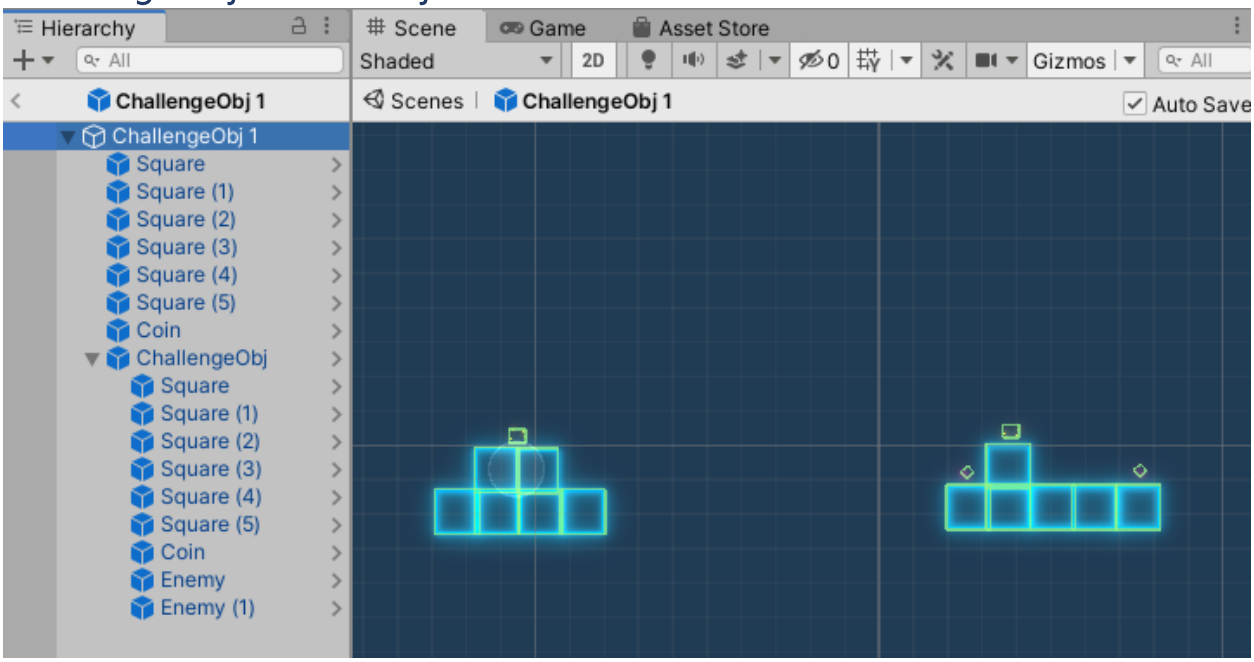


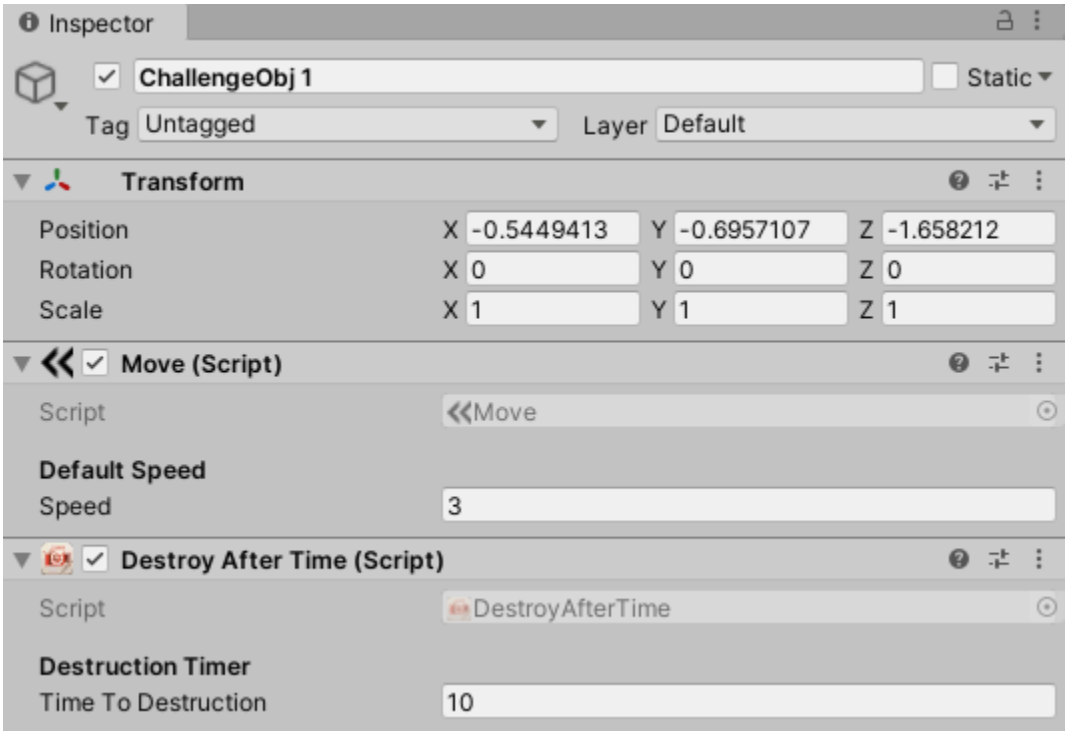
Polyrun v2 Prove Yourself

Hierarchy

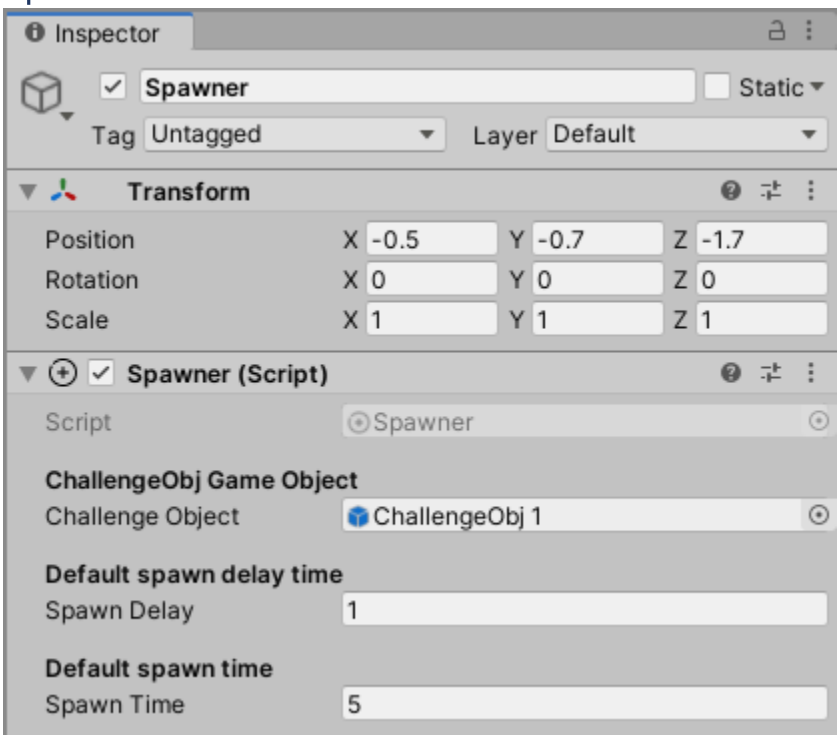


ChallengeObj Prefab Object



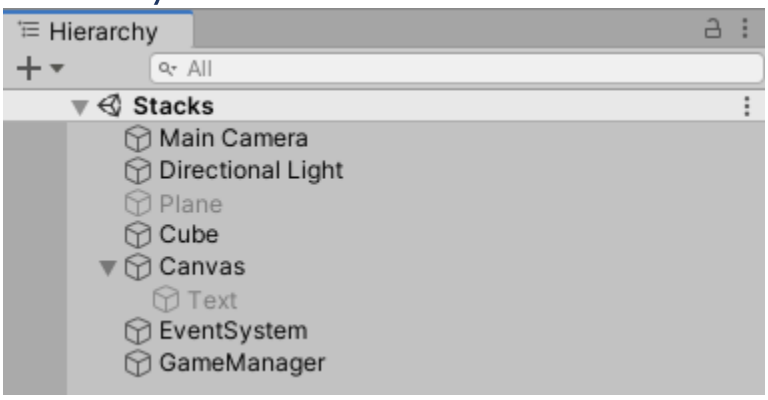


Spawner



Stacks

Hierarchy

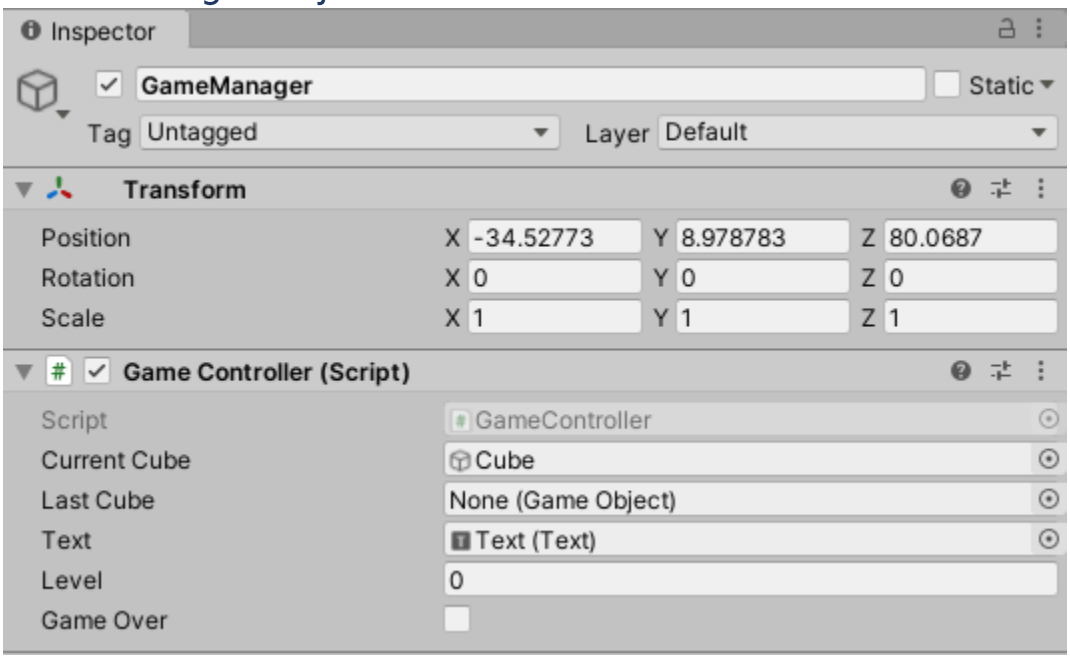


GameController.cs Script

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

public class GameController : MonoBehaviour
{
    public GameObject currentCube;
    public GameObject lastCube;
    public Text text;
    public int Level;
    public bool Done;

    void Start()
    {
        newBlock();
    }

    void newBlock()
    {

        currentCube.transform.position = new Vector3(
            Mathf.Round(currentCube.transform.position.x),
            currentCube.transform.position.y,
            Mathf.Round(currentCube.transform.position.z)
            );
        currentCube.transform.localScale = new Vector3(
            lastCube.transform.localScale.x - Mathf.Abs(currentCube.transform.position.x
- lastCube.transform.position.x),
            lastCube.transform.localScale.y,
            lastCube.transform.localScale.z - Mathf.Abs(currentCube.transform.position.z
- lastCube.transform.position.z));

        currentCube.transform.position = Vector3.Lerp(currentCube.transform.position,
lastCube.transform.position, 0.5f) + Vector3.up * 5f;
```

```
        if (currentCube.transform.localScale.x <= 0f ||
currentCube.transform.localScale.z <= 0f)
        {
            Done = true;
            text.gameObject.SetActive(true);
            text.text = "Final Score: " + Level;
            StartCoroutine(X());
            return;
        }

        lastCube = currentCube;
        currentCube = Instantiate(lastCube);
        currentCube.name = Level + "";
        currentCube.GetComponent<MeshRenderer>().material.
        SetColor("_Color", Color.HSVToRGB((Level / 100f) % 1f, 1f, 1f));
        Level++;
        Camera.main.transform.position = currentCube.transform.position + new
Vector3(100, 100, 100);
        Camera.main.transform.LookAt(currentCube.transform.position);
    }

    void Update()
    {
        if (Done)
        {
            return;
        }
        var time = Mathf.Abs(Time.realtimeSinceStartup % 2f - 1f);
        var pos1 = lastCube.transform.position + Vector3.up * 10f;
        var pos2 = pos1 + ((Level % 2 == 0) ? Vector3.left : Vector3.forward) * 120;

        if (Level % 2 == 0)
        {
            currentCube.transform.position = Vector3.Lerp(pos2, pos1, time);
        }
        else
        {
            currentCube.transform.position = Vector3.Lerp(pos1, pos2, time);
        }

        if (Input.GetMouseButtonDown(0))
        {
            newBlock();
        }
    }
    IEnumerator X()
    {
        yield return new WaitForSeconds(3f);
        SceneManager.LoadScene("Stacks");
    }
}
```

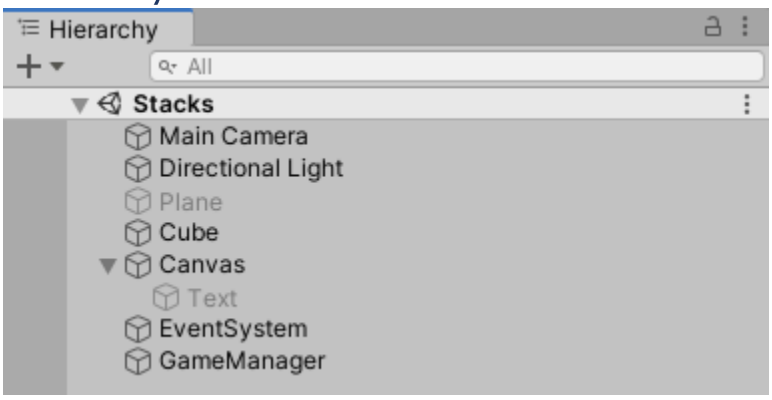
GameManager Object



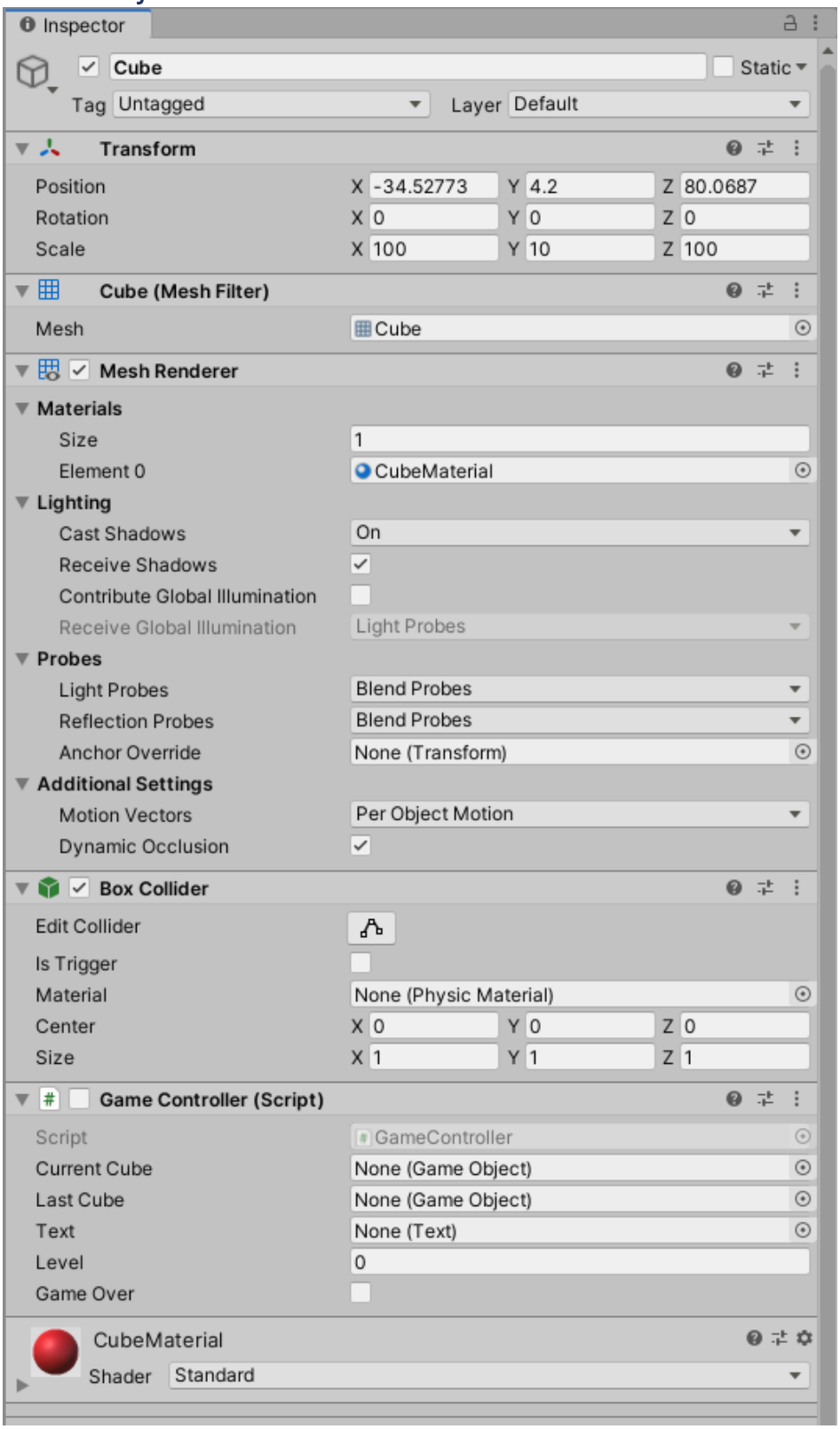
Text Object

Stacks on Stacks Prove Yourself

Hierarchy



Cube Object

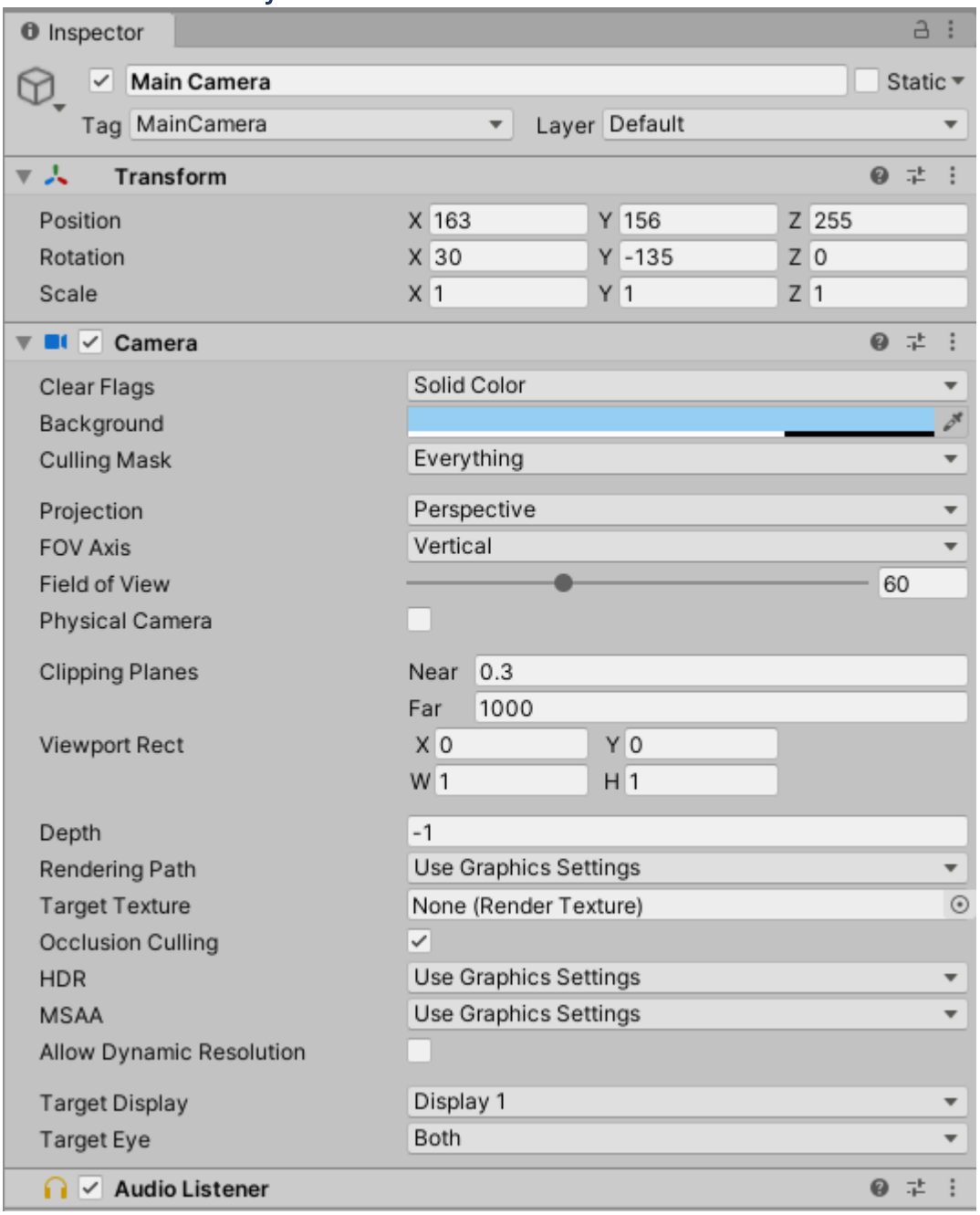


GameController.cs Script

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

public class GameController : MonoBehaviour
{
    public GameObject currentCube;
    public GameObject lastCube;
    public Text text;
    public int Level;
    public bool gameOver;

    void Start()
    {
        newBlock();
        text.gameObject.SetActive(true);
    }

    void newBlock()
    {
        if (lastCube != null)
        {
            currentCube.transform.position = new
Vector3(Mathf.Round(currentCube.transform.position.x),
                currentCube.transform.position.y,
                Mathf.Round(currentCube.transform.position.z));

            if (lastCube.transform.localScale.x <
Mathf.Abs(currentCube.transform.position.x - lastCube.transform.position.x) ||
                lastCube.transform.localScale.z <
Mathf.Abs(currentCube.transform.position.z - lastCube.transform.position.z))
            {
                gameOver = true;
                text.text = "Final Score: " + Level;
                StartCoroutine(X());
                return;
            }
            else
            {
                currentCube.transform.localScale = new Vector3(
                    lastCube.transform.localScale.x -
Mathf.Abs(currentCube.transform.position.x - lastCube.transform.position.x),
                    lastCube.transform.localScale.y,
                    lastCube.transform.localScale.z -
Mathf.Abs(currentCube.transform.position.z - lastCube.transform.position.z)
                    );
                currentCube.transform.position =
Vector3.Lerp(currentCube.transform.position, lastCube.transform.position, 0.5f) +
Vector3.up * 5f;
            }
        }

        lastCube = currentCube;
```

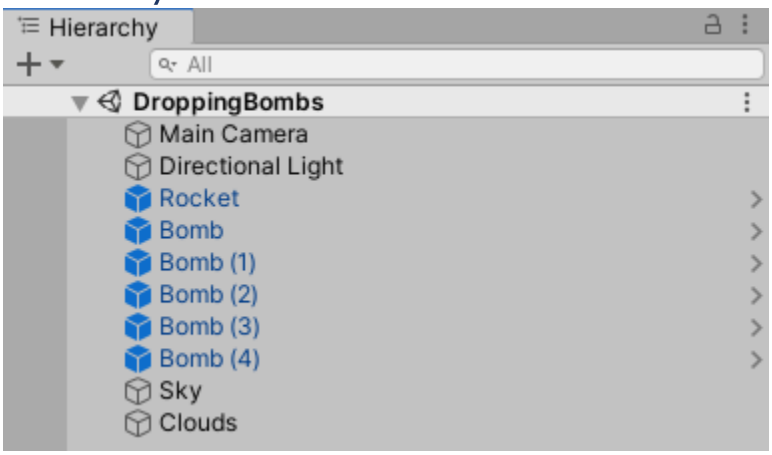
```
        currentCube = Instantiate(lastCube);
        currentCube.name = Level + "";
        currentCube.GetComponent<MeshRenderer>().material.SetColor("_Color",
Color.HSVToRGB((Level / 100f) % 1f, 1f, 1f));

        Level++;
        text.text = "Score: " + Level;

        Camera.main.transform.position = currentCube.transform.position + new
Vector3(100, 100, 100);
        Camera.main.transform.LookAt(currentCube.transform.position);
    }

    void Update()
    {
        if (gameOver)
            return;

        var time = Mathf.Abs(Time.realtimeSinceStartup % 2f - 1f);
        var pos1 = lastCube.transform.position + Vector3.up * 10f;
        var pos2 = pos1 + ((Level % 2 == 0) ? Vector3.left : Vector3.forward) * 120;
        var pos3 = pos1 + ((Level % 2 == 0) ? Vector3.right : Vector3.back) * 120;

        if (Level % 2 == 0)
        {
            currentCube.transform.position = Vector3.Lerp(pos2, pos3, time);
        }
        else
        {
            currentCube.transform.position = Vector3.Lerp(pos3, pos2, time);
        }

        if (Input.GetMouseButtonDown(0))
        {
            newBlock();
        }
    }
    IEnumerator X()
    {
        yield return new WaitForSeconds(3f);
        SceneManager.LoadScene("Stacks");
    }
}
```

