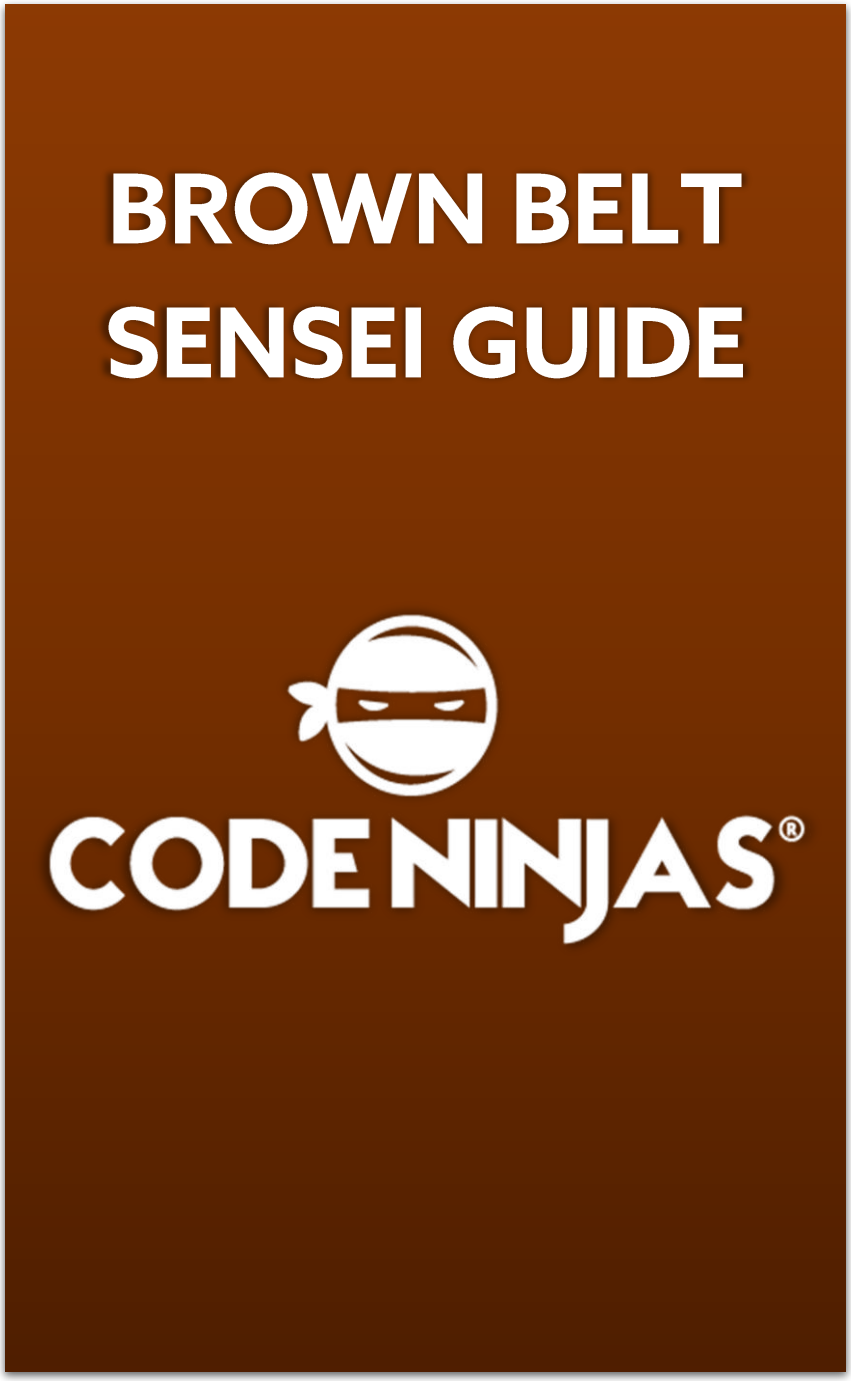
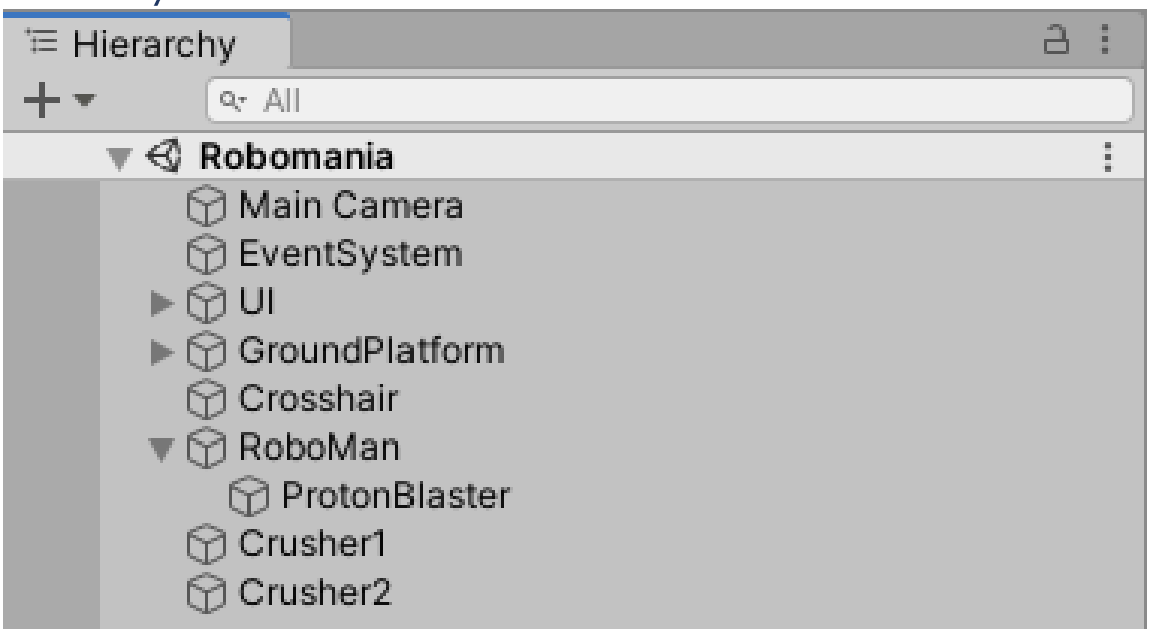
BROWN BELT SENSEI GUIDE

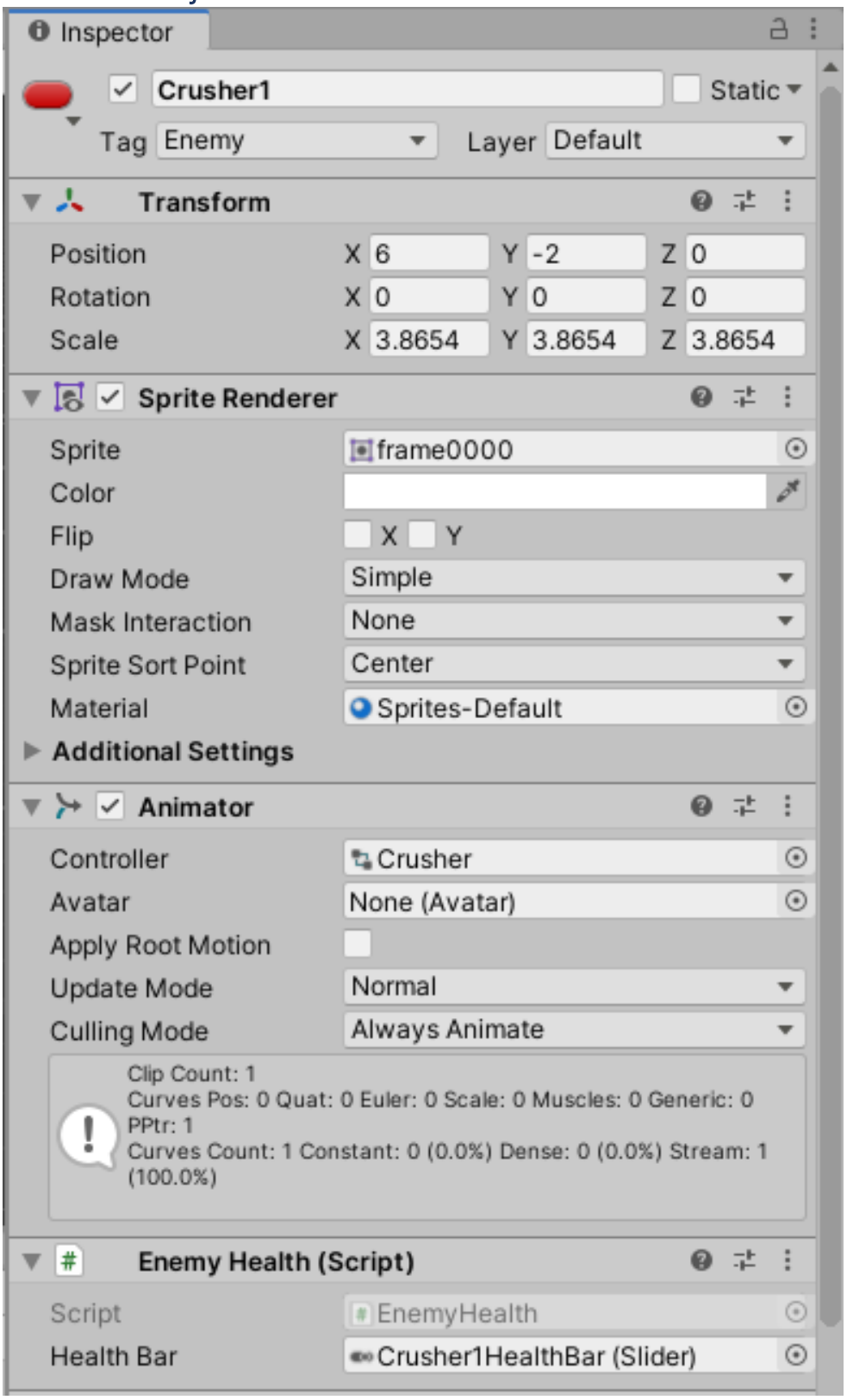


Activity Solution: Robomania

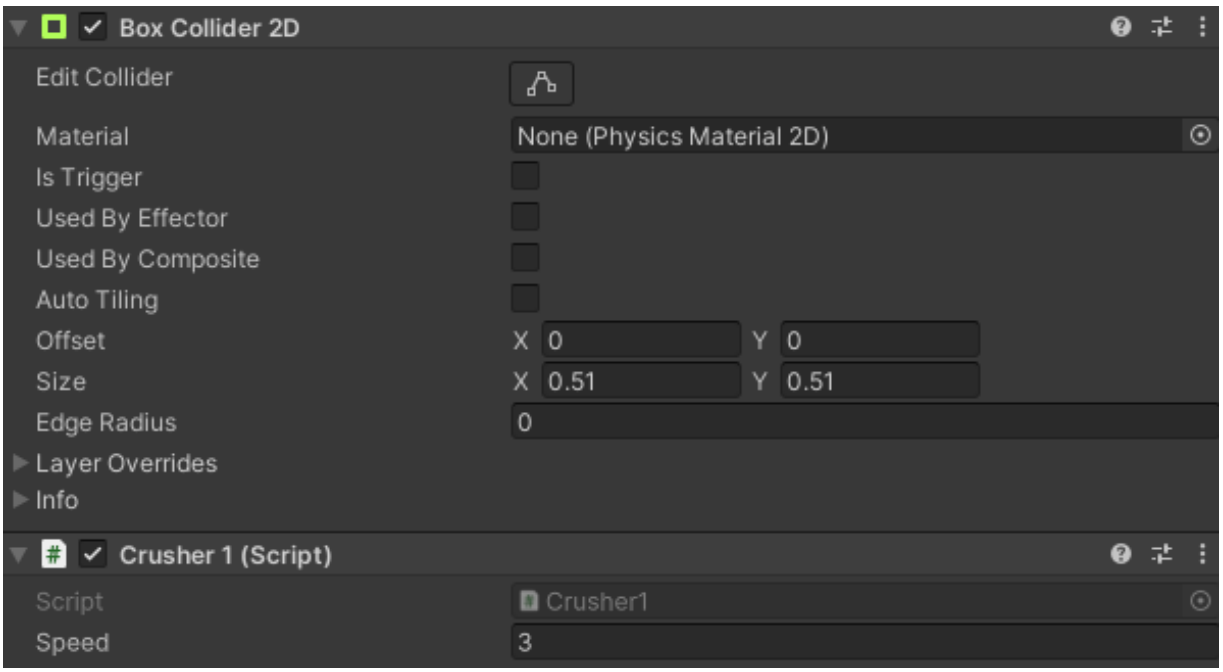
Hierarchy



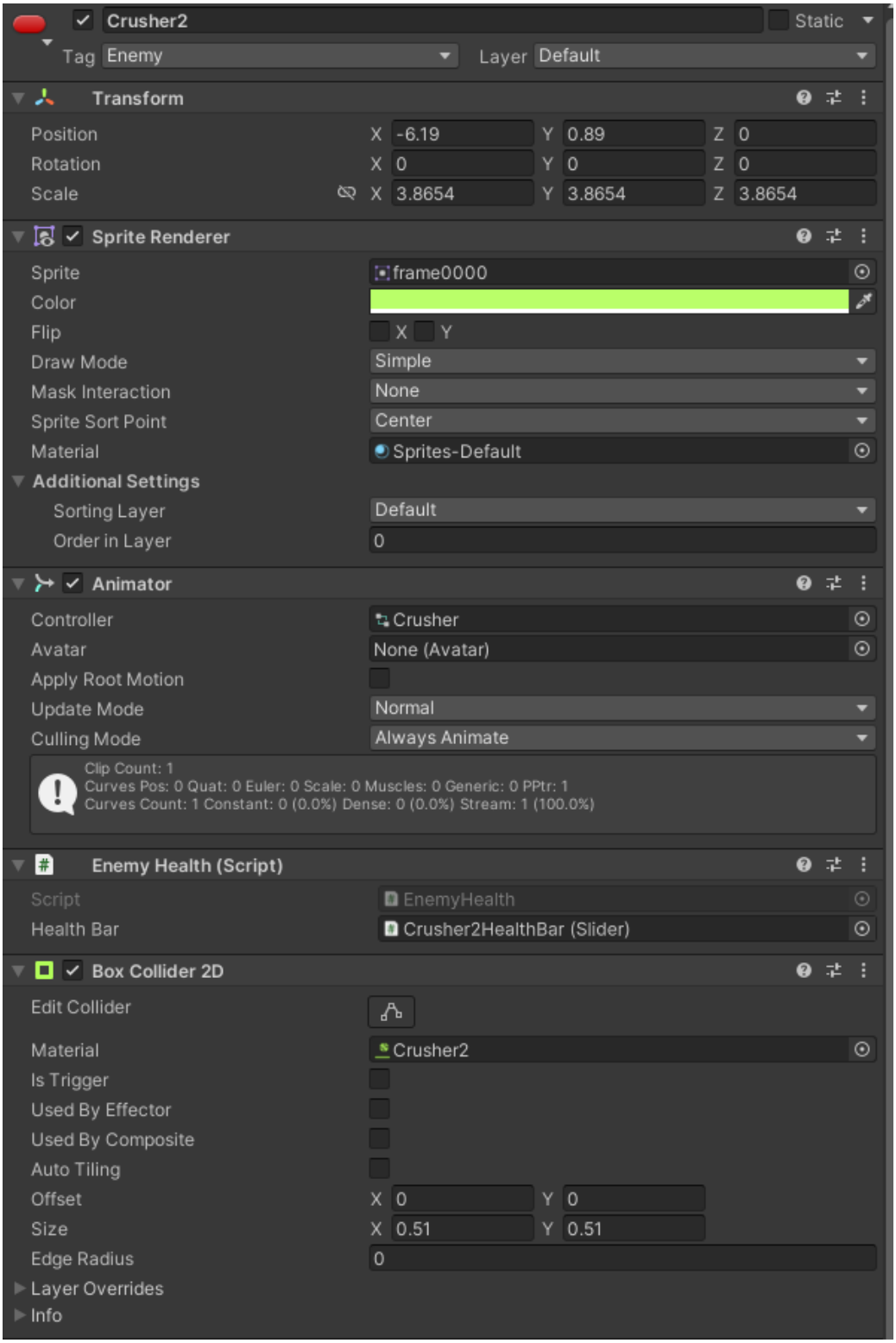
Crusher1 Object



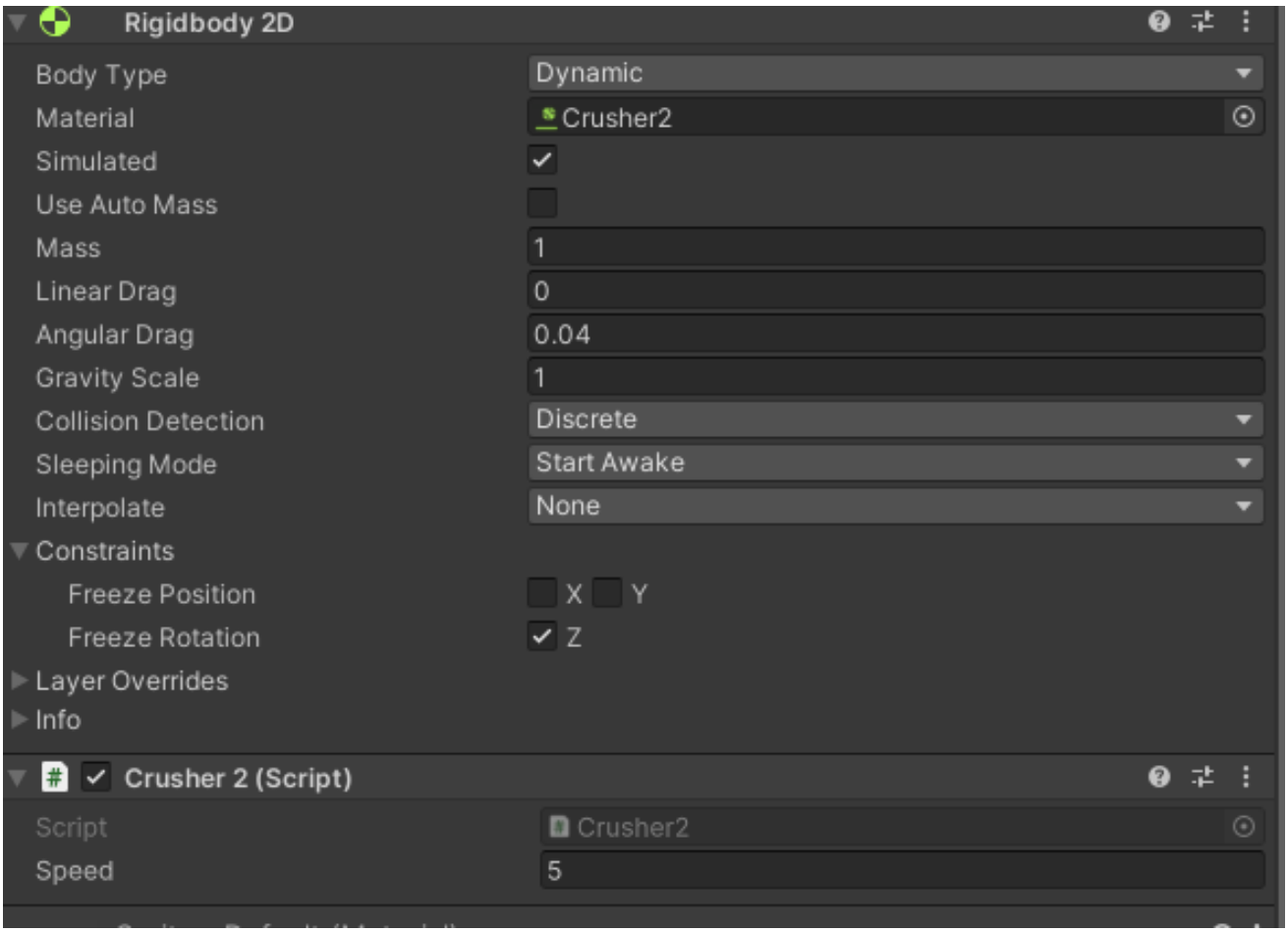
Crusher1 Object Continued



Crusher2 Object



Crusher2 Object Continued



Crusher1.cs Script

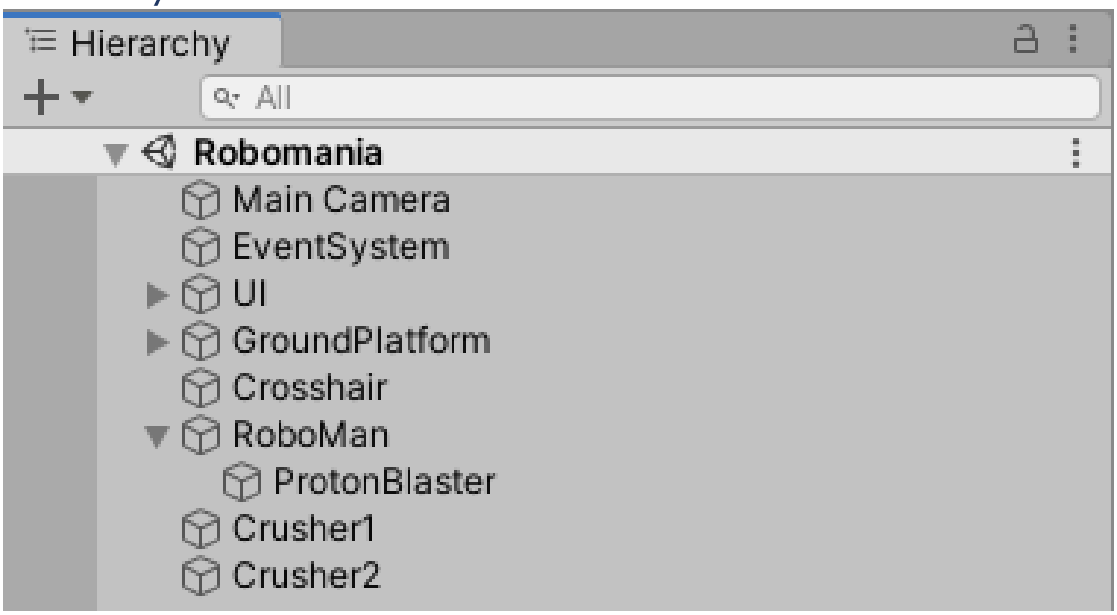
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Crusher1 : MonoBehaviour
{
    public float speed;

    private void FixedUpdate()
    {
        if (transform.position.x <= -8 || transform.position.x >= 8)
        {
            speed *= -1;
        }
        float newXPosition = transform.position.x + speed * Time.fixedDeltaTime;
        float newYPosition = transform.position.y;
        Vector2 newPosition = new Vector2(newXPosition, newYPosition);
        transform.position = newPosition;
    }
}
```

Crusher2.cs Script

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

public class Crusher2 : MonoBehaviour
{
    public float speed = 5f;
    // Start is called before the first frame update
    void Start()
    {
        GetComponent<Rigidbody2D>().AddForce(Vector2.right * speed, ForceMode2D.Impulse);
    }
}
```

Activity Solution: Robomania Prove Yourself

Hierarchy

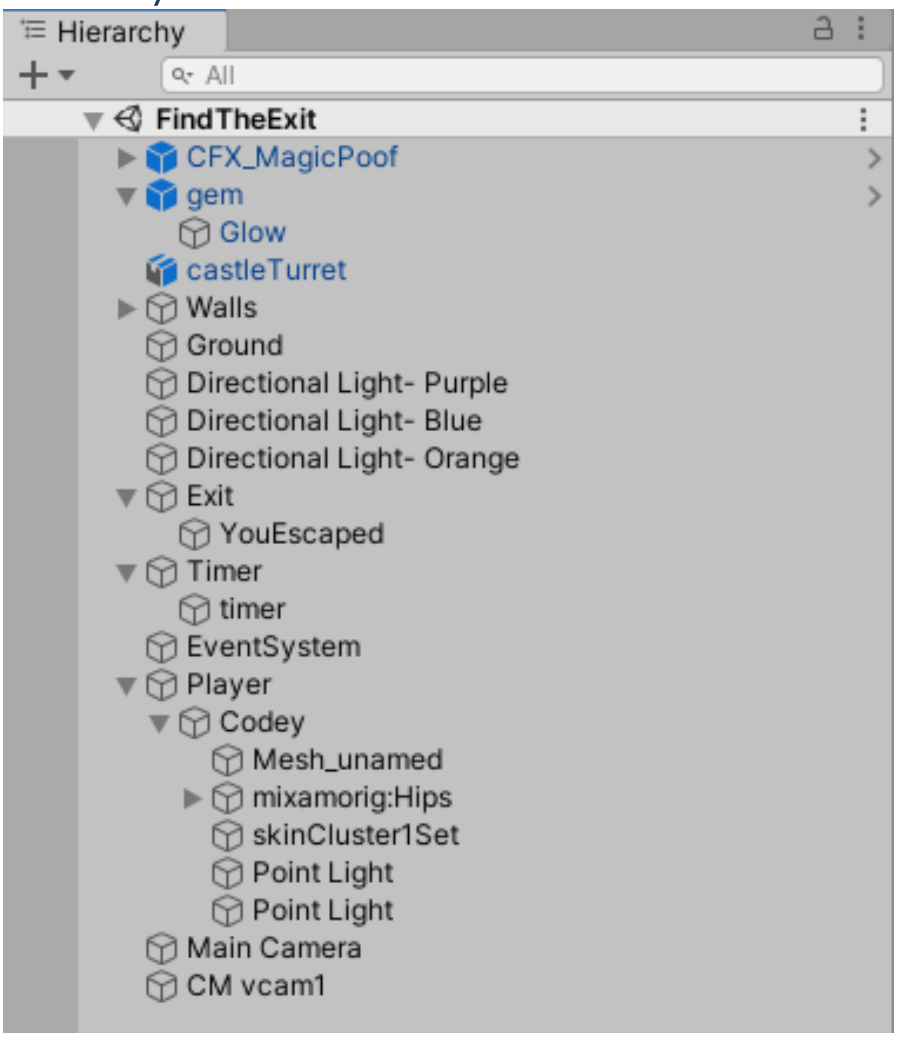


Crusher1.cs Script

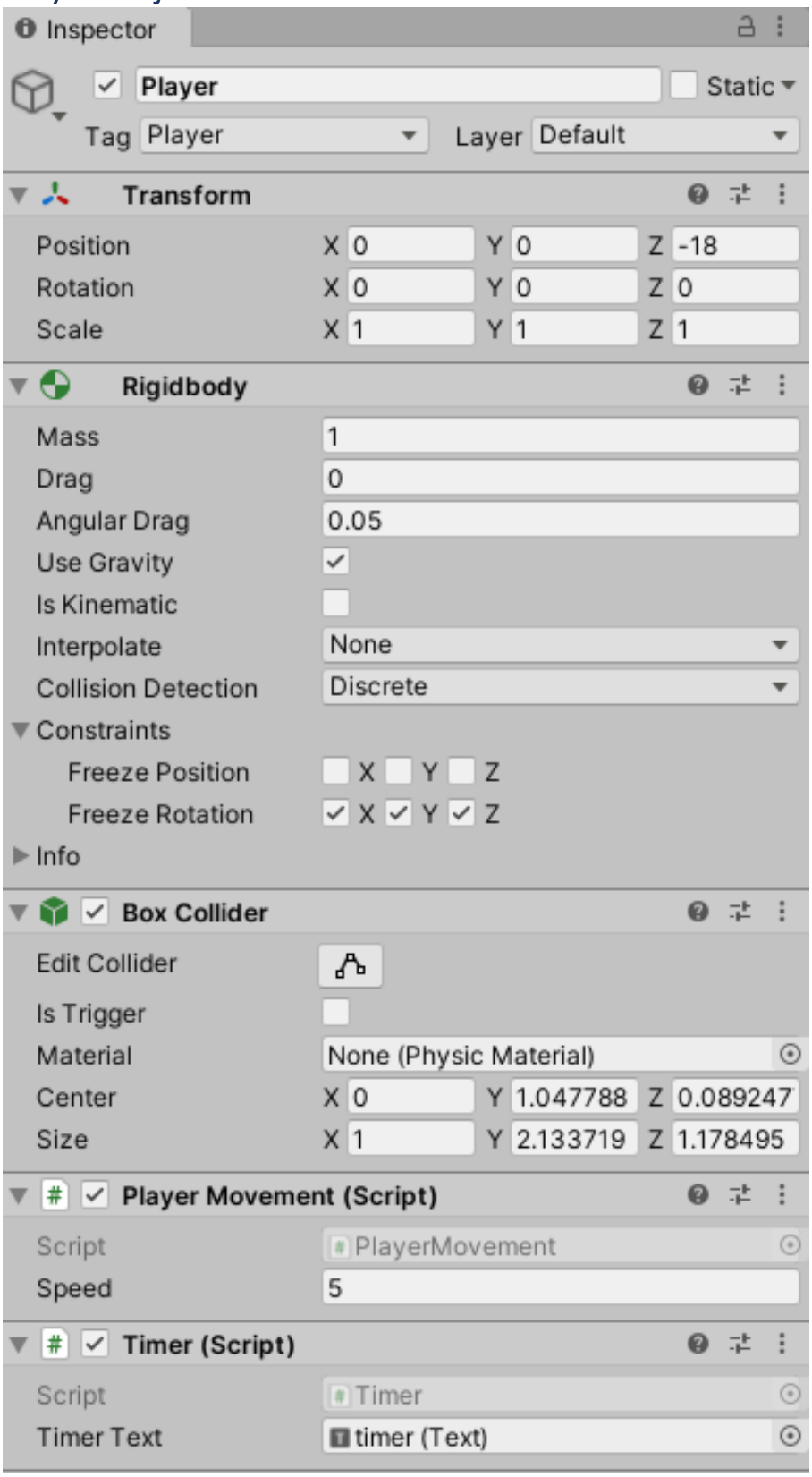
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Crusher1 : MonoBehaviour
{
    public float speed = 5f;
    // Start is called before the first frame update
    void Start()
    {
        GetComponent<Rigidbody2D>().AddForce(Vector2.left * speed, ForceMode2D.Impulse);
    }
}
```

Activity Solution: Find the Exit

Hierarchy



Player Object

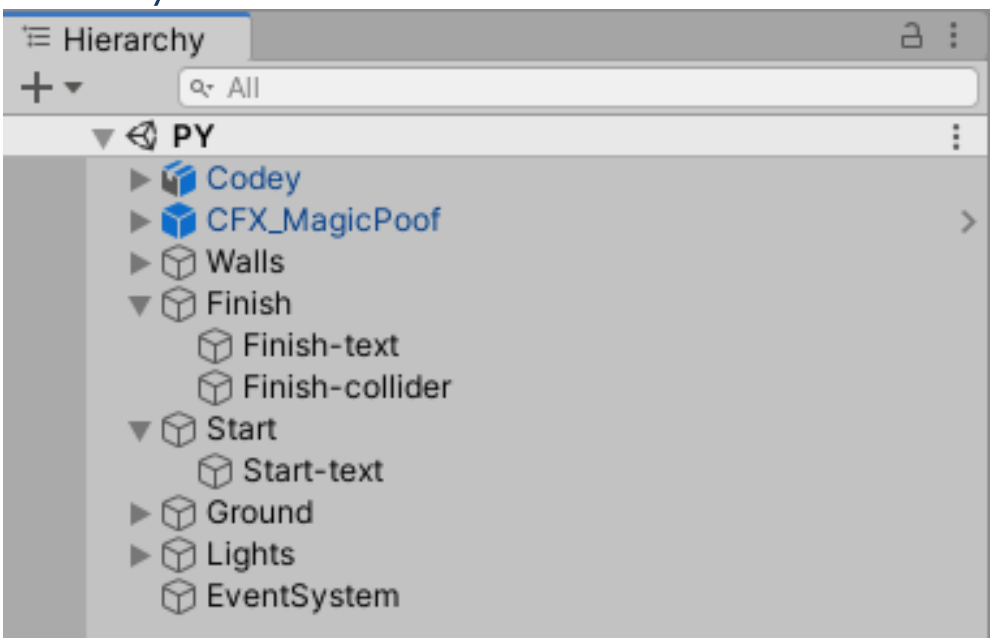


PlayerMovement.cs Script

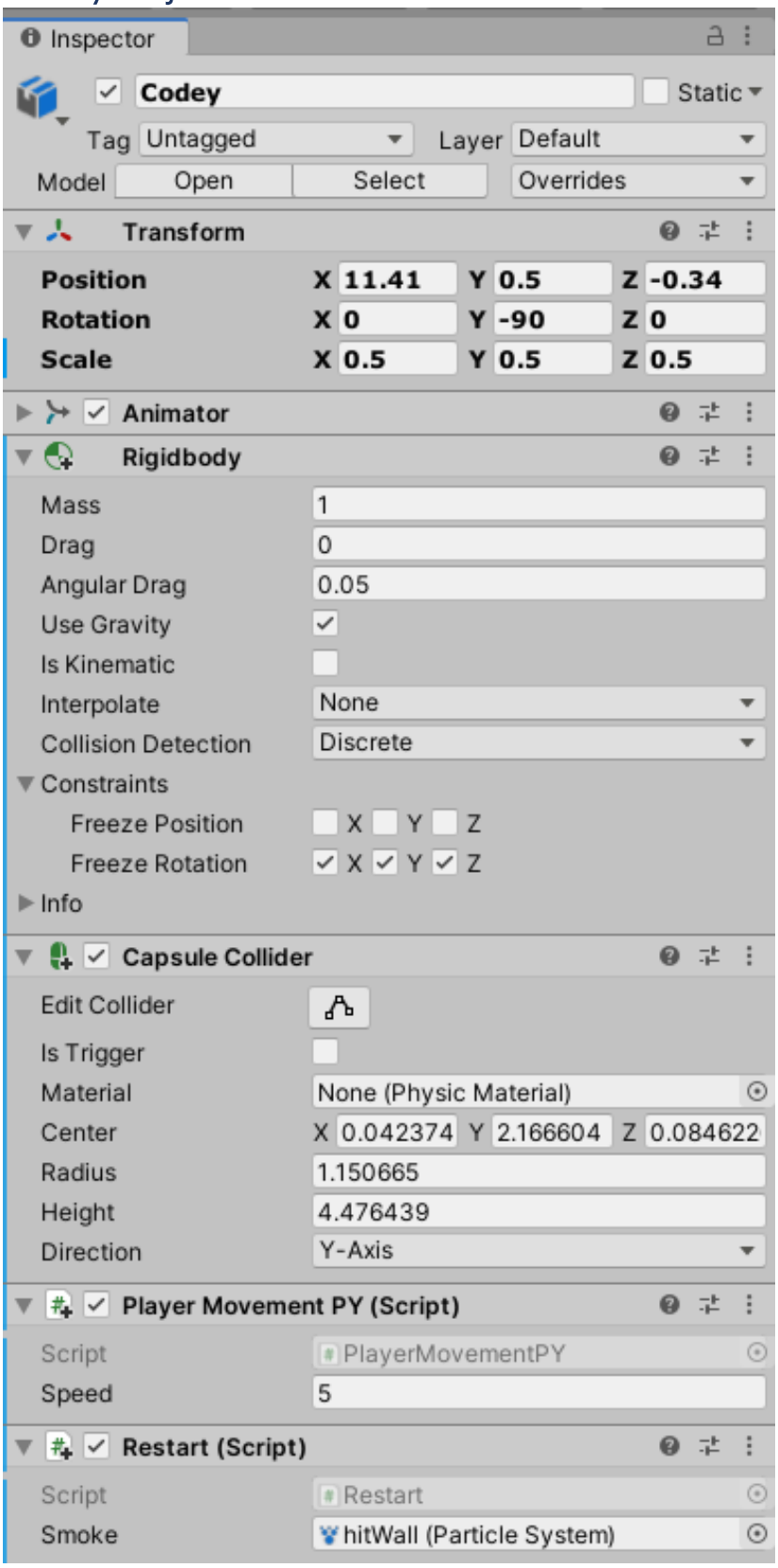
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
    public float speed;

    void Update()
    {
        float horizontal = Input.GetAxis("Horizontal");
        float vertical = Input.GetAxis("Vertical");
        Vector3 destination = new Vector3(horizontal, 0, vertical);
        GetComponent<Rigidbody>().velocity = destination * speed;
    }
}
```

Activity Solution: Find the Exit Prove Yourself

Hierarchy



Codey Object

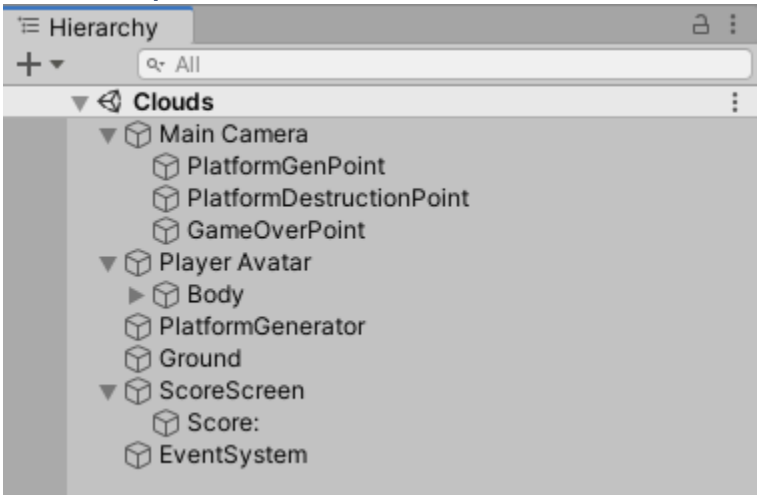


PlayerMovementPY.cs Script

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

public class PlayerMovementPY : MonoBehaviour
{
    public float speed;

    void Update()
    {
        // the horizontal axis determines left and right
        // Codey always moves up
        Vector3 destination = new Vector3(Input.GetAxisRaw("Horizontal"), 0, 1);
        GetComponent<Rigidbody>().velocity = destination * speed;
    }
}
```

Activity Solution: Cloud Hop

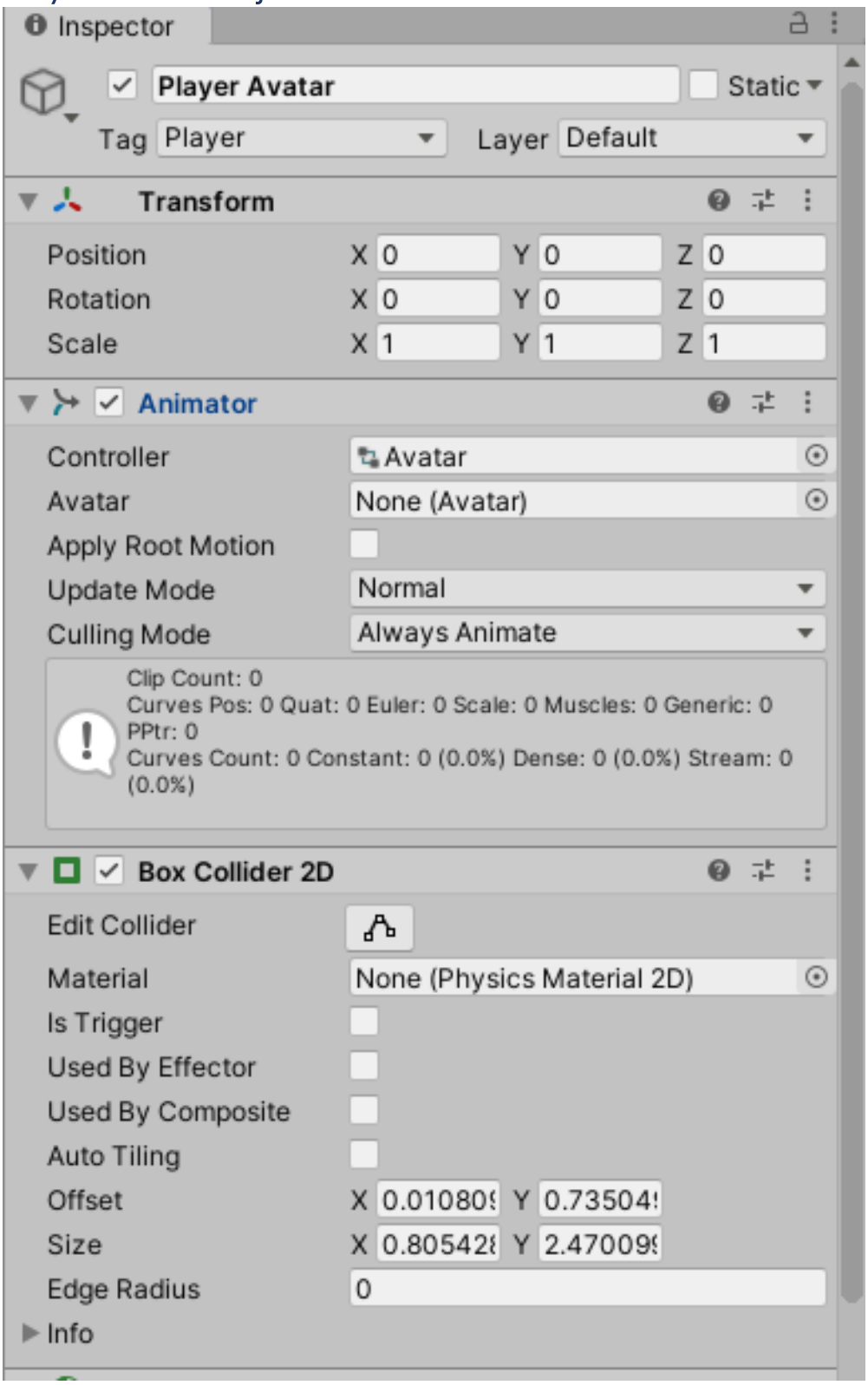
Hierarchy



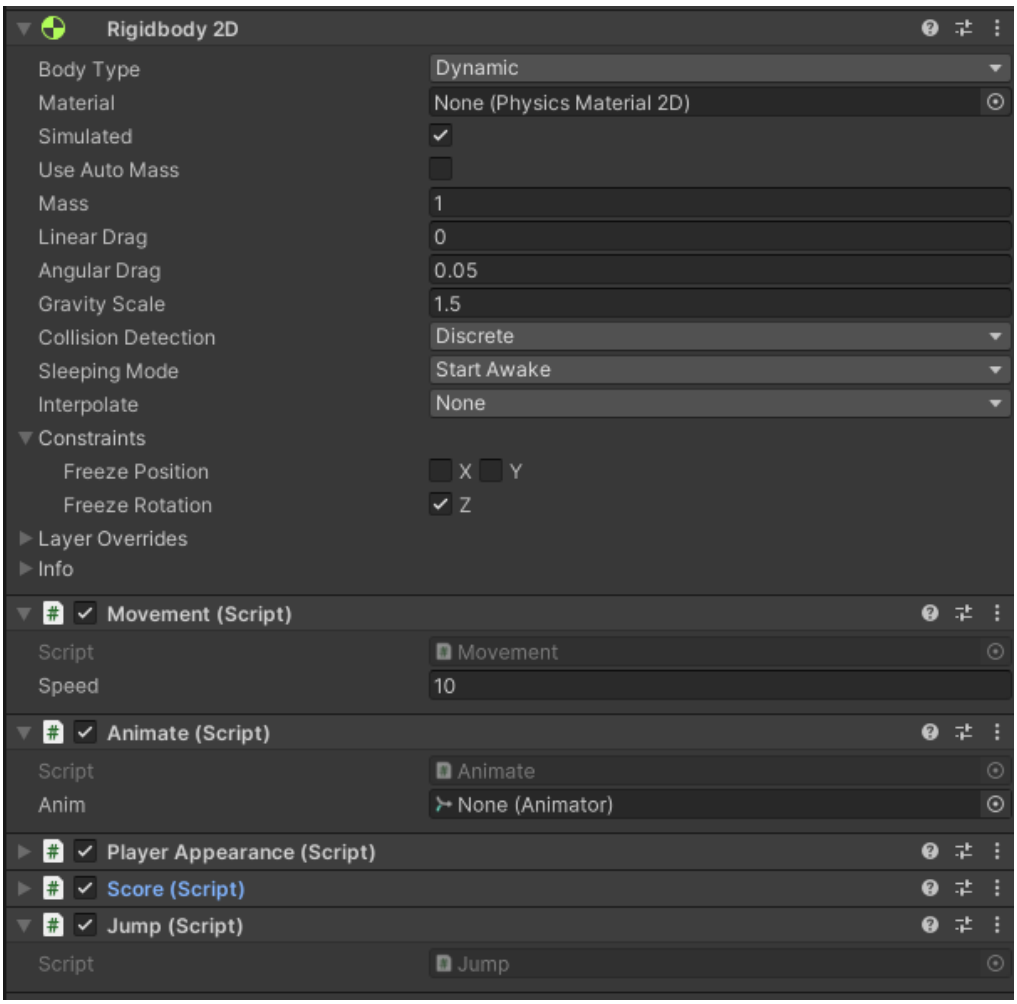
Jump.cs Script

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

public class Jump : MonoBehaviour
{
    private Rigidbody2D rb;
    private float jumpForce = 15;

    // Start is called before the first frame update
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetButtonDown("Jump") && rb.velocity.y == 0)
        {
            rb.AddForce(Vector2.up * jumpForce, ForceMode2D.Impulse);
        }
    }
}
```

Player Avatar Object



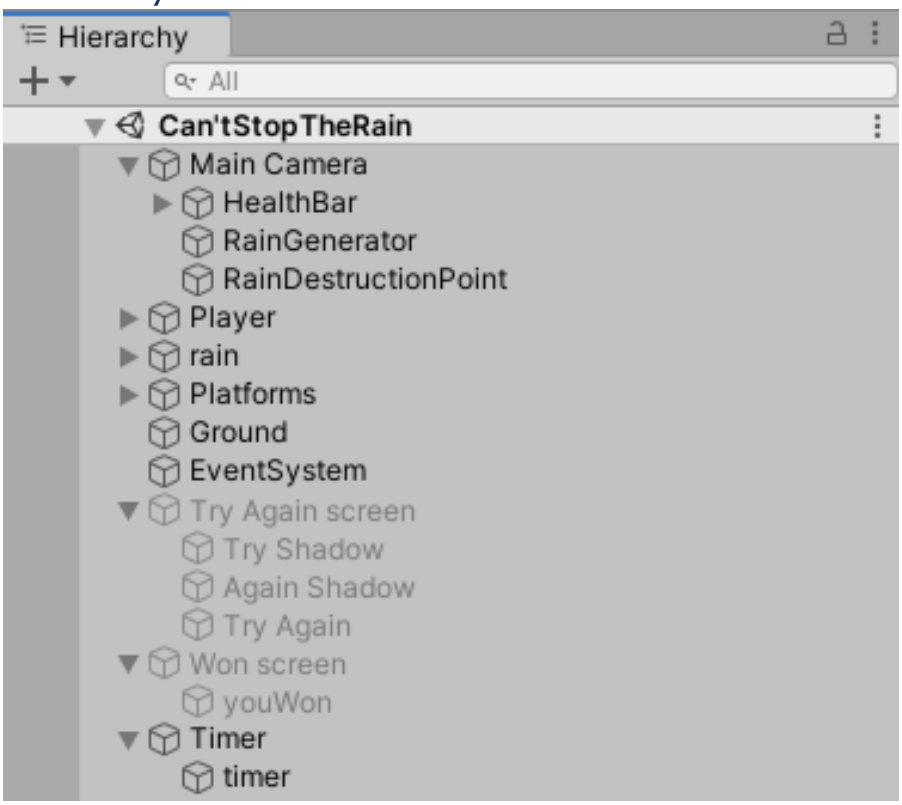
Player Avatar Object Continued



Player Avatar Object Continued

Activity Solution: Cloud Hop Prove Yourself

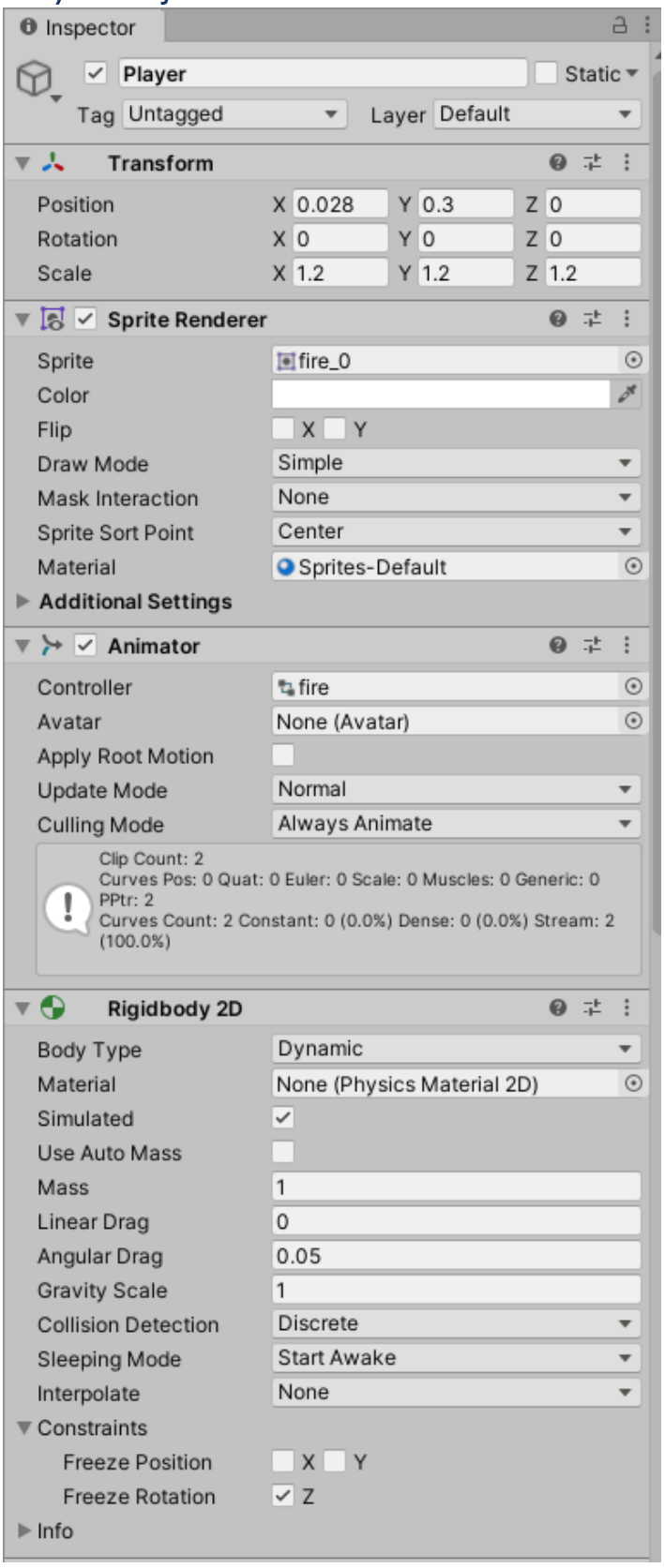
Hierarchy



Jump.cs Script

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

public class Jump : MonoBehaviour
{
    private Rigidbody2D rb;
    private float jumpForce = 15;

    // Start is called before the first frame update
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetButtonDown("Jump") && rb.velocity.y == 0)
        {
            rb.AddForce(Vector2.up * jumpForce, ForceMode2D.Impulse);
        }
    }

}
```

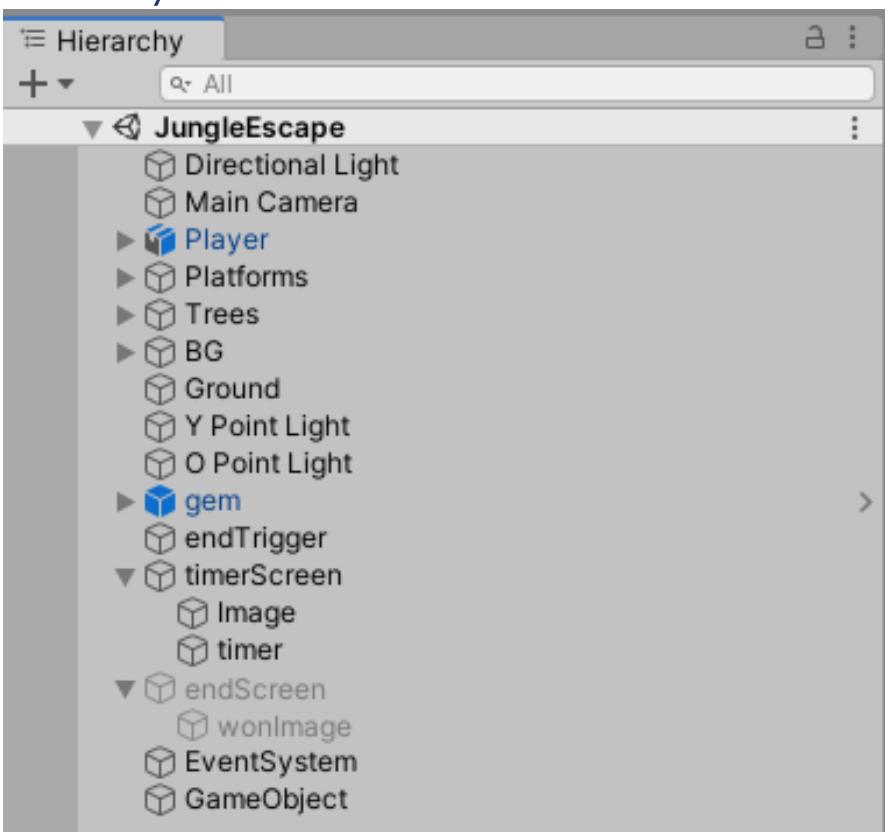
Player Object



Player Object Continued

Activity Solution: Jungle Escape

Hierarchy



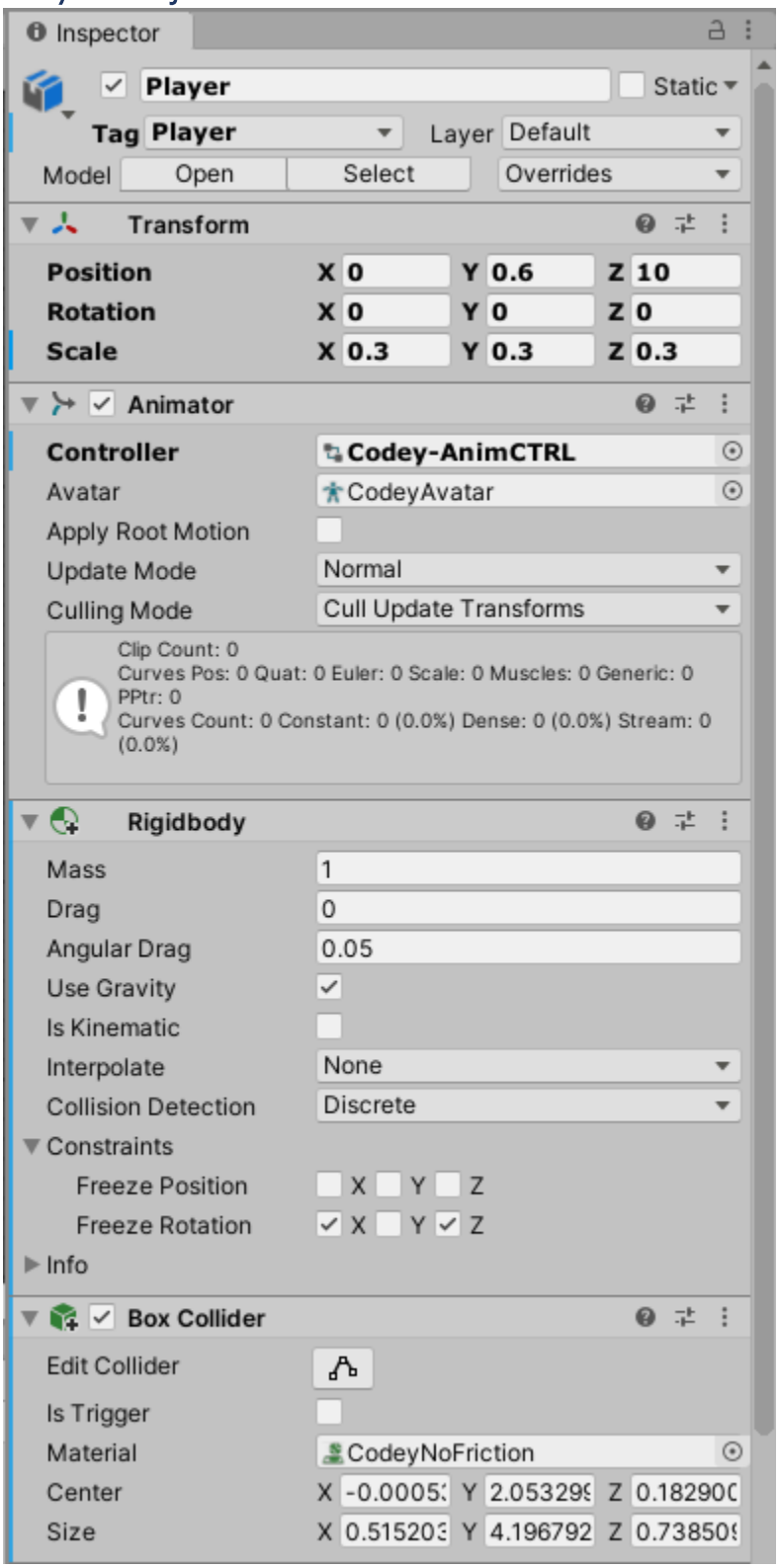
Animate.cs Script

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

public class Animate : MonoBehaviour
{
    Animator animator;
    Jump jump;
    Movement movement;

    void Start()
    {
        animator = GetComponent<Animator>();
        jump = GetComponent<Jump>();
        movement = GetComponent<Movement>();
    }

    void Update()
    {
        animator.SetBool("grounded", jump.isGrounded);
        animator.SetFloat("speed", movement.speed);
    }
}
```