Main Camera Object

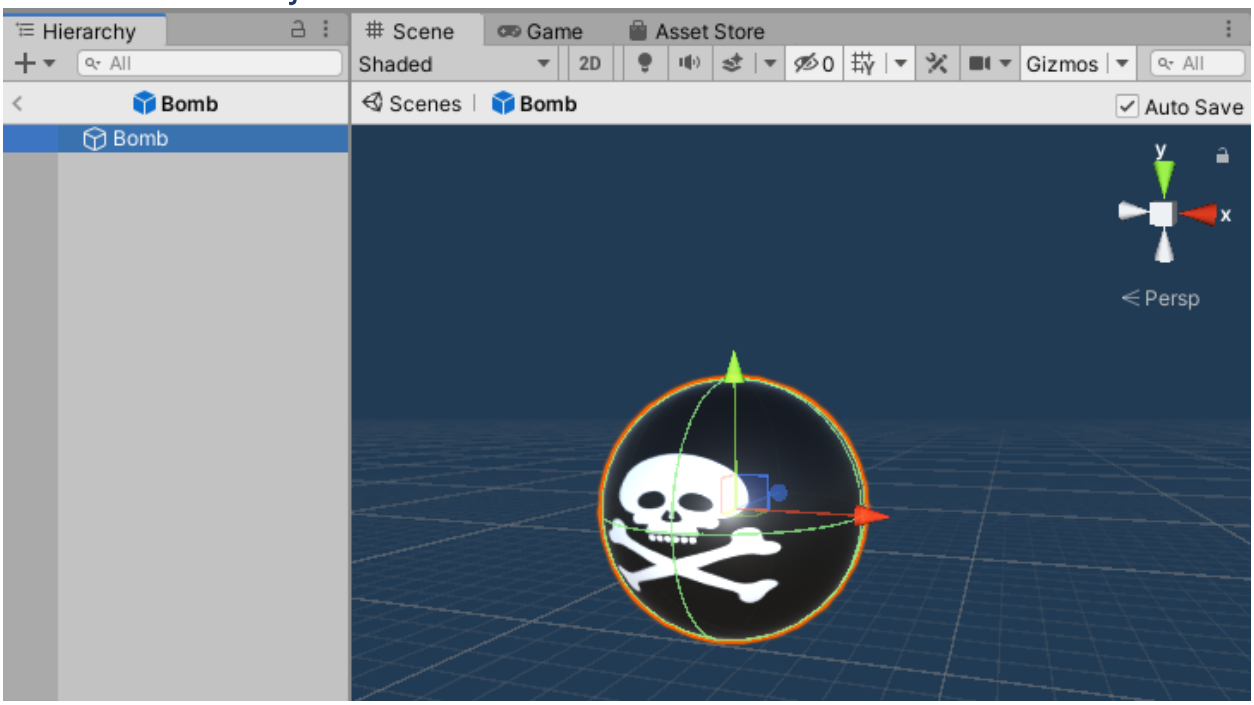


Dropping Bombs Part 2

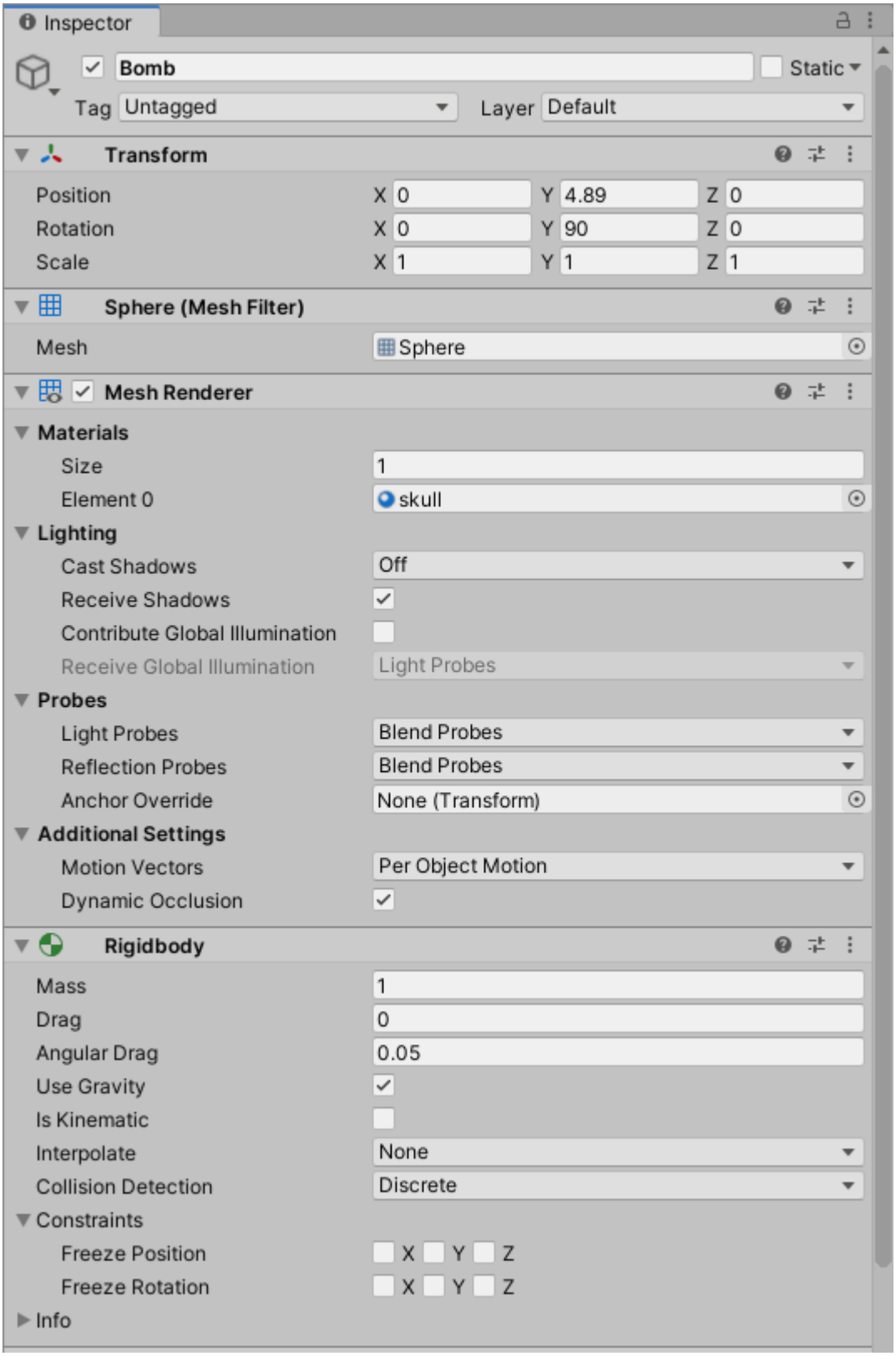
Hierarchy



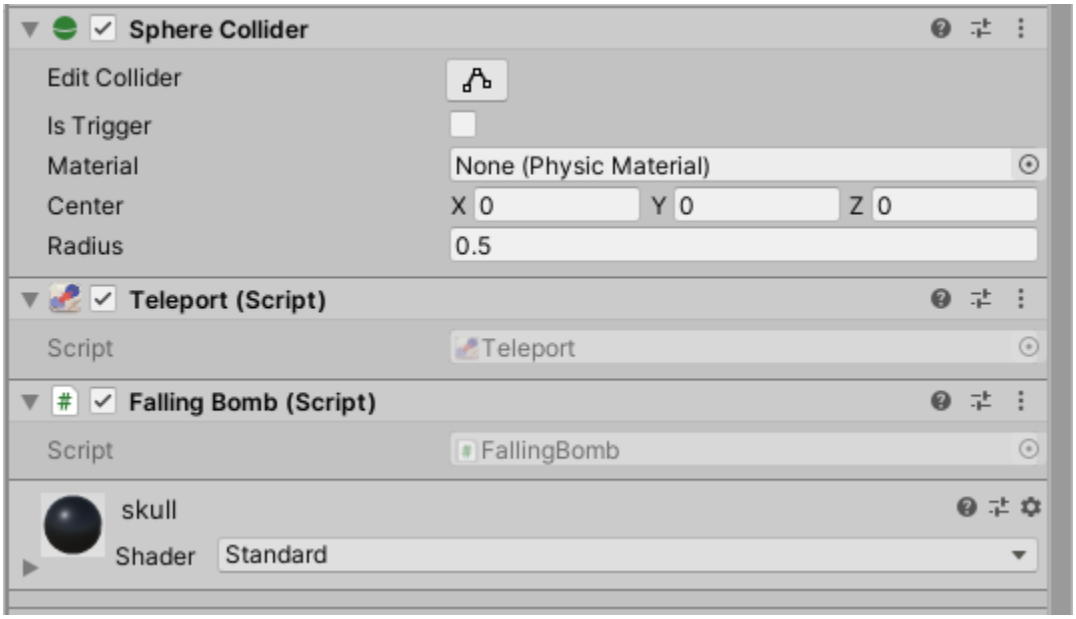
Bomb Prefab Object



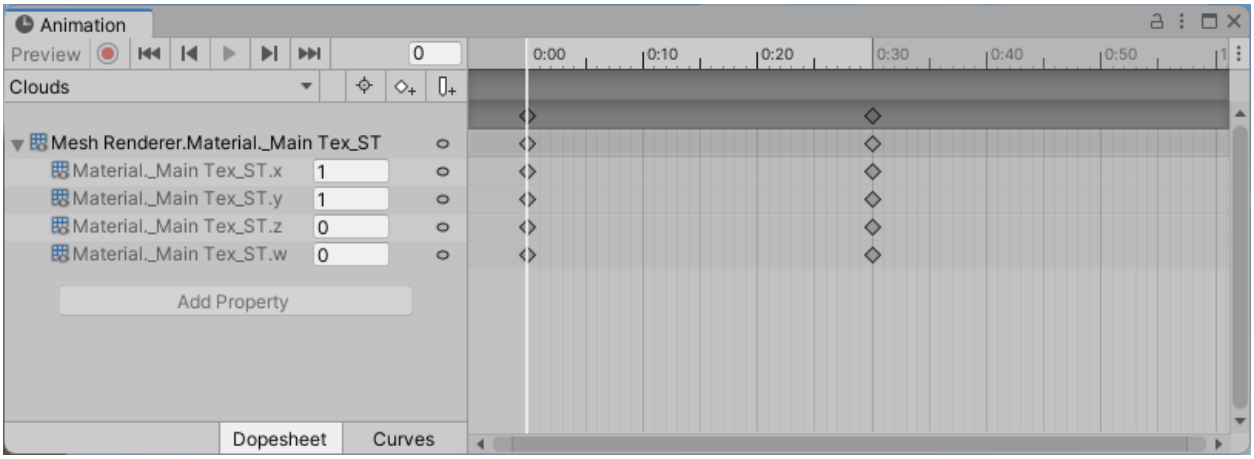
Bomb Prefab Object Continued

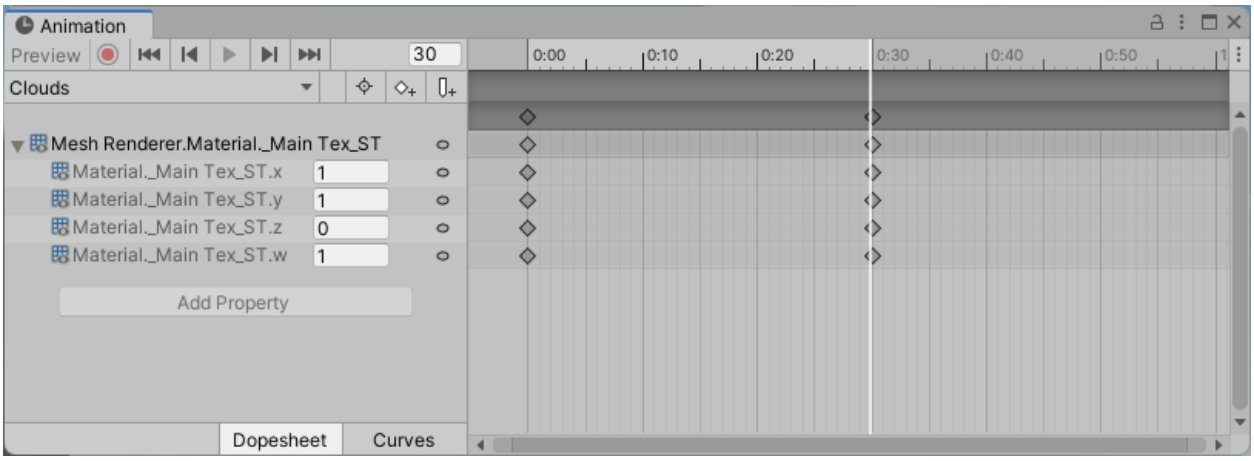


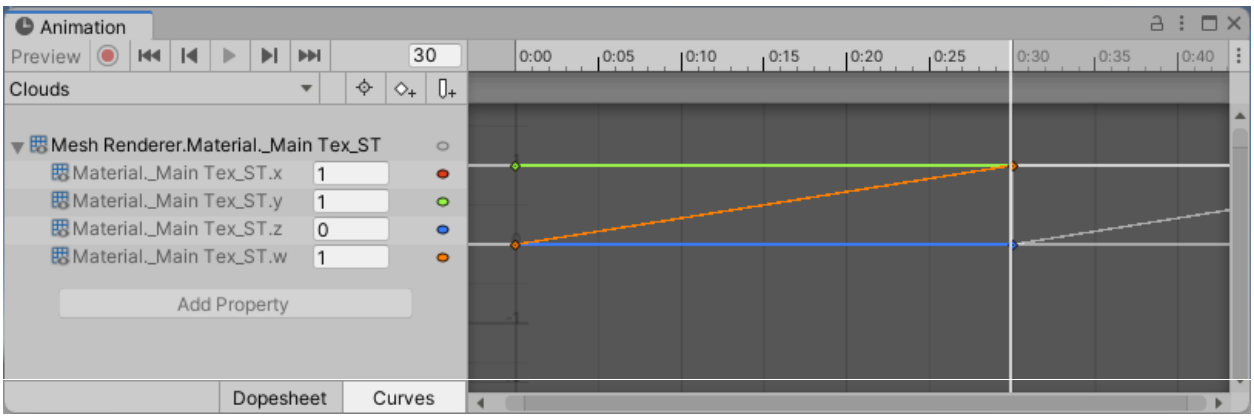
Bomb Prefab Object Continued



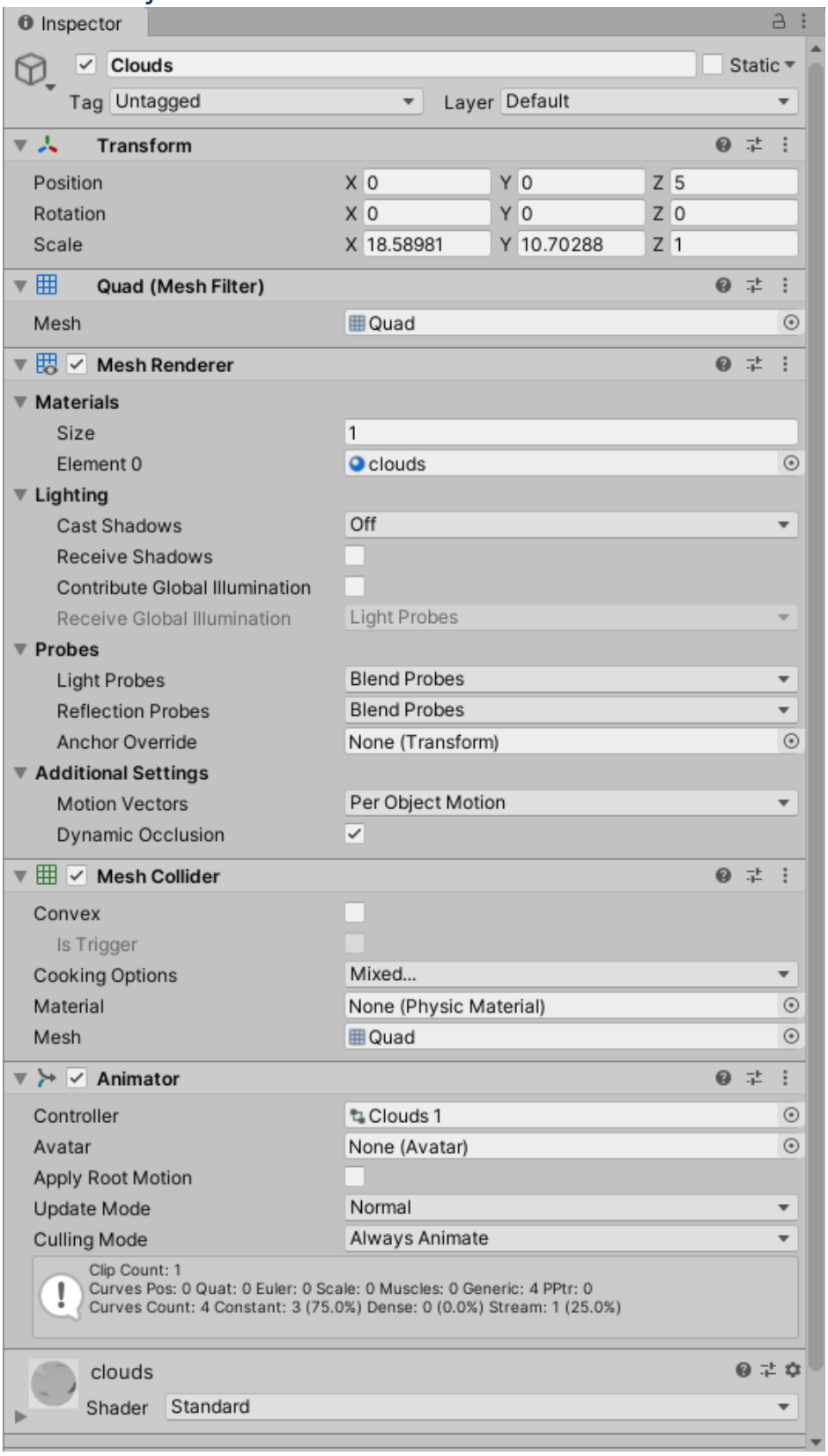
Cloud Animation Frames



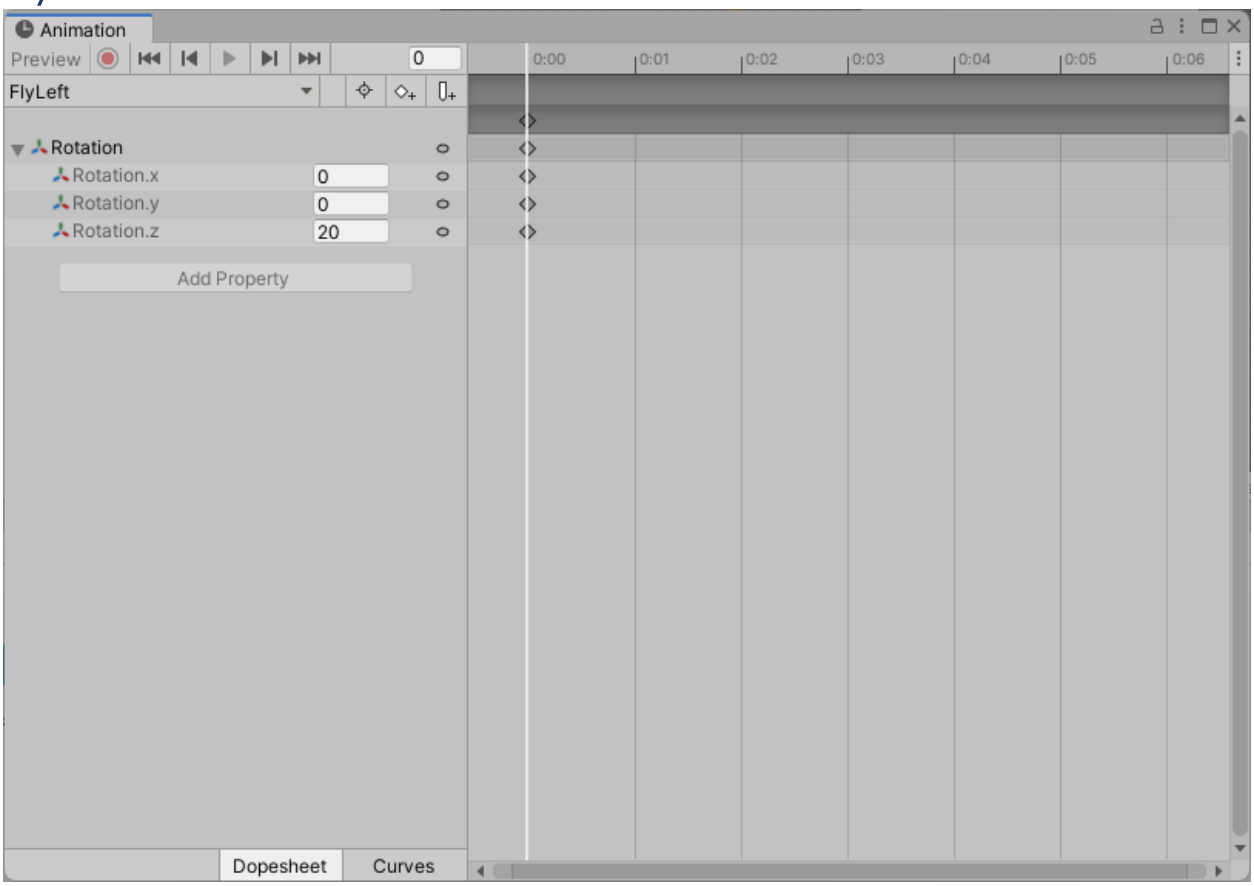




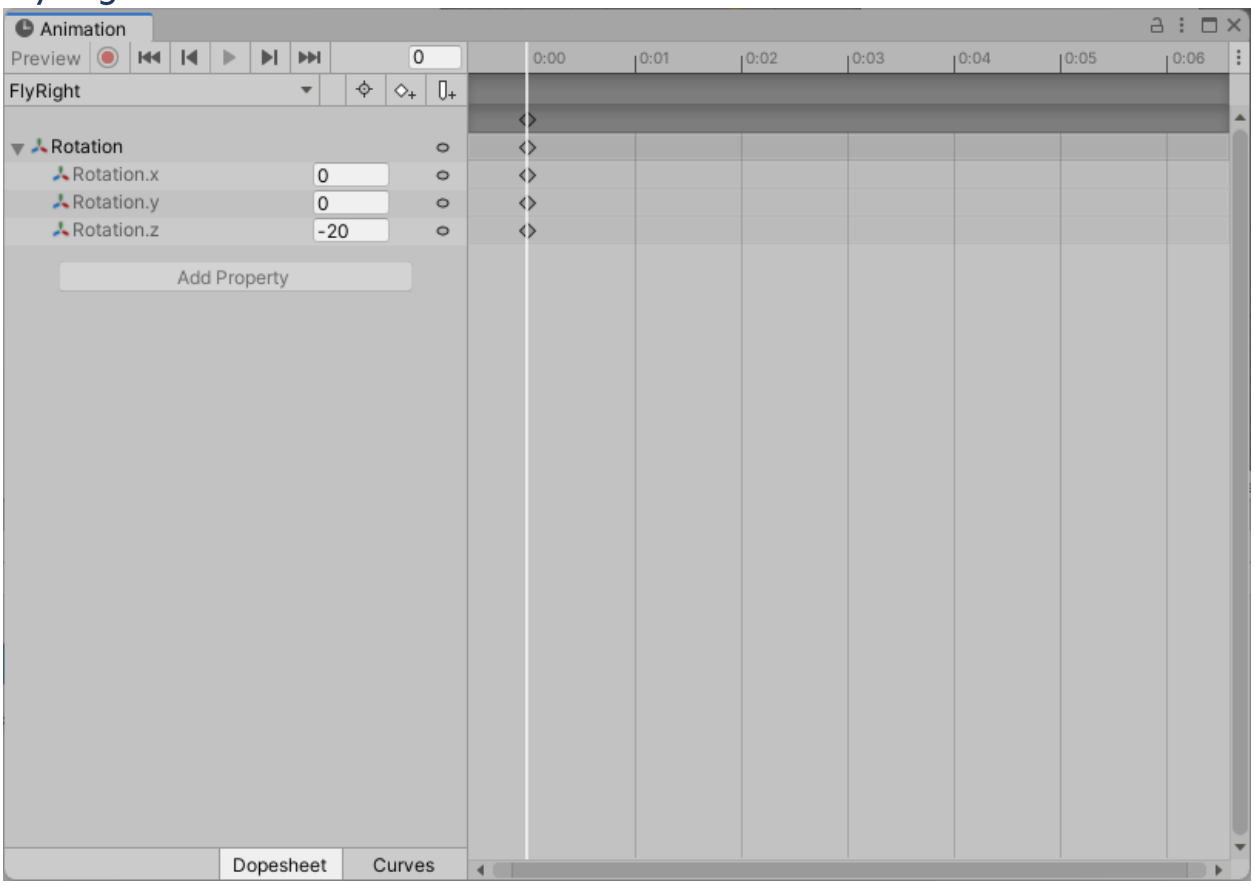
Cloud Object



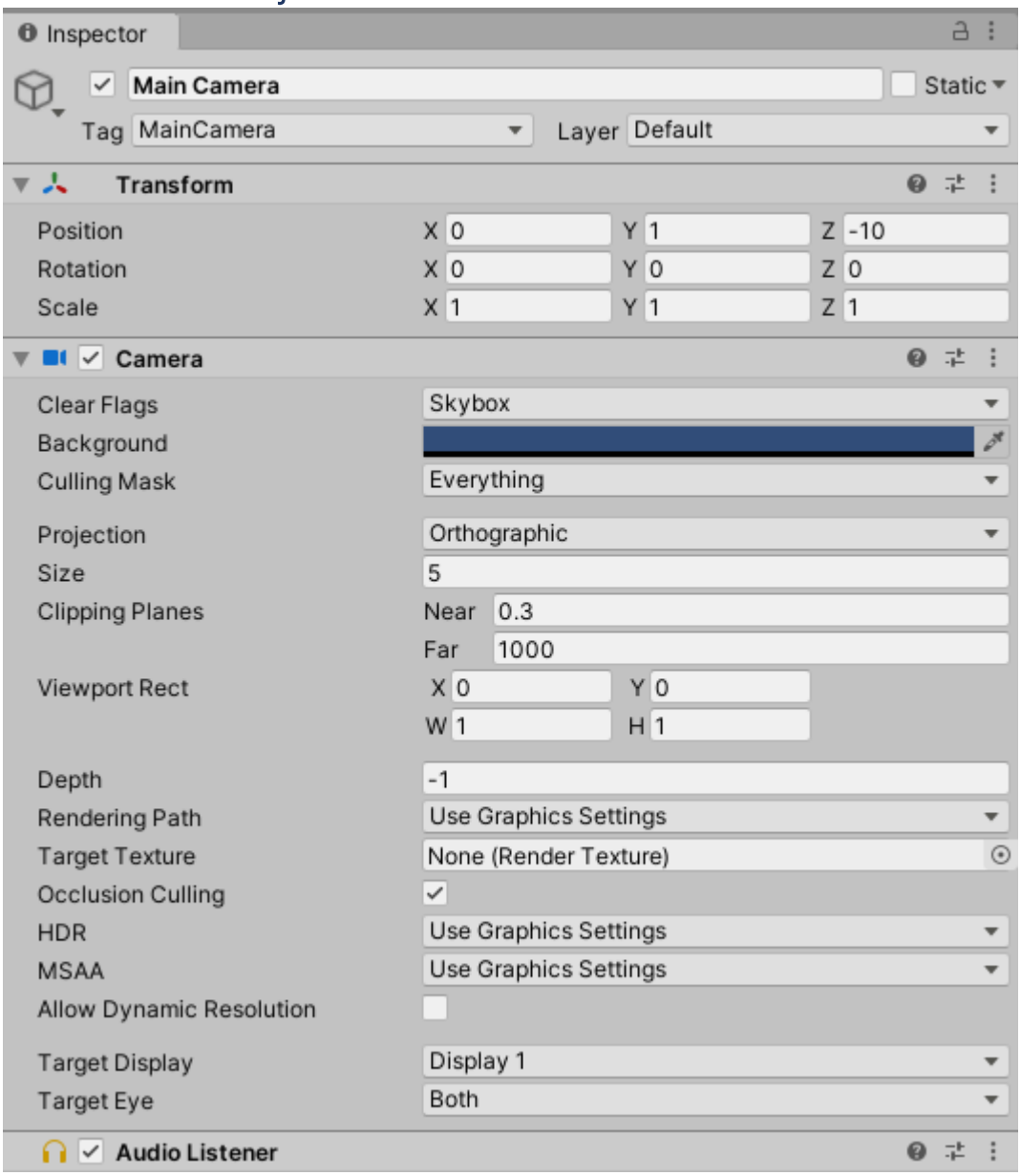
Fly Left Animation Frames



Fly Right Animation Frames



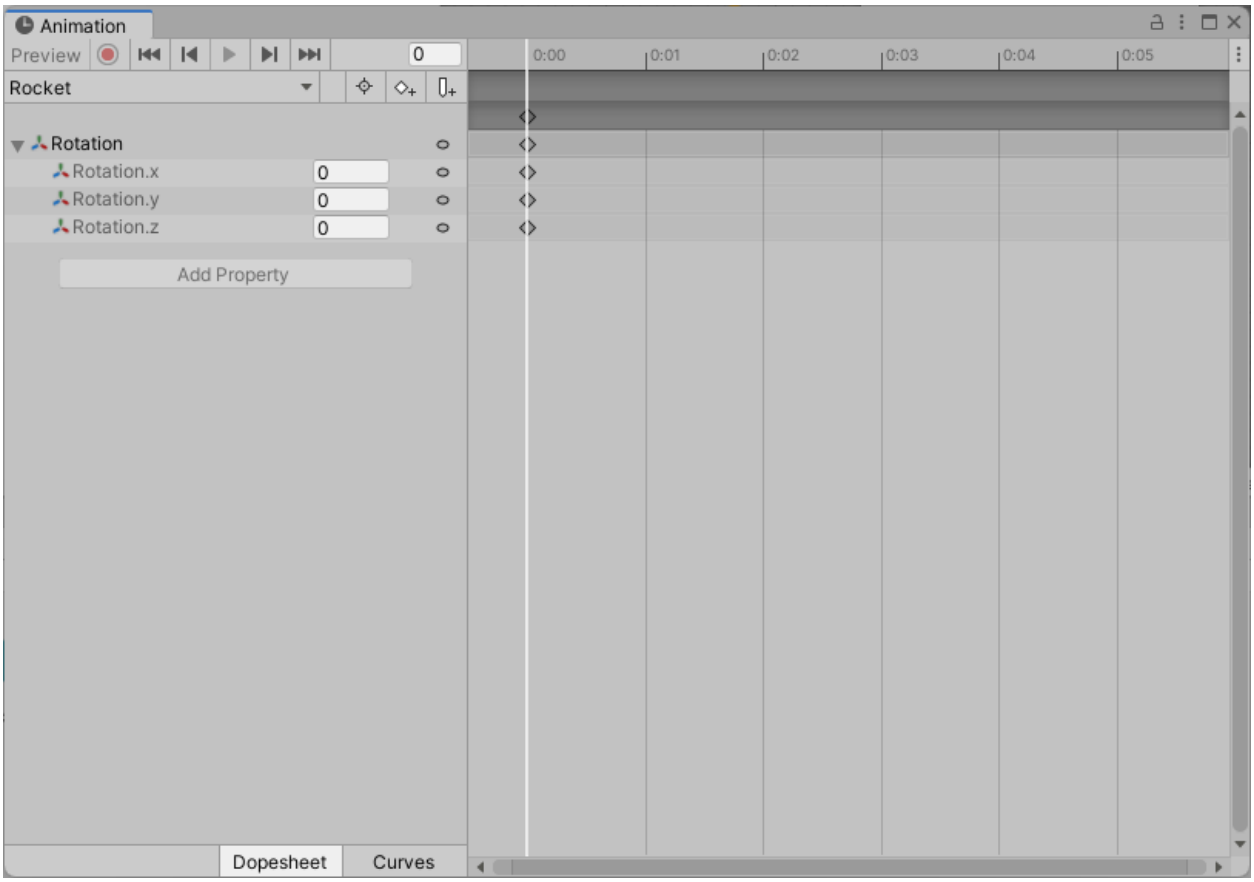
Main Camera Object



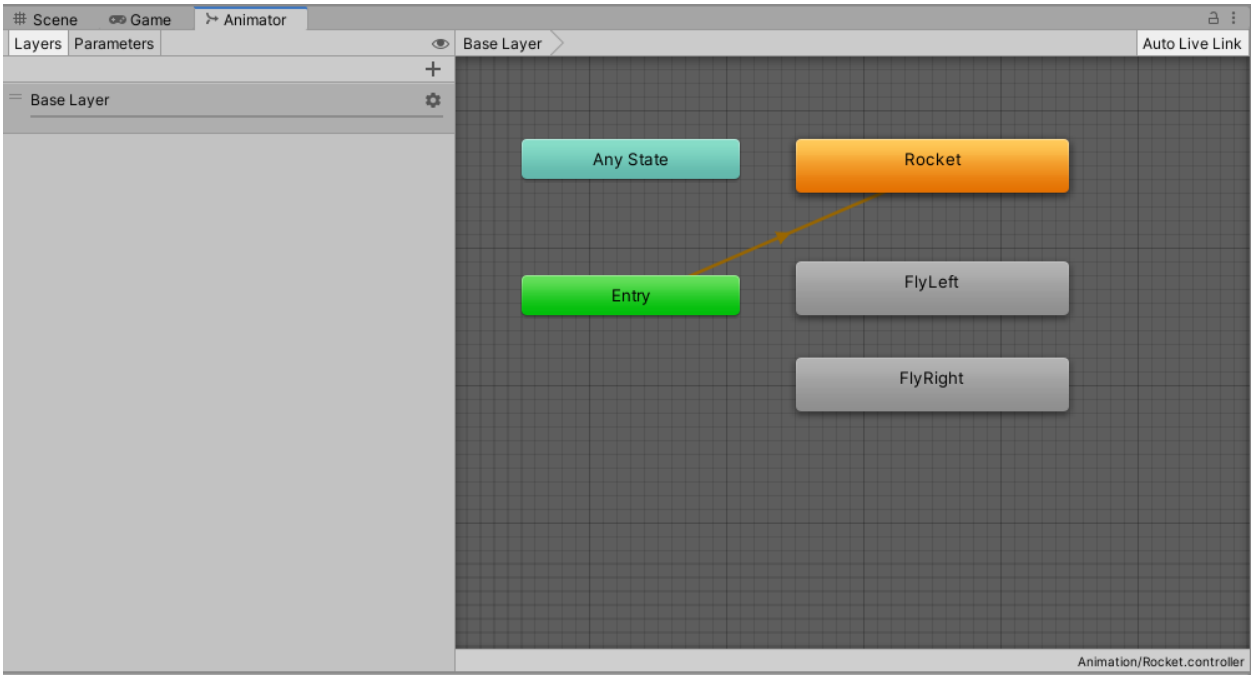
RocketAnimate.cs Script

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

public class RocketAnimate : MonoBehaviour
{
    [Header("Animator")]
    public Animator animator;

    void Start()
    {
        animator = GetComponent<Animator>();
    }

    void FixedUpdate()
    {
        float horizontal = Input.GetAxis("Horizontal");