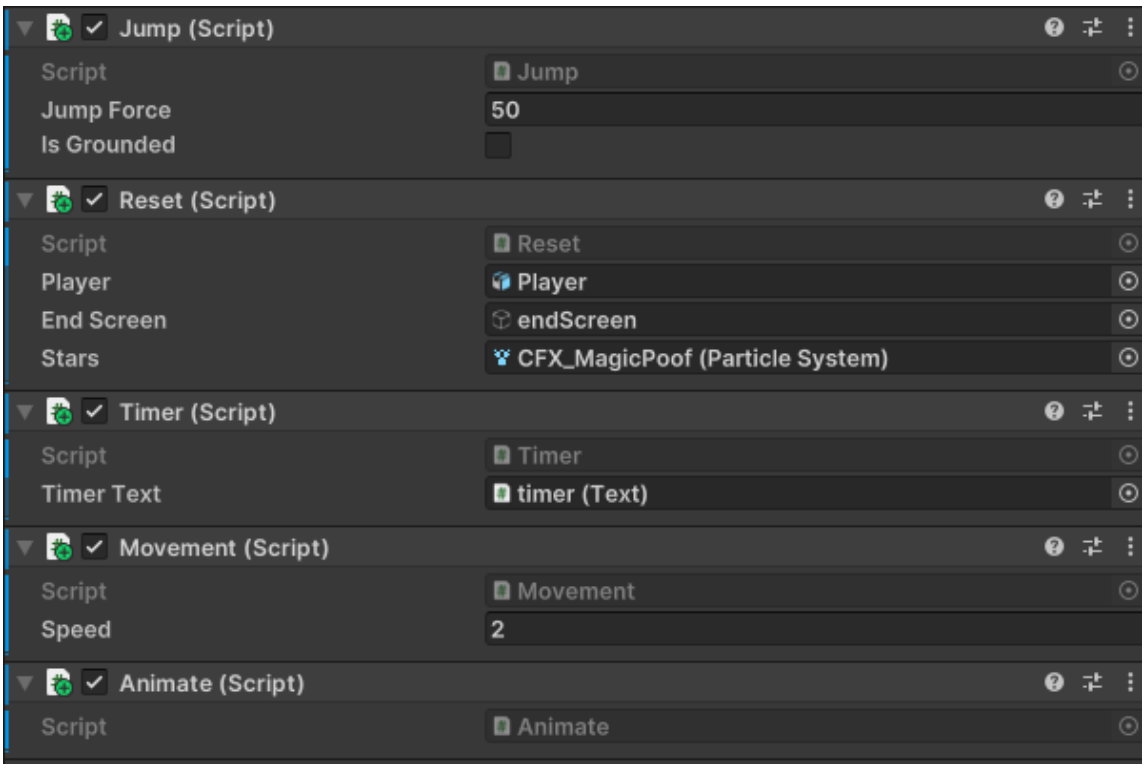
Jump.cs Script

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

public class Jump : MonoBehaviour
{
    Rigidbody rb;

    public float jumpForce = 5.7f;

    public bool isGrounded;

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

    void Update()
    {
        isGrounded = Physics.Raycast(transform.position, Vector3.down, .15f);
        Debug.DrawRay(transform.position, Vector3.down * .15f, Color.red);

        if(Input.GetButtonDown("Jump") && isGrounded){
            rb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
        }
    }
}
```

Player Object

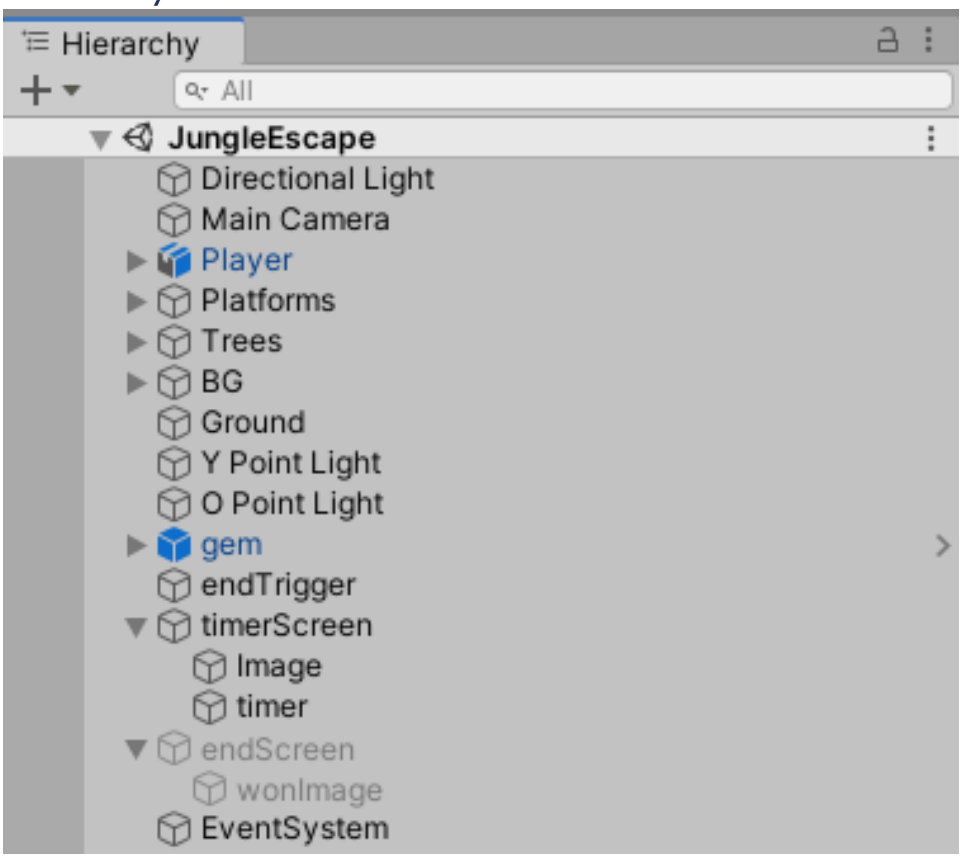


Player Object Continued



Activity Solution: Jungle Escape Prove Yourself

Hierarchy



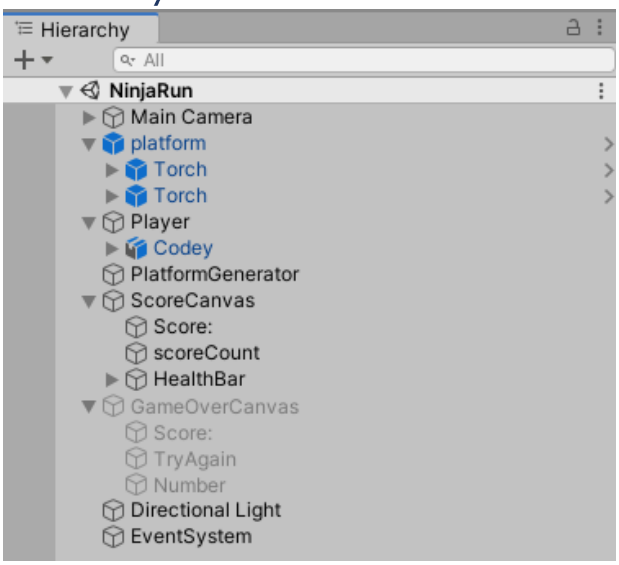
DetectFalsePlatforms.cs Script

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

public class DetectFalsePlatforms : MonoBehaviour
{
    bool hit;
    int FalseLayer = 1 << 8;

    void Update()
    {
        // cast a ray starting at the position that looks forward 3 units
        // only check on layer 8 which only includes the platforms with no colliders
        hit = Physics.Raycast(transform.position, transform.forward, 3, FalseLayer);
        Debug.DrawRay(transform.position, transform.forward * 3, Color.red);

        // if the raycast is true- meaning it has hit a platform with no collider
        // then log a warning saying "Be careful!" to the console
        if (hit)
        {
            Debug.LogWarning("Be careful!");
        }

        // if the raycast is false, it has not hit a platform with no collider
        // then log "All clear!" to the console
        else
        {
            Debug.Log("All clear!");
        }
    }
}
```

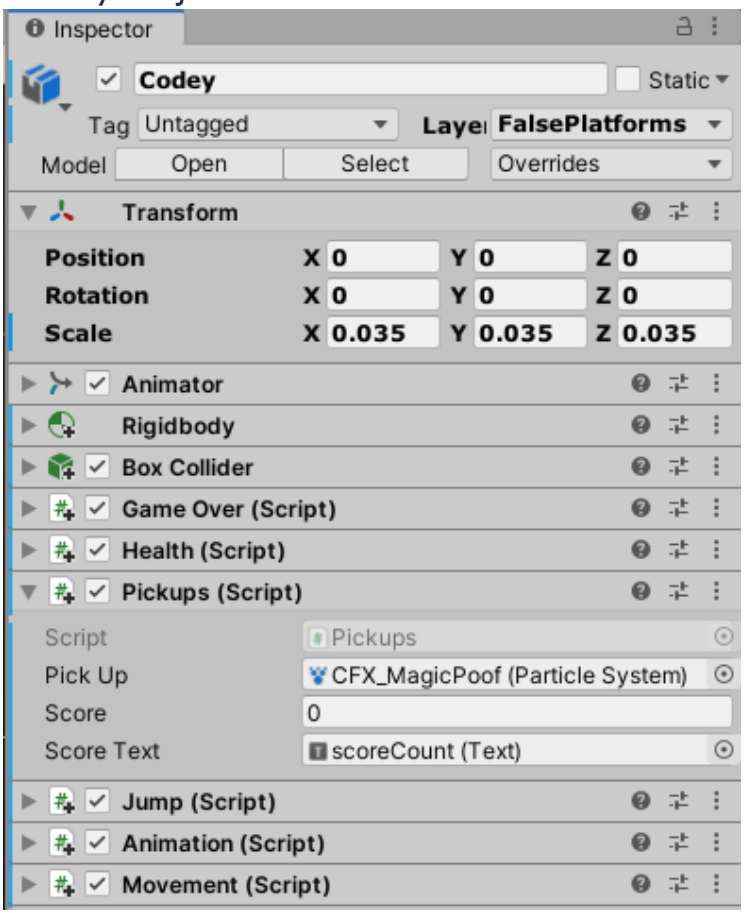
Player Object

Activity Solution: Ninja Run

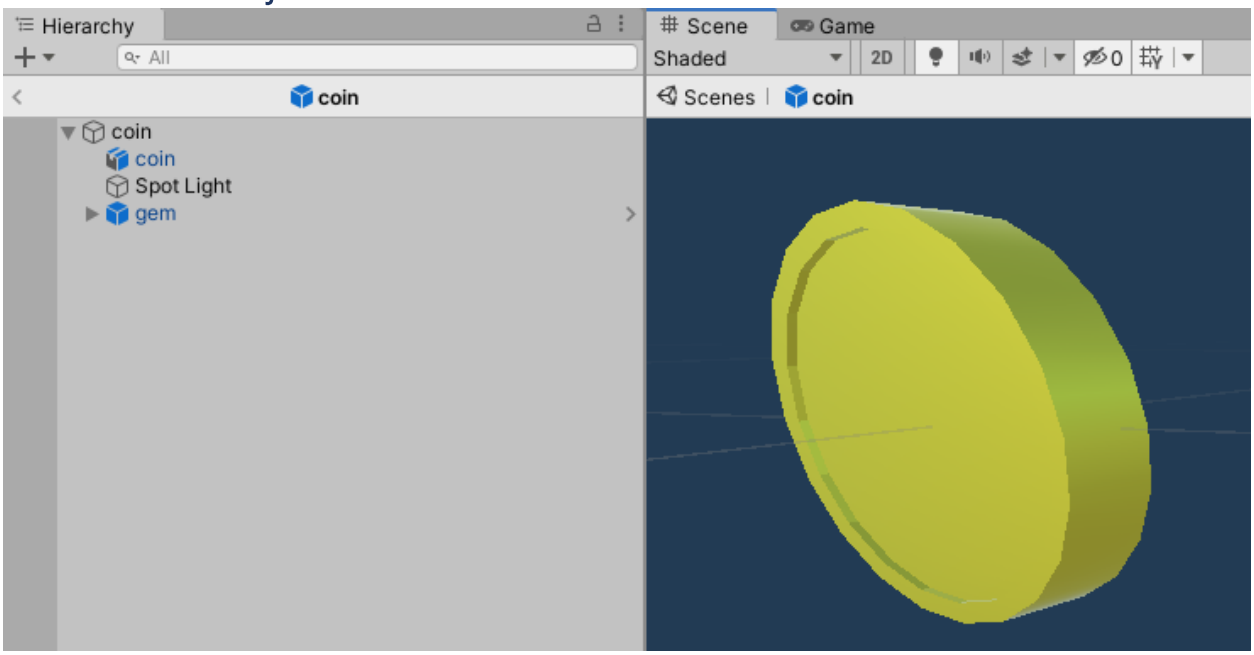
Hierarchy

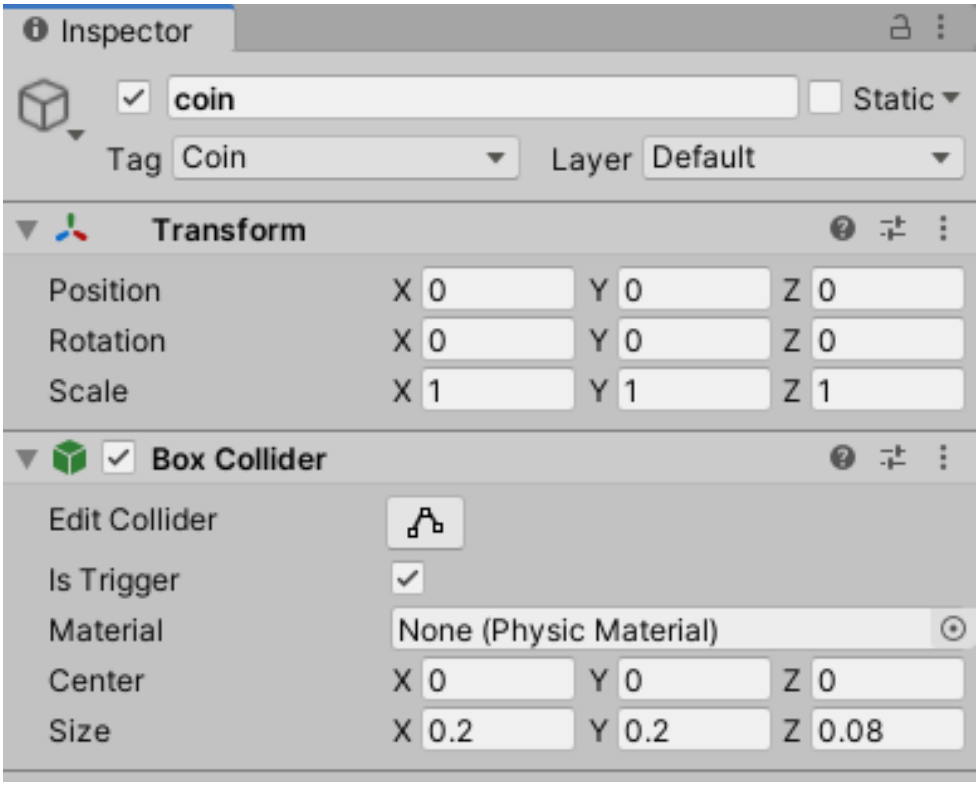


Codey Object



Coin Prefab Object



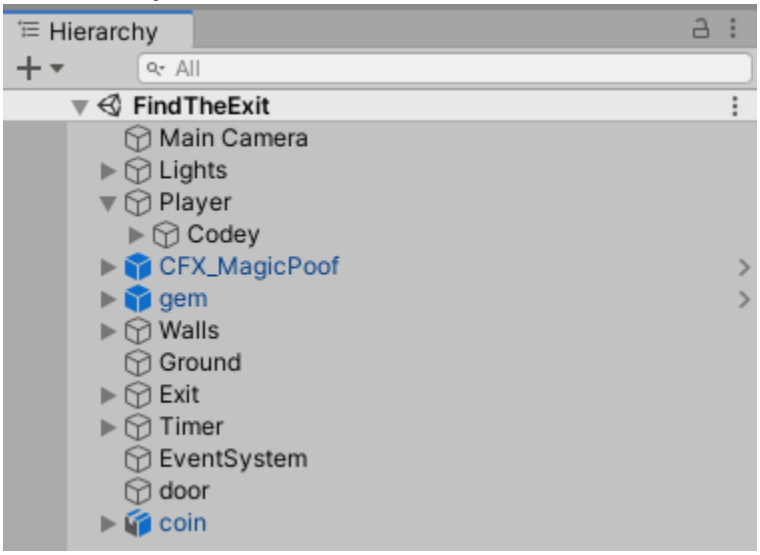


Pickups.cs Script

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

public class Pickups : MonoBehaviour
{
    public ParticleSystem PickUp;

    public int score;
    public Text scoreText;

    void Start()
    {
        PickUp.Stop();
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.CompareTag("Coin"))
        {
            score++;
            scoreText.text = score.ToString();
            Destroy(other.gameObject);
            PickUp.Play();
        }
    }
}
```

Activity Solution: Ninja Run Prove Yourself

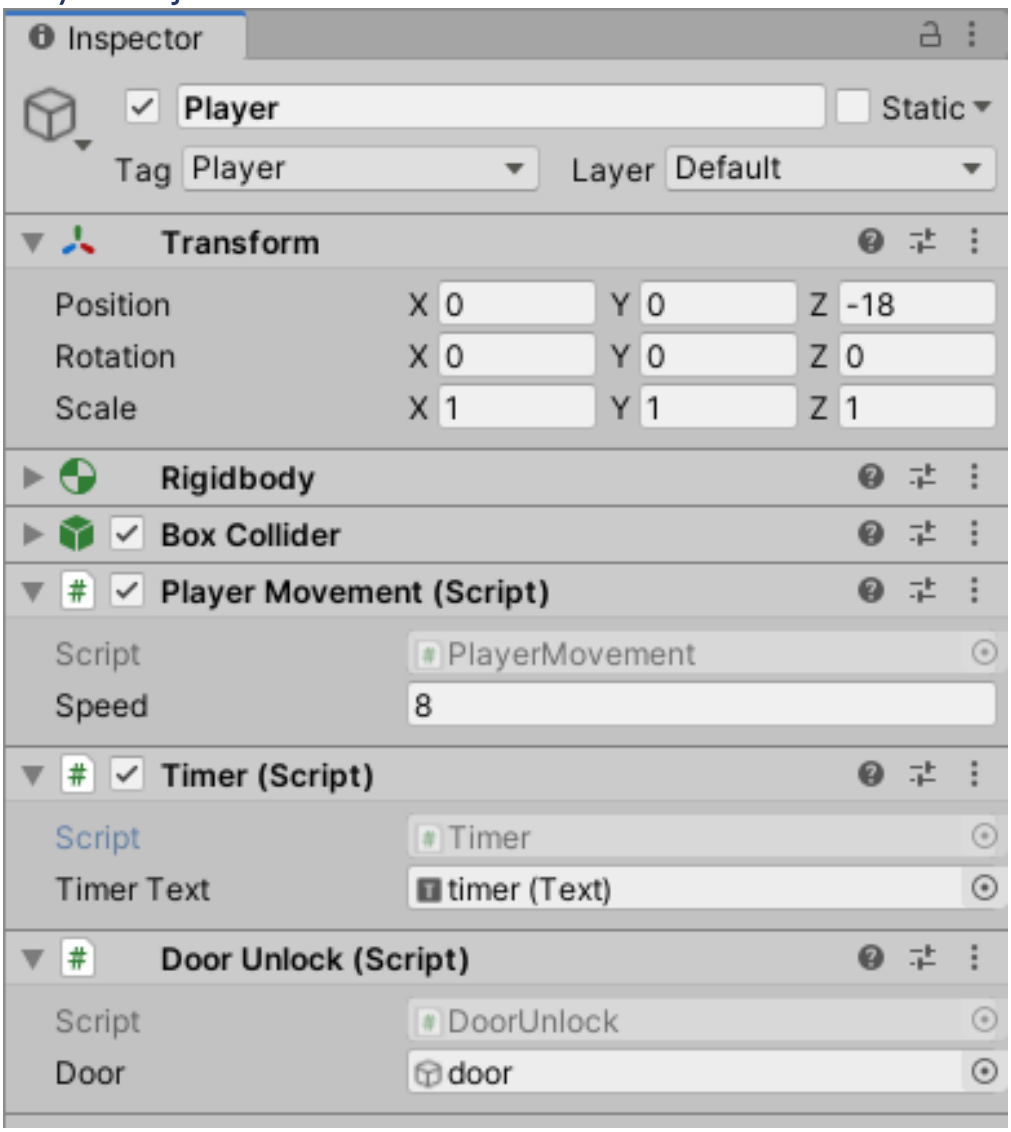
Hierarchy



DoorUnlock.cs Script

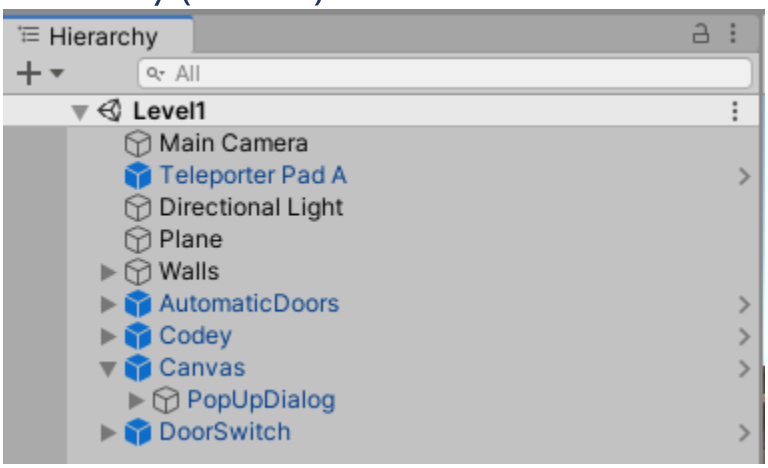
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DoorUnlock : MonoBehaviour
{
    public GameObject door;

    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.CompareTag("coin"))
        {
            Destroy(other.gameObject);
            Destroy(door);
        }
    }
}
```

Player Object

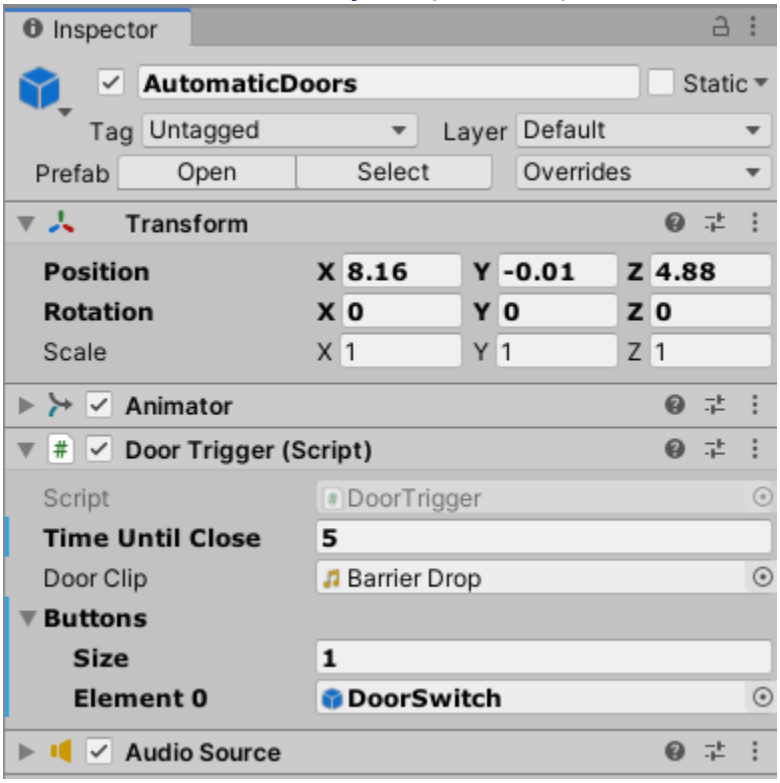


Activity Solution: Evil Fortress of Doctor Worm

Hierarchy (Level 1)



AutomaticDoors Object (Level 1)

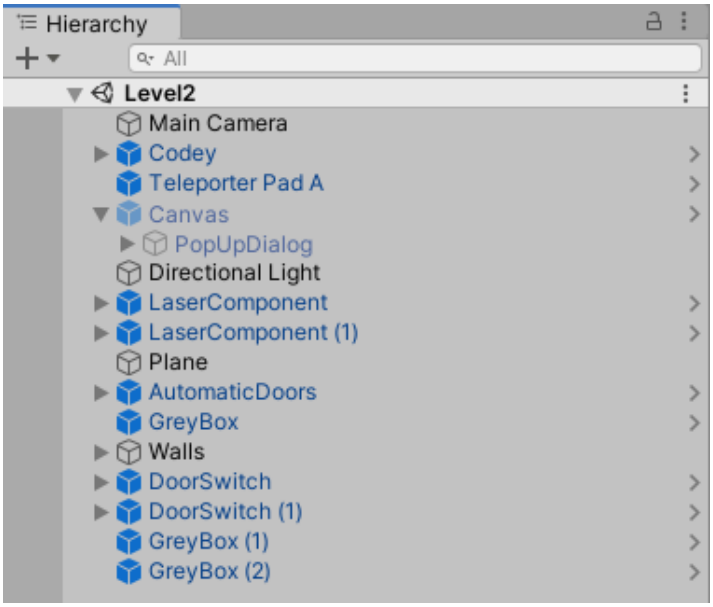


DoorSwitch Object (Level 1)

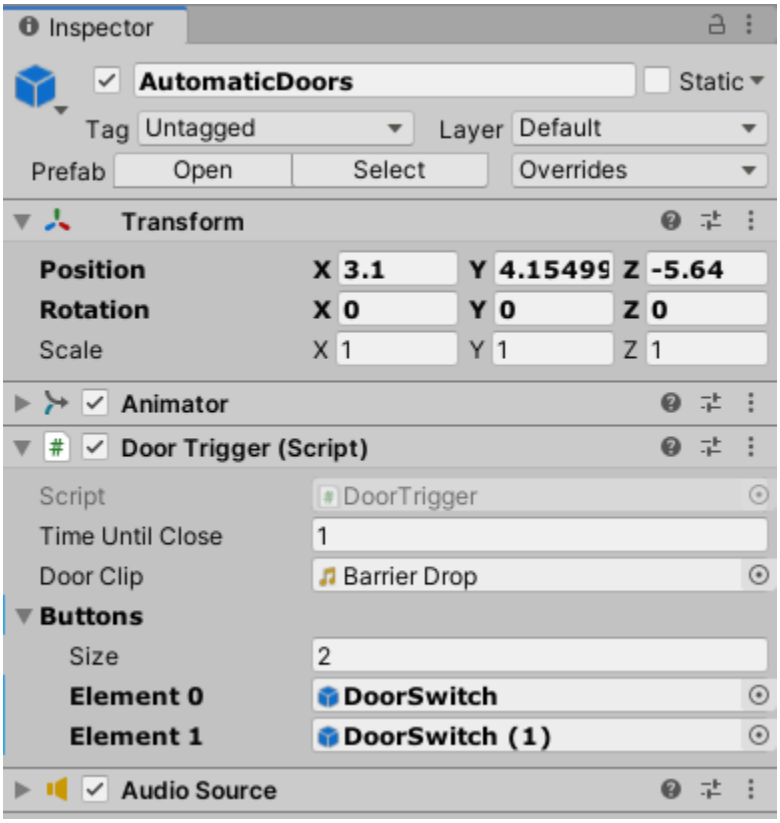
DoorSwitch.cs Script (Level 1)

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

public class DoorSwitch : MonoBehaviour
{
    public DoorTrigger door;
    private GameObject switchIcon;
    private AudioSource playBeep;

    private void Start()
    {
        switchIcon = this.transform.Find("red").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = true;
        switchIcon = this.transform.Find("green").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = false;
        playBeep = GetComponent<AudioSource>();
    }

    public void OnTriggerEnter(Collider other)
    {
        playBeep.Play(0);
    }

    public void OnTriggerStay(Collider other)
    {
        StopAllCoroutines();
        switchIcon = this.transform.Find("green").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = true;
        switchIcon = this.transform.Find("red").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = false;
        door.SwitchCheck();
    }

    public void OnTriggerExit(Collider other)
    {
        switchIcon = this.transform.Find("red").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = true;
        switchIcon = this.transform.Find("green").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = false;
        playBeep.Play(0);
    }
}
```

Hierarchy (Level 2)



AutomaticDoors Object (Level 2)



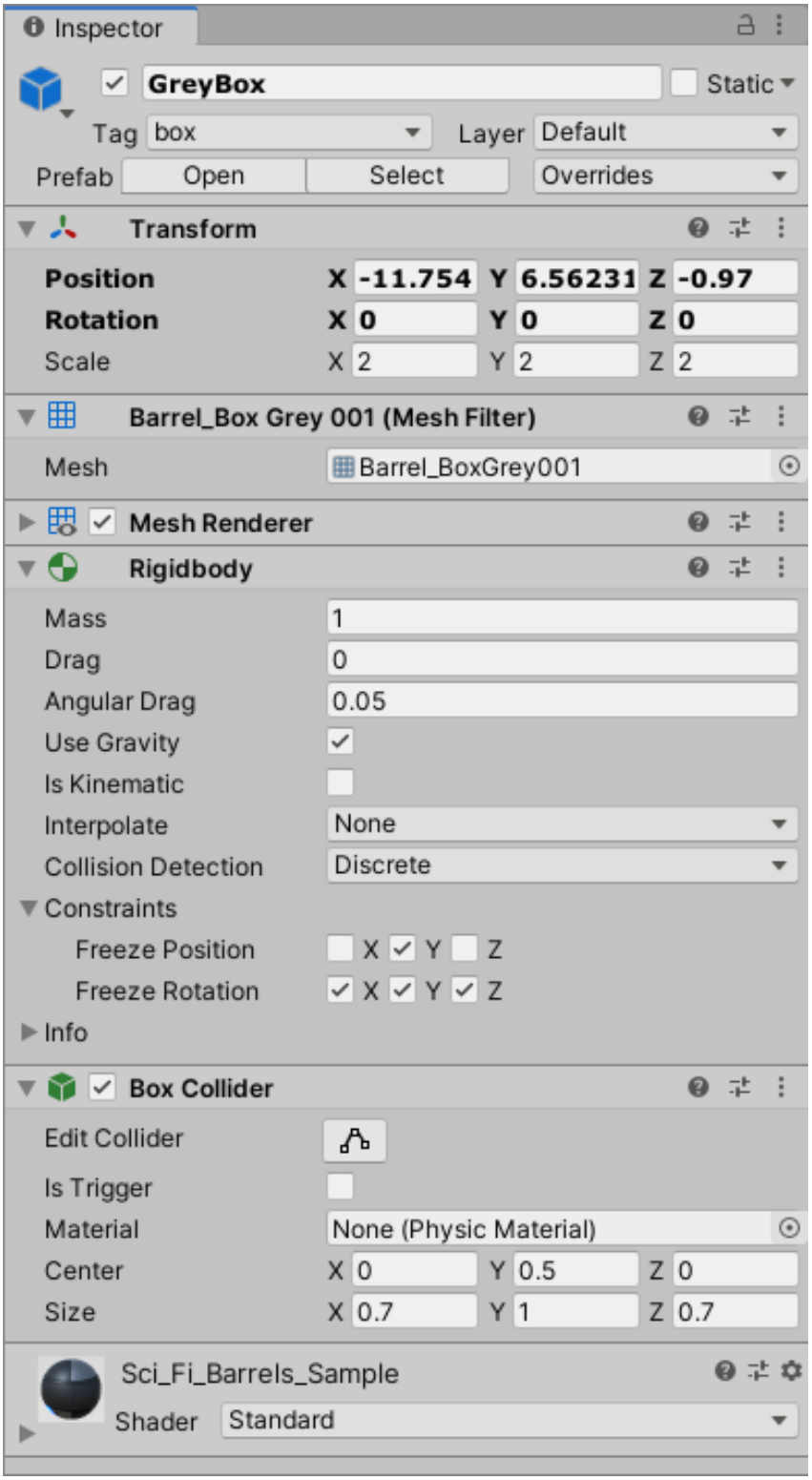
GameOver.cs Script (Level 2)

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

public class GameOver : MonoBehaviour
{
    public static bool isPlayerHit;
    public GameObject player;
    private Animator anim;
    public AudioClip teleportHum;
    public AudioClip laserHit;
    private AudioSource[] allAudioSources;
    private AudioSource teleporterAudio;

    private void Awake()
    {
        teleporterAudio = GetComponent<AudioSource>();
        teleporterAudio.clip = teleportHum;
        teleporterAudio.volume = 0.4f;
        teleporterAudio.Play();
    }

    void Start()
    {
        anim = player.GetComponent<Animator>();
        isPlayerHit = Laser.playerDeath;
        allAudioSources = FindObjectsOfType<AudioSource>();
    }

    private void Update()
    {
        if (!teleporterAudio.isPlaying)
        {
            teleporterAudio.Play();
        }
    }

    public void PlayerHit()
    {
        teleporterAudio.PlayOneShot(laserHit, 1.0f);
        StopAllAudio();
        if (!isPlayerHit)
        {
            anim.SetTrigger("PlayerHit");
            Invoke("RestartLevel", 5);
        }
    }
```

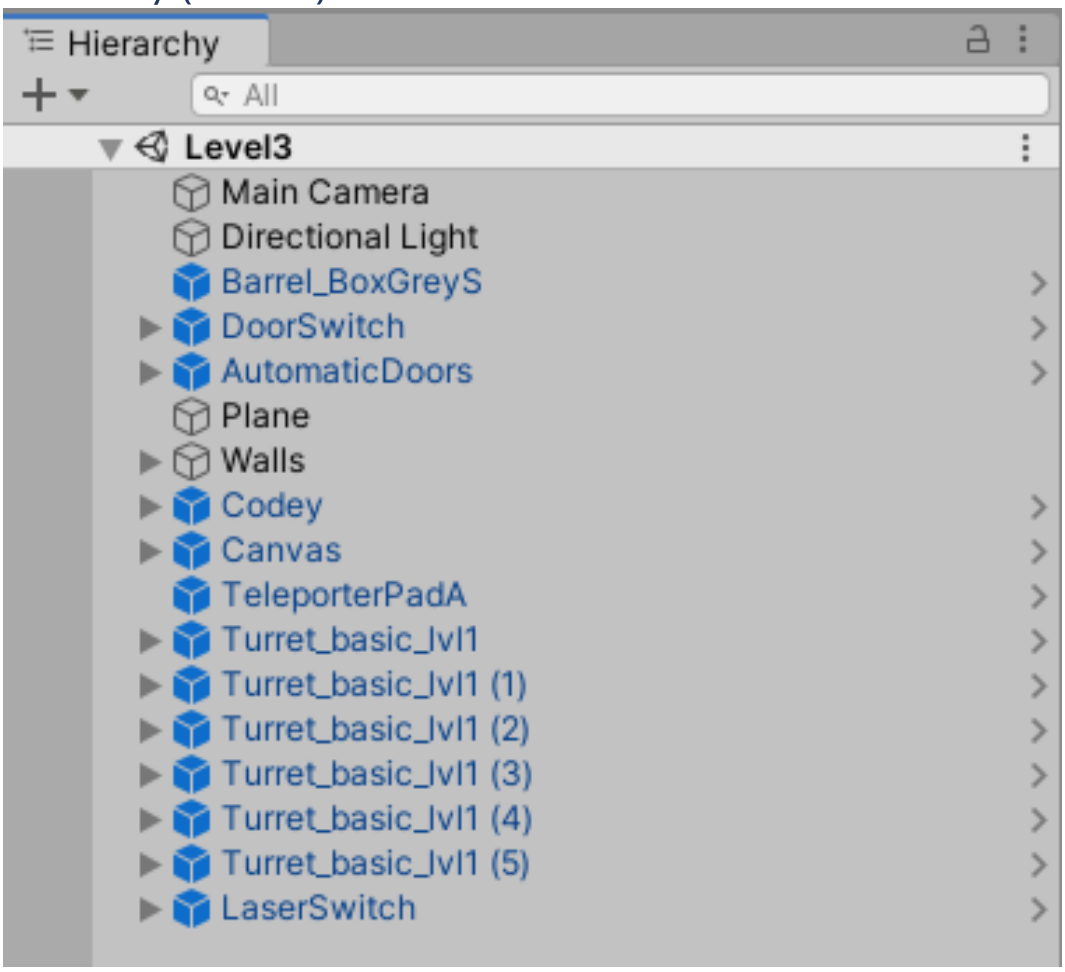
GameOver.cs Script (Level 2) Continued

```
    public void StopAllAudio()
    {
        for (var audioS = 0; audioS < allAudioSources.Length; audioS++)
        {
            if (allAudioSources[audioS] != teleporterAudio)
            {
                allAudioSources[audioS].Stop();
            }
        }

    }

    public void RestartLevel()
    {
        Scene scene = SceneManager.GetActiveScene();
        SceneManager.LoadScene(scene.name);
    }
}
```

GrayBox Object (Level 2)



Hierarchy (Level 3)



LaserSwitch.cs Script (Level 3)

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

public class LaserSwitch : MonoBehaviour
{
    private GameObject switchIcon;
    private AudioSource playBeep;
    public bool lasersAreOff = false;

    private void Start()
    {
        switchIcon = this.transform.Find("red").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = true;
        switchIcon = this.transform.Find("green").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = false;

        playBeep = GetComponent<AudioSource>();
    }

    public void OnTriggerEnter(Collider other)
    {
        playBeep.Play();
        lasersAreOff = true;
    }

    public void OnTriggerStay(Collider other)
    {
        StopAllCoroutines();
        switchIcon = this.transform.Find("green").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = true;
        switchIcon = this.transform.Find("red").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = false;

    }

    public void OnTriggerExit(Collider other)
    {
        switchIcon = this.transform.Find("red").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = true;
        switchIcon = this.transform.Find("green").gameObject;
        switchIcon.GetComponent<SpriteRenderer>().enabled = false;
    }
}
```

TurretLook.cs Script (Level 3)

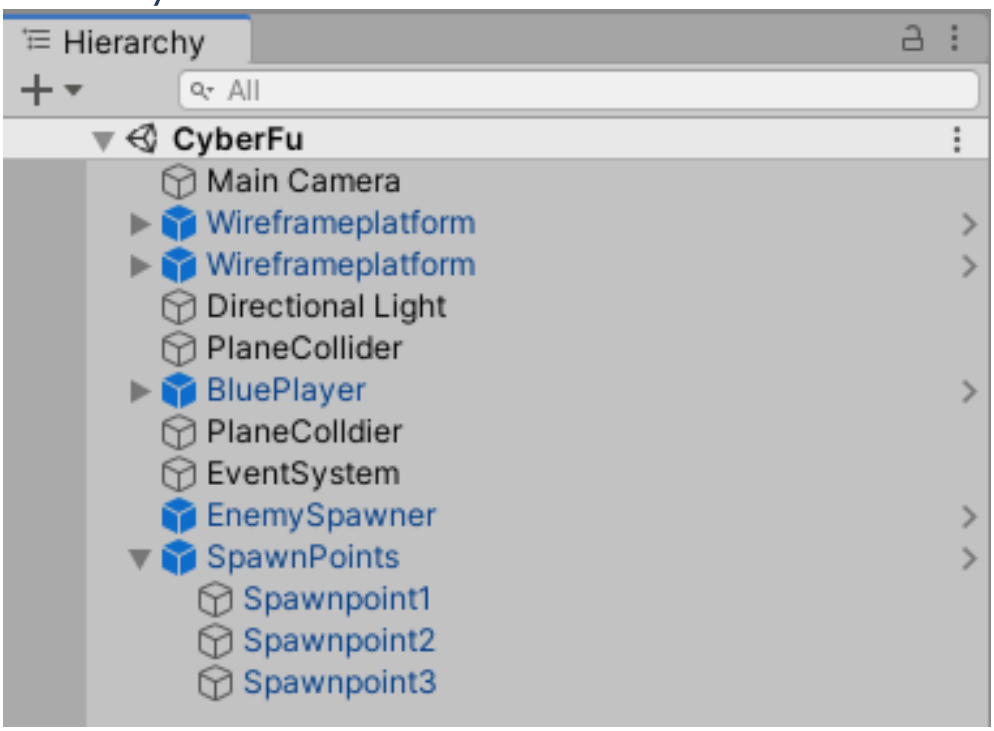
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TurretLook : MonoBehaviour
{
    public Transform player;
    private LaserSwitch laserSwitch;
    private GameObject laser;

    private void Start()
    {
        laserSwitch = GameObject.Find("LaserSwitch").GetComponent<LaserSwitch>();
        laser = transform.Find("Laser").gameObject;
    }

    void Update()
    {
        if (laserSwitch.lasersAreOff)
        {
            laser.setActive(false);
            return;
        }
        transform.LookAt(player);
    }
}
```

Activity Solution: Evil Fortress of Doctor Worm Prove Yourself

There is no provided solution for this activity. Each ninja must use what they learned in parts 1, 2, and 3 to compose their very own unique level.

Activity Solution: CyberFu Part 1

Hierarchy

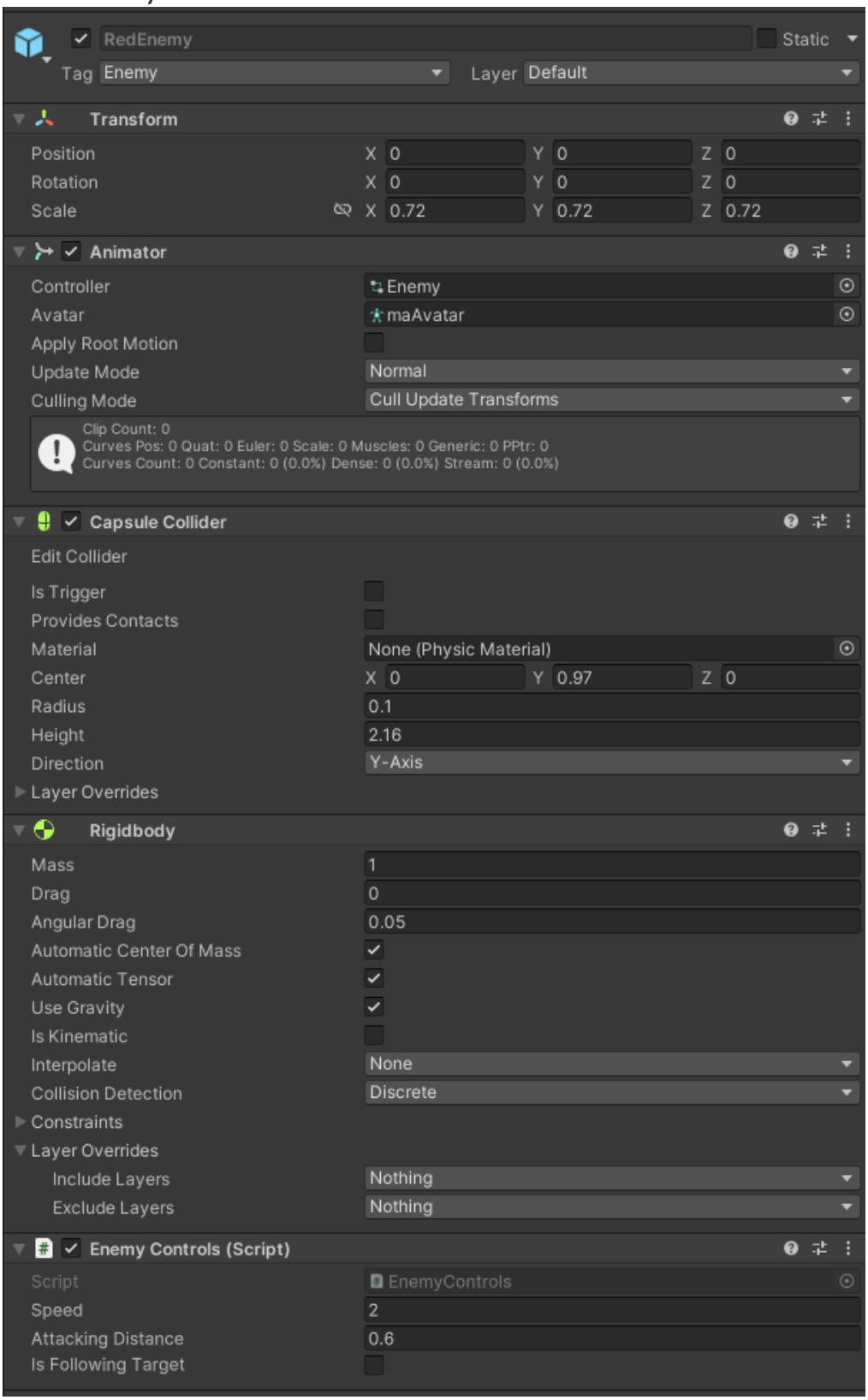


EnemyControls.cs Script

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

public class EnemyControls : MonoBehaviour
{
    public float speed = 2f;
    public float attackingDistance = 0.6f;

    private Animator animatorEnemy;
    private Rigidbody rigidbodyEnemy;
    private Transform target;

    [SerializeField]
    private bool isFollowingTarget;

    private float currentAttackingTime = 0f;
    private float maxAttackingTime = 2f;

    // Start is called before the first frame update
    void Start()
    {
        animatorEnemy = GetComponent<Animator>();
        rigidbodyEnemy = GetComponent<Rigidbody>();
        target = GameObject.FindGameObjectWithTag("Player").transform;
    }

    // Update is called once per frame
    void Update()
    {
        transform.LookAt(target.position);

        isFollowingTarget = Vector3.Distance(transform.position, target.position) >=
attackingDistance;

        if (isFollowingTarget)
        {
            rigidbodyEnemy.velocity = transform.forward * speed;
        }
        else
        {
            Attack();
        }

        animatorEnemy.SetBool("Walk", isFollowingTarget);
    }

    void Attack()
    {
        rigidbodyEnemy.velocity = Vector3.zero;

        currentAttackingTime += Time.deltaTime;
```

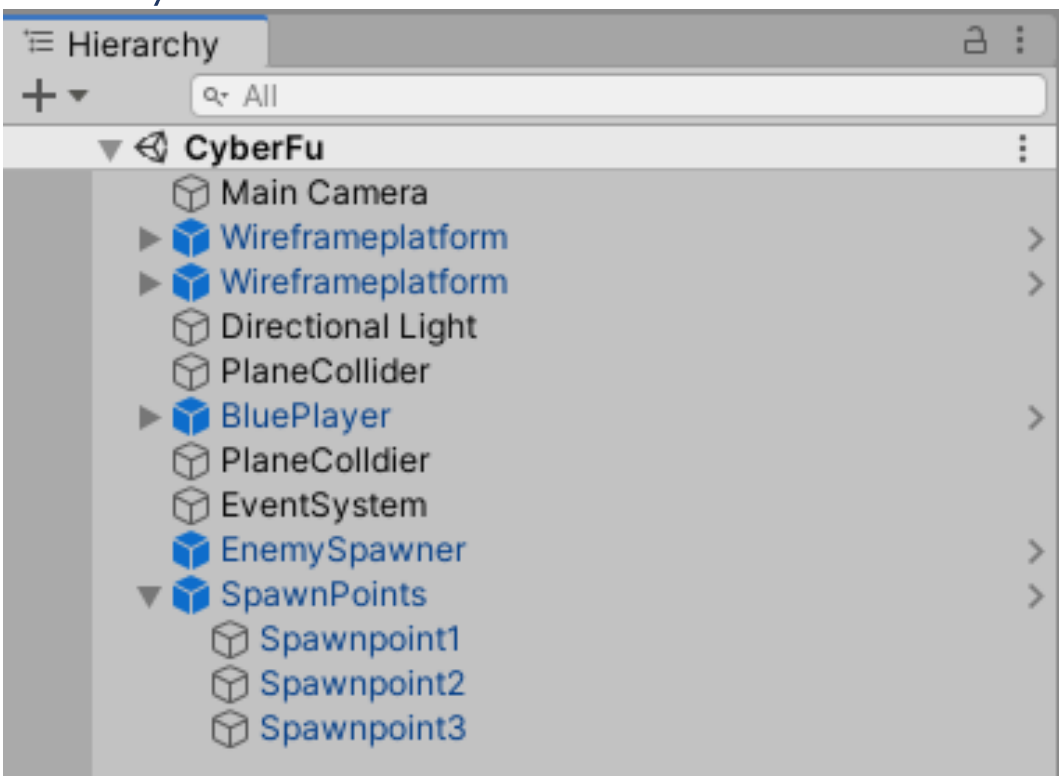
```
        if (currentAttackingTime > maxAttackingTime)
        {
            currentAttackingTime = 0f;
            int rand = Random.Range(1, 4);
            animatorEnemy.SetTrigger("Attack" + rand);
        }
    }
}
```

RedEnemy Prefab



Activity Solution: CyberFu Part 1 Prove Yourself

Hierarchy



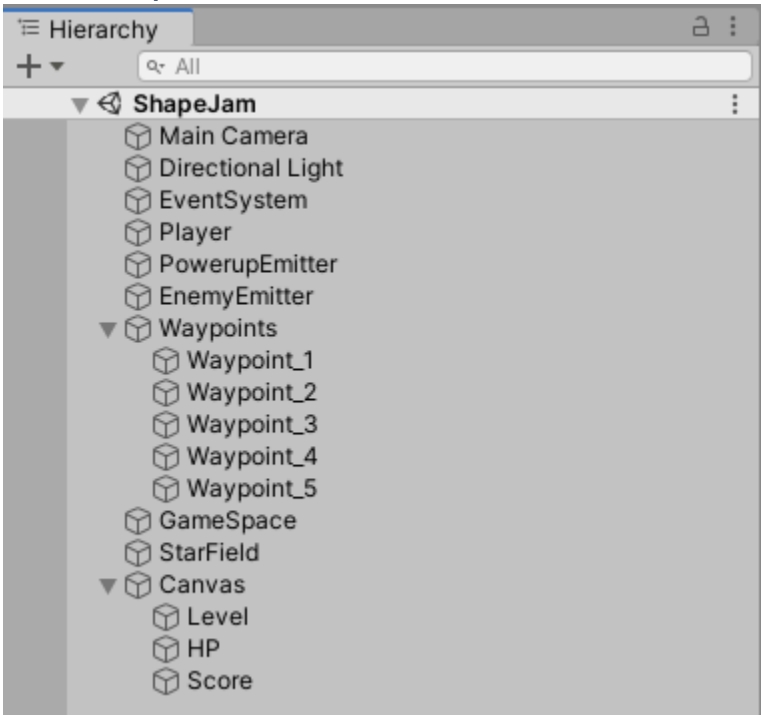
EnemyControls.cs Script

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

public class EnemyControls : MonoBehaviour
{
    public float speed = 2f;
    public float attackingDistance = 0.6f;

    private Animator animatorEnemy;
    private Rigidbody rigidbodyEnemy;
    private Transform target;

    [SerializeField]
    private bool isFollowingTarget;

    private float currentAttackingTime = 0f;
    private float maxAttackingTime = 2f;

    // Start is called before the first frame update
    void Start()
    {
        animatorEnemy = GetComponent<Animator>();
        rigidbodyEnemy = GetComponent<Rigidbody>();
        target = GameObject.FindGameObjectWithTag("Player").transform;
    }

    // Update is called once per frame
    void Update()
    {
        transform.LookAt(target.position);

        isFollowingTarget = Vector3.Distance(transform.position, target.position) >=
attackingDistance;

        if (isFollowingTarget)
        {
            rigidbodyEnemy.velocity = transform.forward * speed;
        }
        else
        {
            Attack();
        }

        animatorEnemy.SetBool("Walk", isFollowingTarget);
    }

    void Attack()
    {
        rigidbodyEnemy.velocity = Vector3.zero;

        currentAttackingTime += Time.deltaTime;

        if (currentAttackingTime > maxAttackingTime)
        {
```

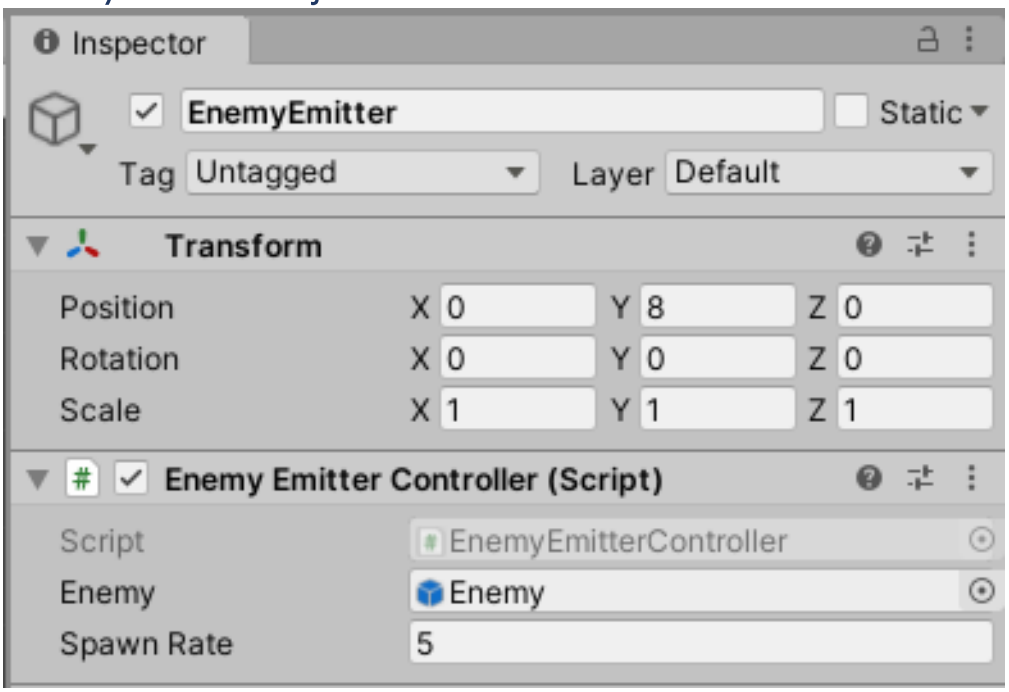
```
            currentAttackingTime = 0f;
            int rand = Random.Range(1, 7);
            animatorEnemy.SetTrigger("Attack" + rand);
        }
    }
}
```

Activity Solution: Shape Jam

Hierarchy



EnemyEmitter Object



EnemyEmitterController.cs Script

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

public class EnemyEmitterController : MonoBehaviour
{
    public GameObject Enemy;
    private PlayerControls playerController;
    public float spawnRate;
    private float nextSpawn = 5.0f;

    void Start()
    {
        GameObject player = GameObject.Find("Player");
        playerController = player.GetComponent<PlayerControls>();