        animator.SetFloat("HAxis", horizontal);
    }
}
```

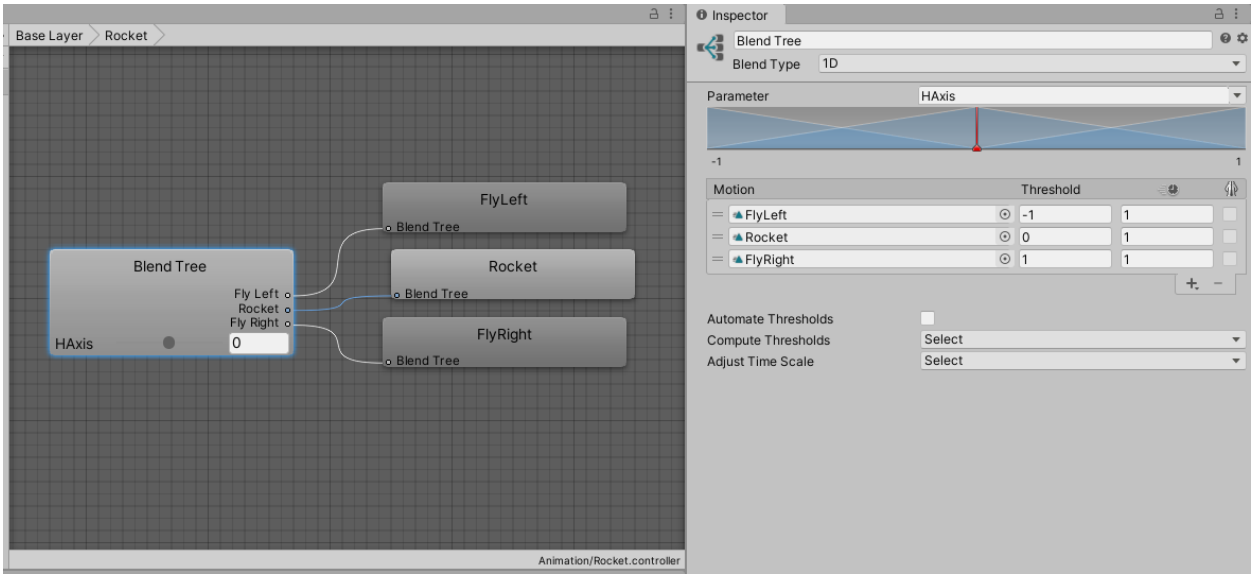
Rocket Animation Frames



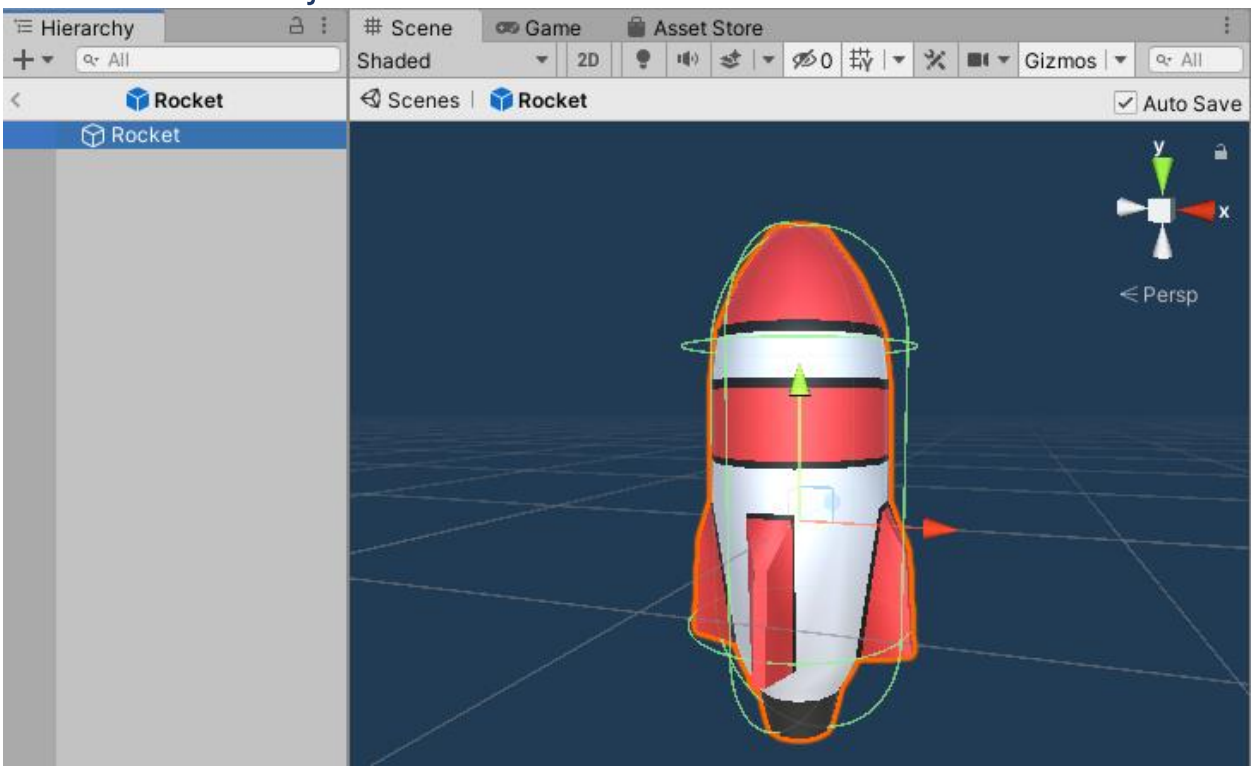
Rocket Animator Tree



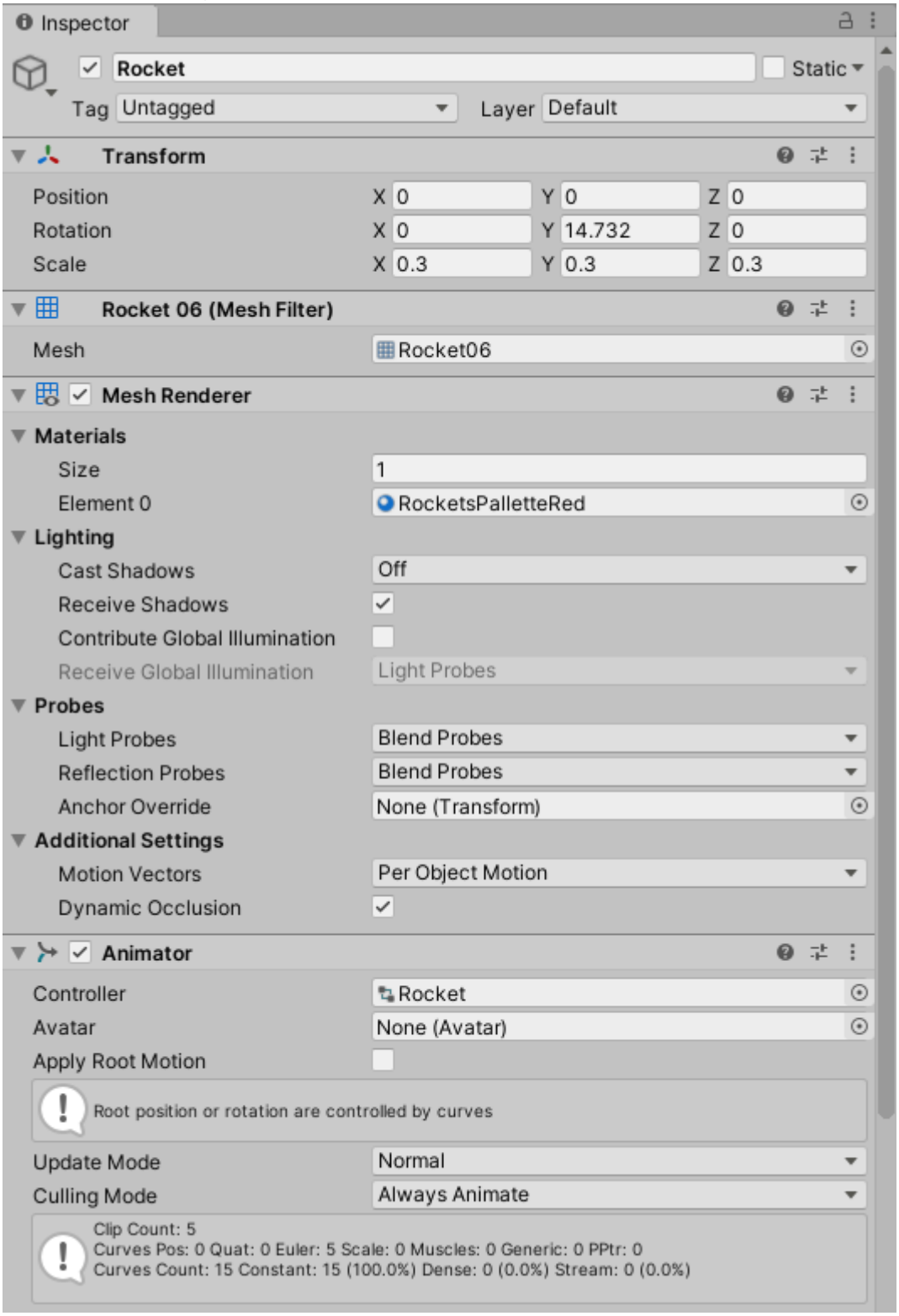
Rocket Blend Tree



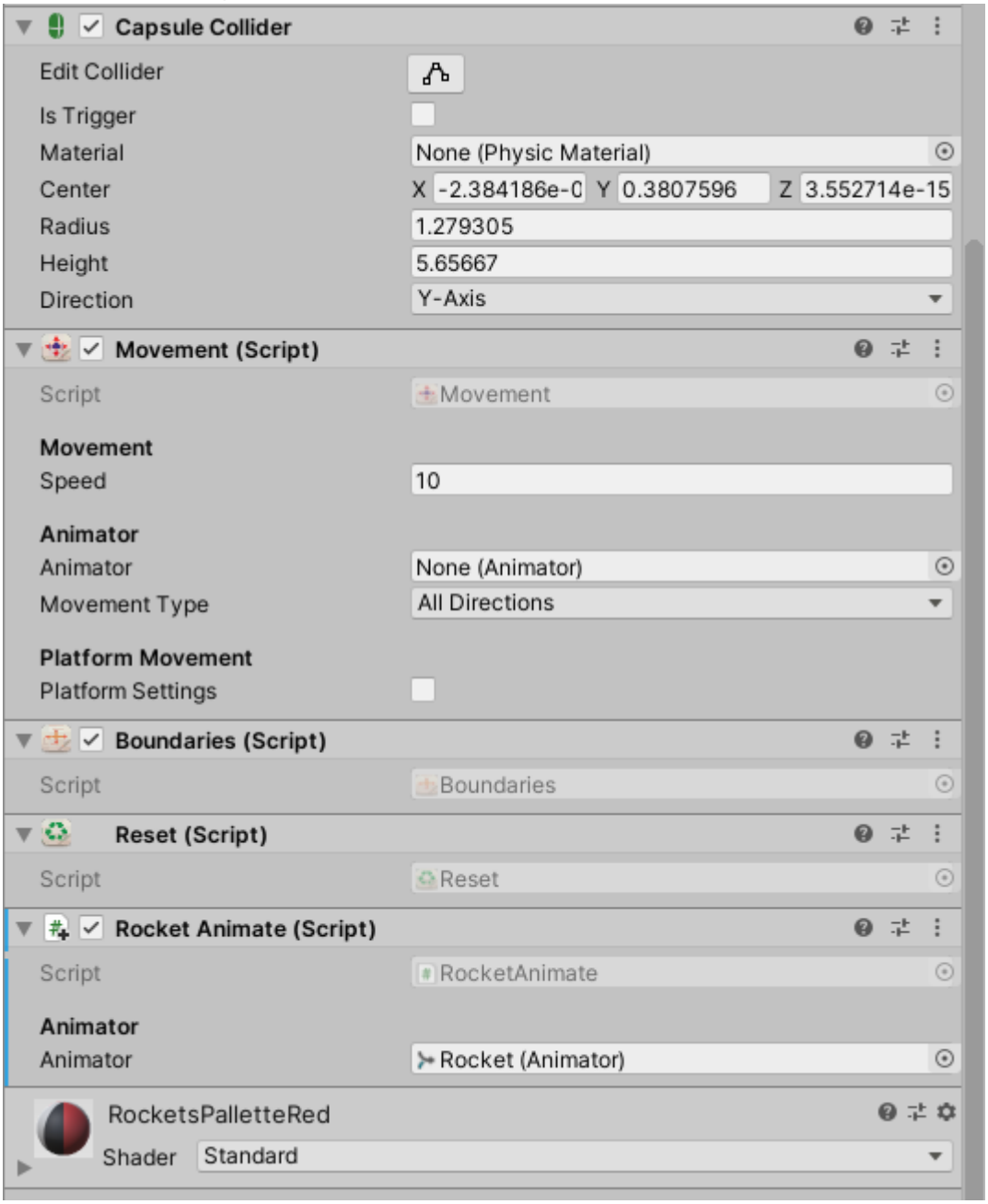
Rocket Prefab Object



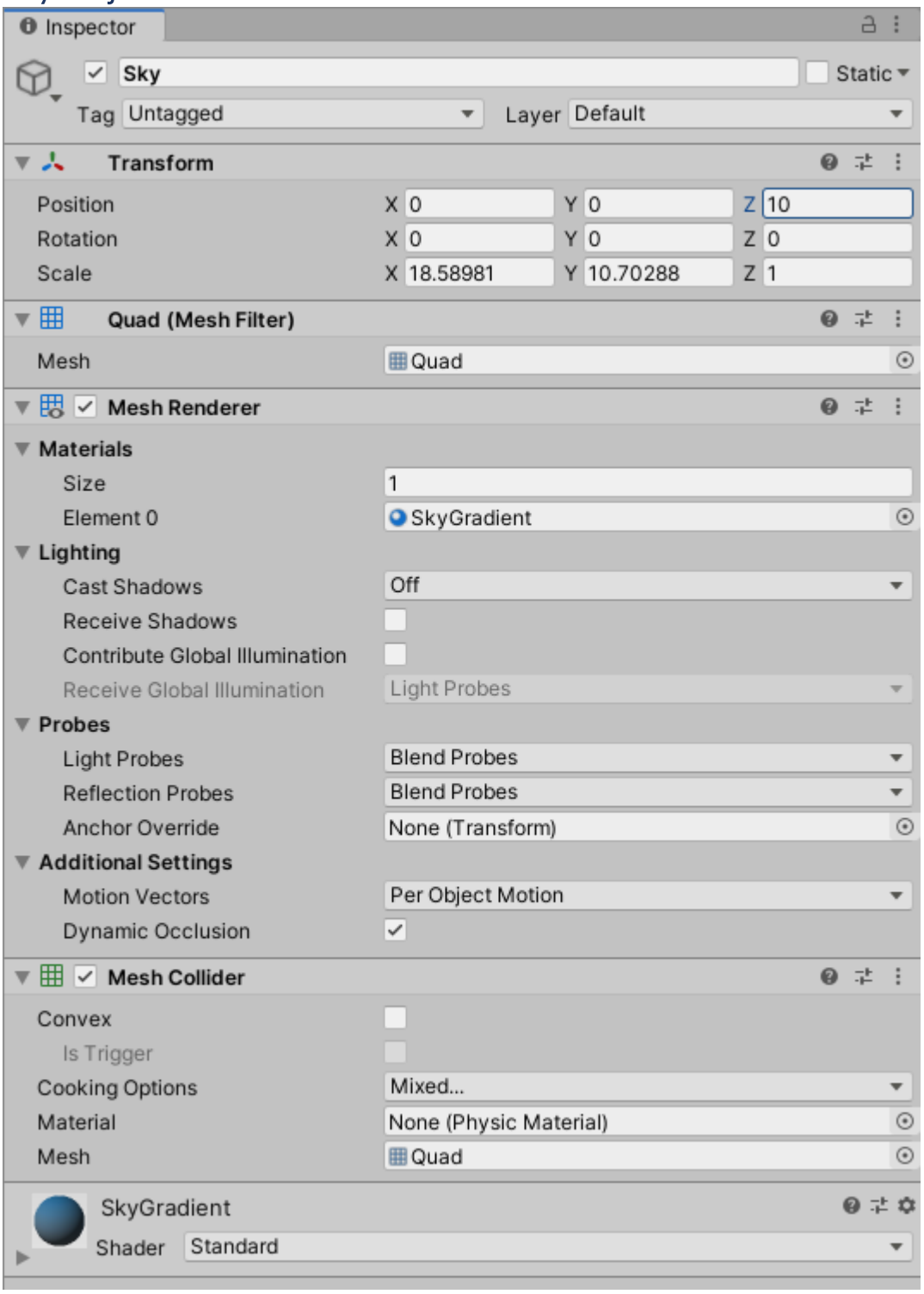
Rocket Prefab Object Continued



Rocket Prefab Object Continued

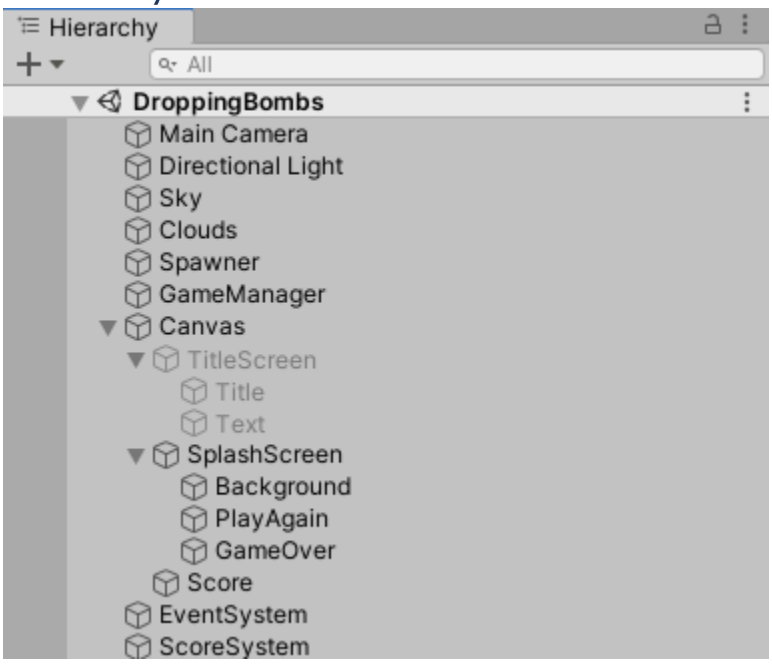


Sky Object

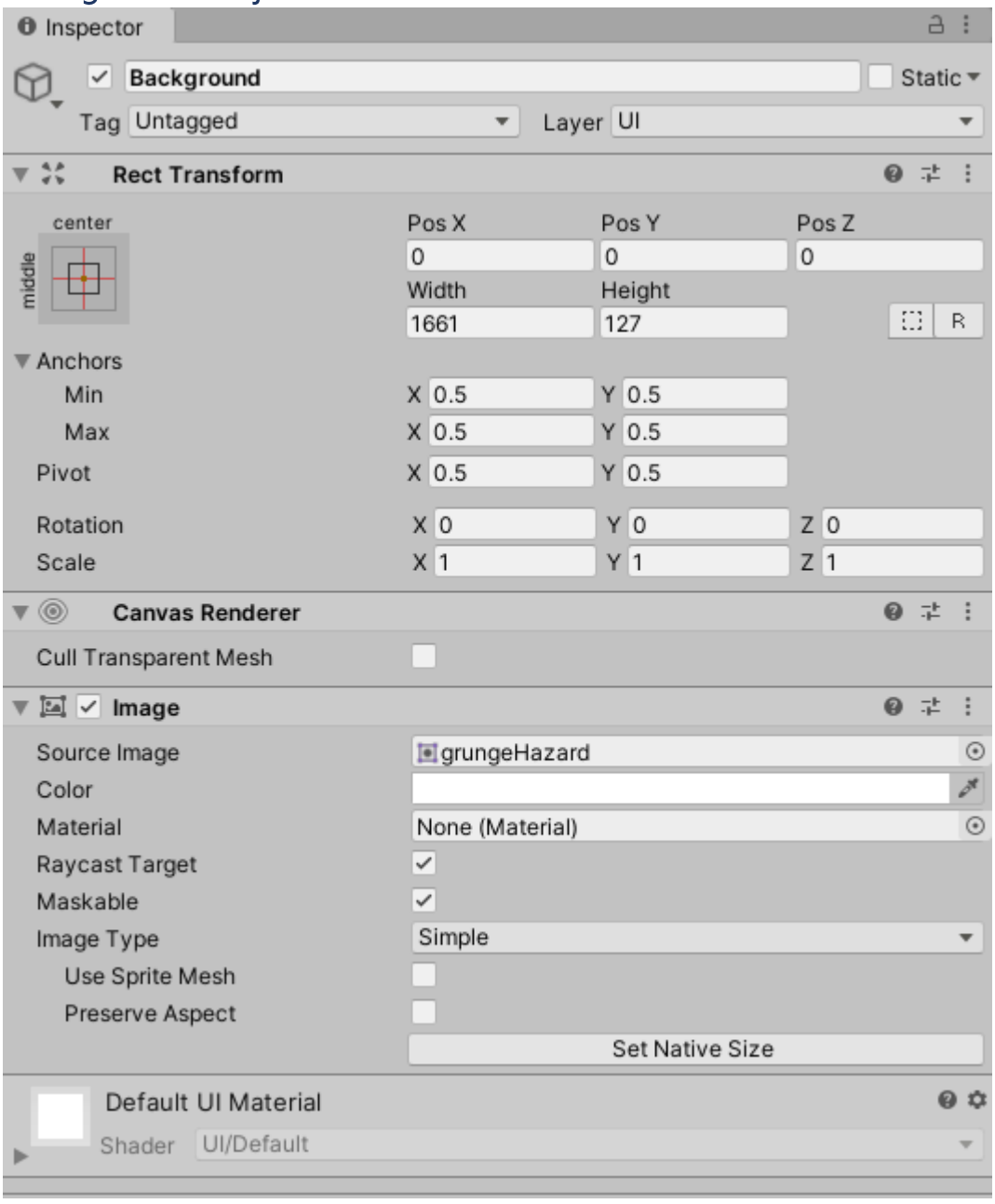


Dropping Bombs Part 3

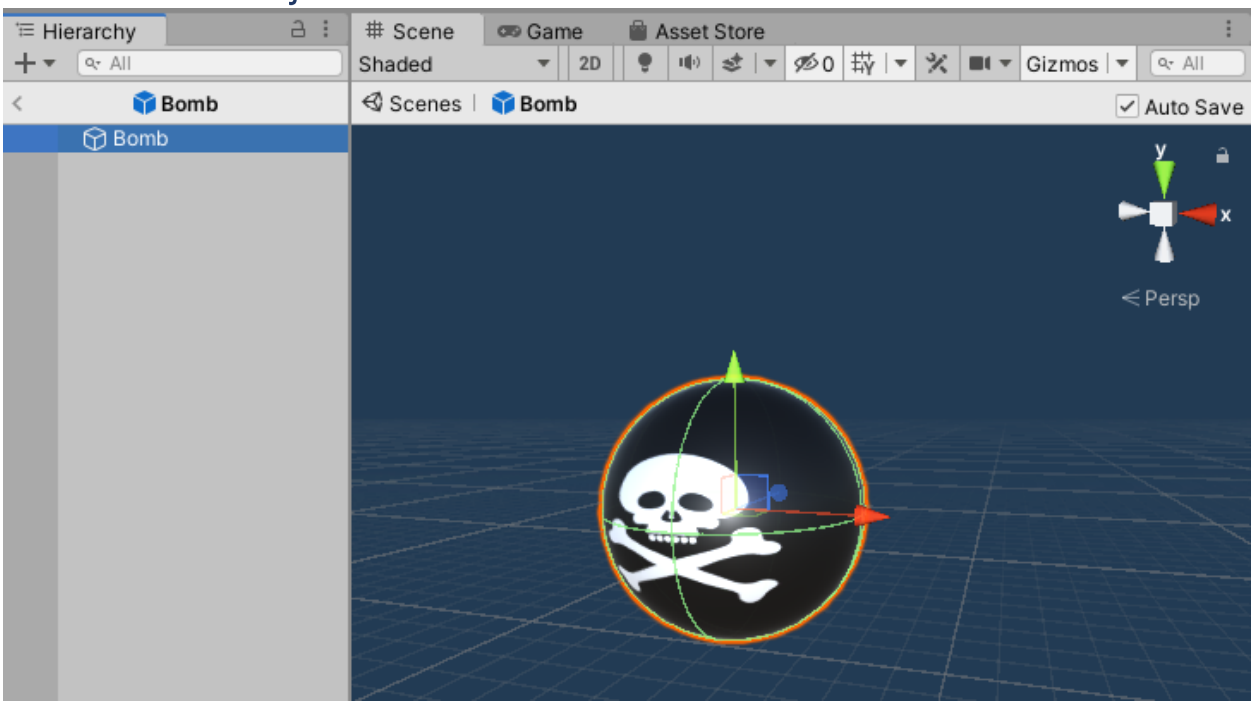
Hierarchy



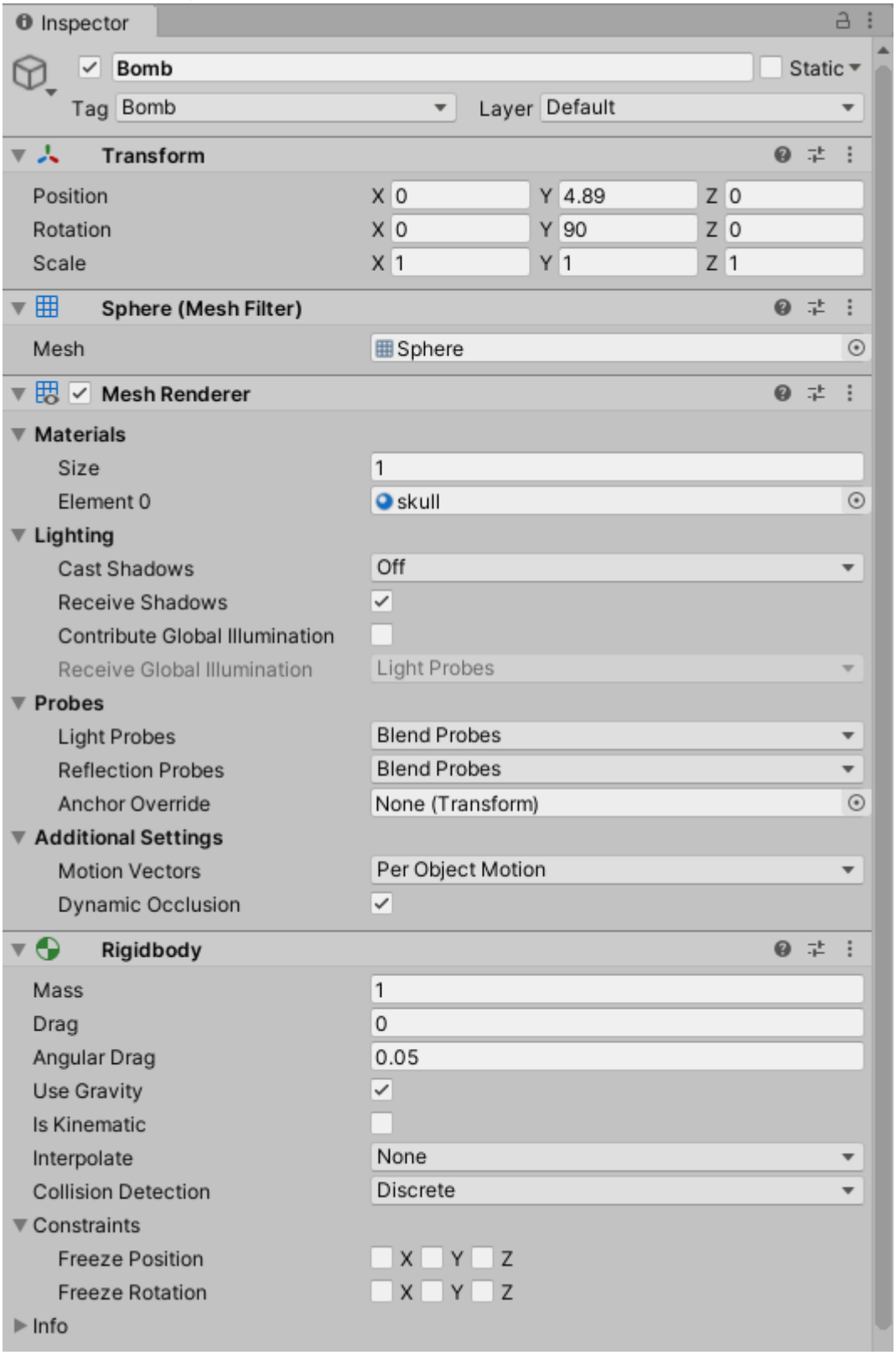
Background Object



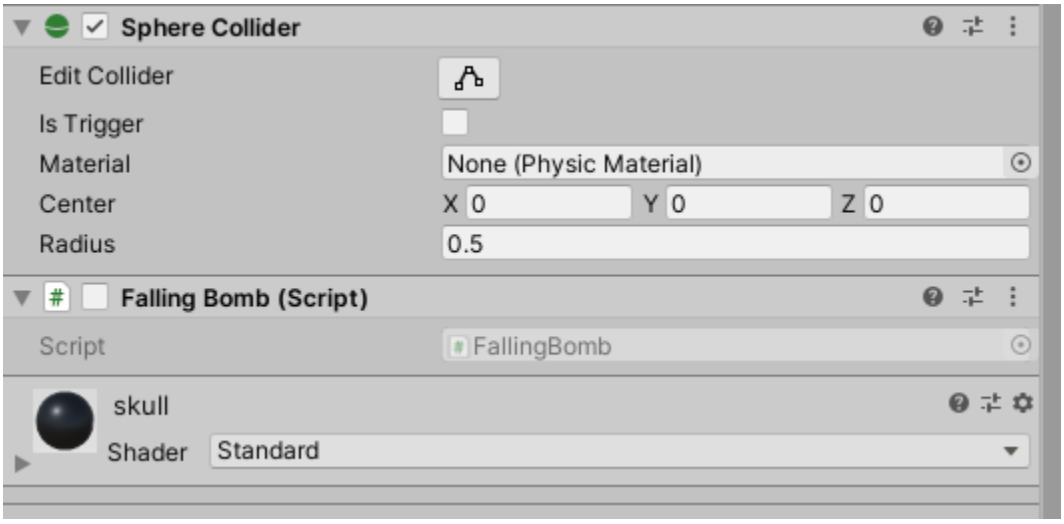
Bomb Prefab Object



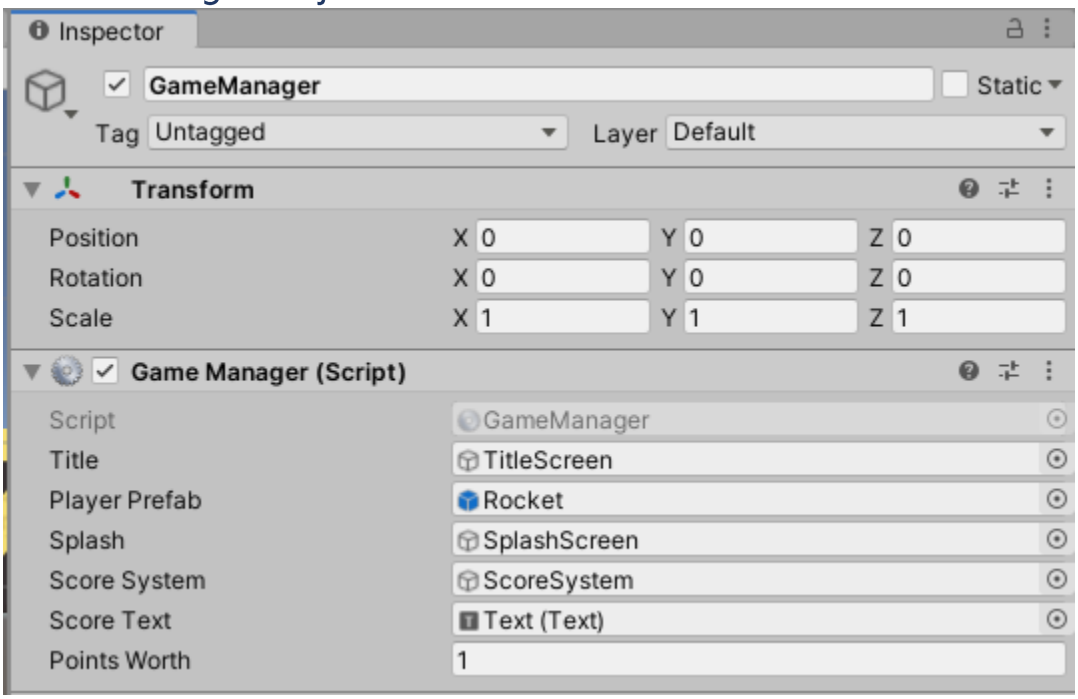
Bomb prefab Object Continued



Bomb Prefab Object Continued



GameManager Object

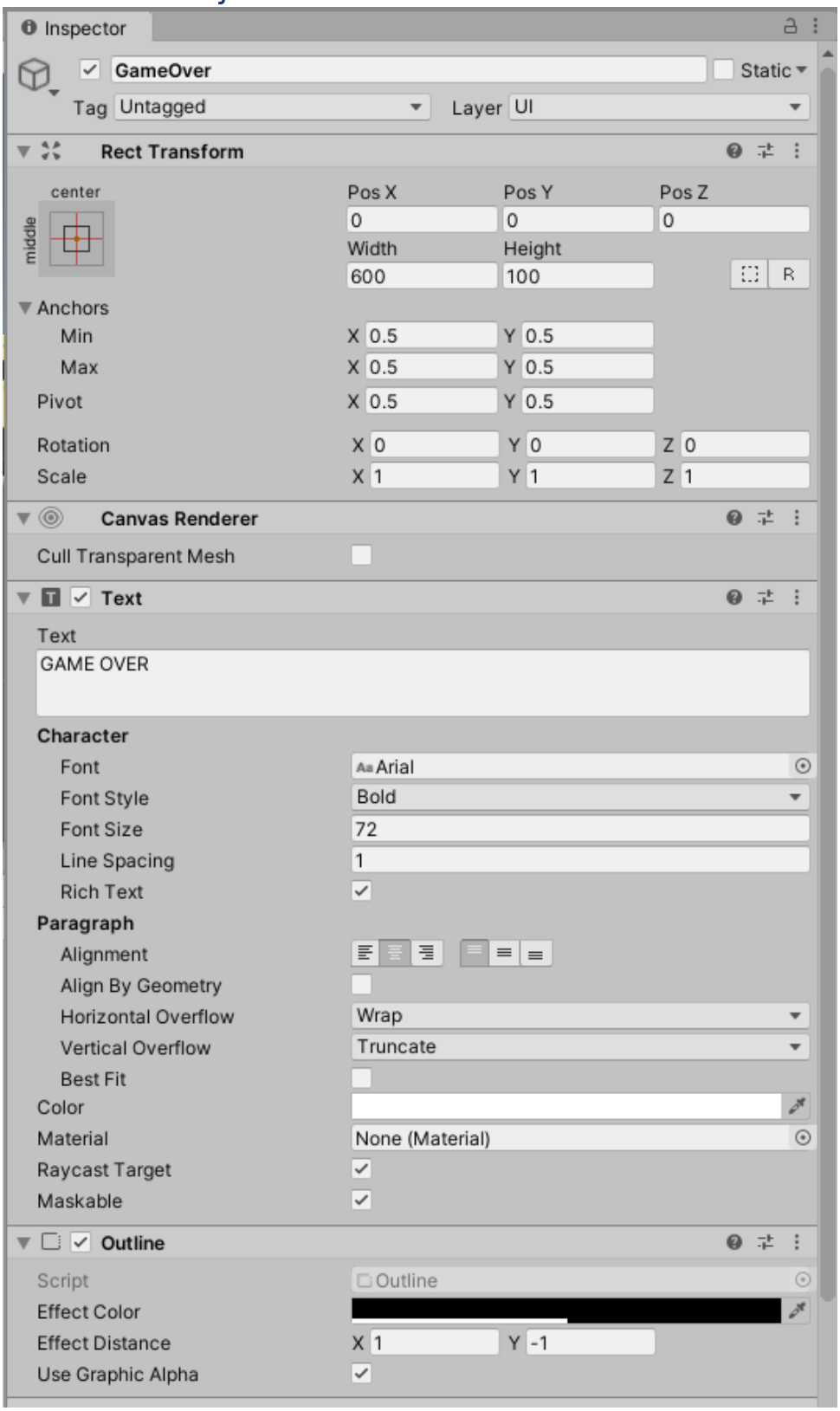


GameManager.cs Script

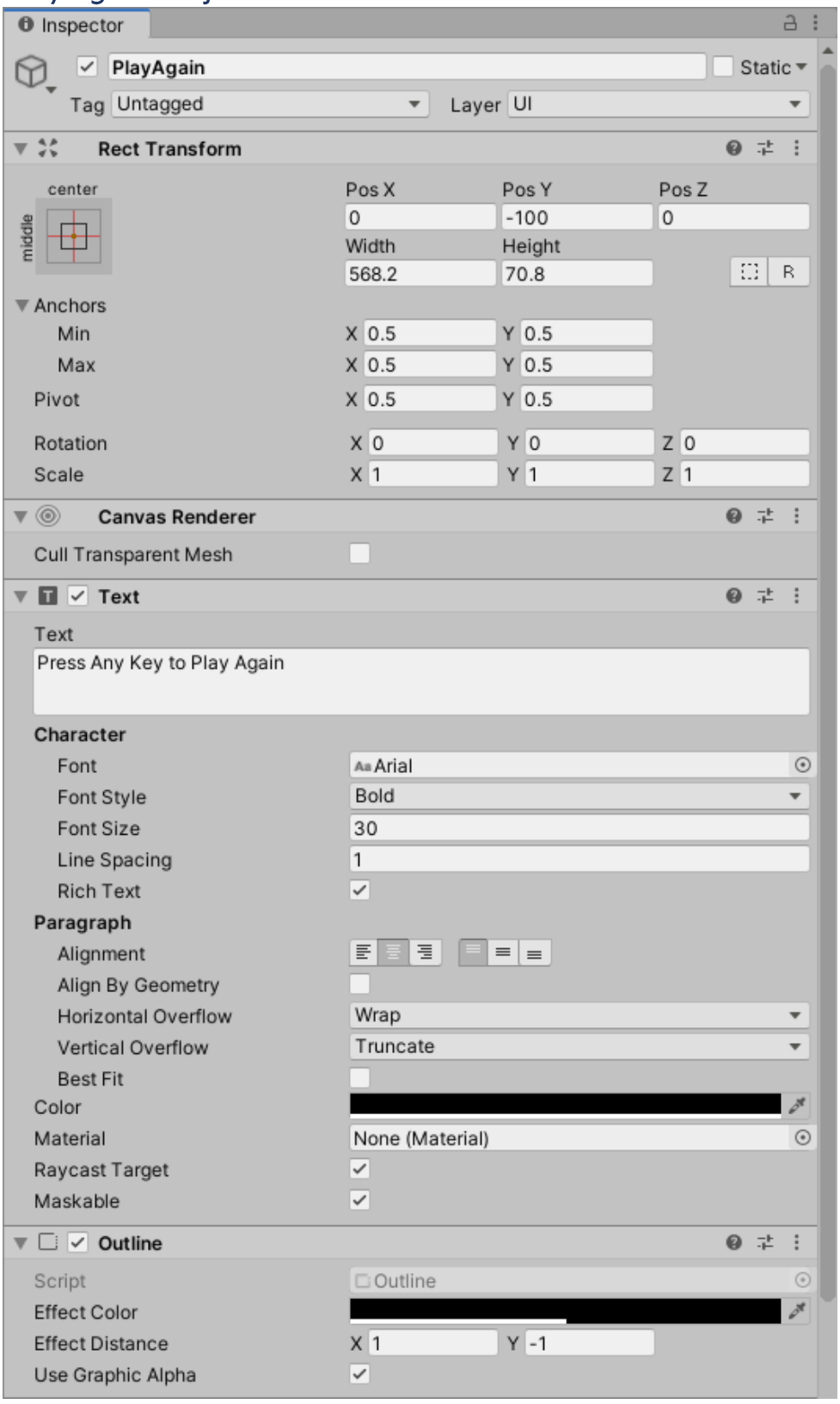
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class GameManager : MonoBehaviour
{
    private Spawner spawner;
    public GameObject title;
    private Vector2 screenBounds;
    public GameObject playerPrefab;
    private GameObject player;
    private bool gameStarted = false;
    public GameObject splash;
    public GameObject scoreSystem;
    public Text scoreText;
    public int pointsWorth = 1;
    private int score;

    void Awake()
    {
        spawner = GameObject.Find("Spawner").GetComponent<Spawner>();
        screenBounds = Camera.main.ScreenToWorldPoint(
            new Vector3(Screen.width, Screen.height, Camera.main.transform.position.z)