    }
    void Update()
    {
        if (playerController.gameOver)
        {
            return;
        }
        if (Time.time > nextSpawn)
        {
            nextSpawn = Time.time + spawnRate;
            /*****************************\
            |**** Add your code below ****|
            \*****************************/
            for (int i = 0; i < playerController.currentLevel; i++)
            {
                float randomX = Random.Range(-6.0f, 6.0f);
                Vector3 enemyPosition = new Vector3(randomX, 6, 0);
            }
            /*****************************\
            |**** Add your code above ****|
            \*****************************/
        }
    }
}
```

HazardFiring.cs Script

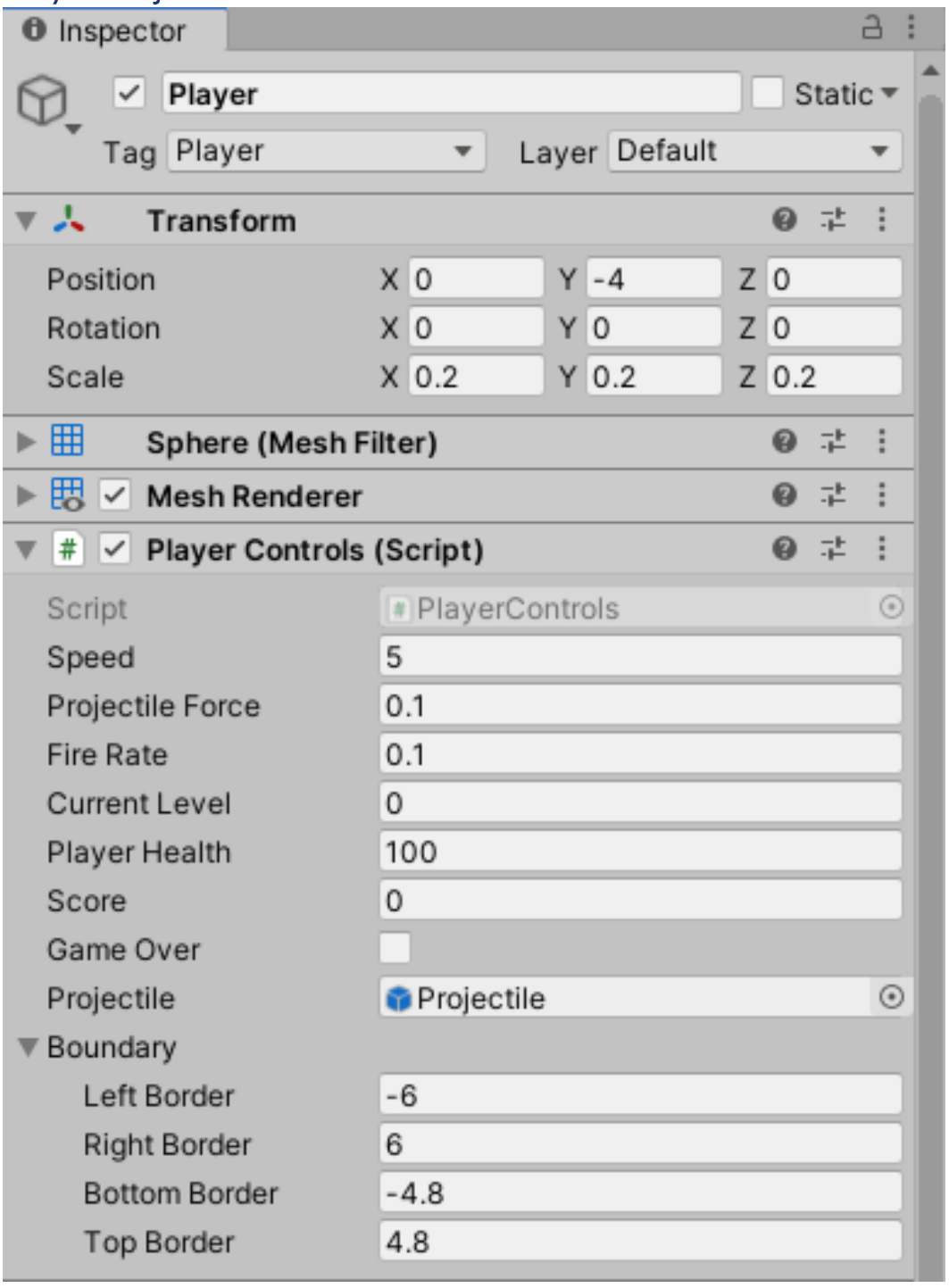
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class HazardFiring : MonoBehaviour
{
    public GameObject hazard;
    public GameObject player;
    public Vector3 target;
    public Vector3 HazardMoveDirection;
    public Transform hazardSpawn;
    public float fireRate;
    private float nextFire;
    public float speed;
    private PlayerControls playerController;

    void Start()
    {
        player = GameObject.Find("Player");
        playerController = player.GetComponent<PlayerControls>();
    }
    void Update()
    {
        if (playerController.gameOver)
        {
            return;
        }

        target = player.transform.position;

        HazardMoveDirection = target - transform.position;

        if (Time.time > nextFire)
        {
            nextFire = Time.time + fireRate;

            /*****************************\
            |**** Add your code below ****|
            \*****************************/
            GameObject hazardClone = Instantiate(hazard, transform.position,
transform.rotation);

            Rigidbody hazardrigidBody = hazardClone.GetComponent<Rigidbody>();
            hazardrigidBody.velocity = HazardMoveDirection.normalized * speed;

            /*****************************\
            |**** Add your code above ****|
            \*****************************/
        }
    }
}
```

PlayerControls.cs Script

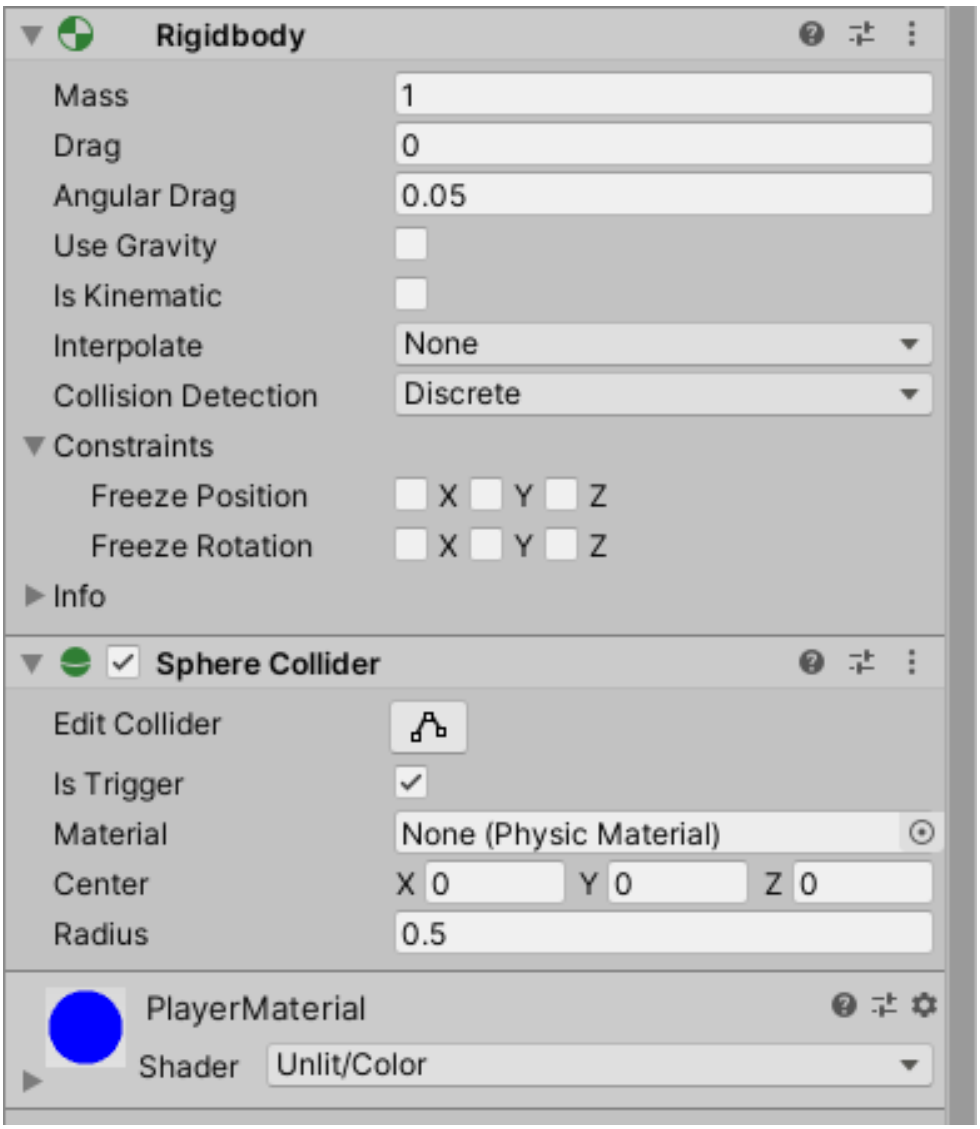
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[System.Serializable]
public class Boundaries
{
    public float leftBorder, rightBorder, bottomBorder, topBorder;
}
public class PlayerControls : MonoBehaviour
{
    public float speed;
    public float projectileForce;
    public float fireRate;
    public int currentLevel;
    private float nextFire;
    public int playerHealth;
    public int score;
    public bool gameOver = false;
    private Vector2 moveDirection = Vector2.zero;
    public GameObject projectile;
    public Boundaries boundary;
    private Rigidbody myRigidBody;

    void Start()
    {
        currentLevel = 1;
        score = 0;
        myRigidBody = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        moveDirection = new Vector2(
              Input.GetAxis("Horizontal"), Input.GetAxis("Vertical"));
        if (!gameOver)
        {
            myRigidBody.velocity = moveDirection * speed;
        }
        myRigidBody.position = new Vector2(
            Mathf.Clamp(
              myRigidBody.position.x, boundary.leftBorder, boundary.rightBorder),
            Mathf.Clamp(
              myRigidBody.position.y, boundary.bottomBorder, boundary.topBorder)
        );
    }
```

```
    void Update()
    {
        if (gameOver)
        {
            return;
        }

        if ((Input.GetKey("space") || Input.GetKey("z")) && (Time.time > nextFire))
        {
            nextFire = Time.time + fireRate;

            /*****************************\
            |**** Add your code below ****|
            \*****************************/
            Instantiate(projectile, transform.position, transform.rotation);
            if (currentLevel >= 3)
            {
                Vector3 rightOffset = new Vector3(0.2f, 0, 0);
                Vector3 leftOffset = new Vector3(-0.2f, 0, 0);
                Instantiate(projectile, transform.position + rightOffset,
transform.rotation);
                Instantiate(projectile, transform.position + leftOffset,
transform.rotation);
            }
            if (currentLevel >= 3)
            {
                Vector3 rightOffset = new Vector3(0.2f, 0.2f, 0);
                Vector3 leftOffset = new Vector3(-0.2f, 0.2f, 0);
                Instantiate(
                    projectile, transform.position + rightOffset, transform.rotation);
                Instantiate(
                    projectile, transform.position + leftOffset, transform.rotation);
            }
            /*****************************\
            |**** Add your code above ****|
            \*****************************/
        }
        if (playerHealth <= 0)
        {
            gameOver = true;
        }
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "Hazard")
        {
            Debug.Log("Player hit!");
            Destroy(other.gameObject);
            playerHealth--;
        }
    }
}
```

Player Object



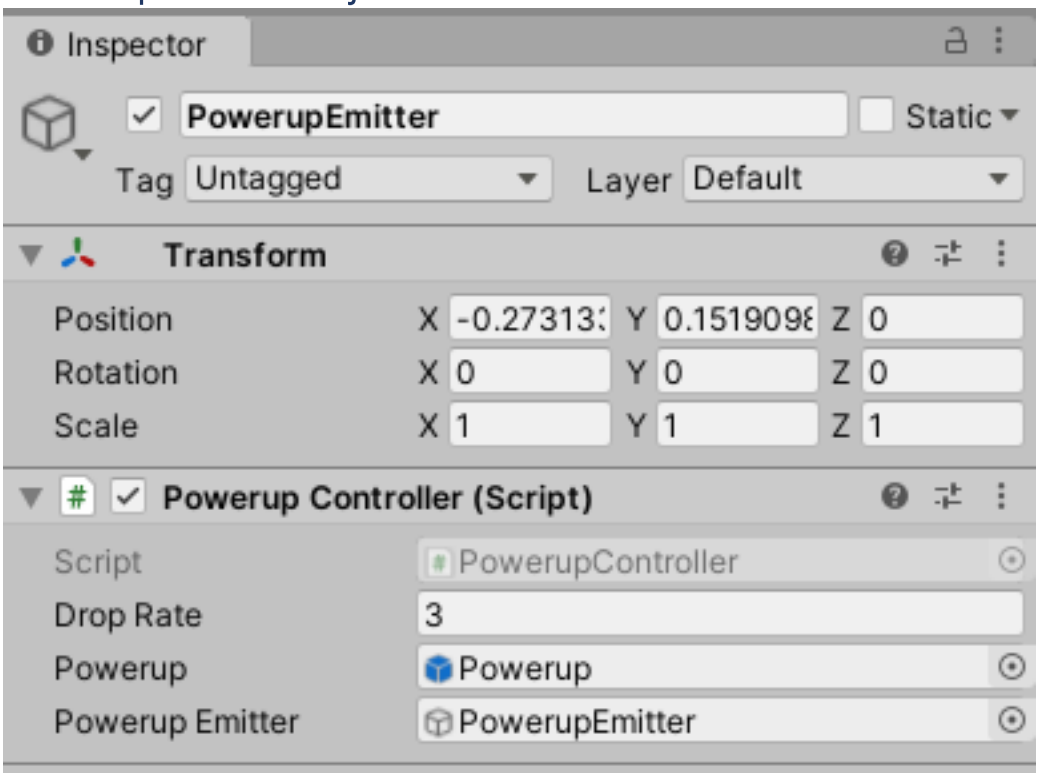
Player Object Continued



PowerupController.cs Script

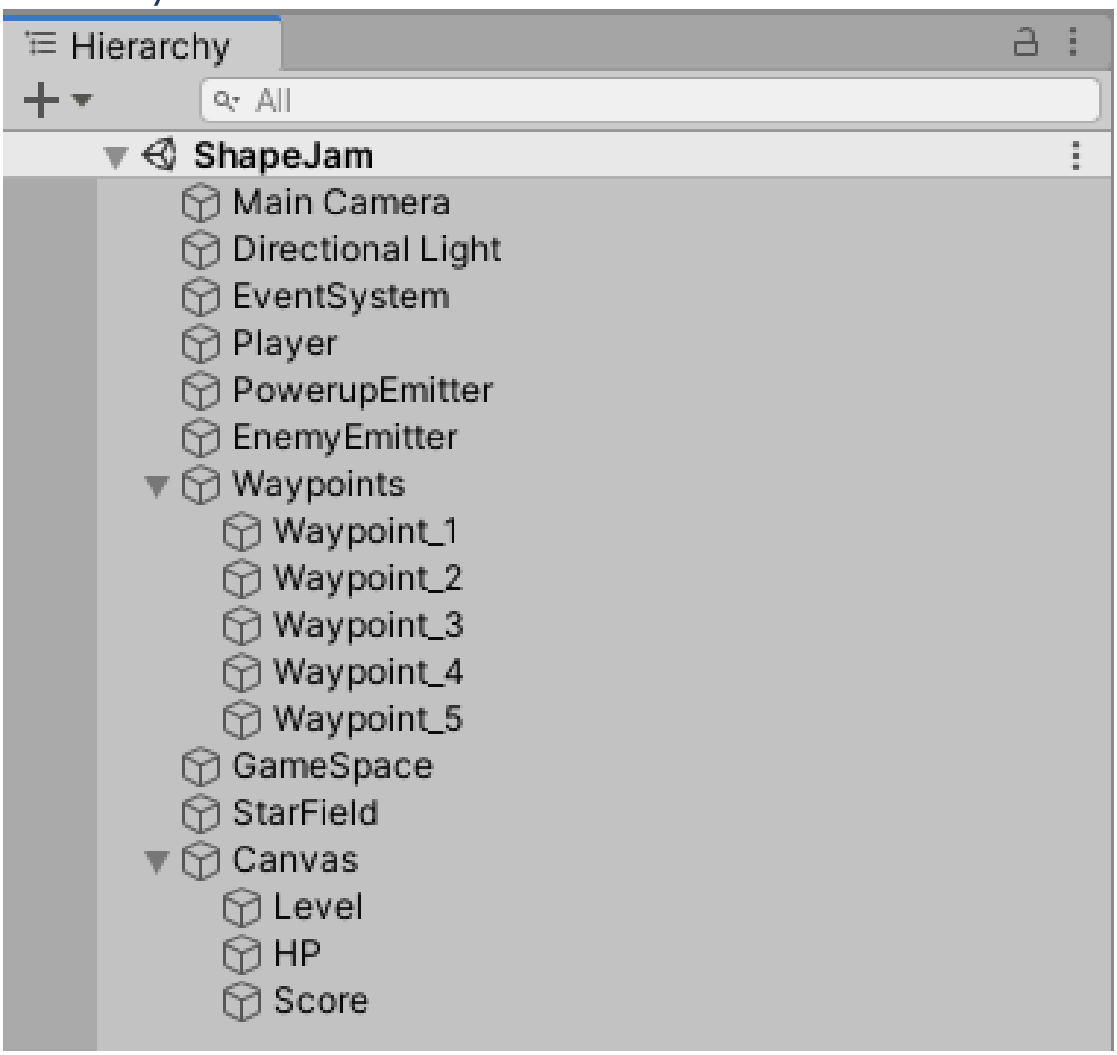
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PowerupController : MonoBehaviour
{
    private float nextDrop = 1.0f;
    public float dropRate;
    public GameObject powerup;
    public GameObject powerupEmitter;
    private GameObject player;
    private PlayerControls playerController;

    void Start()
    {
        powerupEmitter = GameObject.Find("PowerupEmitter");
        player = GameObject.Find("Player");
        playerController = player.GetComponent<PlayerControls>();
    }

    void Update()
    {
        if (playerController.gameOver)
        {
            return;
        }
        if (Time.time > nextDrop)
        {
            float randomX = Random.Range(-6.0f, 6.0f);
            Vector3 powerupPosition = new Vector3(randomX, 6.0f, 0.0f);
            transform.position = powerupPosition;
            nextDrop = Time.time + dropRate;

            /*****************************\
            |**** Add your code below ****|
            \*****************************/

            Instantiate(powerup, transform.position, transform.rotation);

            /*****************************\
            |**** Add your code above ****|
            \*****************************/
        }
    }
}
```

PowerupEmitter Object



Activity Solution: Shape Jam Prove Yourself

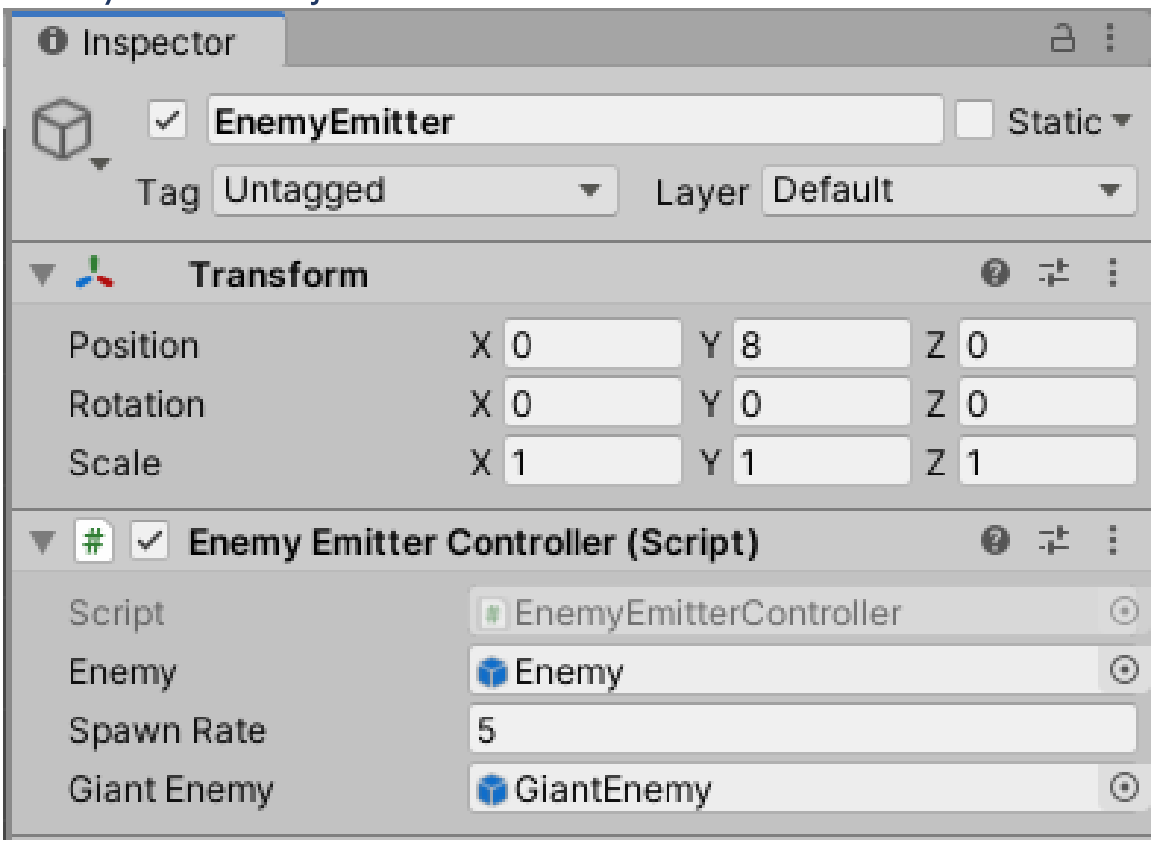
Hierarchy



EnemyEmitterController.cs Script

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

public class EnemyEmitterController : MonoBehaviour
{
    public GameObject Enemy;
    private PlayerControls playerController;
    public float spawnRate;
    private float nextSpawn = 5.0f;
    public GameObject GiantEnemy;

    void Start()
    {
        GameObject player = GameObject.Find("Player");
        playerController = player.GetComponent<PlayerControls>();

    }

    void Update()
    {
        if (playerController.gameOver)
        {
            return;
        }
        if (Time.time > nextSpawn)
        {
            nextSpawn = Time.time + spawnRate;
            for (int i = 0; i < playerController.currentLevel; i++)
            {
                float randomX = Random.Range(-6.0f, 6.0f);
                Vector3 enemyPosition = new Vector3(randomX, 6, 0);
                transform.position = enemyPosition;
                Instantiate(Enemy, transform.position, transform.rotation);
            }
            if (Random.value > 0.5)
            {
                float randomX = Random.Range(-6.0f, 6.0f);
                Vector3 enemyPosition = new Vector3(randomX, 6, 0);
                transform.position = enemyPosition;
                Instantiate(GiantEnemy, transform.position, transform.rotation);
            }
        }
    }
}
```

EnemyEmitter Object



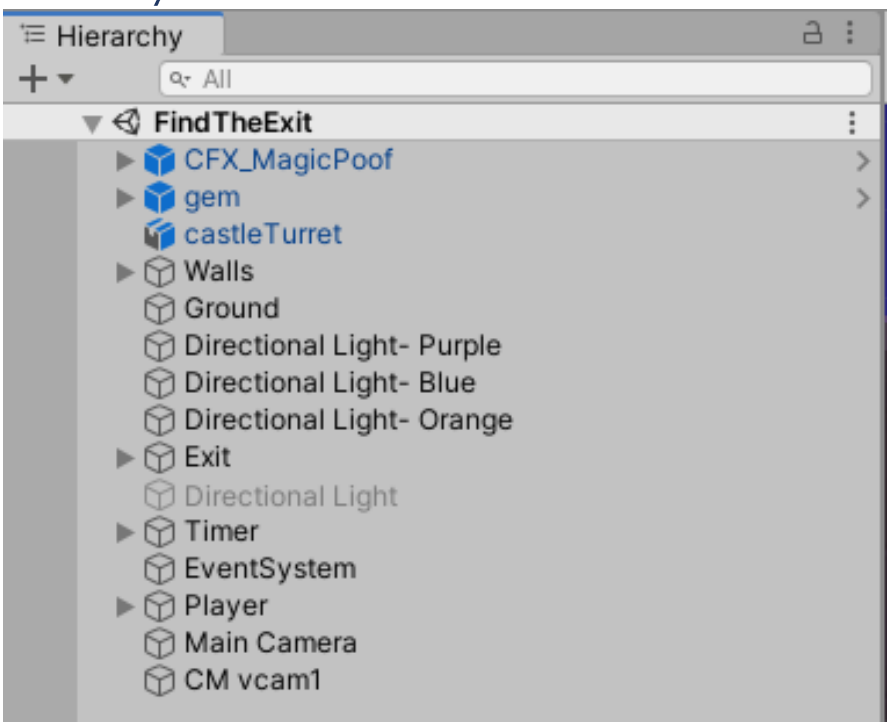
PlayerControls.cs Script

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

[System.Serializable]
public class Boundaries
{
    public float leftBorder, rightBorder, bottomBorder, topBorder;
}
public class PlayerControls : MonoBehaviour
{
    public float speed;
    public float projectileForce;
    public float fireRate;
    public int currentLevel;
    private float nextFire;
    public int playerHealth;
    public int score;
    public bool gameOver = false;
    private Vector2 moveDirection = Vector2.zero;
    public GameObject projectile;
    public Boundaries boundary;
    private Rigidbody myRigidBody;

    void Start()
    {
        currentLevel = 1;
        score = 0;
        myRigidBody = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        moveDirection = new Vector2(
              Input.GetAxis("Horizontal"), Input.GetAxis("Vertical"));
        if (!gameOver)
        {
            myRigidBody.velocity = moveDirection * speed;
        }
        myRigidBody.position = new Vector2(
            Mathf.Clamp(
              myRigidBody.position.x,
              boundary.leftBorder,
              boundary.rightBorder),
            Mathf.Clamp(myRigidBody.position.y,
              boundary.bottomBorder,
              boundary.topBorder)
        );
    }
```

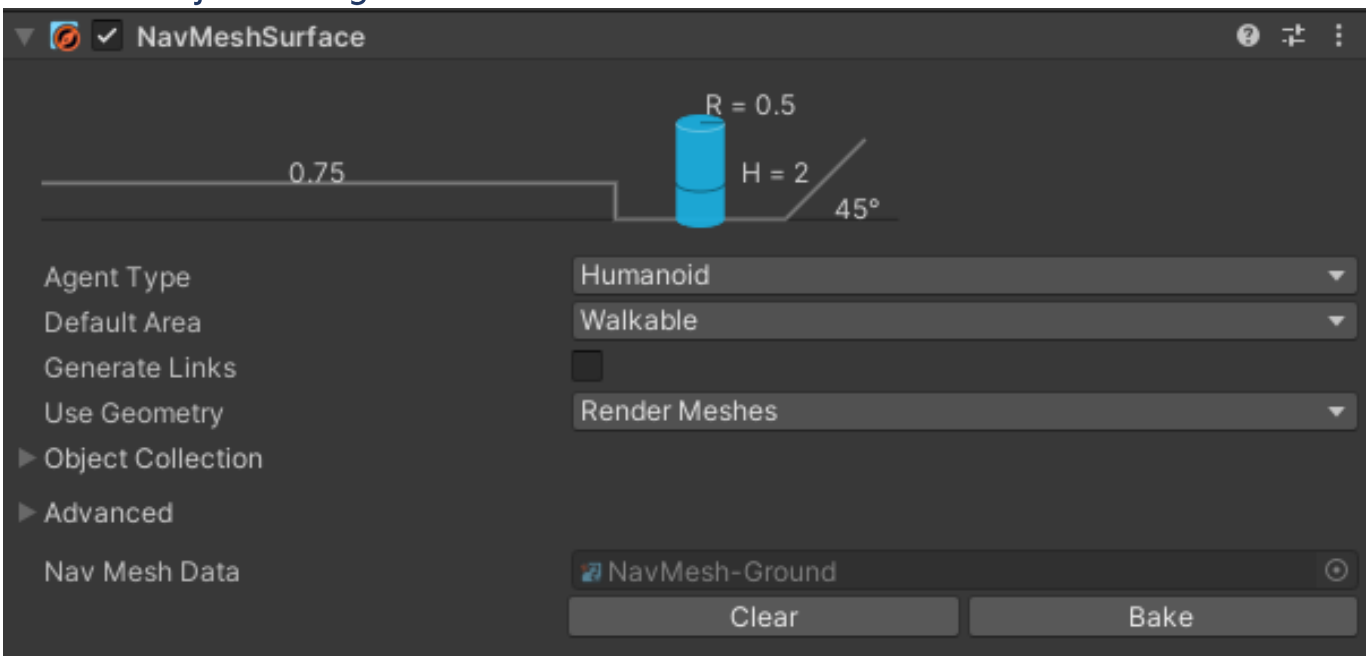
```
    void Update()
    {
        if (gameOver)
        {
            return;
        }
        if ((Input.GetKey("space") || Input.GetKey("z")) && (Time.time > nextFire))
        {
            nextFire = Time.time + fireRate;

            Instantiate(projectile, transform.position, transform.rotation);
            if (currentLevel >= 1)
            {
                Vector3 rightOffset = new Vector3(0.2f, -0.2f, 0);
                Vector3 leftOffset = new Vector3(-0.2f, -0.2f, 0);
                Instantiate(
                    projectile, transform.position + rightOffset, transform.rotation);
                Instantiate(
                    projectile, transform.position + leftOffset, transform.rotation);
            }
            if (currentLevel >= 3)
            {
                Vector3 rightOffset = new Vector3(0.2f, 0.2f, 0);
                Vector3 leftOffset = new Vector3(-0.2f, 0.2f, 0);
                Instantiate(
                    projectile, transform.position + rightOffset, transform.rotation);
                Instantiate(
                    projectile, transform.position + leftOffset, transform.rotation);
            }
            if (currentLevel >= 5)
            {
                Vector3 rightOffset = new Vector3(0.4f, 0.4f, 0);
                Vector3 leftOffset = new Vector3(-0.4f, 0.4f, 0);
                Instantiate(
                    projectile, transform.position + rightOffset, transform.rotation);
                Instantiate(
                    projectile, transform.position + leftOffset, transform.rotation);
            }
        }
        if (playerHealth <= 0)
        {
            gameOver = true;
        }
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "Hazard")
        {
            Debug.Log("Player hit!");
            Destroy(other.gameObject);
            playerHealth--;
        }
    }
}
```