            );
        player = playerPrefab;
        scoreText.enabled = false;
    }

    void Start()
    {
        spawner.active = false;
        title.SetActive(true);
        splash.SetActive(false);
    }

    void Update()
    {
        if (!gameStarted)
        {
            if (Input.anyKeyDown)
            {
                ResetGame();
            }
        }
        else
        {
            if (!player)
            {
                OnPlayerKilled();
            }
        }

        var nextBomb = GameObject.FindGameObjectsWithTag("Bomb");
```

```
        foreach (GameObject bombObject in nextBomb)
        {
            if (!gameStarted)
            {
                Destroy(bombObject);
            }
            else if (bombObject.transform.position.y < (-screenBounds.y))
            {
                scoreSystem.GetComponent<Score>().AddScore(pointsWorth);
                Destroy(bombObject);
            }
        }
    }

    void ResetGame()
    {
        spawner.active = true;
        title.SetActive(false);
        splash.SetActive(false);
        player = Instantiate(playerPrefab, new Vector3(0, 0, 0),
playerPrefab.transform.rotation);
        gameStarted = true;
        scoreText.enabled = true;
        scoreSystem.GetComponent<Score>().score = 0;
        scoreSystem.GetComponent<Score>().Start();
    }

    void OnPlayerKilled()
    {
        spawner.active = false;
        gameStarted = false;
        splash.SetActive(true);
        score = scoreSystem.GetComponent<Score>().score;
    }
}
```

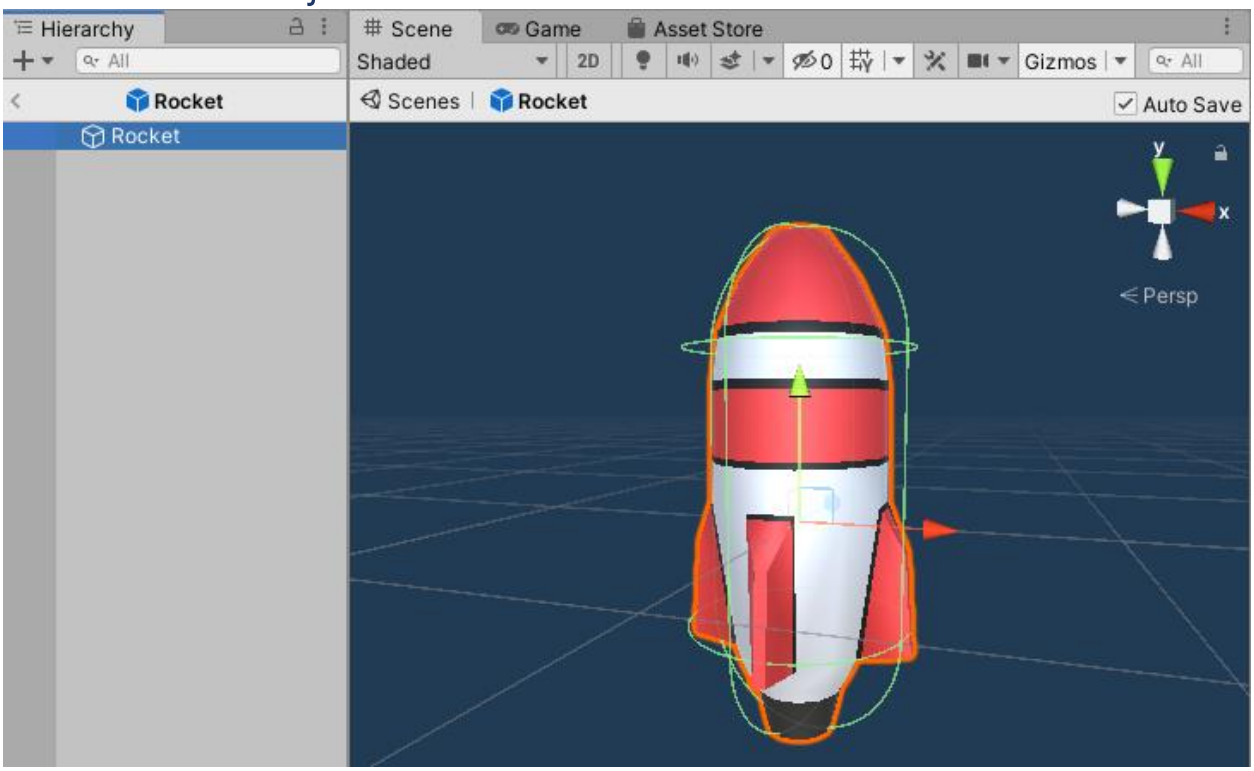
GameOver Object



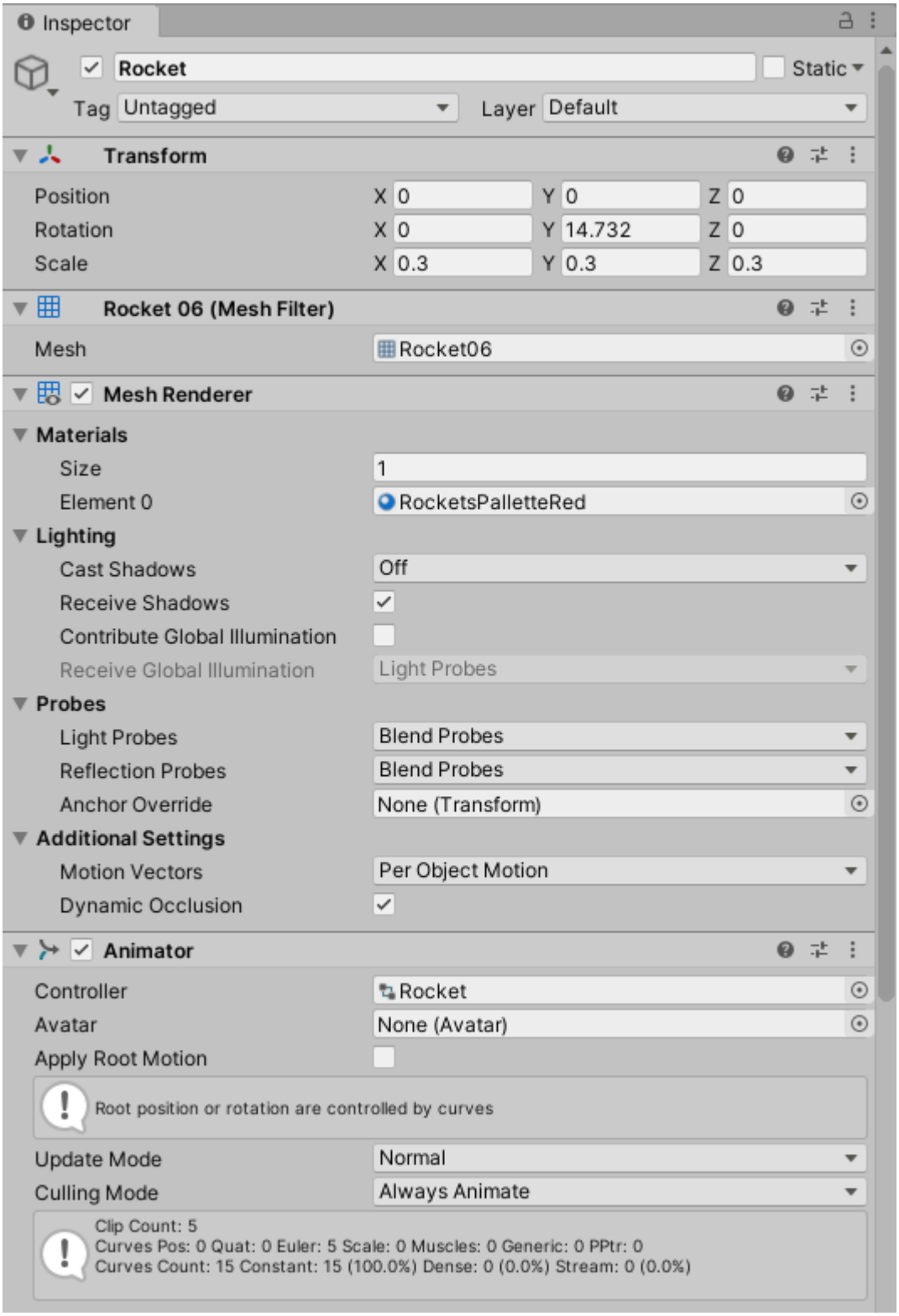
PlayAgain Object



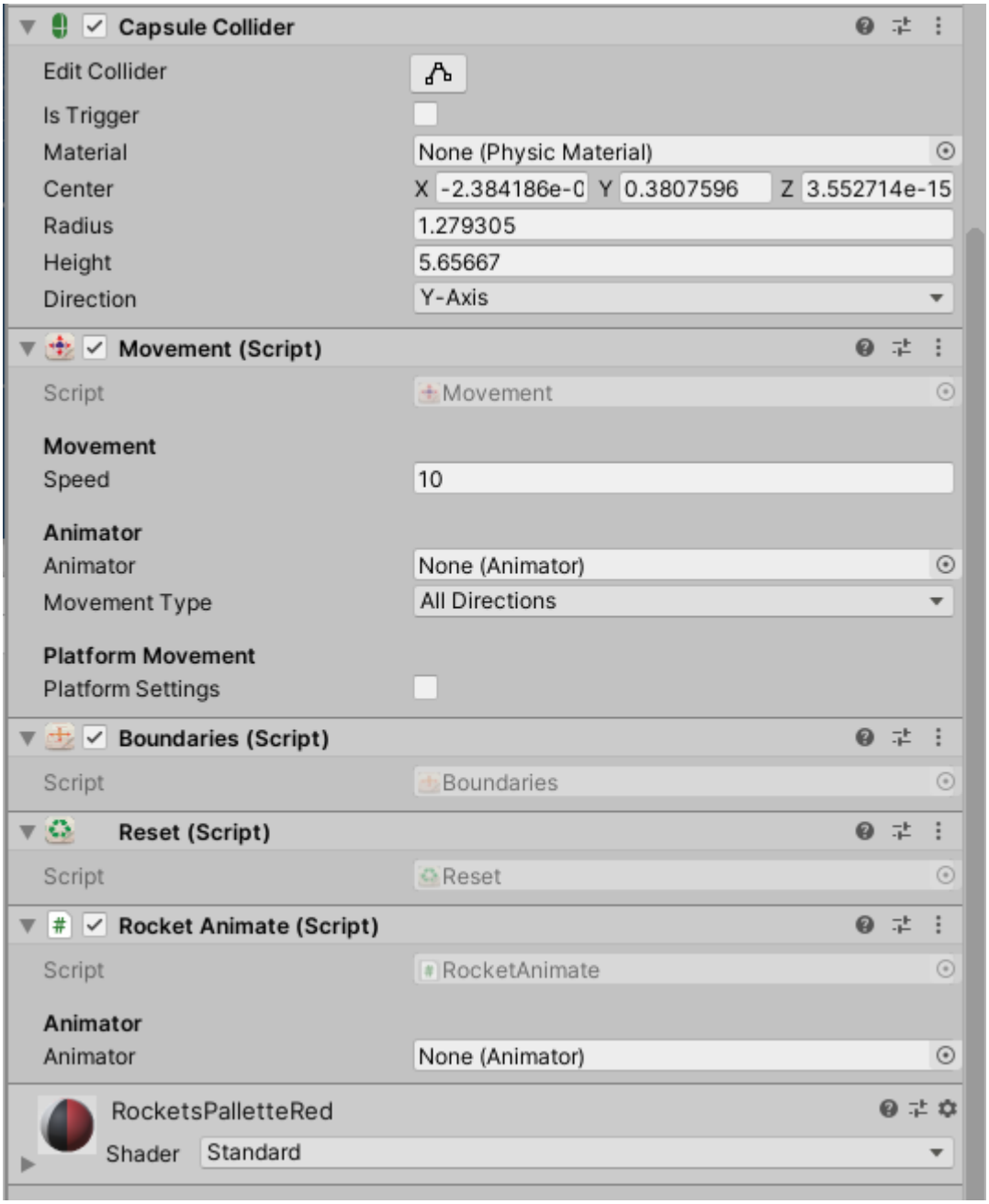
Rocket Prefab Object



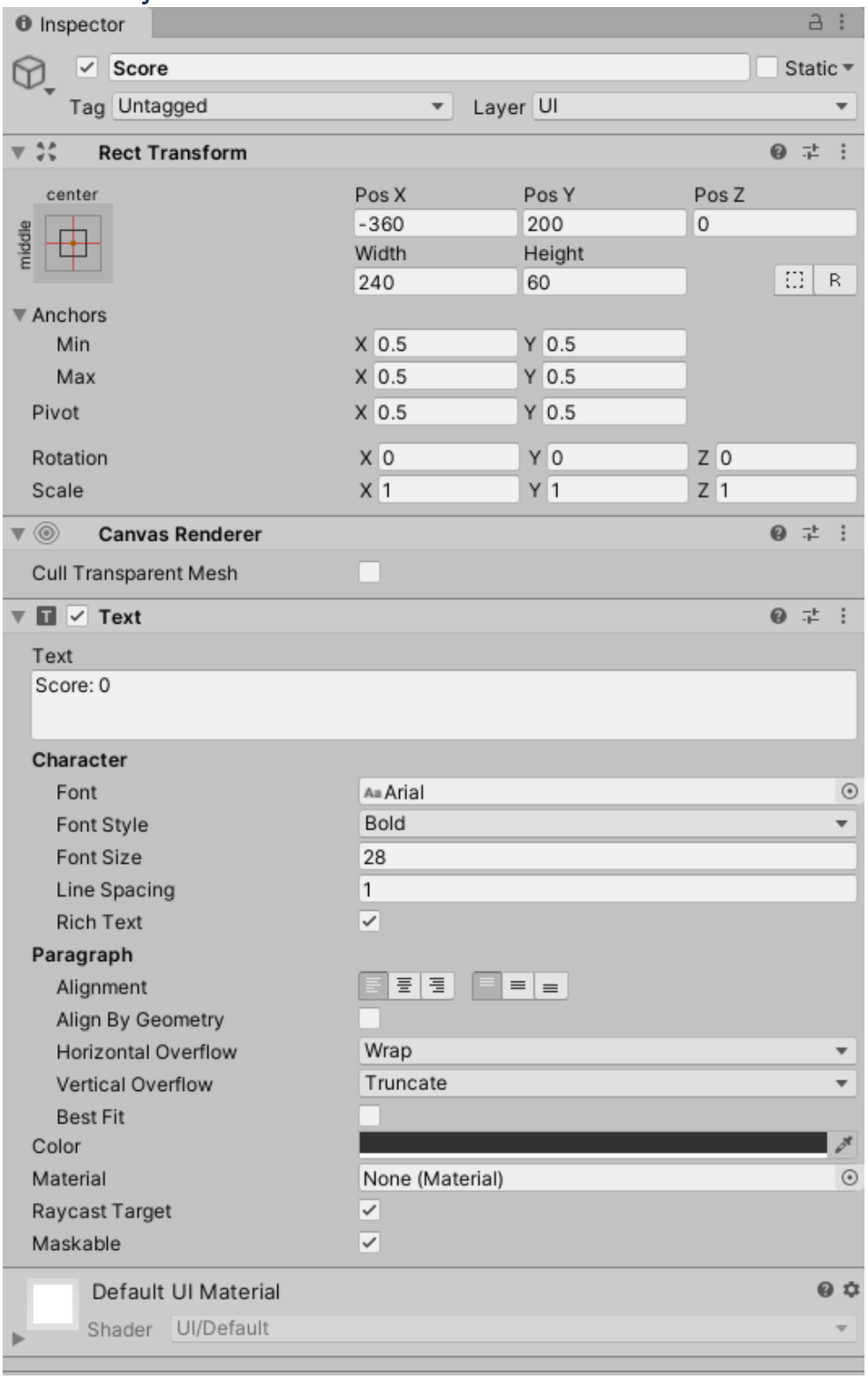
Rocket Prefab Object Continued



Rocket Prefab Object Continued



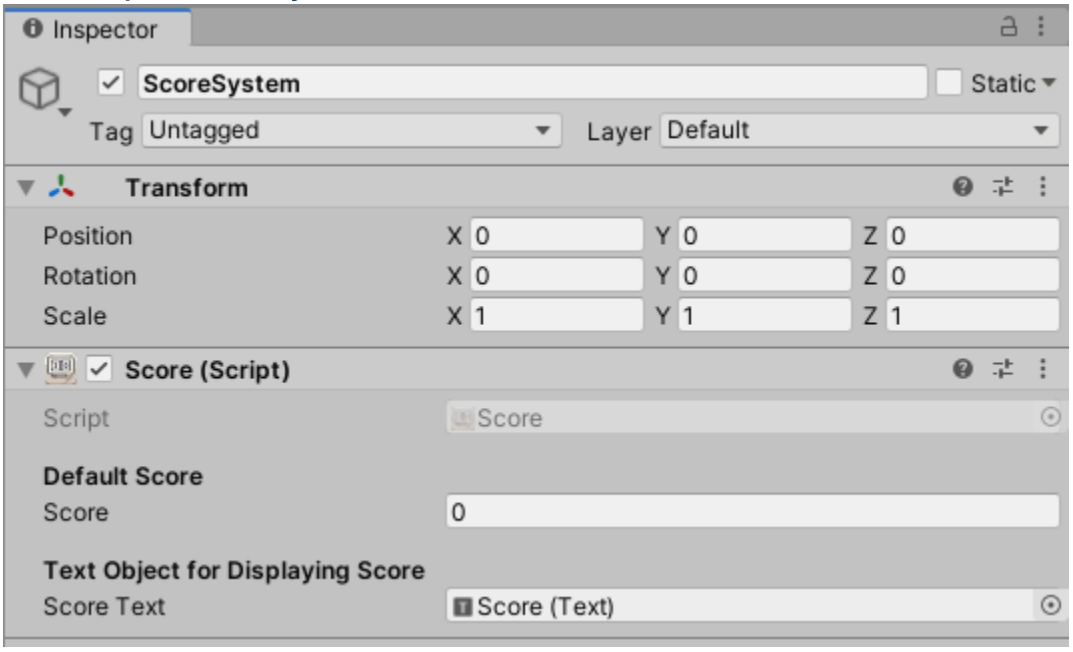
Score Object



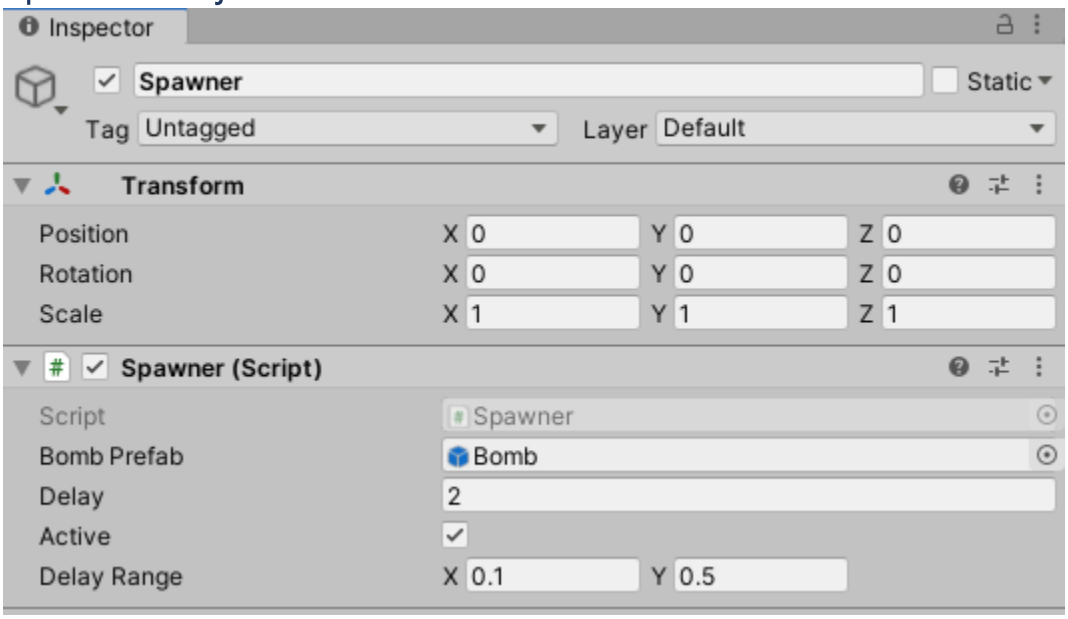
Score.cs Script

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

public class Score : MonoBehaviour
{
    [Header("Default Score")]
    public int score = 0;
    [Header("Text Object for Displaying Score")]
    public Text scoreText;

    public void Start()
    {
        scoreText.text = "Score: " + score.ToString();
    }

    public void AddScore(int points)
    {
        score = score + points;
        scoreText.text = "Score: " + score.ToString();
    }
}
```

ScoreSystem Object



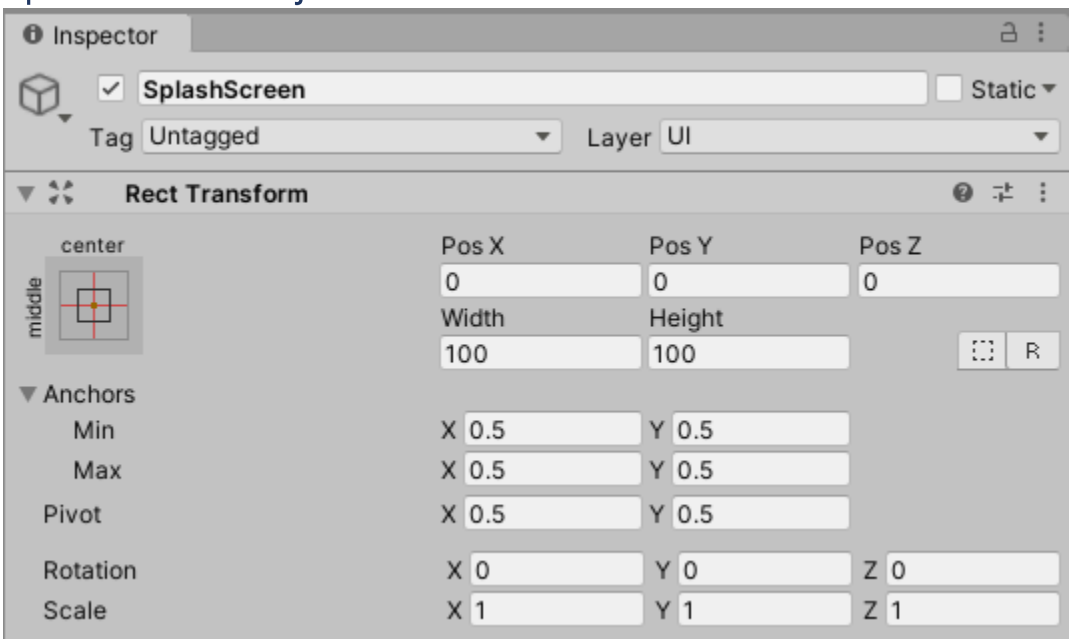
Spawner Object

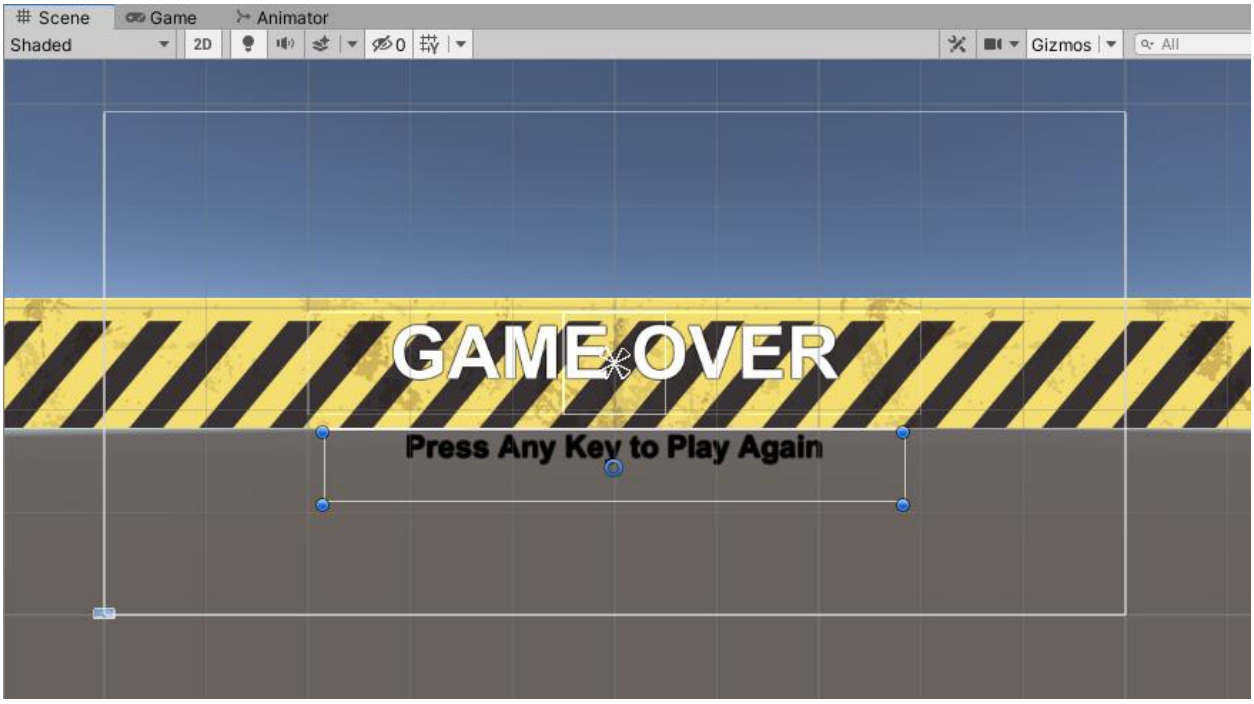


Spawner.cs Script

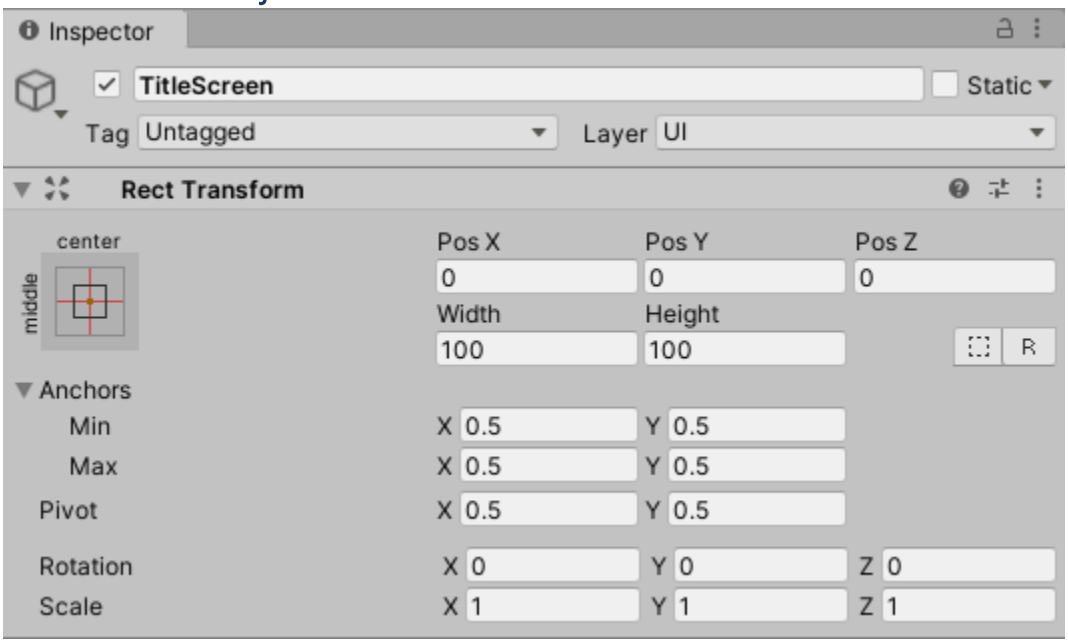
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Spawner : MonoBehaviour
{
    public GameObject bombPrefab;
    public float delay = 2.0f;
    public bool active = true;
    public Vector2 delayRange = new Vector2(1, 2);

    private Vector2 screenBounds;
    private float objectWidth;
    private float objectHeight;

    void Start()
    {
        ResetDelay();
        StartCoroutine(EnemyGenerator());

        screenBounds = Camera.main.ScreenToWorldPoint(
            new Vector3(Screen.width, Screen.height, Camera.main.transform.position.z)
            );
        objectWidth = bombPrefab.GetComponent<MeshRenderer>().bounds.size.x / 2;
        objectHeight = bombPrefab.GetComponent<MeshRenderer>().bounds.size.y / 2;
    }

    IEnumerator EnemyGenerator()
    {
        yield return new WaitForSeconds(delay);
        if (active)
        {
            float randomX = Random.Range(screenBounds.x - objectWidth, screenBounds.x * -
1 + objectWidth);
            float spawnY = (screenBounds.y + objectHeight) + 5;

            Instantiate(bombPrefab, new Vector3(randomX, spawnY, 0),
bombPrefab.transform.rotation);
            ResetDelay();
        }

        StartCoroutine(EnemyGenerator());
    }

    void ResetDelay()
    {
        delay = Random.Range(delayRange.x, delayRange.y);
    }
}
```