Activity Solution: Labyrinth

Hierarchy



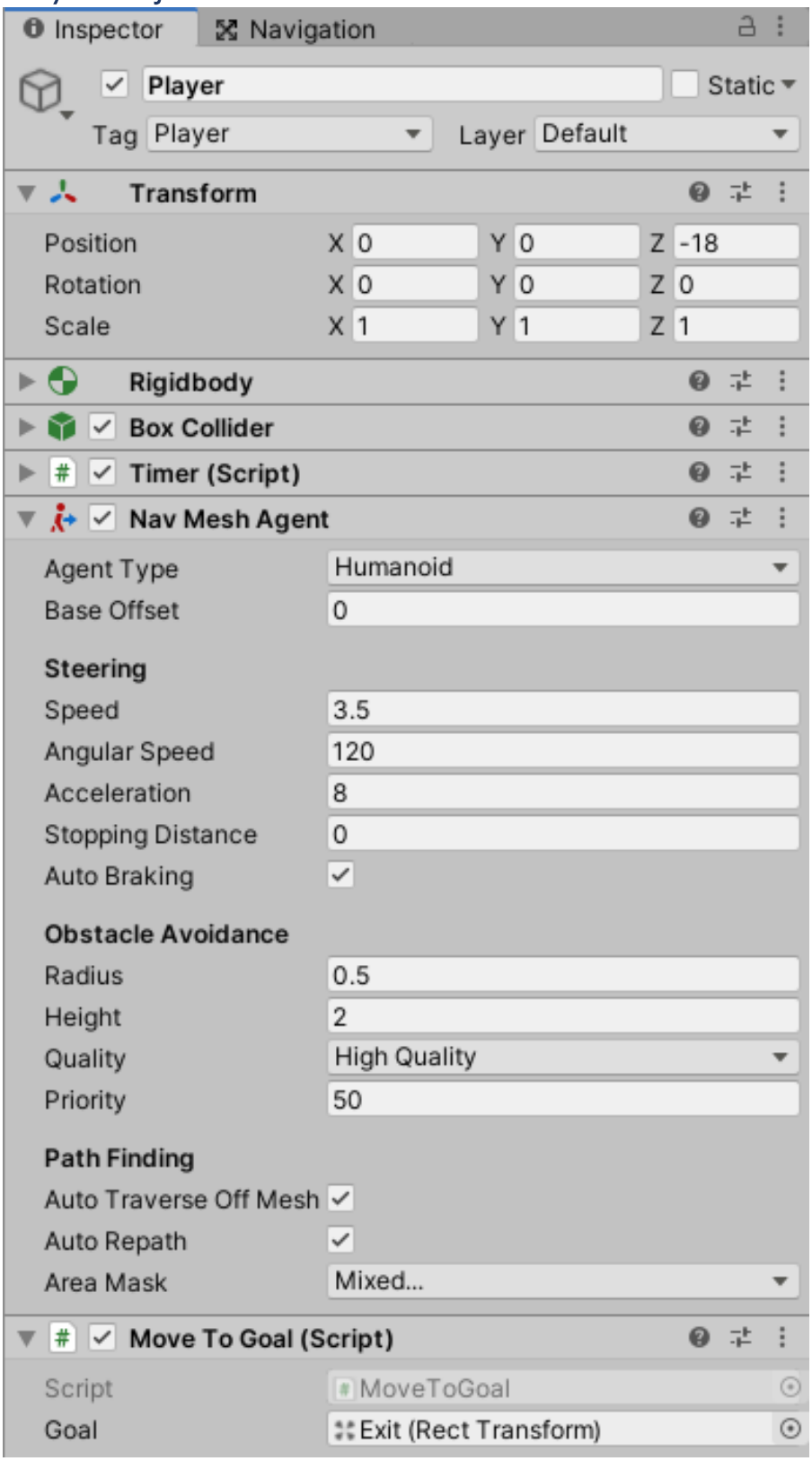
Ground Object Navigation



MoveToGoal.cs Script

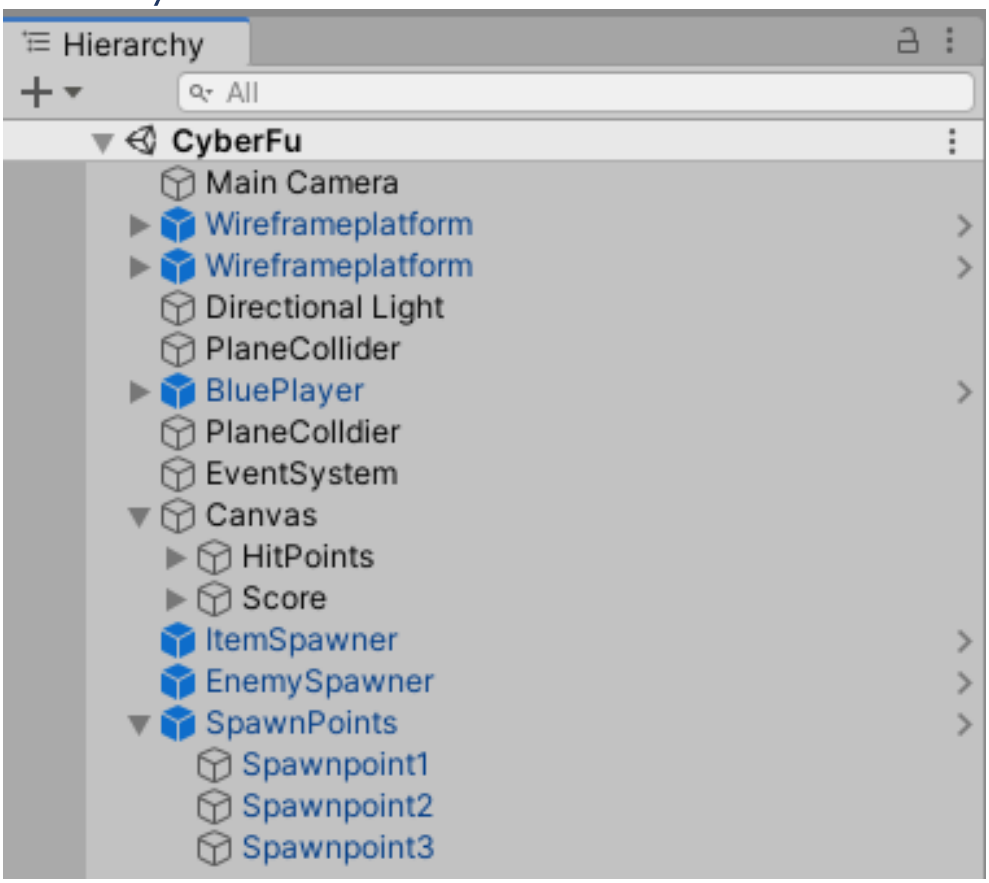
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

public class MoveToGoal : MonoBehaviour
{
    // goal is the destination for the NavMeshAgent,
    // animator is the Player object's Animator
    // agent contains the NavMeshAgent
    public Transform goal;
    Animator animator;
    NavMeshAgent agent;

    void Start()
    {
        // Animator and Agent initialized, Agent's destination set
        // goal.position is set in interface.
        animator = GetComponentInChildren<Animator>();
        agent = GetComponent<NavMeshAgent>();
        agent.destination = goal.position;
    }

    void Update()
    {
        if (agent.hasPath)
        {
            Animator.SetBool("isRunning", true);
        }
        else
        {
            Animator.SetBool("isRunning", false);
        }
    }
}
```

Player Object

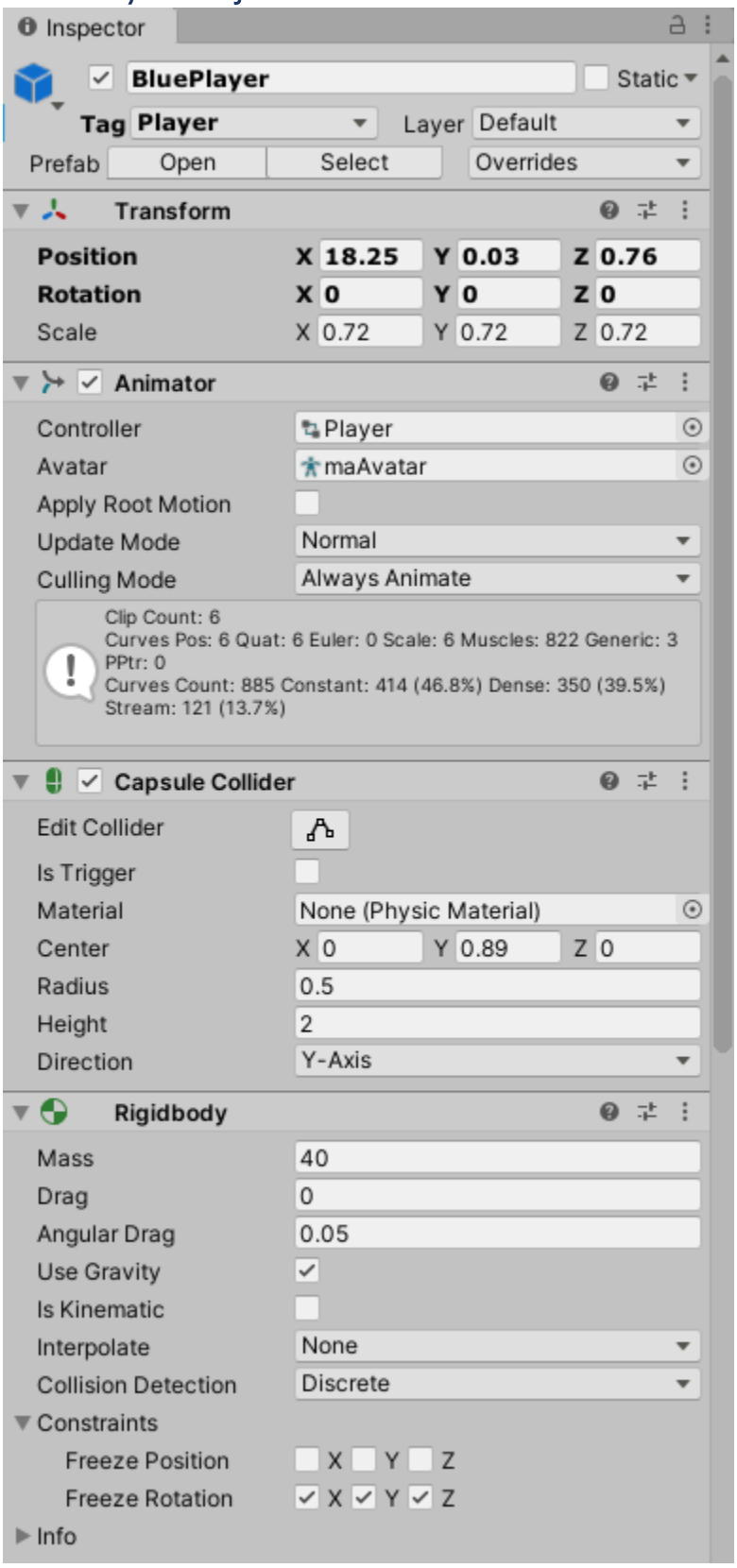


Activity Solution: CyberFu Part 2

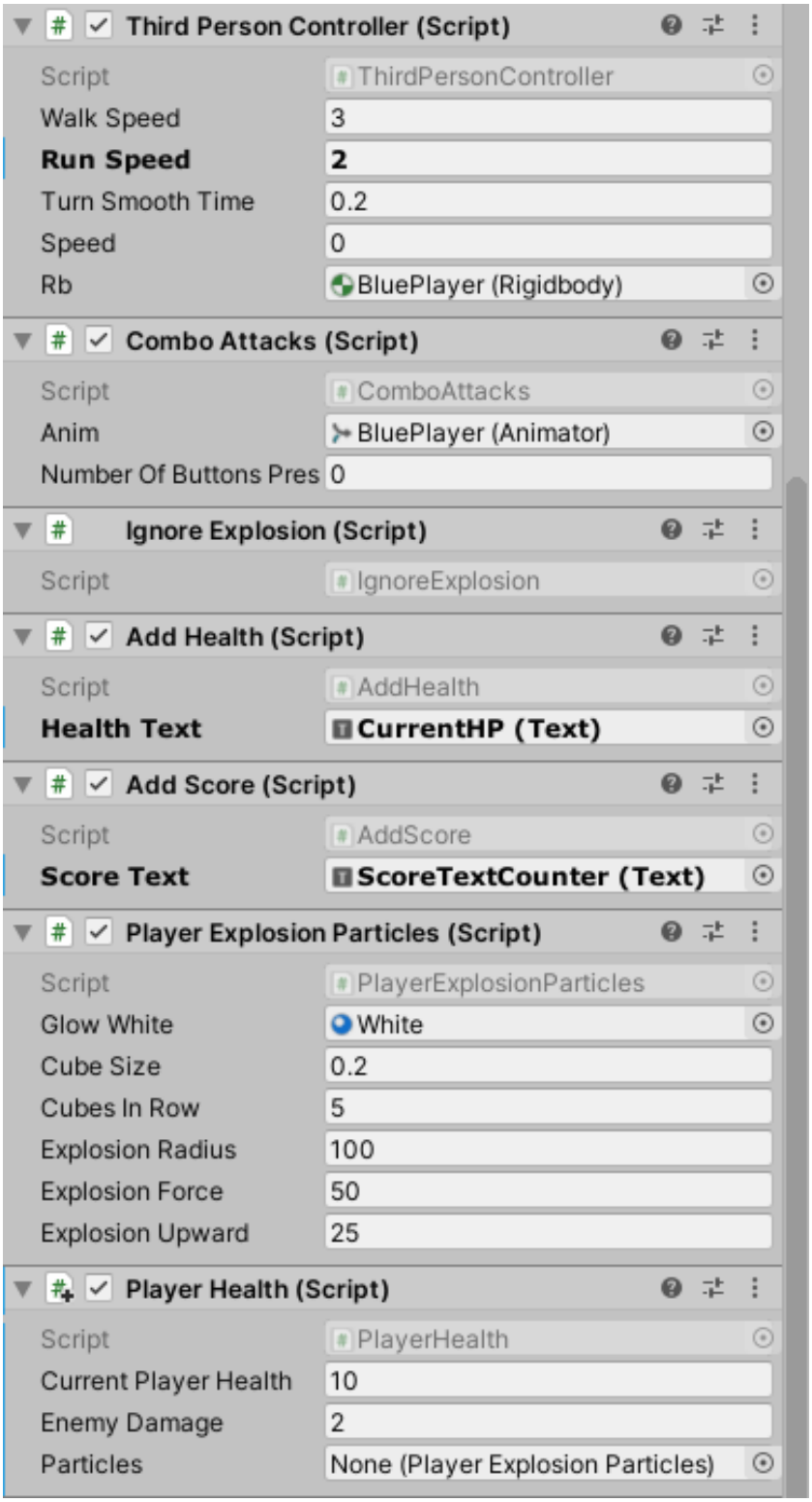
Hierarchy



BluePlayer Object



BluePlayer Object Continued

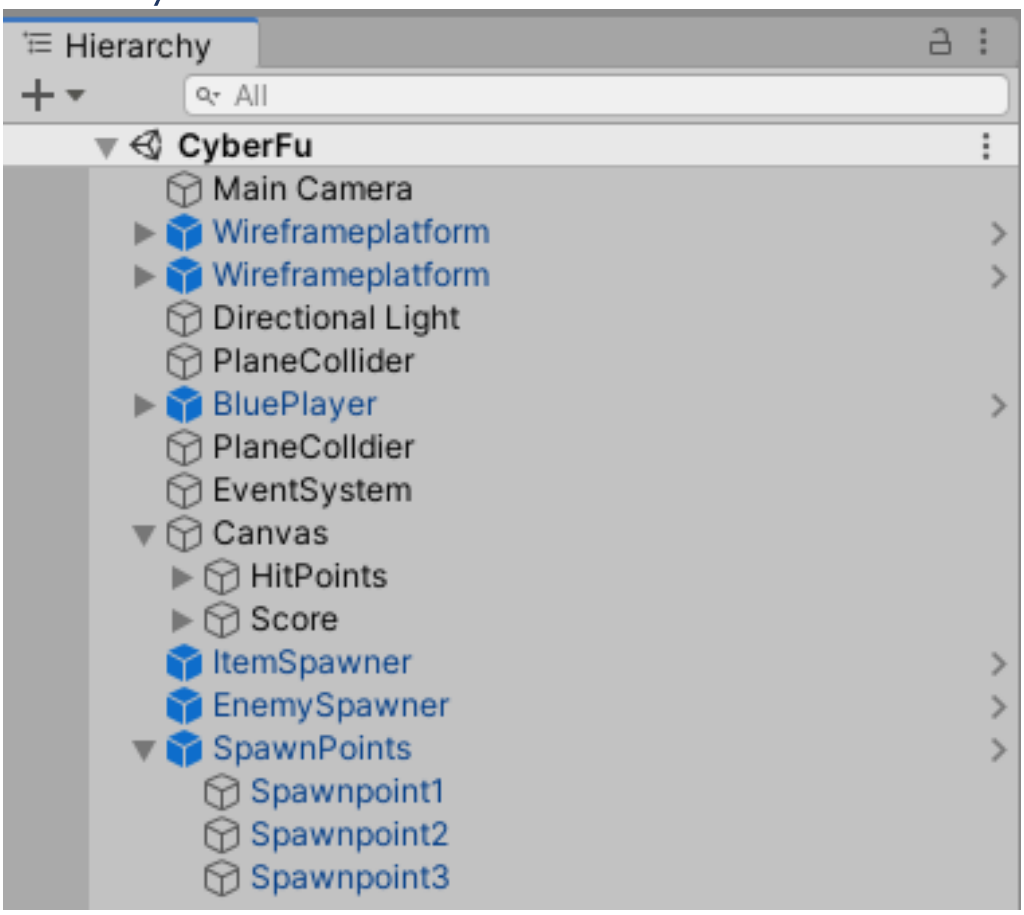


PlayerHealth.cs Script

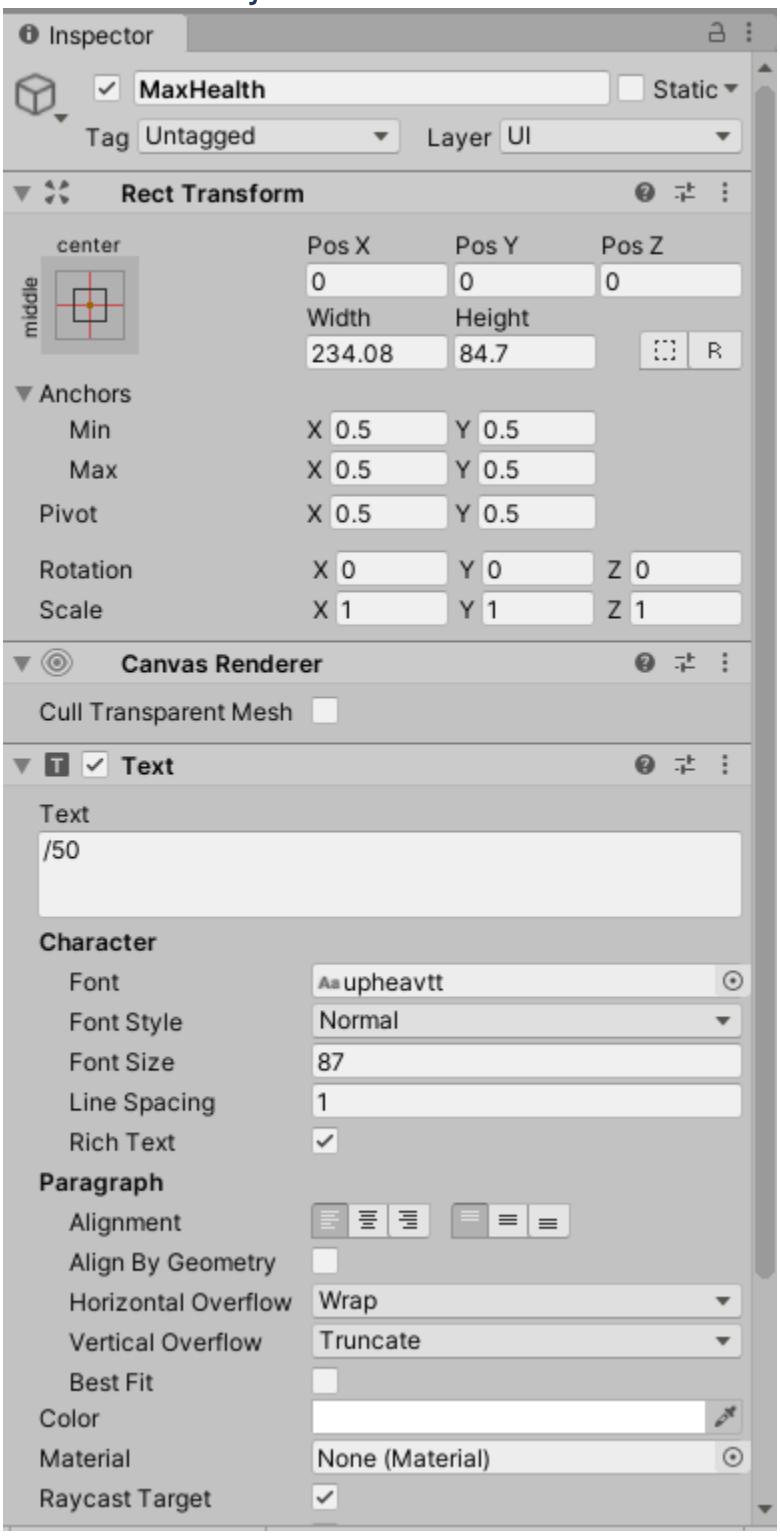
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class PlayerHealth : MonoBehaviour
{
    public int currentPlayerHealth = 10;
    public int enemyDamage = 2;
    public PlayerExplosionParticles particles;
    private Animator playerAnimator;

    void Start()
    {
        playerAnimator = GetComponent<Animator>();
        particles = GetComponent<PlayerExplosionParticles>();
    }

    public void HurtPlayer()
    {
        currentPlayerHealth -= enemyDamage;
        playerAnimator.SetTrigger("Hit");
        if (currentPlayerHealth <= 0)
        {
            particles.Explode();
            Invoke("ReloadScene", 5);
        }
    }

    private void ReloadScene()
    {
        SceneManager.LoadScene("CyberFu");
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.tag == "HitCollider")
        {
            HurtPlayer();
        }
    }
}
```

Activity Solution: CyberFu Part 2 Prove Yourself

Hierarchy



MaxHealth Object

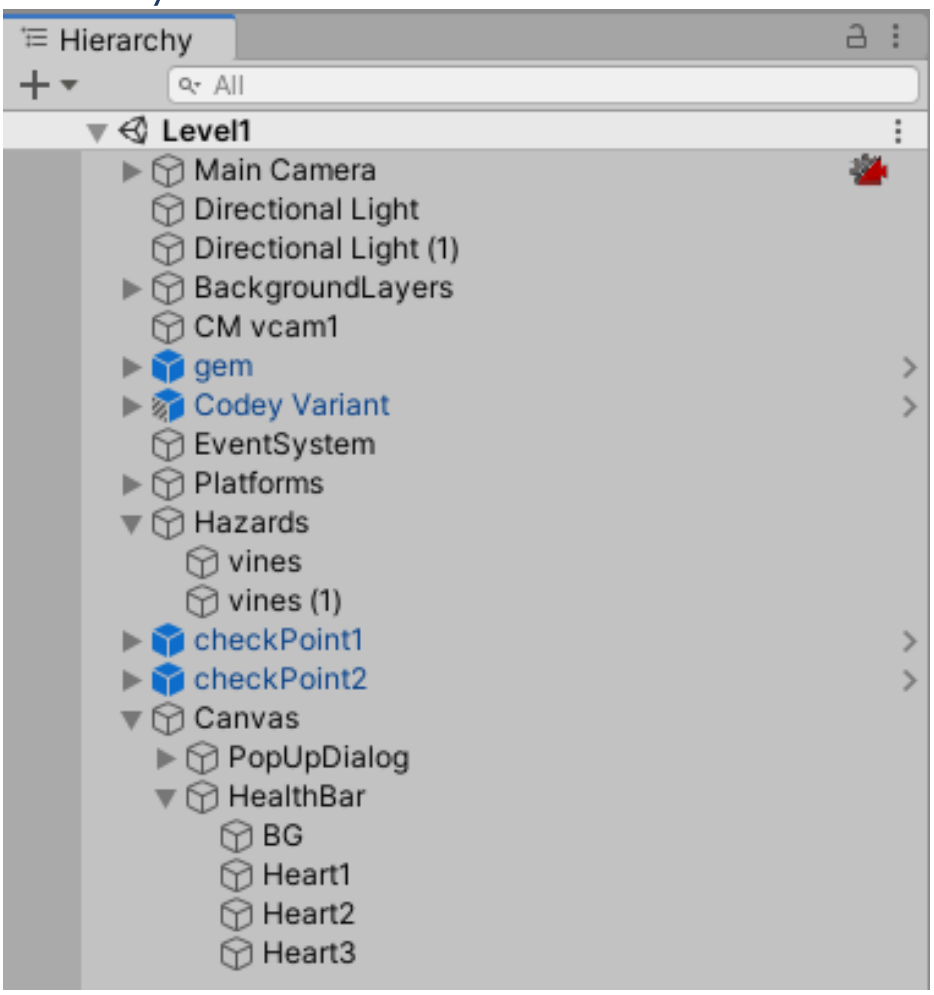


PlayerHealth.cs Script

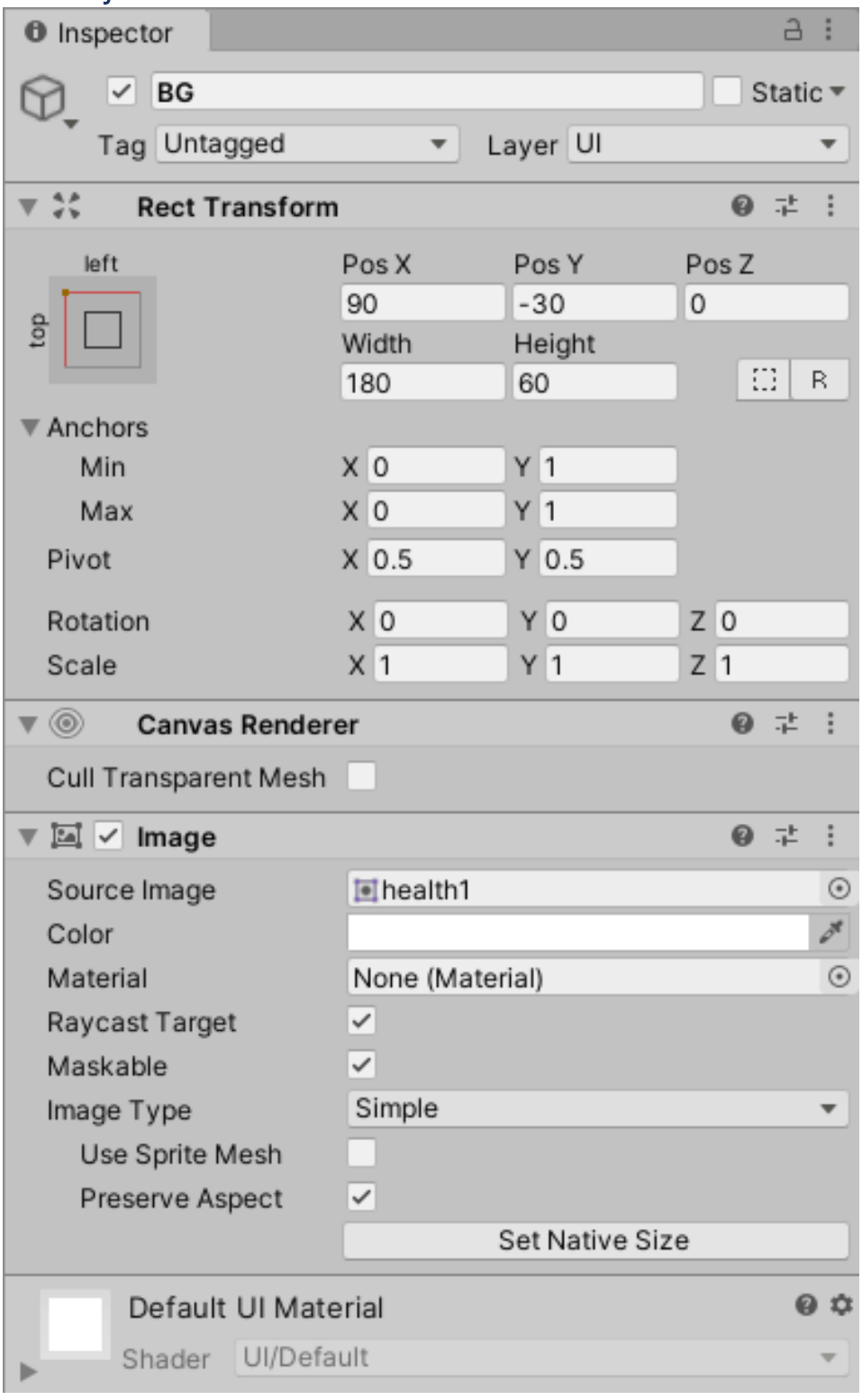
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class PlayerHealth : MonoBehaviour
{
    public int currentPlayerHealth = 50;
    public int enemyDamage = 5;
    public PlayerExplosionParticles particles;
    private Animator playerAnimator;

    void Start()
    {
        playerAnimator = GetComponent<Animator>();
        particles = GetComponent<PlayerExplosionParticles>();
    }

    public void HurtPlayer()
    {
        currentPlayerHealth -= enemyDamage;
        playerAnimator.SetTrigger("Hit");
        if (currentPlayerHealth <= 0)
        {
            particles.Explode();
            Invoke("ReloadScene", 5);
        }
    }

    private void ReloadScene()
    {
        SceneManager.LoadScene("CyberFu");
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("HitCollider"))
        {
            HurtPlayer();
        }
    }
}
```

Activity Solution: Amazing Ninja Worlds Part 1

Hierarchy



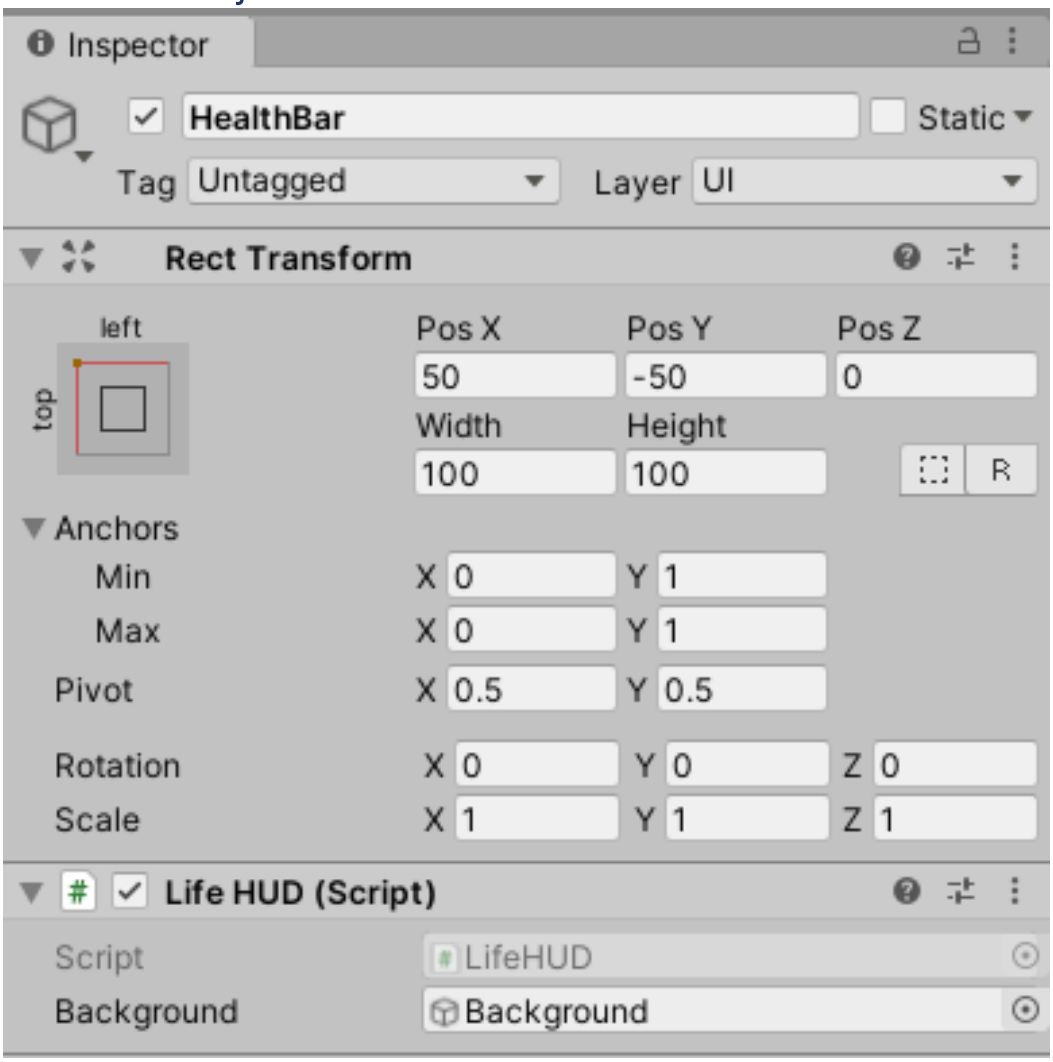
BG Object



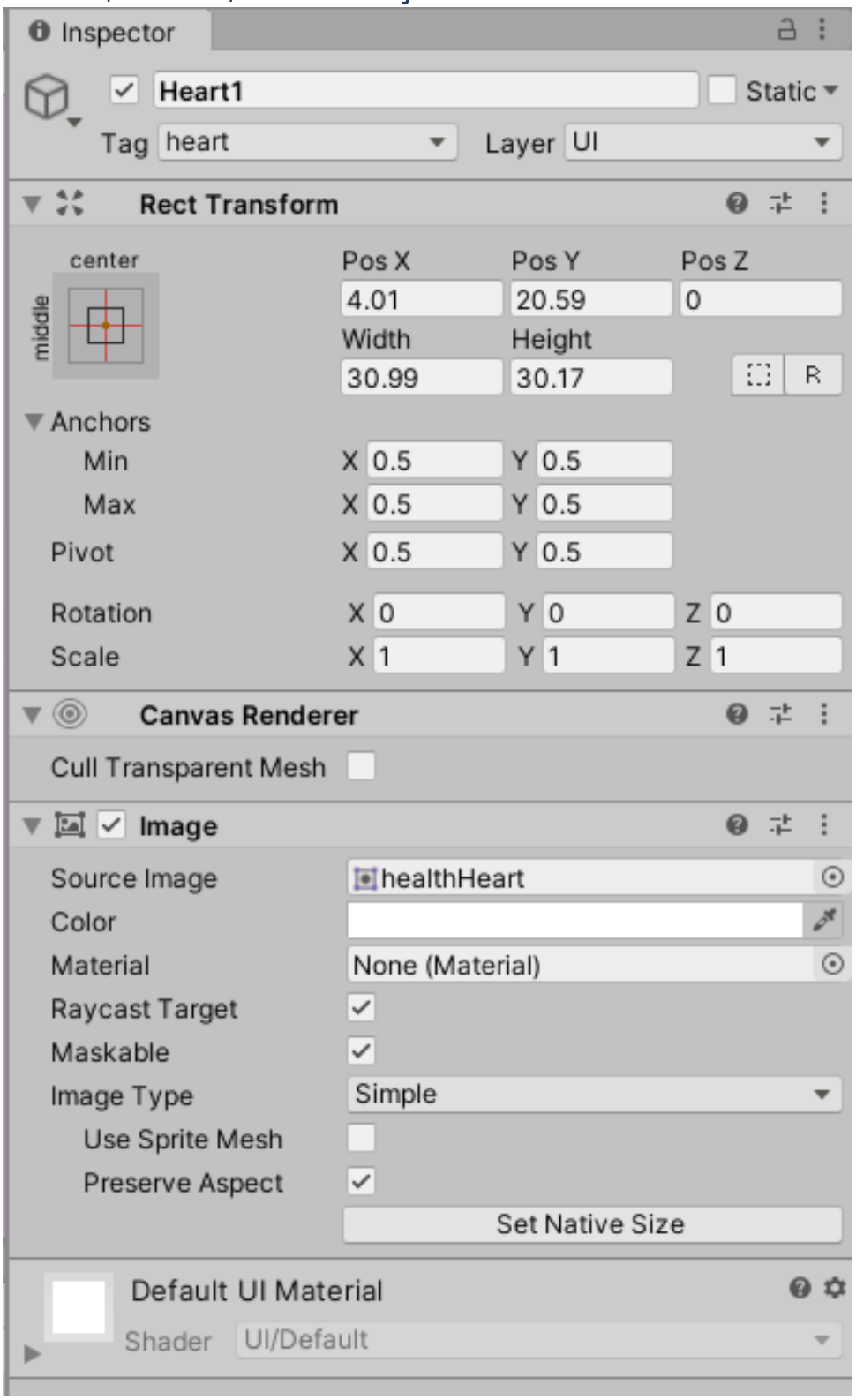
Hazard.cs Script

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

public class Hazard : MonoBehaviour
{
    public GameObject background;
    public GameObject HealthBar;

    private void OnTriggerEnter(Collider other)
    {
        background.GetComponent<GameManager>().moveToCheckPoint();
        HealthBar.GetComponent<LifeHUD>().HurtPlayer();
    }
}
```

HealthBar Object



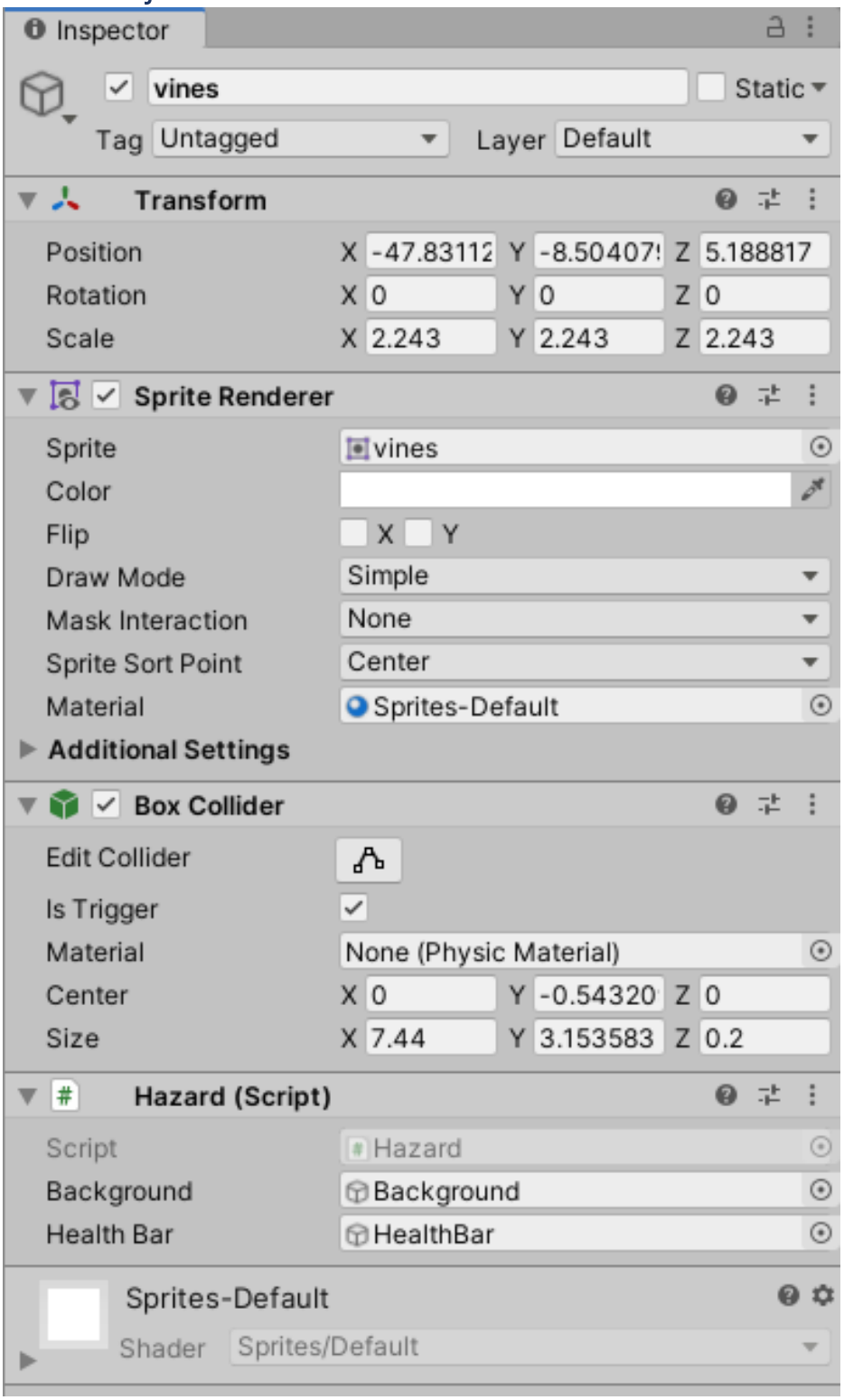
Heart1, Heart2, Heart3 Objects



LifeHUD.cs Script

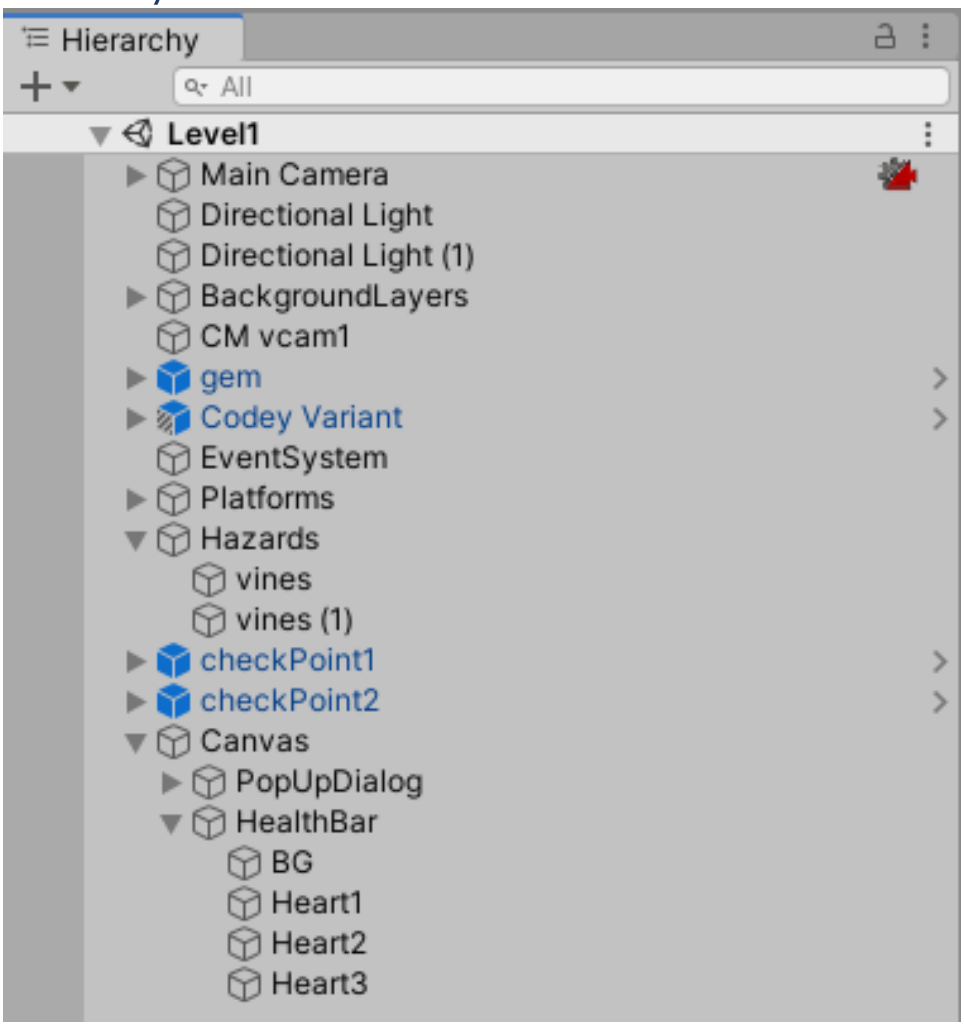
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LifeHUD : MonoBehaviour
{
    private GameObject[] hearts;
    private int lives = 3;
    public GameObject background;

    void Awake()
    {
        hearts = GameObject.FindGameObjectsWithTag("heart");
    }

    public void HurtPlayer()
    {
        Debug.Log("Ouch!");
        lives -= 1;
        hearts[lives].SetActive(false);
        if (lives == 0)
        {
            background.GetComponent<GameManager>().GameOver();
        }
    }
}
```

Vines Object

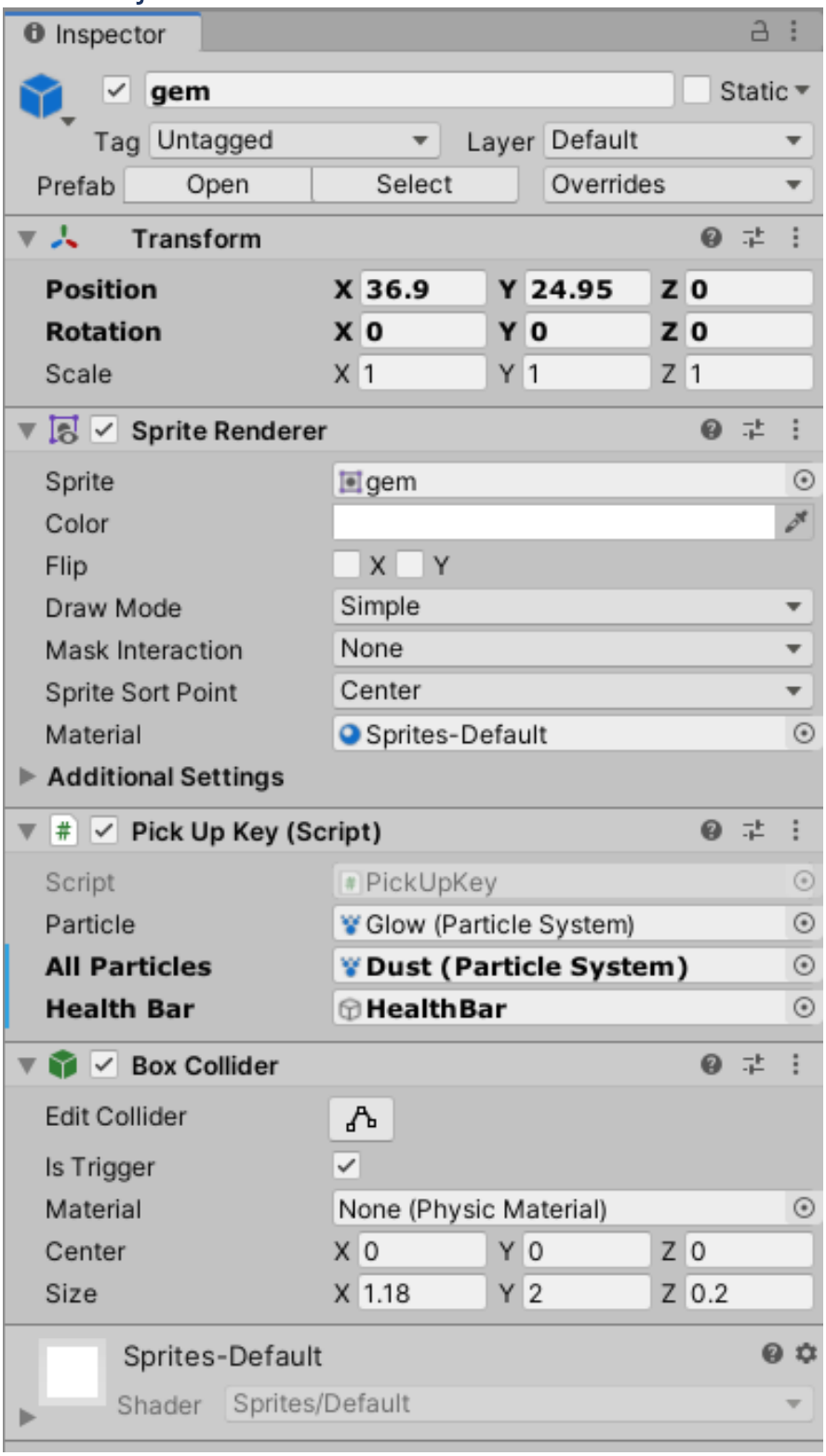


Activity Solution: Amazing Ninja Worlds Part 1 Prove Yourself

Hierarchy



Gem Object



LifeHUD.cs Script

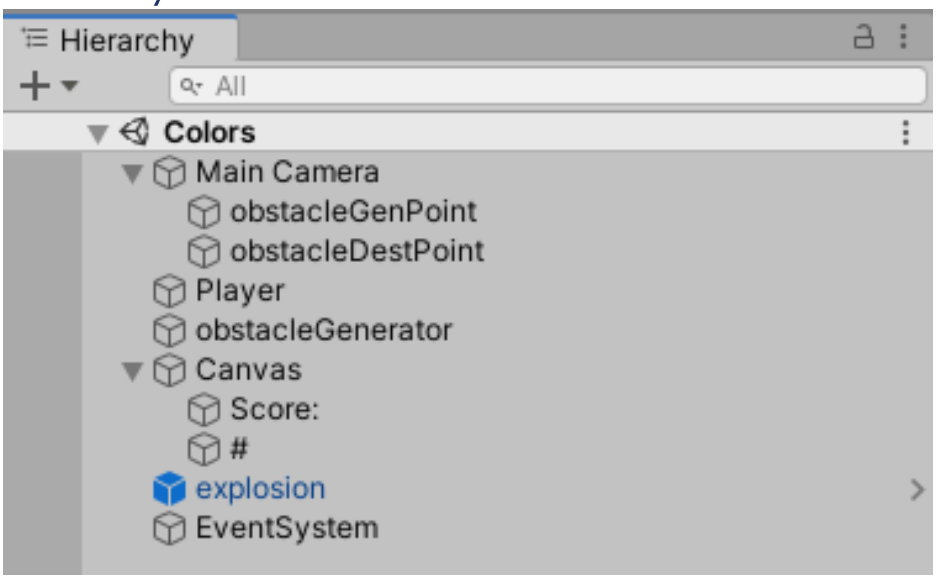
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LifeHUD : MonoBehaviour
{
    private GameObject[] hearts;
    private int lives = 3;
    public GameObject background;

    void Awake()
    {
        hearts = GameObject.FindGameObjectsWithTag("heart");
    }

    public void HurtPlayer()
    {
        Debug.Log("Ouch!");
        lives -= 1;
        hearts[lives].SetActive(false);
        if (lives == 0)
        {
            background.GetComponent<GameManager>().GameOver();
        }
    }

    public void HealPlayer()
    {
        Debug.Log("Yay!");
        if (lives < 3)
        {
            hearts[lives].SetActive(true);
            lives += 1;
        }
    }
}
```

PickUpKey.cs Script

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

public class PickUpKey : MonoBehaviour
{
    SpriteRenderer m_renderer;
    public ParticleSystem m_particle;
    public ParticleSystem allParticles;
    public GameObject HealthBar;

    private void Start()
    {
        m_renderer = GetComponentInParent<SpriteRenderer>();
    }

    private void OnTriggerEnter(Collider other)
    {
        m_renderer.enabled = false;
        m_particle.Stop();
        allParticles.Play();
        HealthBar.GetComponent<LifeHUD>().HealPlayer();
    }
}
```

World of Color

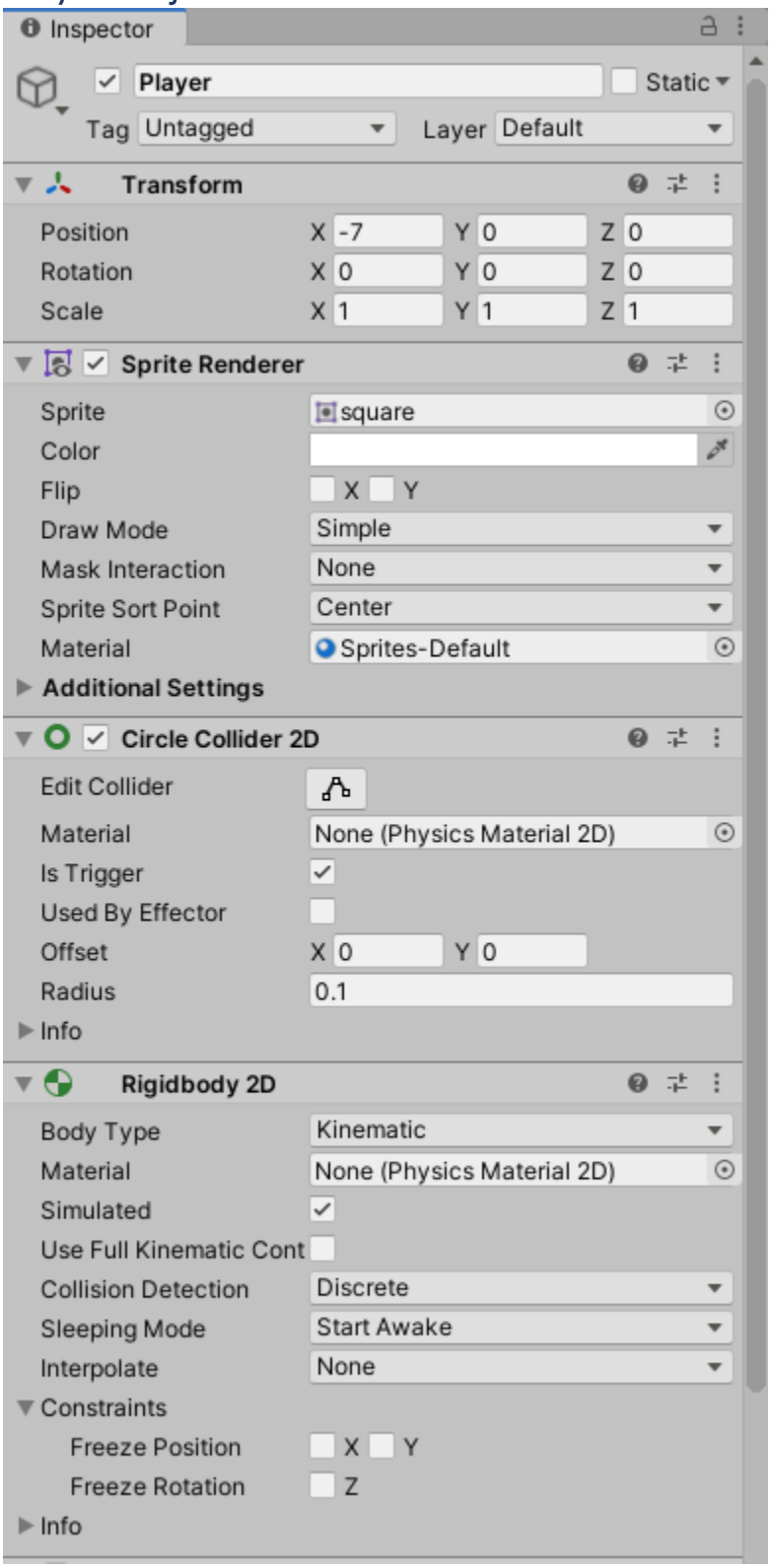
Hierarchy



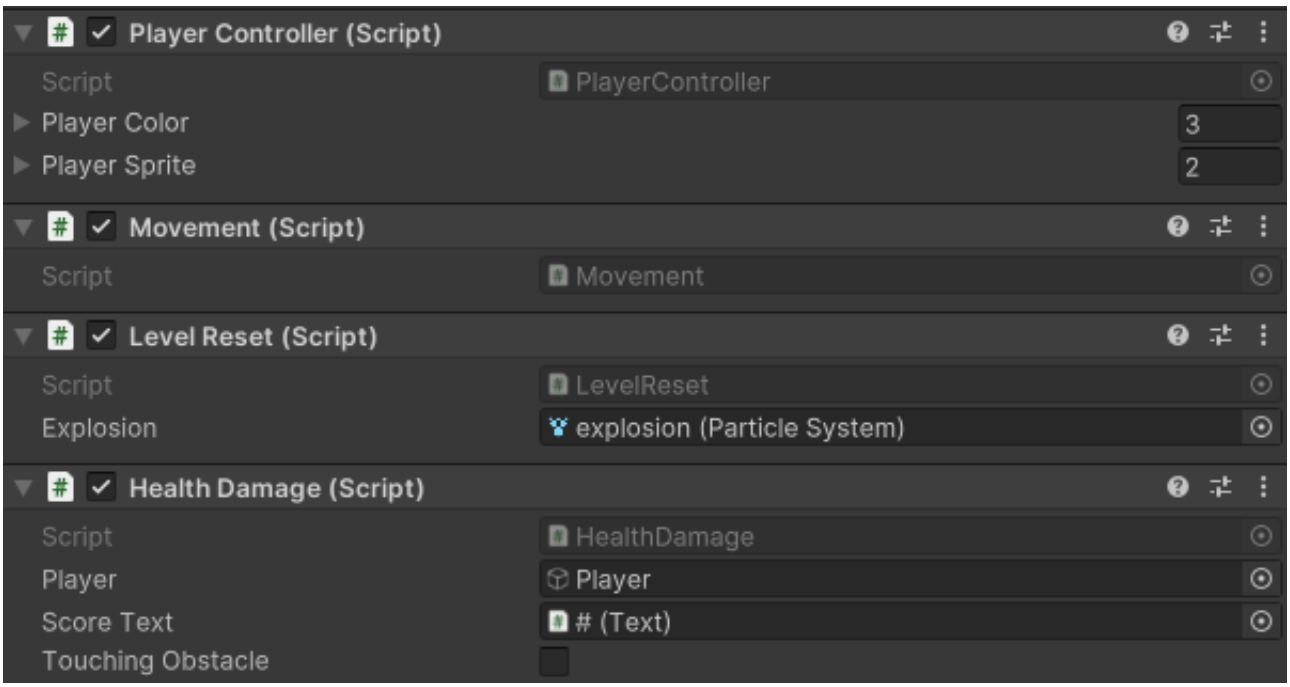
LevelReset.cs Script

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

public class LevelReset : MonoBehaviour
{
    ParticleSystem explosion;

    private void Start()
    {
        explosion.Stop();
    }

    public void GameOver()
    {
        gameObject.SetActive(false);
        Invoke("Reload", 2);
        explosion.transform.position = transform.position;
        explosion.Play();
    }

    void Reload()
    {
        SceneManager.LoadScene(0);
    }
}
```