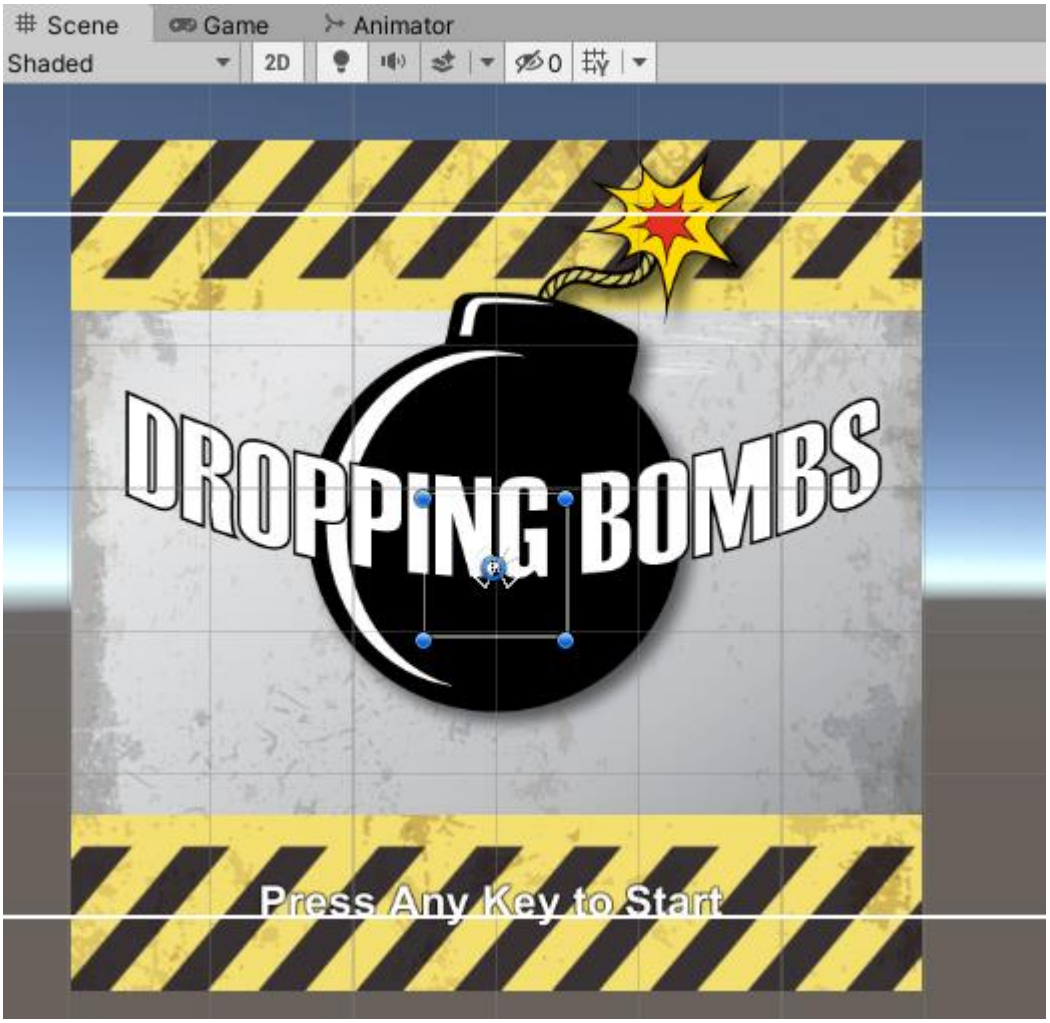
SplashScreen Object



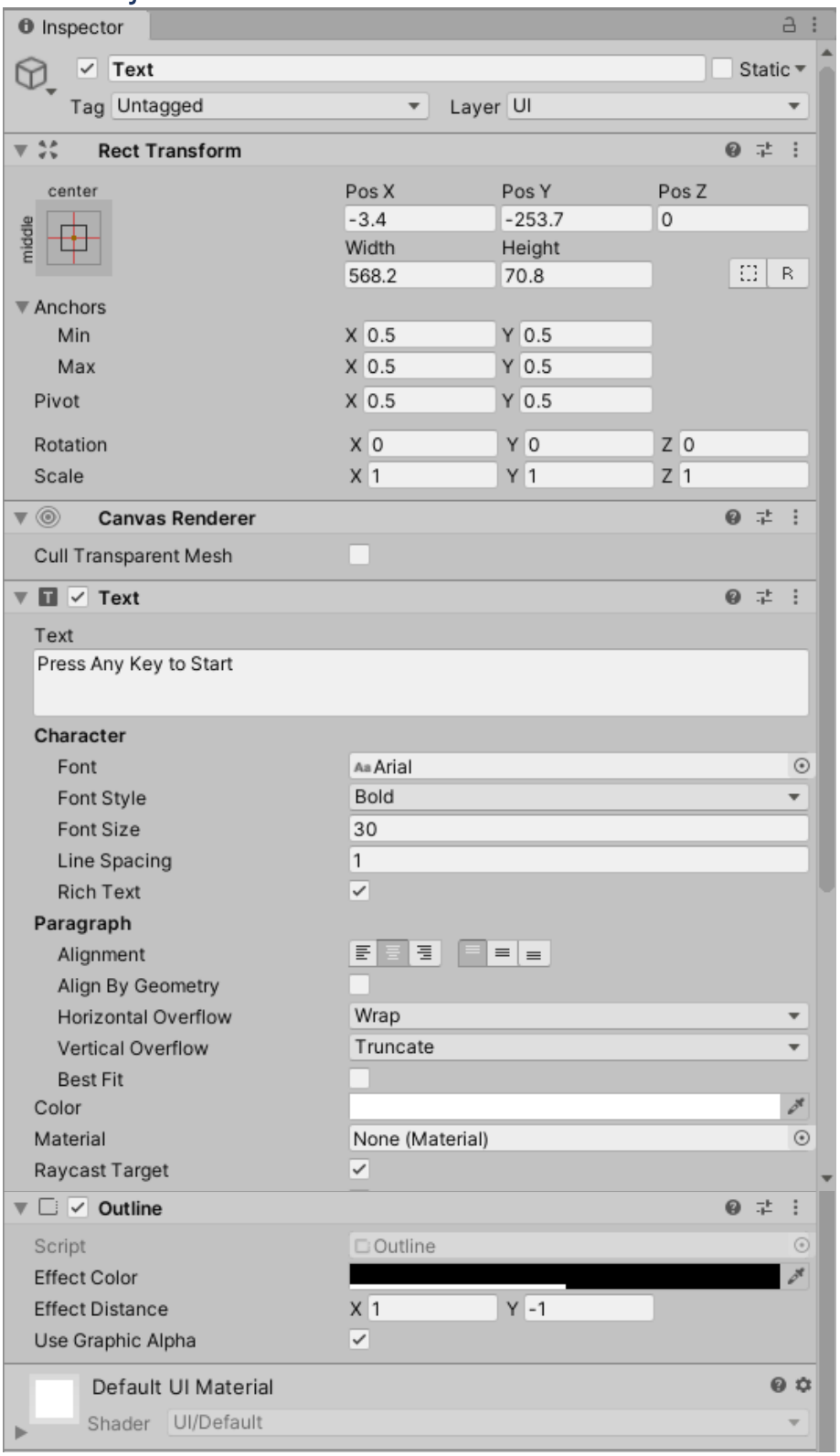


TitleScreen Object





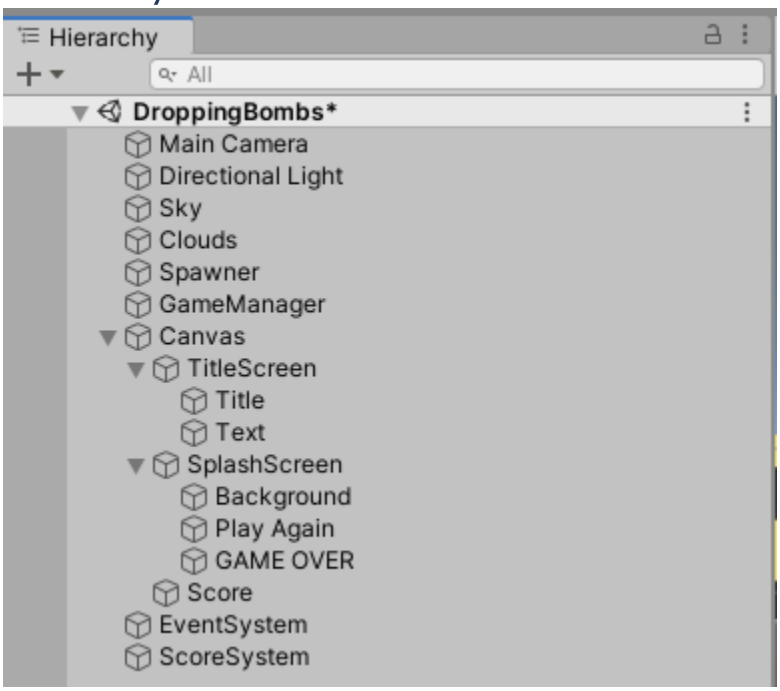
Text Object



Title Object

Dropping Bombs Part 4

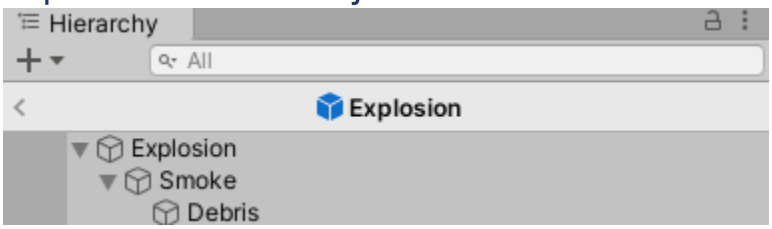
Hierarchy

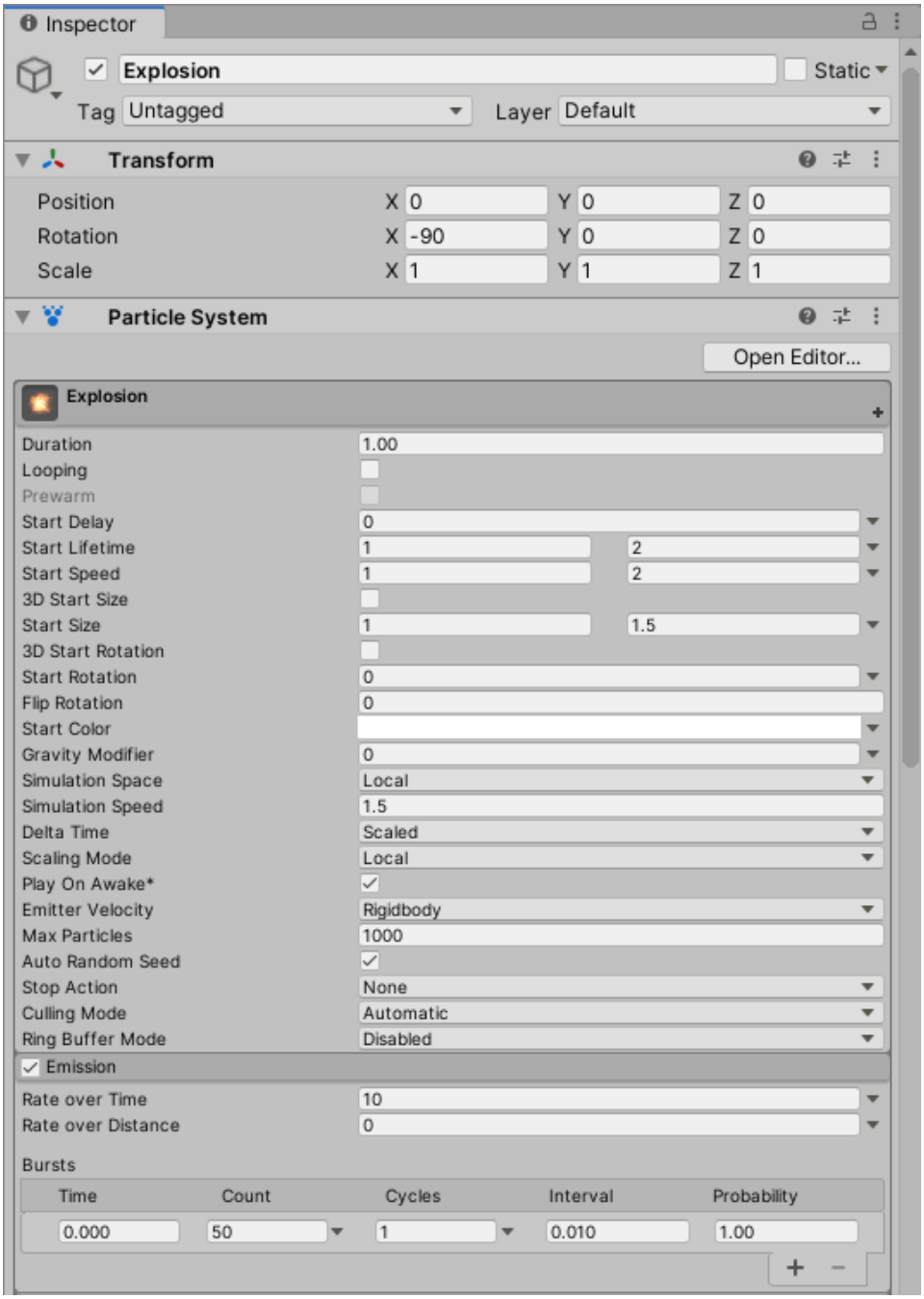


ExplosionClear.cs Script

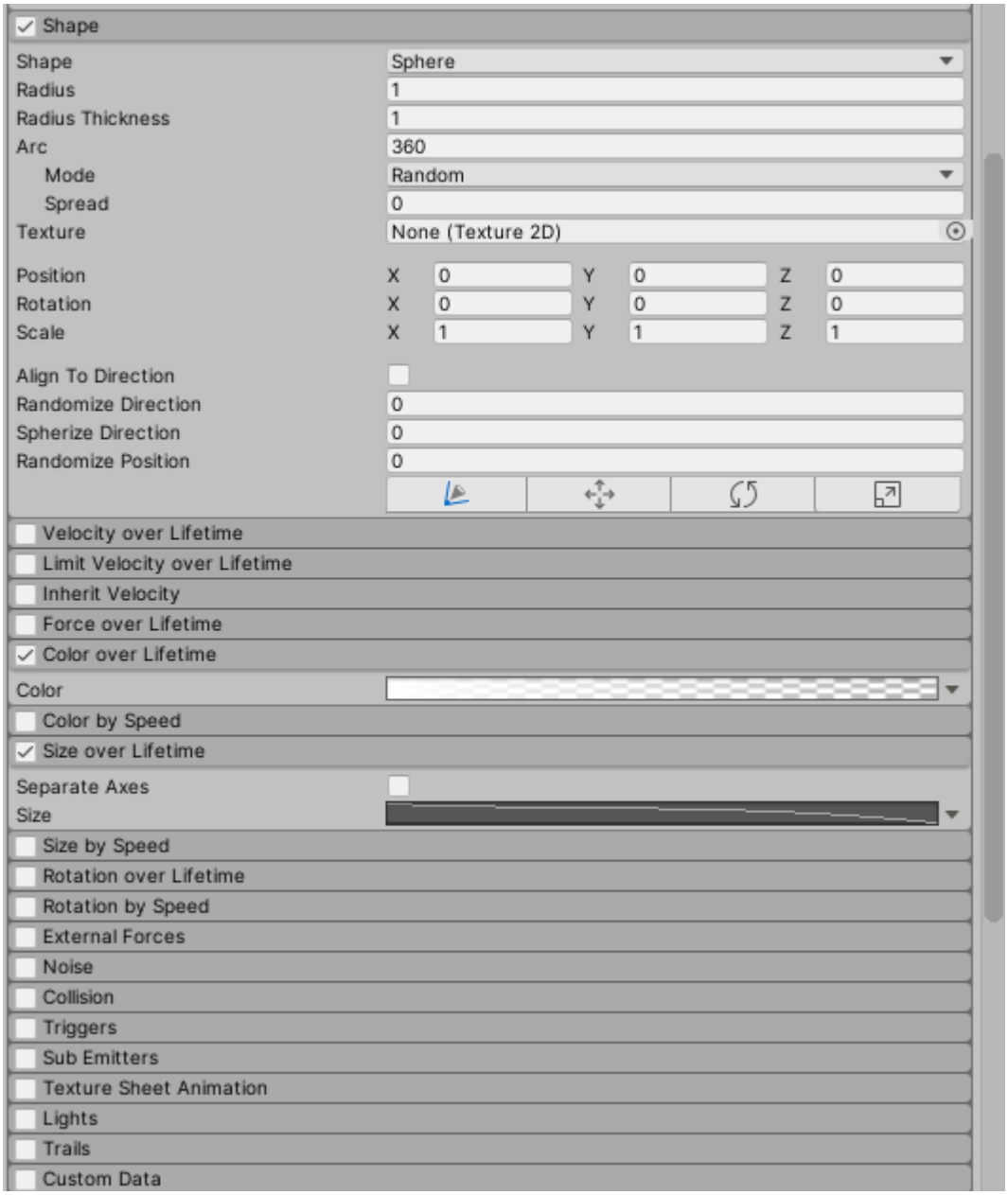
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ExplosionClear : MonoBehaviour
{
    private ParticleSystem particleSmoke;
    private void Awake()
    {
        particleSmoke = gameObject.GetComponentInChildren<ParticleSystem>();
    }

    void Update()
    {
        if (!particleSmoke.IsAlive())
        {
            Destroy(gameObject);
        }
    }
}
```

Explosion Prefab Object

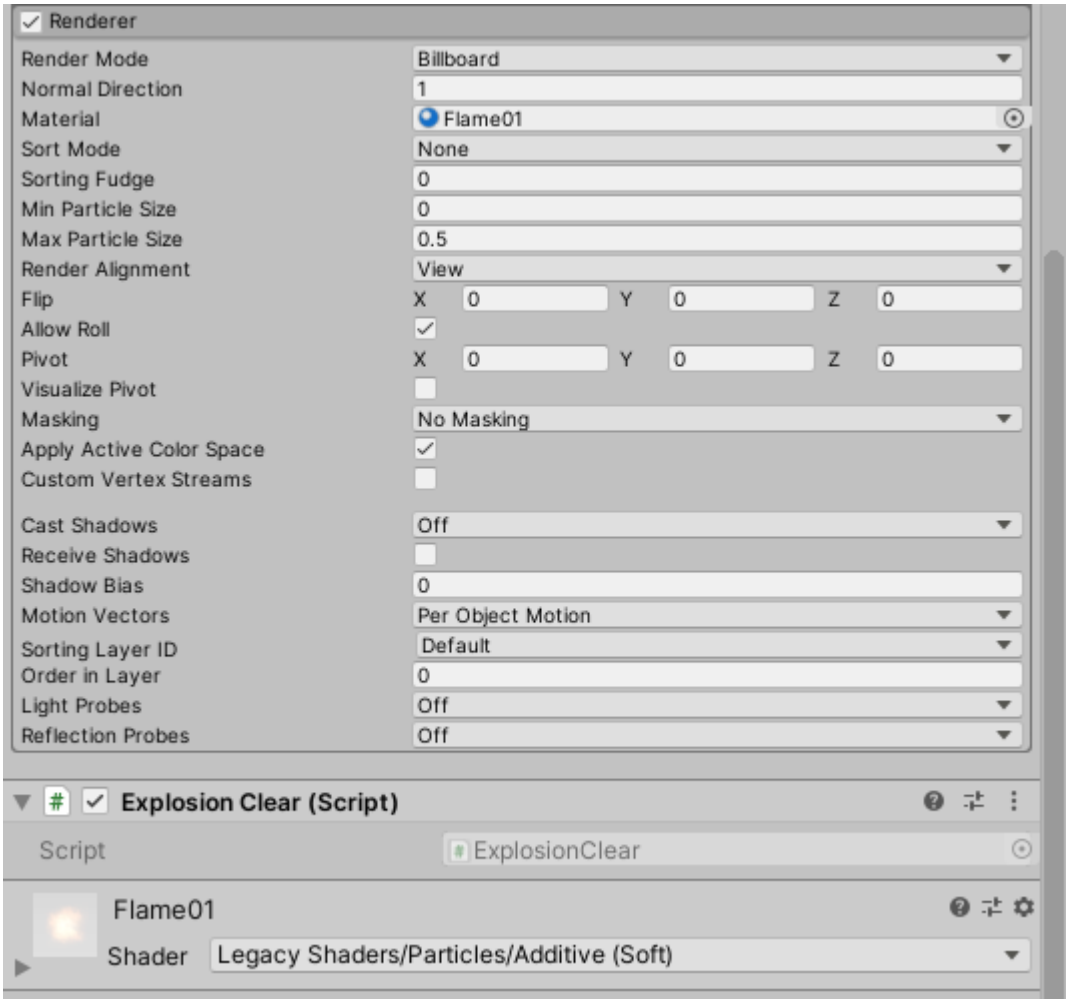




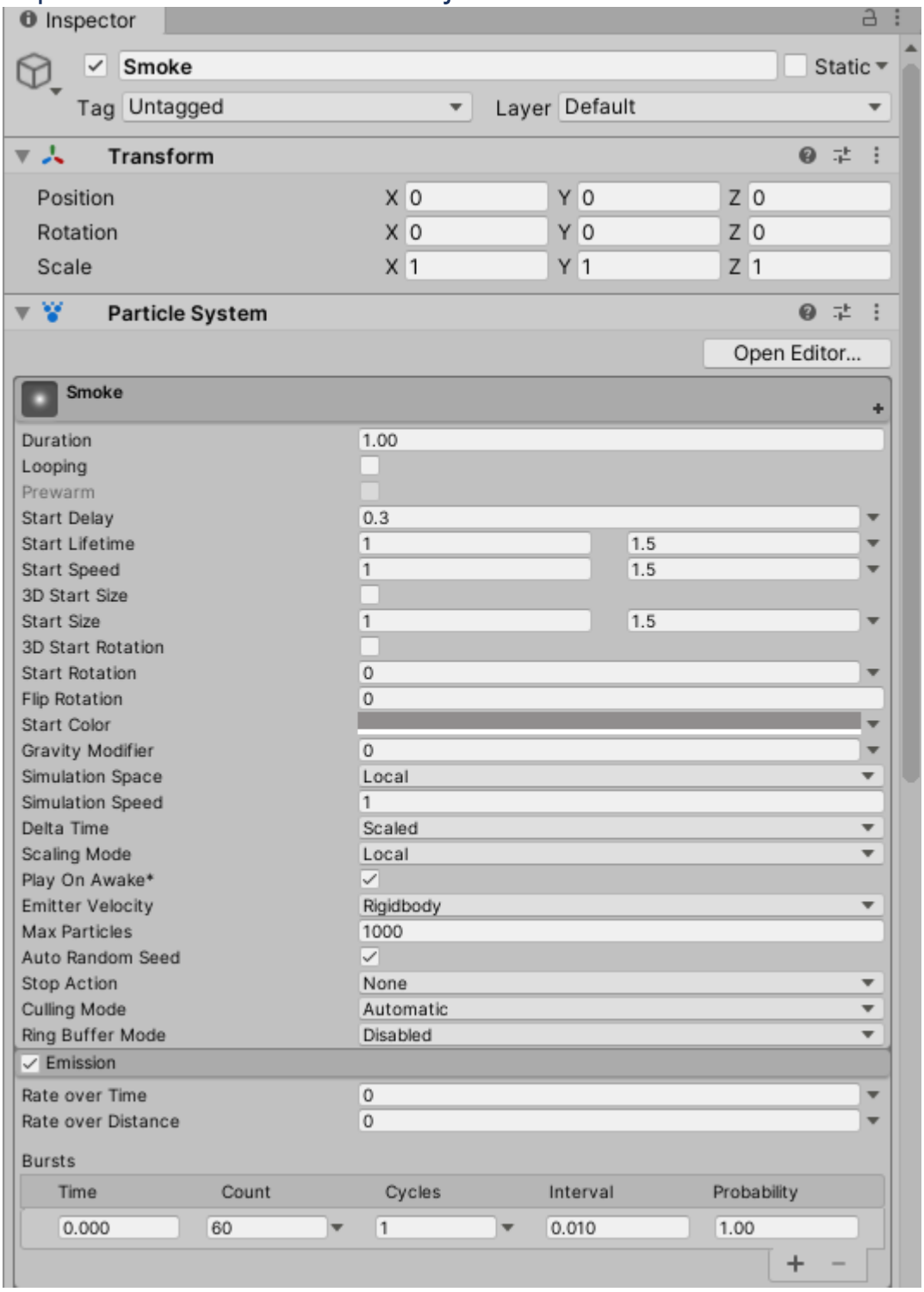
Explosion Prefab Object Continued



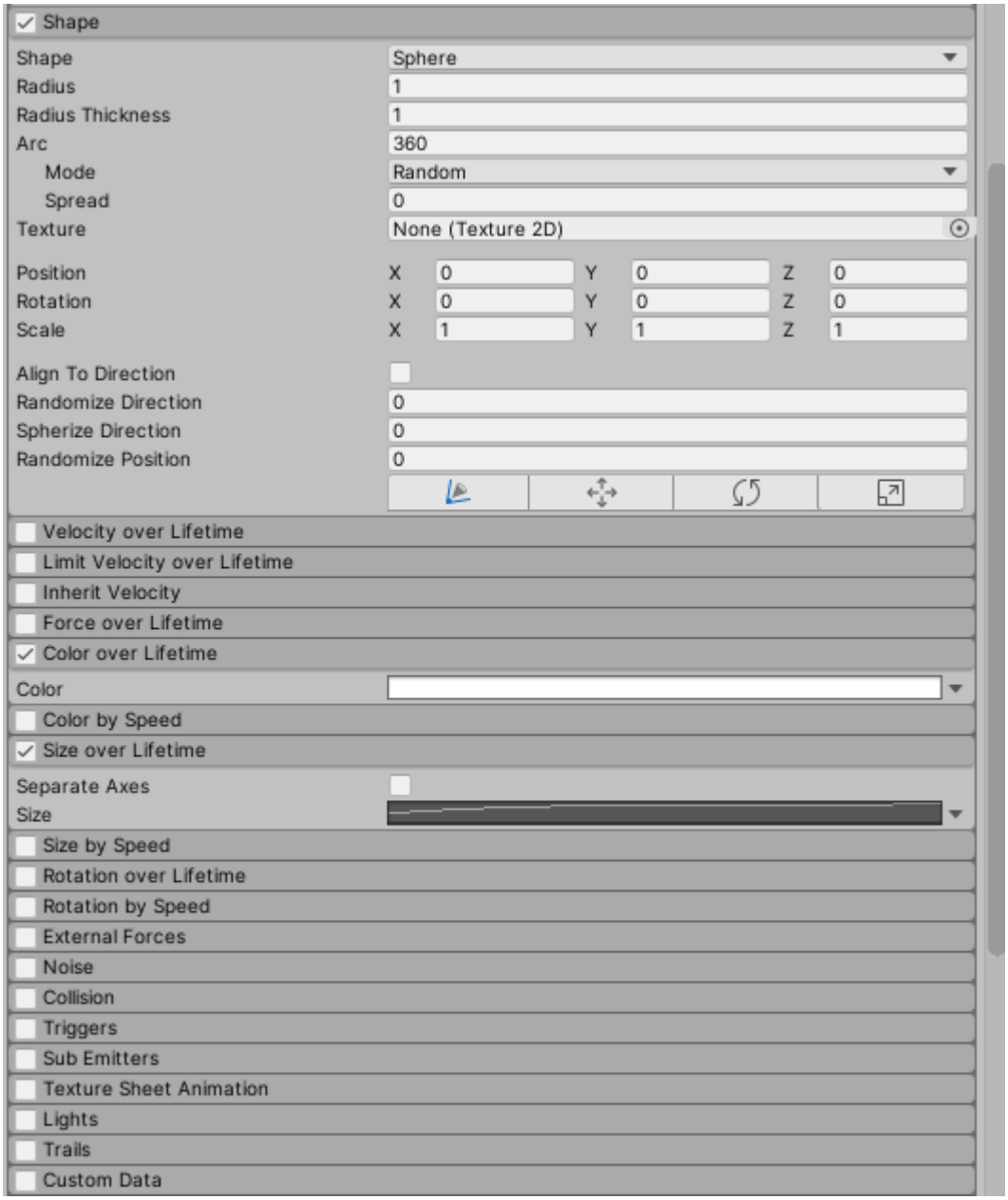
Explosion Prefab Object Continued



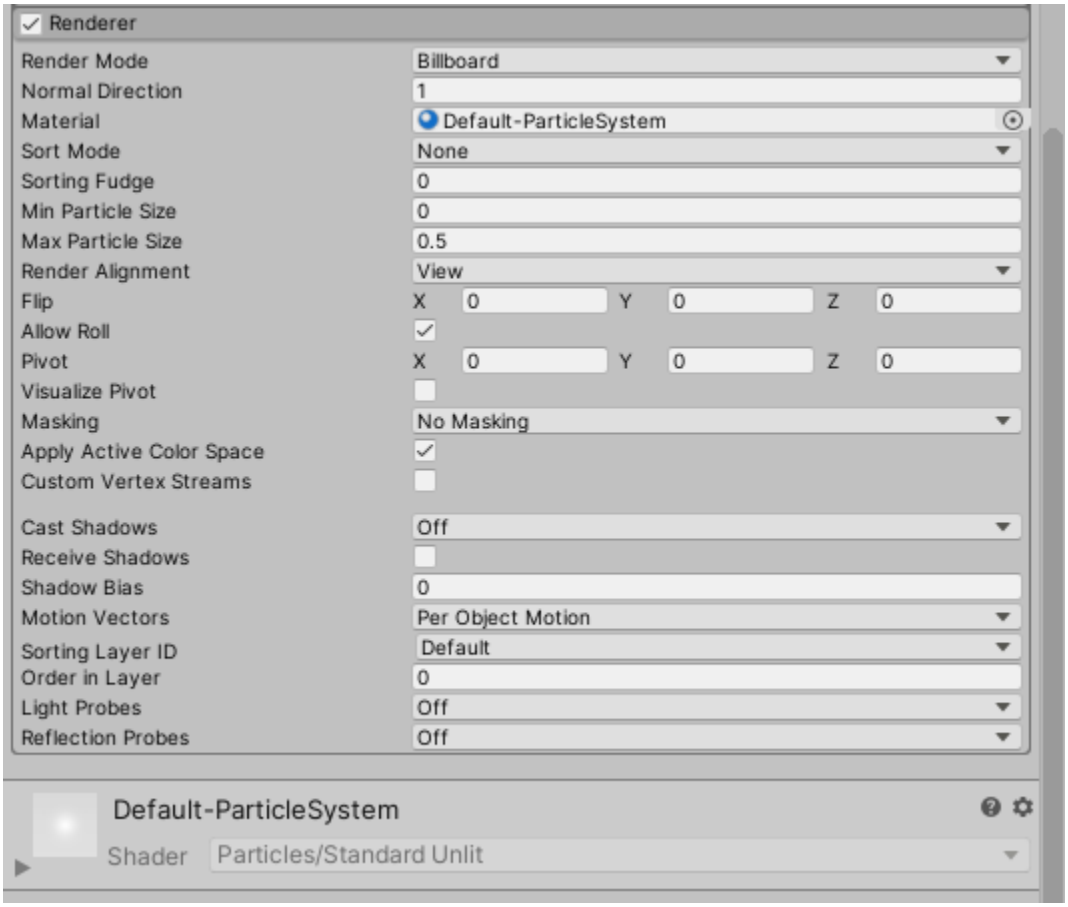
Explosion Prefab > Smoke Object



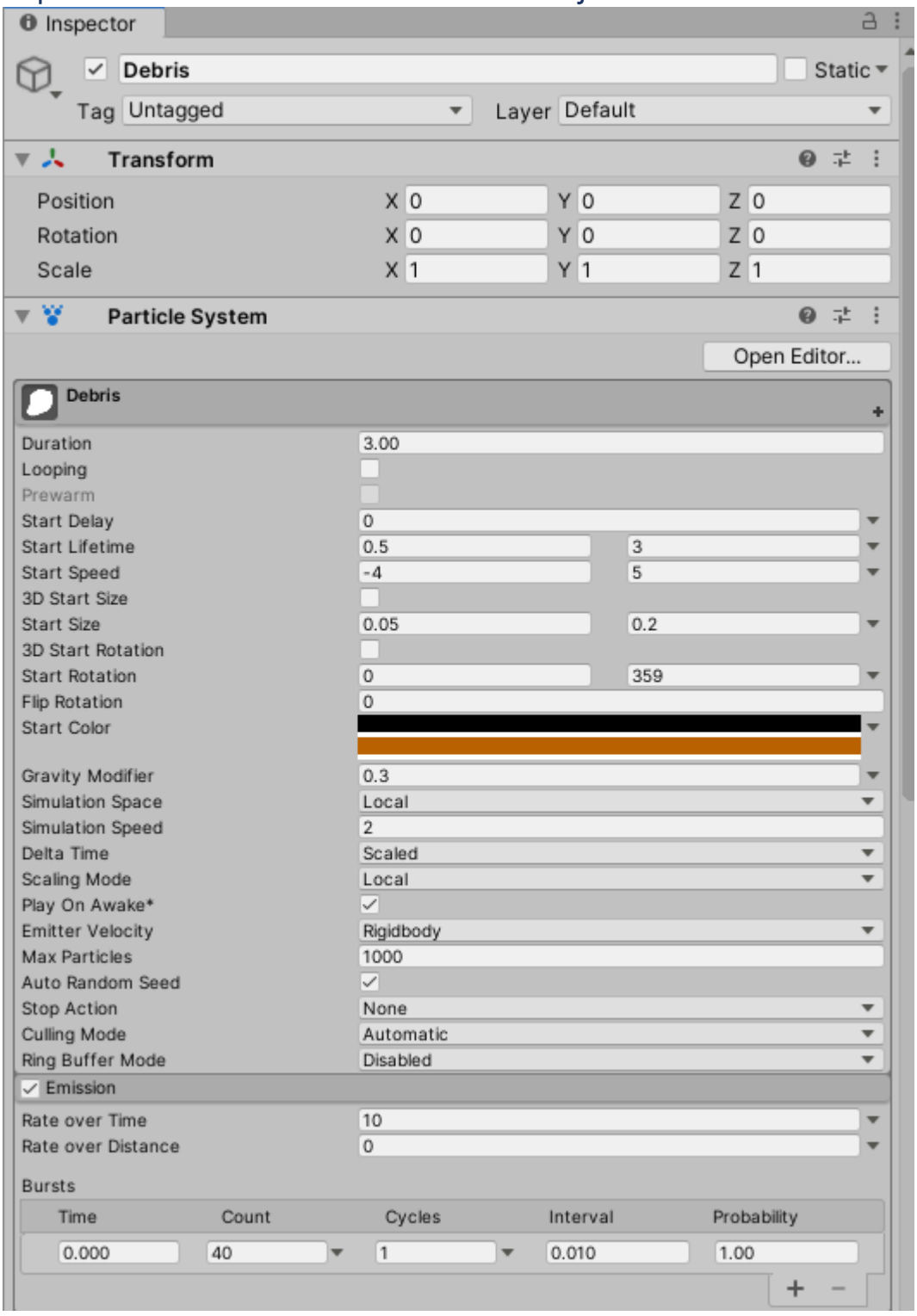
Explosion Prefab > Smoke Object Continued



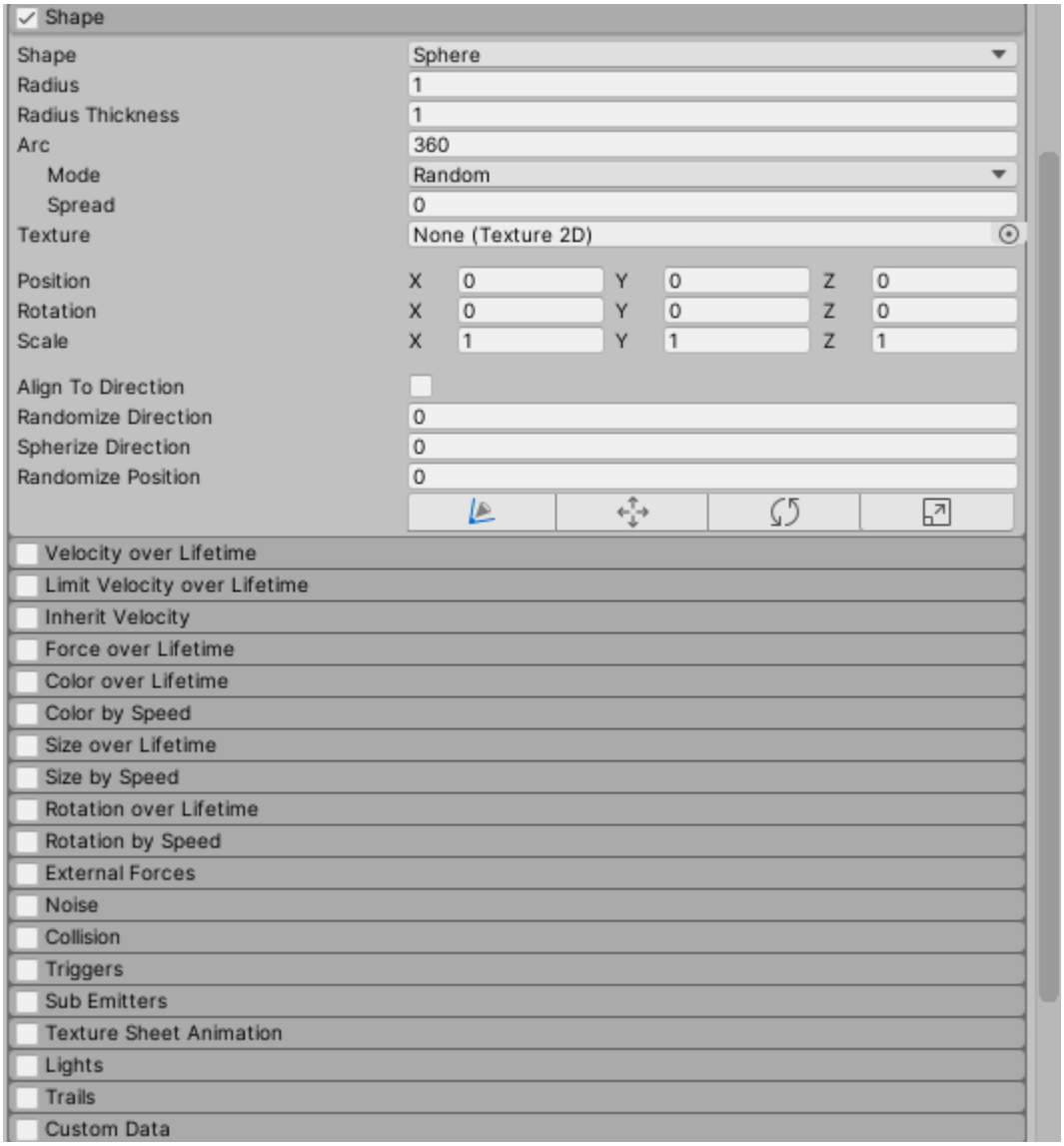
Explosion Prefab > Smoke Object Continued



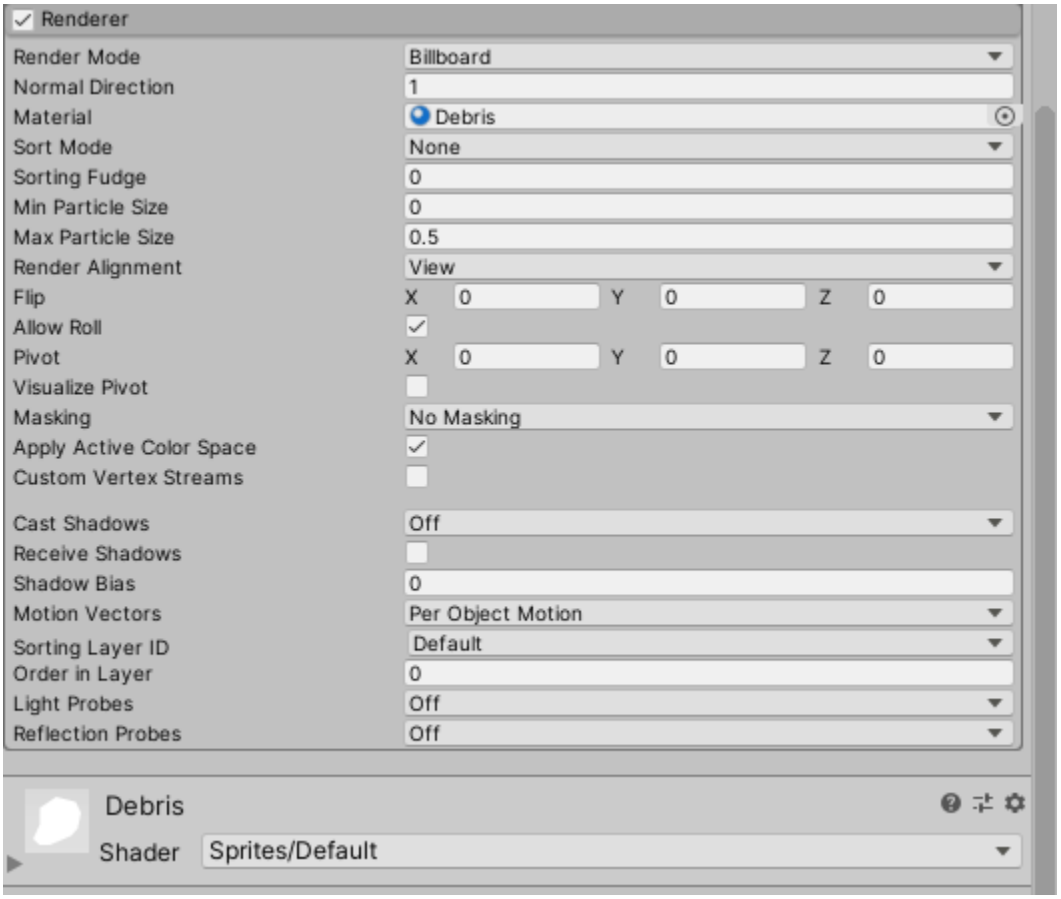
Explosion Prefab > Smoke > Debris Object



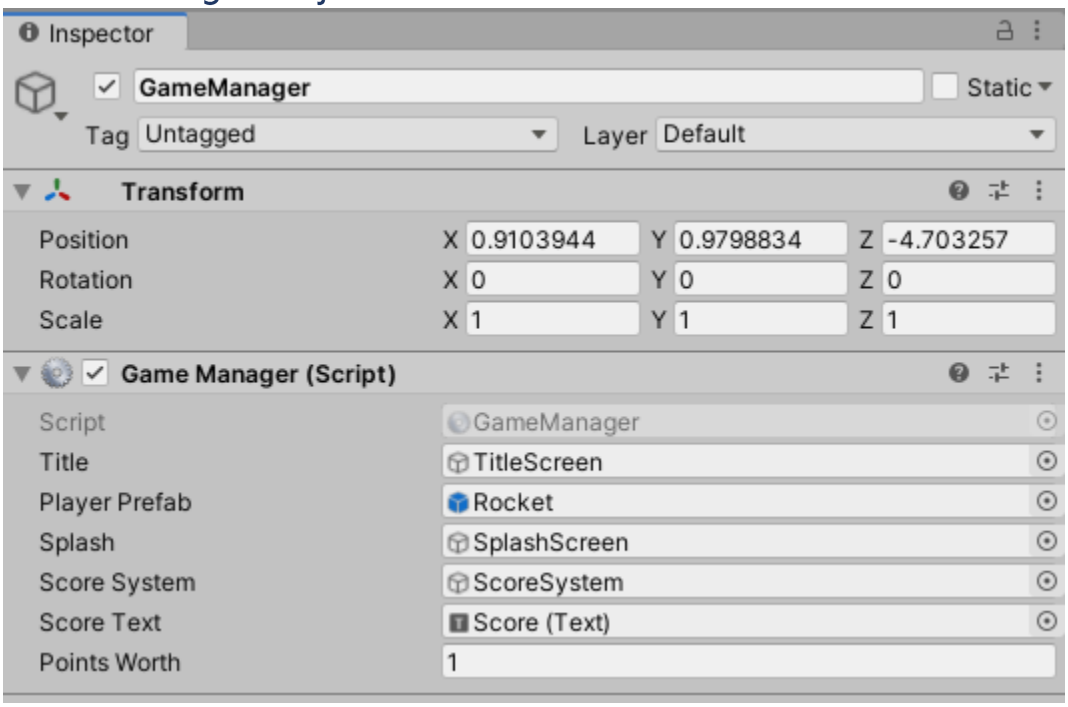
Explosion Prefab > Smoke > Debris Object Continued



Explosion Prefab > Smoke > Debris Object Continued



GameManager Object

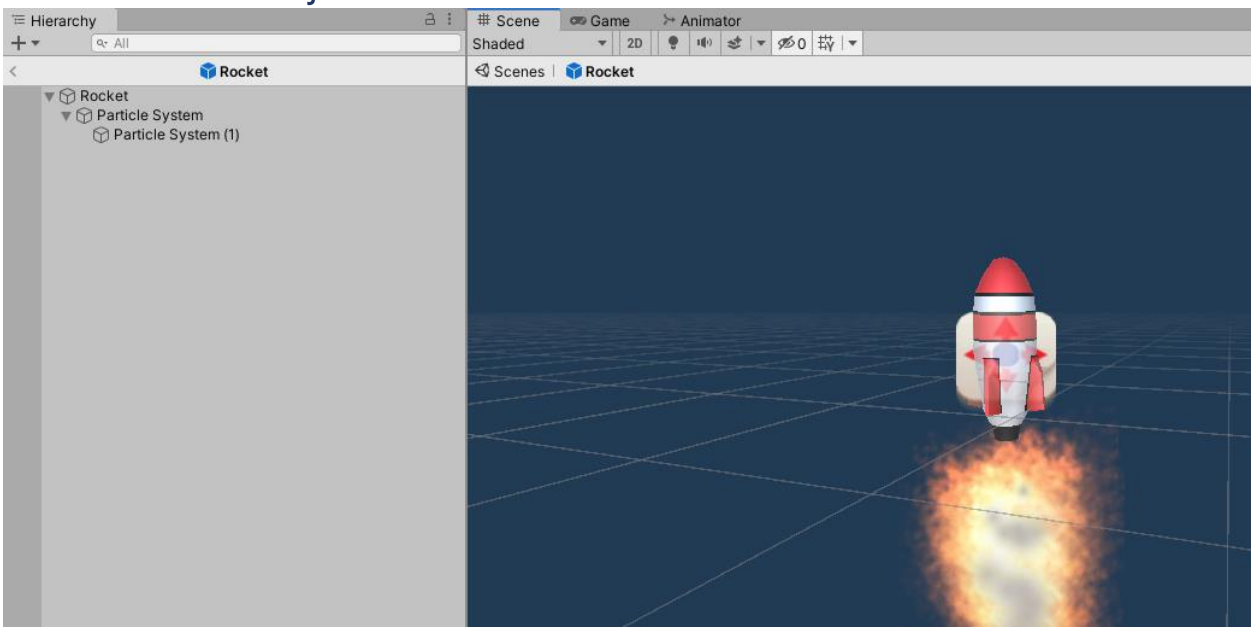


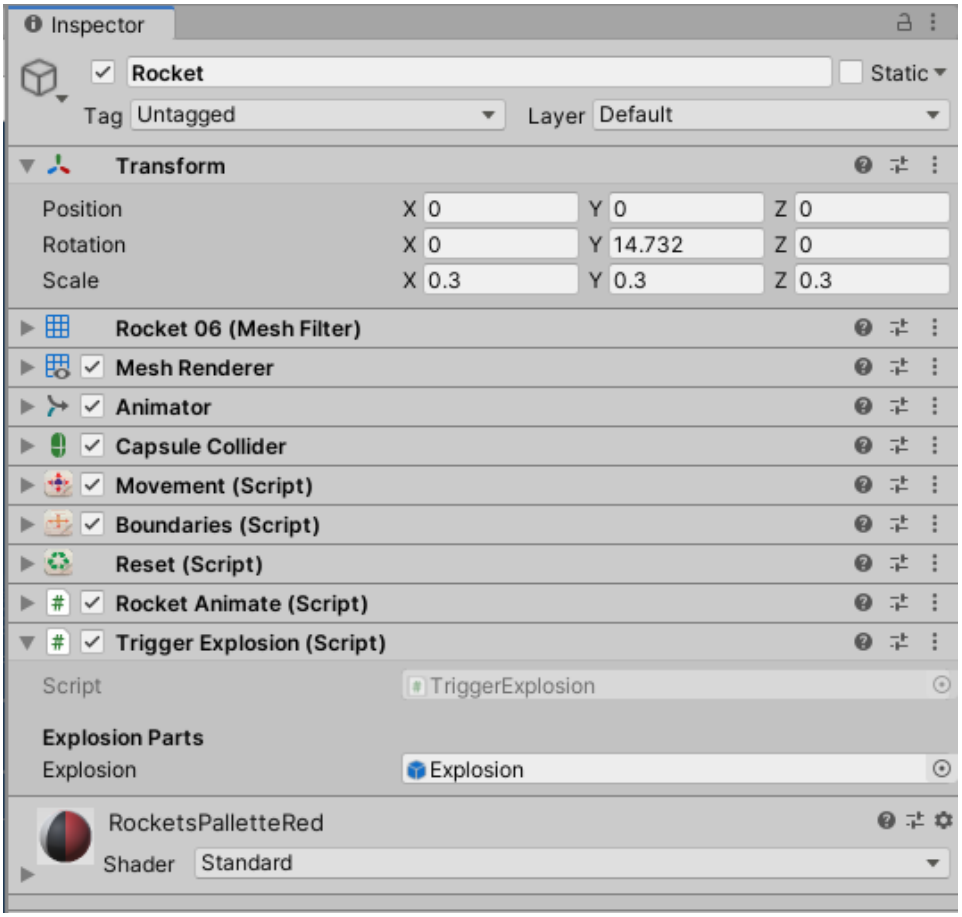
GameManager.cs Script

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

public class GameManager : MonoBehaviour
{
    private Spawner spawner;
    public GameObject title;
    private Vector2 screenBounds;
    public GameObject playerPrefab;
    private GameObject player;
    private bool gameStarted = false;
    public GameObject splash;
    public GameObject scoreSystem;
    public Text scoreText;
    public int pointsWorth = 1;
    private int score;
    private bool smokeCleared = true;

    void Awake()
    {
        spawner = GameObject.Find("Spawner").GetComponent<Spawner>();
        screenBounds = Camera.main.ScreenToWorldPoint(
              new Vector3(
                     Screen.width, Screen.height, Camera.main.transform.position.z
                     ));
        player = playerPrefab;
        scoreText.enabled = false;
    }
    void Start()
    {
        spawner.active = false;
        title.SetActive(true);
        splash.SetActive(false);
    }
    void Update()
    {
        if (!gameStarted)
        {

            if (Input.anyKeyDown && smokeCleared)
            {
                smokeCleared = false;
                ResetGame();
            }
        }
        else
        {
            if (!player)
            {
                OnPlayerKilled();
            }
        }
```

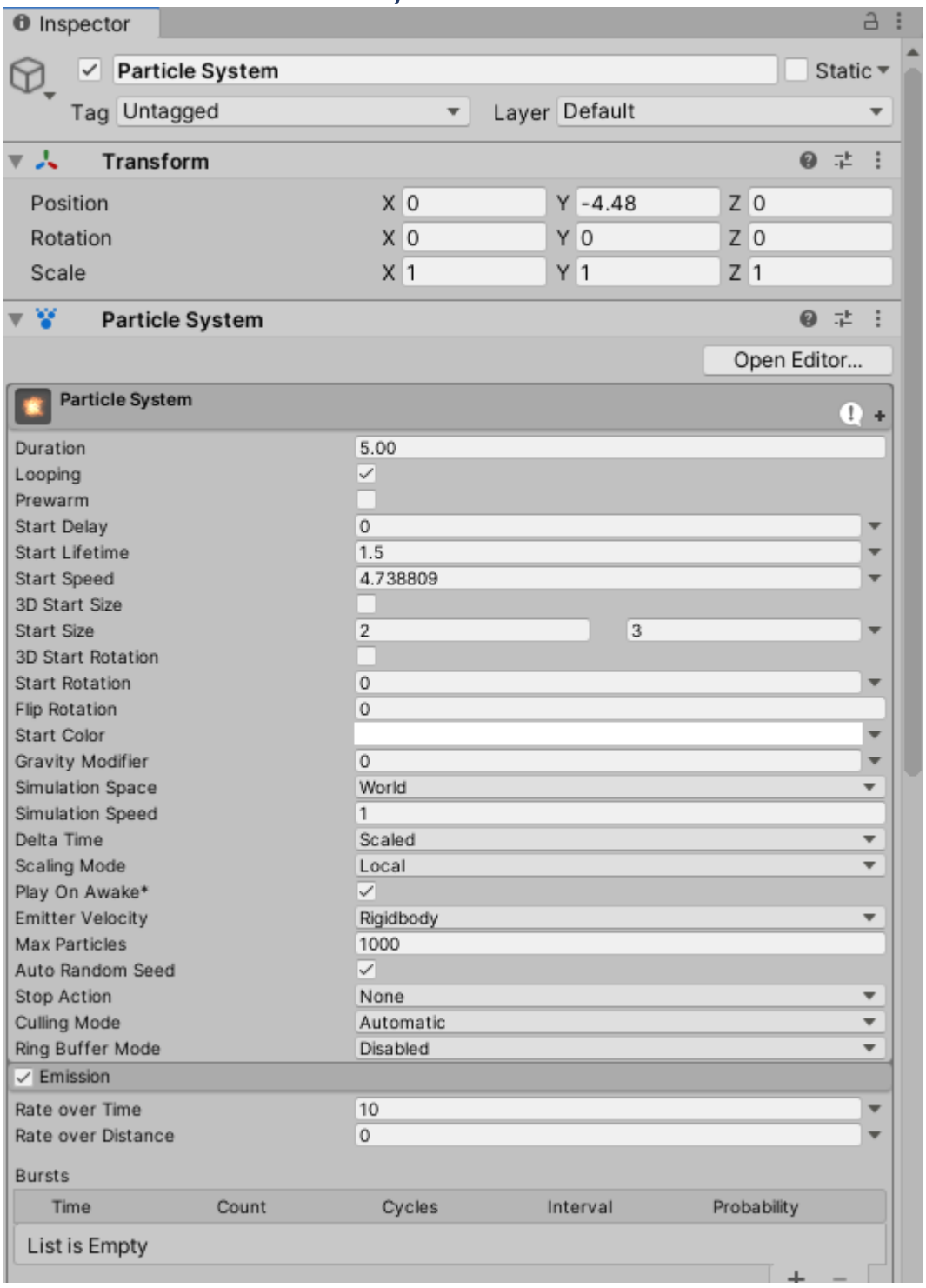
```
        var nextBomb = GameObject.FindGameObjectsWithTag("Bomb");

        foreach (GameObject bombObject in nextBomb)
        {
            if (!gameStarted)
            {
                Destroy(bombObject);
            }
            else if (bombObject.transform.position.y < (-screenBounds.y))
            {
                scoreSystem.GetComponent<Score>().AddScore(pointsWorth);
                Destroy(bombObject);
            }
        }
    }

    void ResetGame()
    {
        spawner.active = true;
        title.SetActive(false);
        splash.SetActive(false);
        player = Instantiate(playerPrefab, new Vector3(0, 0, 0),
playerPrefab.transform.rotation);
        gameStarted = true;
        scoreText.enabled = true;
        scoreSystem.GetComponent<Score>().score = 0;
        scoreSystem.GetComponent<Score>().Start();
    }

    void OnPlayerKilled()
    {
        spawner.active = false;
        gameStarted = false;
        Invoke("SplashScreen", 2f);
    }

    void SplashScreen()
    {
        smokeCleared = true;
        splash.SetActive(true);
    }
}
```

Rocket Prefab Object

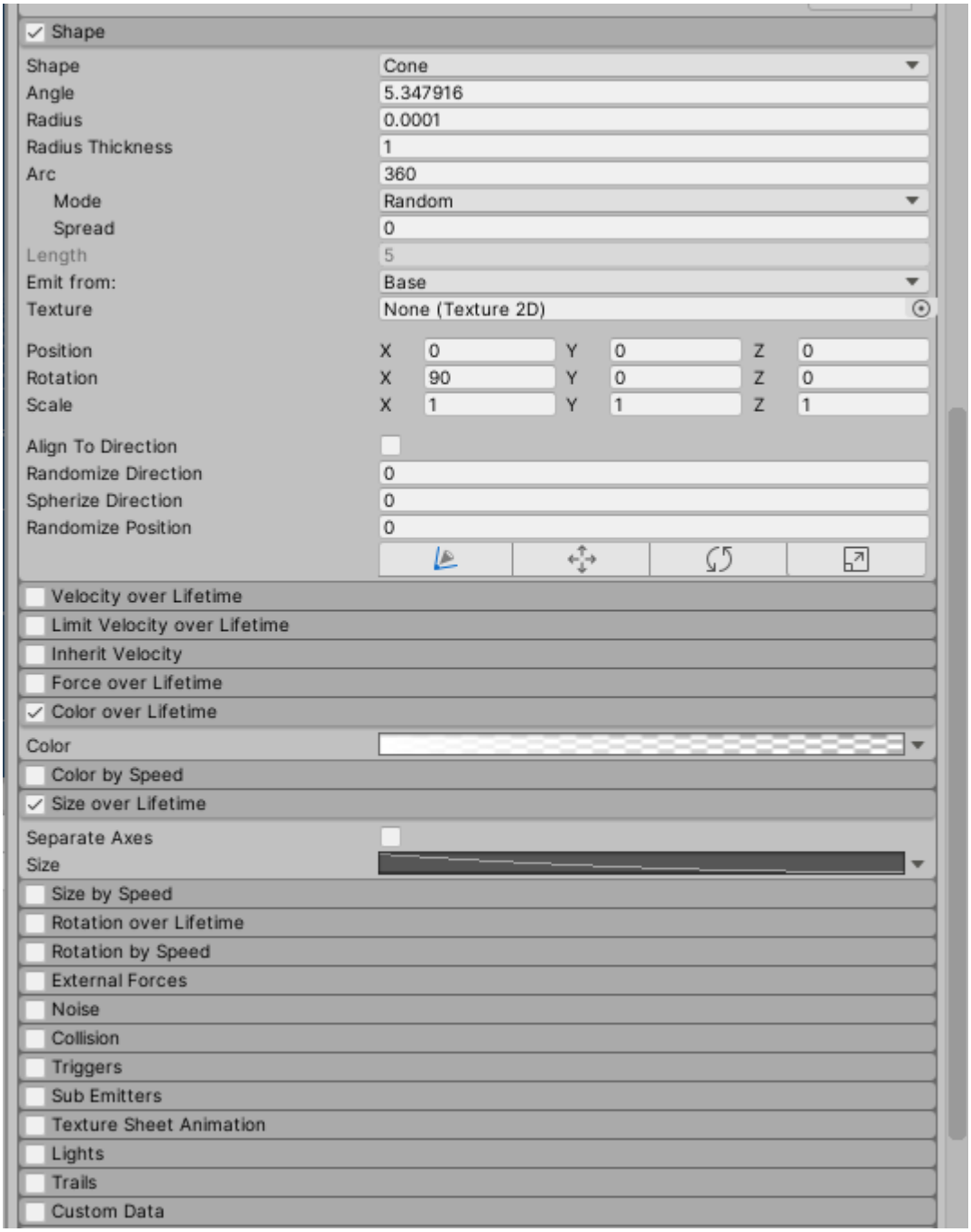




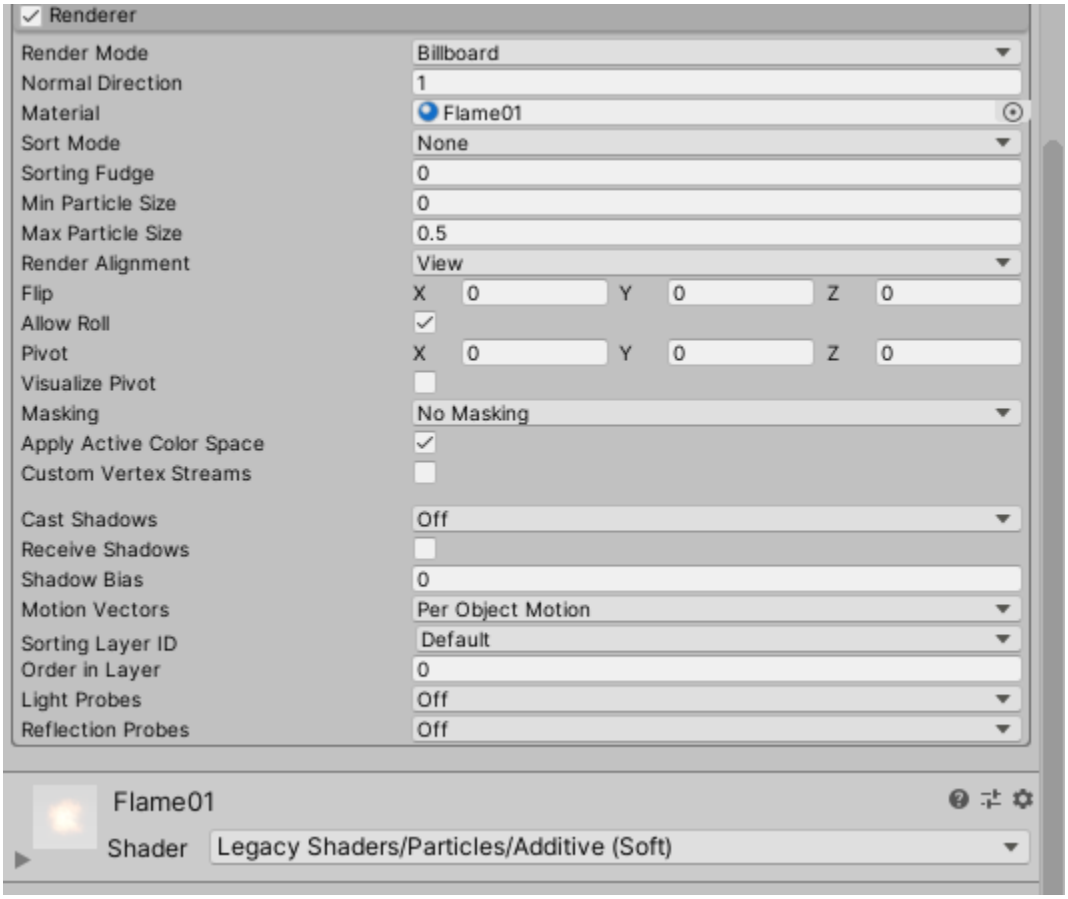
Rocket Prefab > Particle System



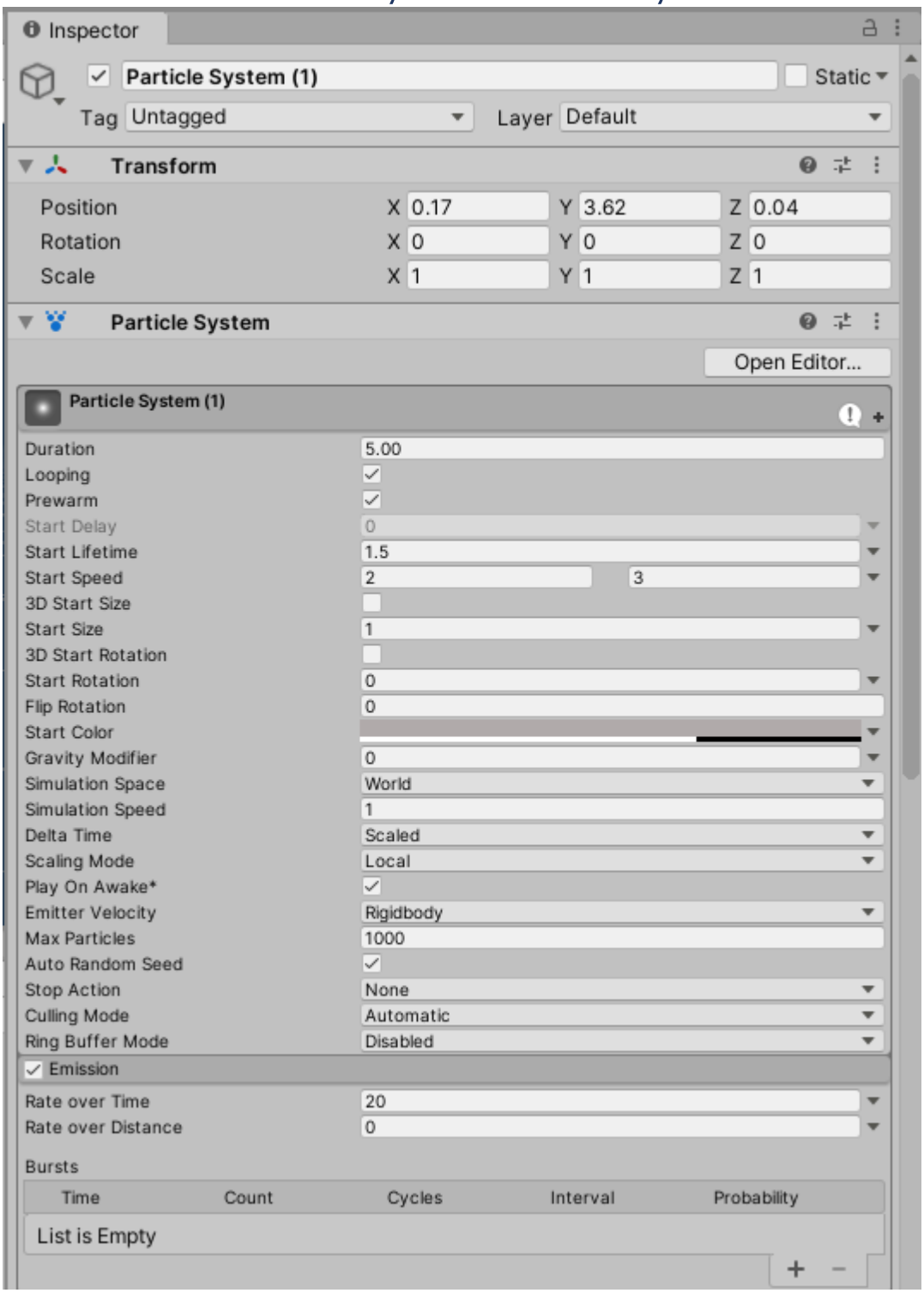
Rocket Prefab > Particle System Continued



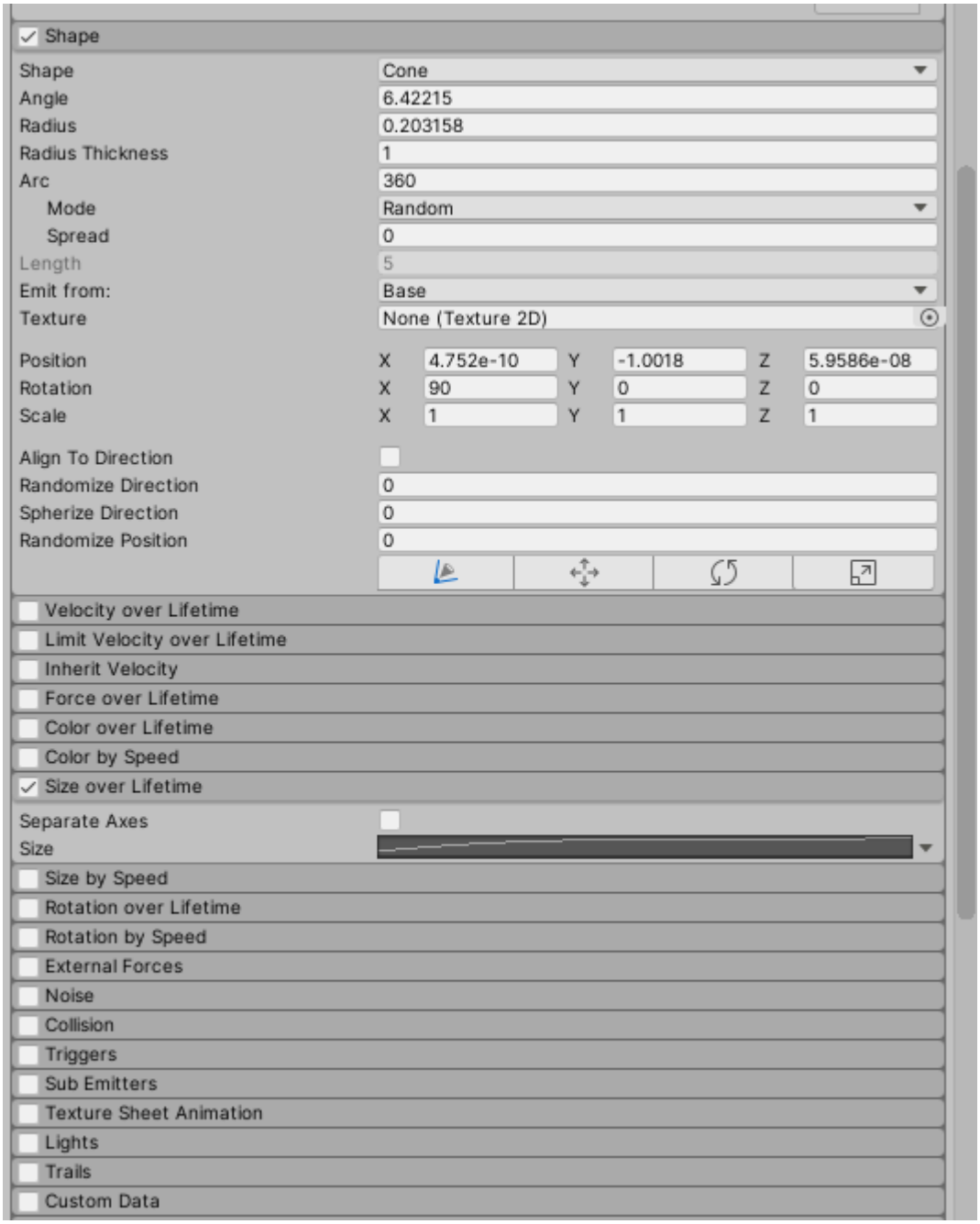
Rocket Prefab > Particle System Continued



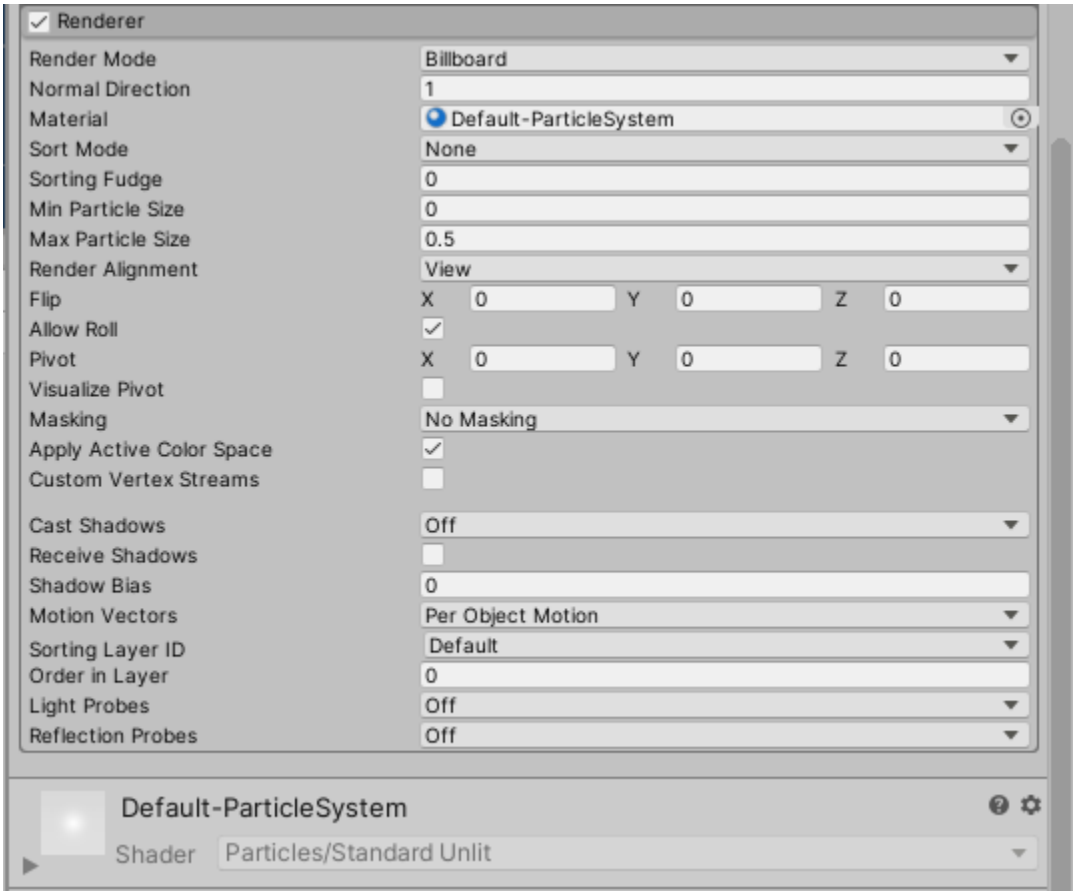
Rocket Prefab > Particle System > Particle System



Rocket Prefab > Particle System > Particle System Continued



Rocket Prefab > Particle System > Particle System Continued

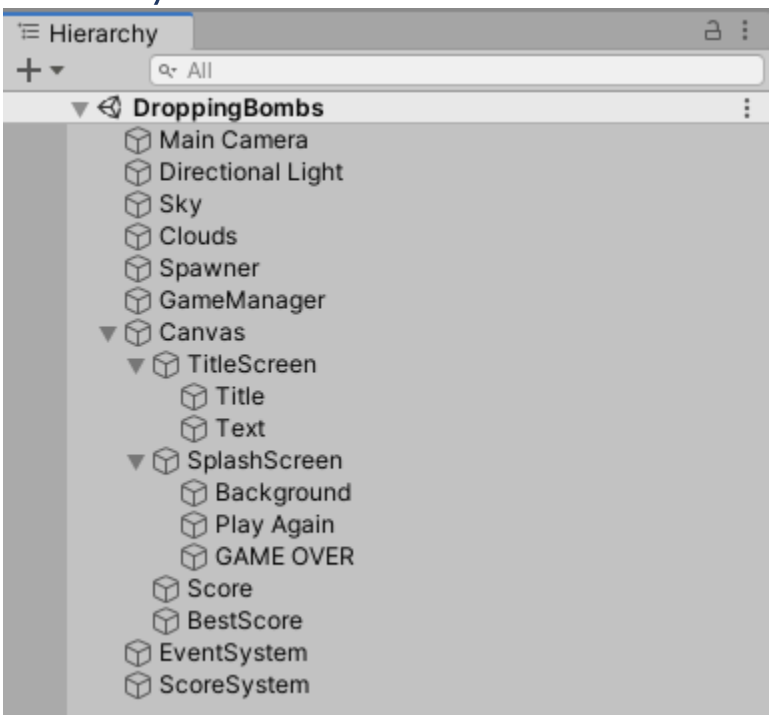


TriggerExplosion.cs Script

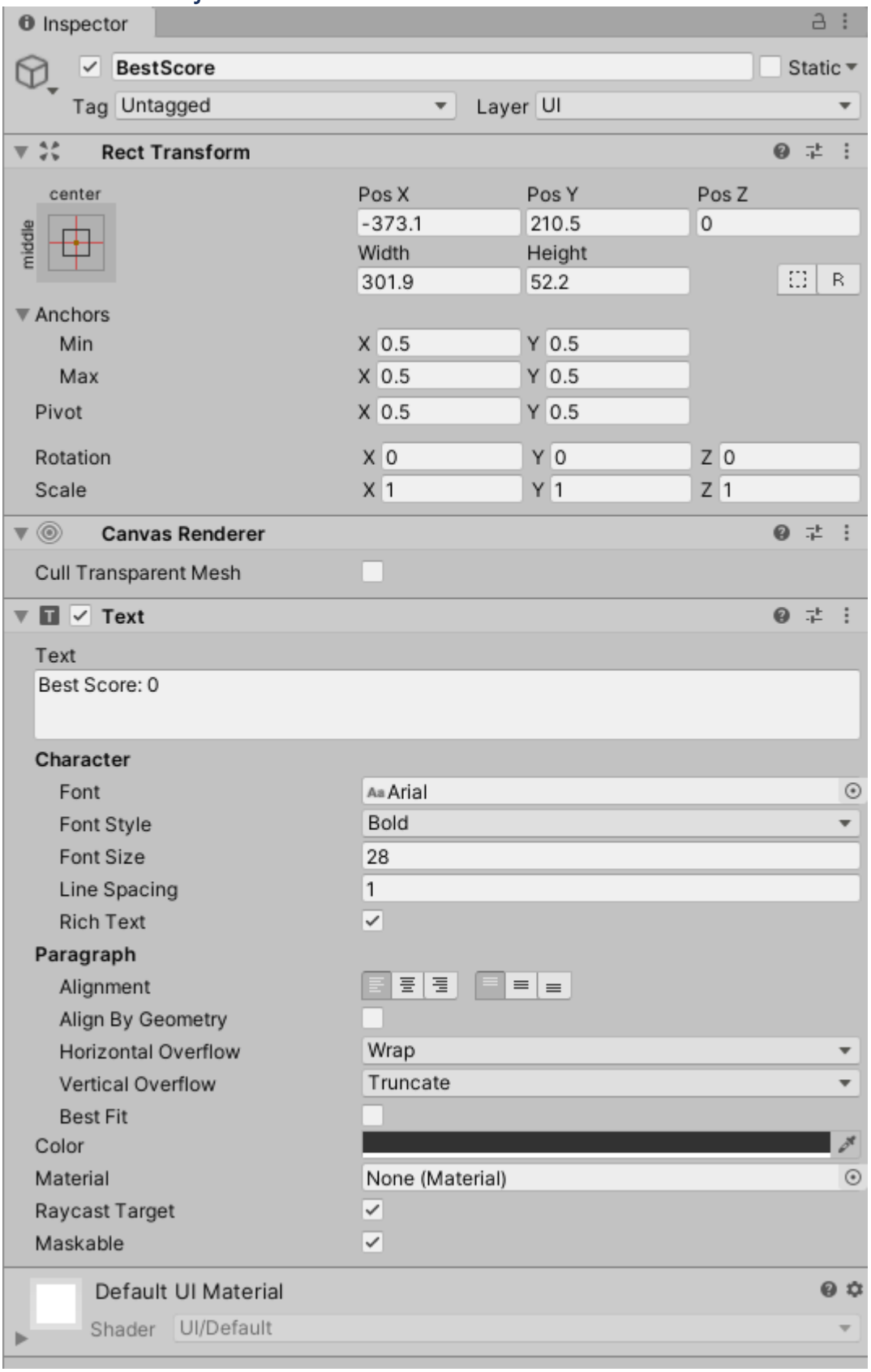
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TriggerExplosion : MonoBehaviour
{
    [Header("Explosion Parts")]
    public GameObject explosion;

    private void OnCollisionEnter(Collision collision)
    {
        Instantiate(explosion, transform.position, transform.rotation);
    }

}
```

Dropping Bombs Part 5

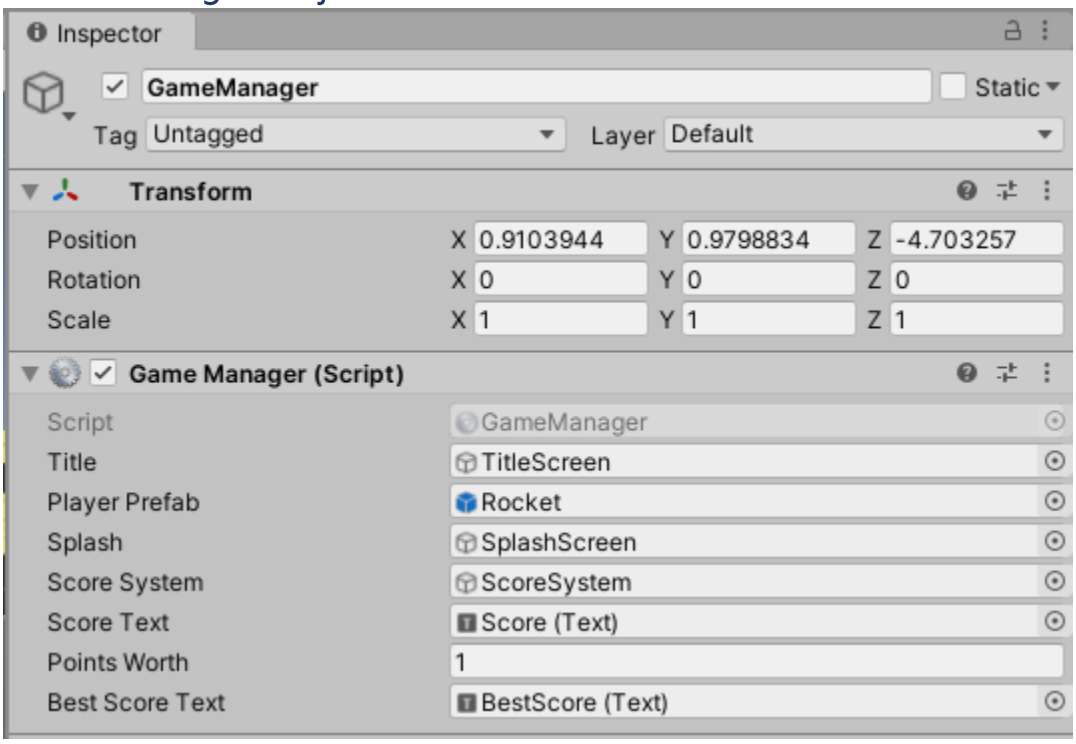
Hierarchy



BestScore Object



GameManager Object



GameManager.cs Script

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

public class GameManager : MonoBehaviour
{
    private Spawner spawner;
    public GameObject title;
    private Vector2 screenBounds;
    public GameObject playerPrefab;
    private GameObject player;
    private bool gameStarted = false;
    public GameObject splash;
    public GameObject scoreSystem;
    public Text scoreText;
    public int pointsWorth = 1;
    private int score;
    private bool smokeCleared = true;
    private int bestScore = 0;
    public Text bestScoreText;
    private bool beatBestScore;

    void Awake()
    {
        spawner = GameObject.Find("Spawner").GetComponent<Spawner>();
        screenBounds = Camera.main.ScreenToWorldPoint(
              new Vector3(Screen.width, Screen.height, Camera.main.transform.position.z)
              );
        player = playerPrefab;
        scoreText.enabled = false;
        bestScoreText.enabled = false;
    }

    void Start()
    {
        spawner.active = false;
        title.SetActive(true);
        splash.SetActive(false);
        bestScore = PlayerPrefs.GetInt("BestScore");
        bestScoreText.text = "Best Score: " + bestScore.ToString();
    }

    void Update()
    {
        if (!gameStarted)
        {
            var textColor = "#323232";
            if (beatBestScore)
            {
                textColor = "#F00";
            }
            bestScoreText.text = "<color=" + textColor + ">Best Score: "
                                    + bestScore.ToString() + "</color>";
```

```
            if (Input.anyKeyDown && smokeCleared)
            {
                smokeCleared = false;
                ResetGame();
            }
        }
        else
        {
            bestScoreText.text = "";
            if (!player)
            {
                OnPlayerKilled();
            }
        }

        var nextBomb = GameObject.FindGameObjectsWithTag("Bomb");

        foreach (GameObject bombObject in nextBomb)
        {
            if (!gameStarted)
            {
                Destroy(bombObject);
            }
            else if (bombObject.transform.position.y < (-screenBounds.y))
            {
                scoreSystem.GetComponent<Score>().AddScore(pointsWorth);
                Destroy(bombObject);
            }
        }
    }

    void ResetGame()
    {
        spawner.active = true;
        title.SetActive(false);
        splash.SetActive(false);
        player = Instantiate(
              playerPrefab,
              new Vector3(0, 0, 0),
              playerPrefab.transform.rotation
              );
        gameStarted = true;
        scoreText.enabled = true;
        scoreSystem.GetComponent<Score>().score = 0;
        scoreSystem.GetComponent<Score>().Start();
        beatBestScore = false;
        bestScoreText.enabled = true;
    }


    void OnPlayerKilled()
    {
        spawner.active = false;
        gameStarted = false;
        Invoke("SplashScreen", 2f);
        score = scoreSystem.GetComponent<Score>().score;
```

```
        if (score > bestScore)
        {
            bestScore = score;
            PlayerPrefs.SetInt("BestScore", bestScore);
            beatBestScore = true;
            bestScoreText.text = "Best Score: " + bestScore.ToString();
        }
    }

    void SplashScreen()
    {
        smokeCleared = true;
        splash.SetActive(true);
    }
}
```