Player Object

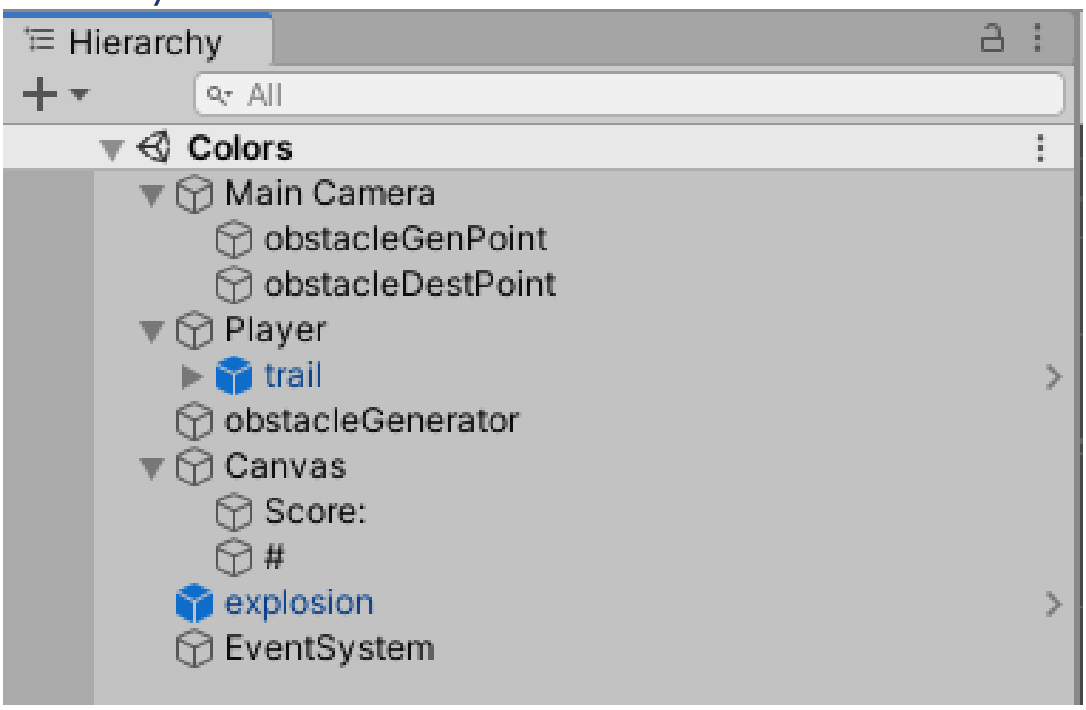


Player Object Continued

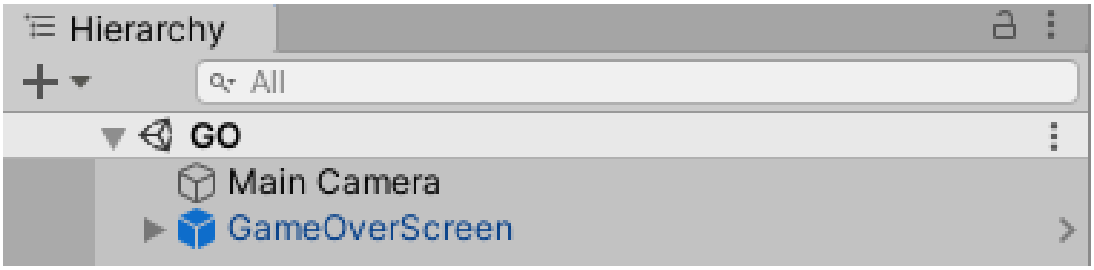


Activity Solution: World of Color Prove Yourself

Hierarchy



Game Over Scene

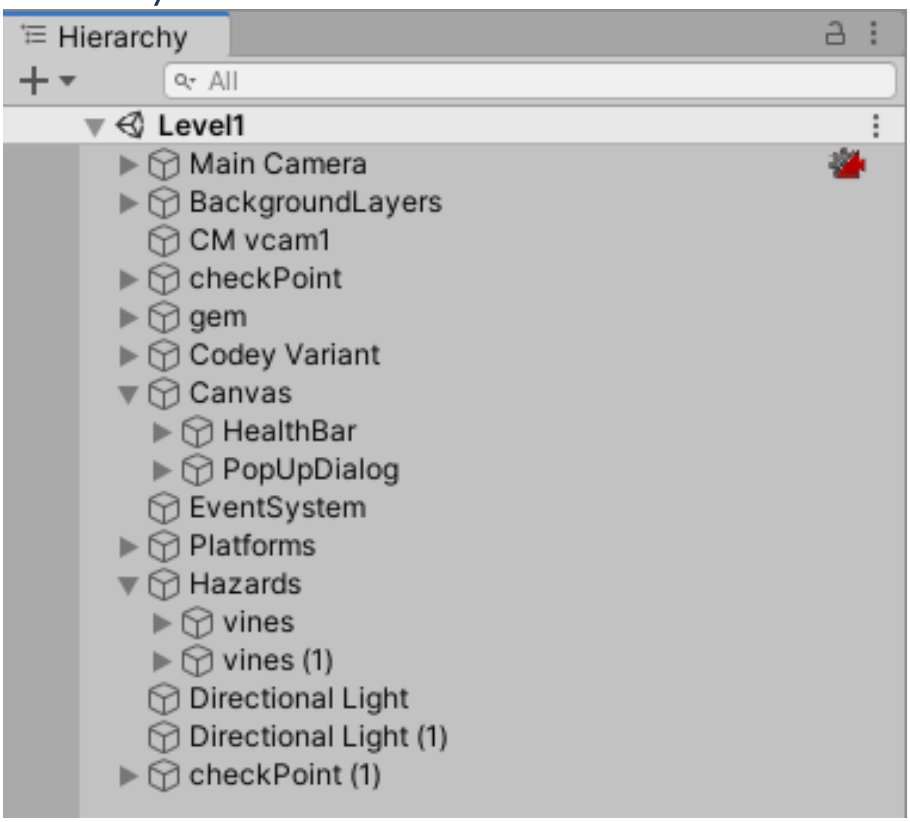


LevelReset.cs Script

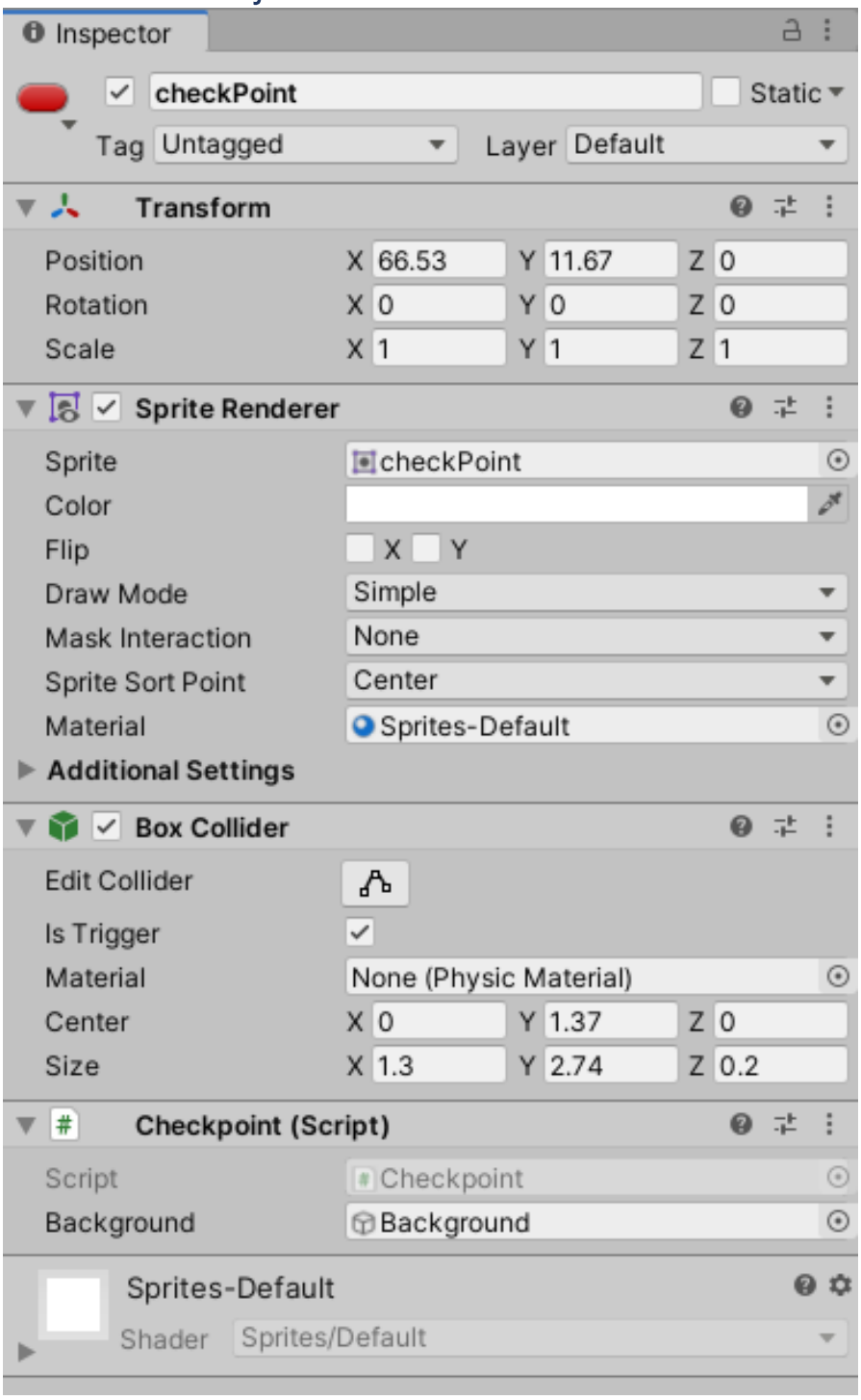
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class LevelReset : MonoBehaviour
{
    ParticleSystem explosion;

    private void Start()
    {
        explosion.Stop();
    }

    public void GameOver()
    {
        gameObject.SetActive(false);
        Invoke("Reload", 2);
        explosion.transform.position = transform.position;
        explosion.Play();
    }

    void Reload()
    {
        SceneManager.LoadScene("GO");
    }
}
```

Activity Solution: Amazing Ninja Worlds Part 2

Hierarchy



CheckPoint Object



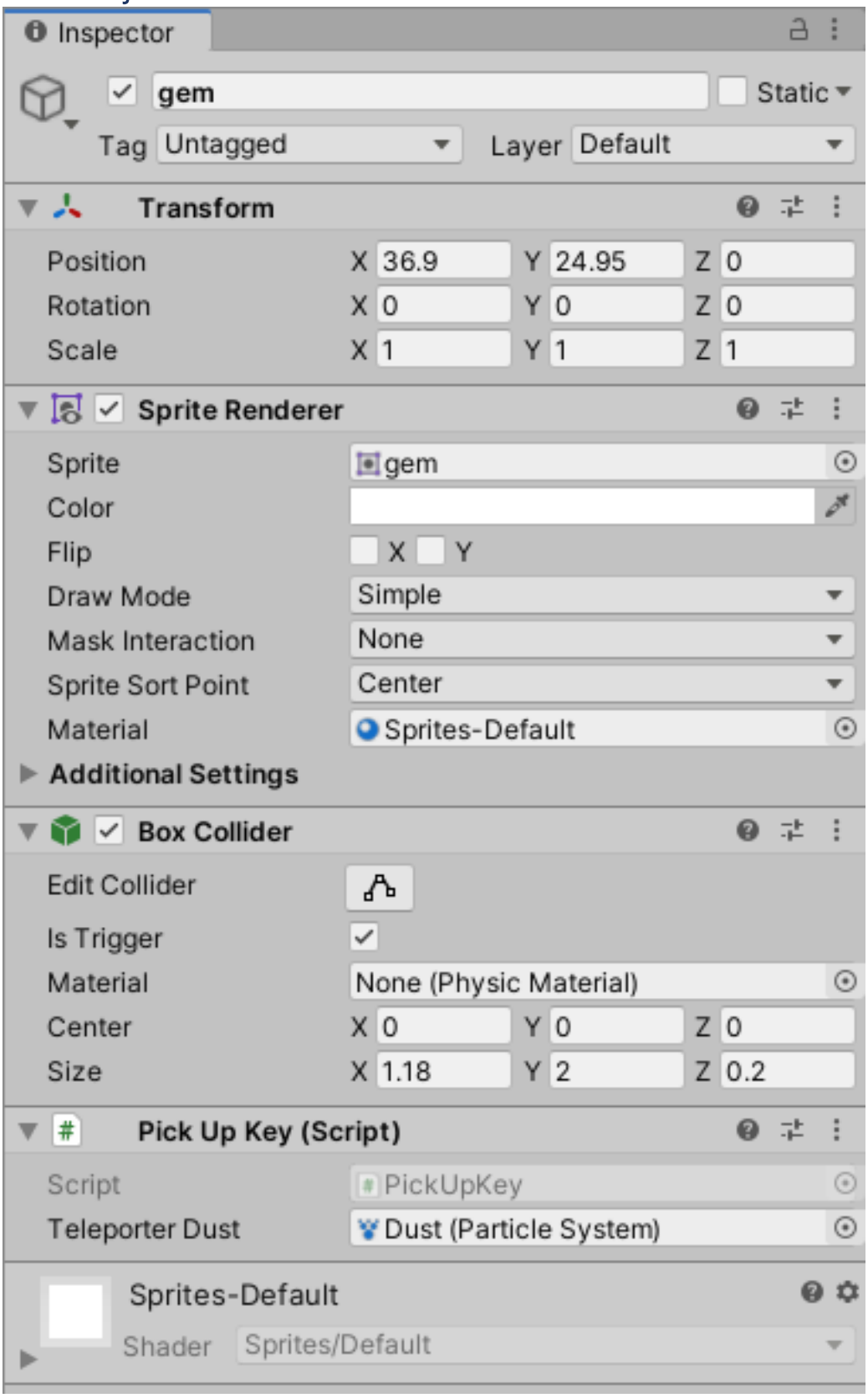
Checkpoint.cs Script

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

public class Checkpoint : MonoBehaviour
{
    public GameObject background;

    private void OnTriggerEnter(Collider other)
    {
        Vector3 newCheckpoint = transform.position;
        background.GetComponent<GameManager>().checkPoint = newCheckpoint;
    }
}
```

Exit.cs Script

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

public class Exit : MonoBehaviour
{
    public GameObject gem;
    public GameObject background;
    public string teleportDestination;

    private void OnTriggerEnter(Collider other)
    {
        if (gem.activeInHierarchy == false)
        {
            background.GetComponent<GameManager>().TeleportOpen(teleportDestination);
        }
    }
}
```

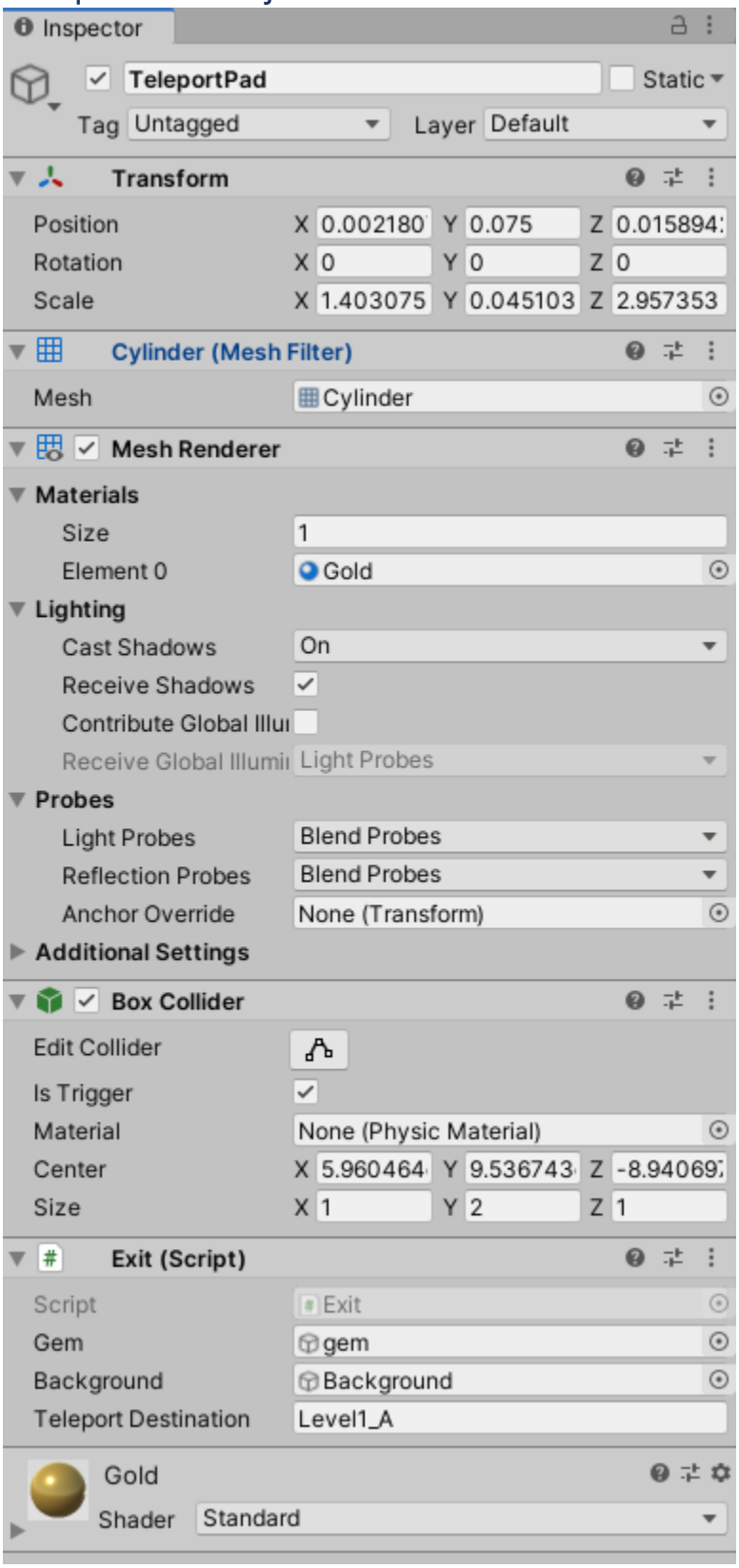
Gem Object



PickUpKey.cs Script

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

public class PickUpKey : MonoBehaviour
{
    public ParticleSystem teleporterDust;
    private void OnTriggerEnter(Collider other)
    {
        gameObject.SetActive(false);
        teleporterDust.Play();
    }
}
```

TeleportPad Object

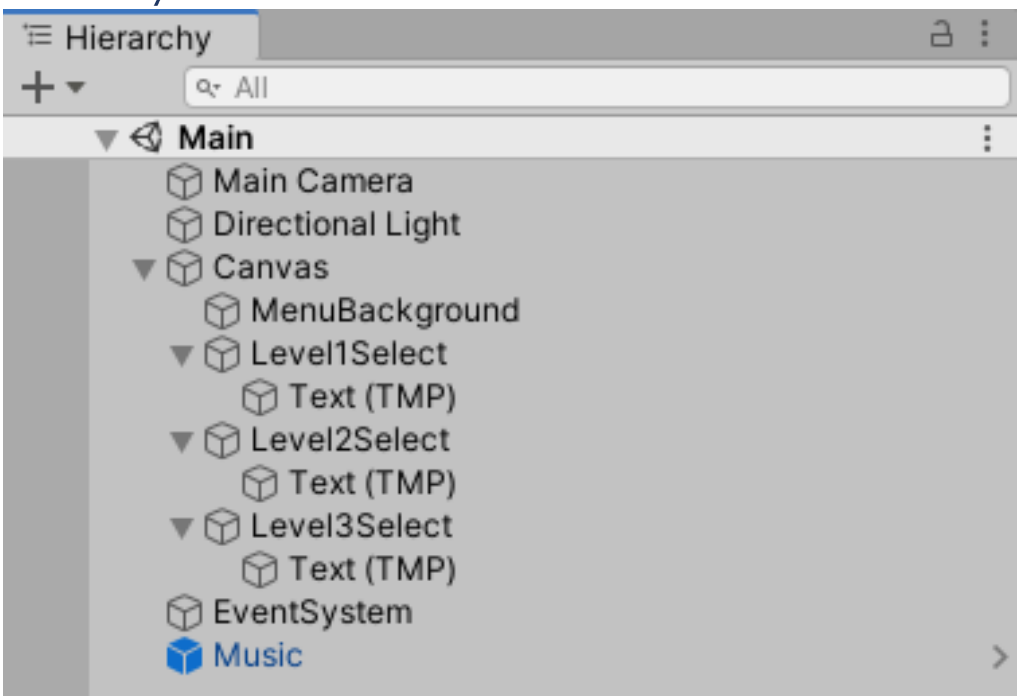


Activity Solution: Amazing Ninja Worlds Part 2 Prove Yourself

There is no provided solution for this activity. Each ninja must use what they learned to create a custom Level1_B.

Activity Solution: Amazing Ninja Worlds Part 3

Hierarchy

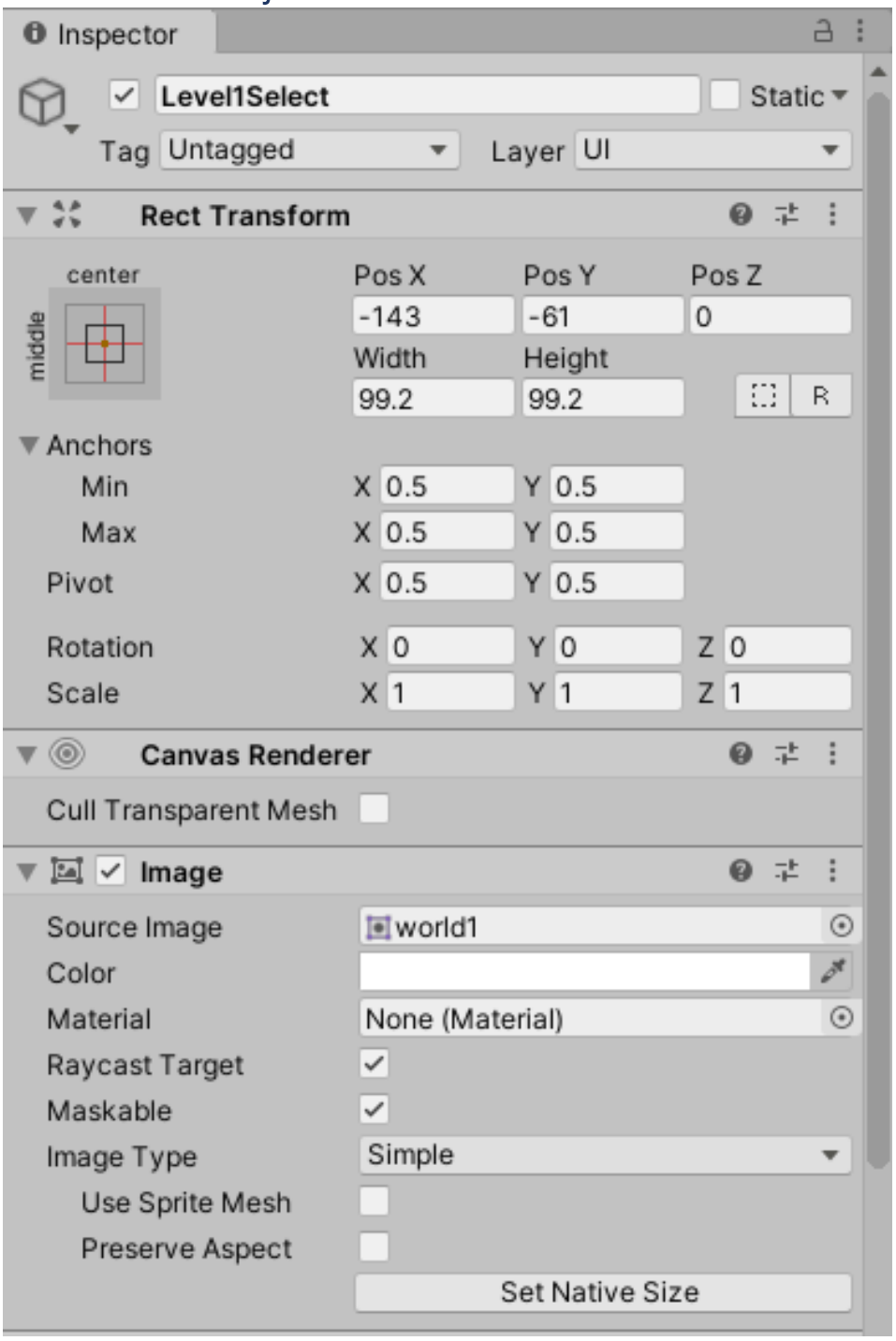


GameManager.cs Script

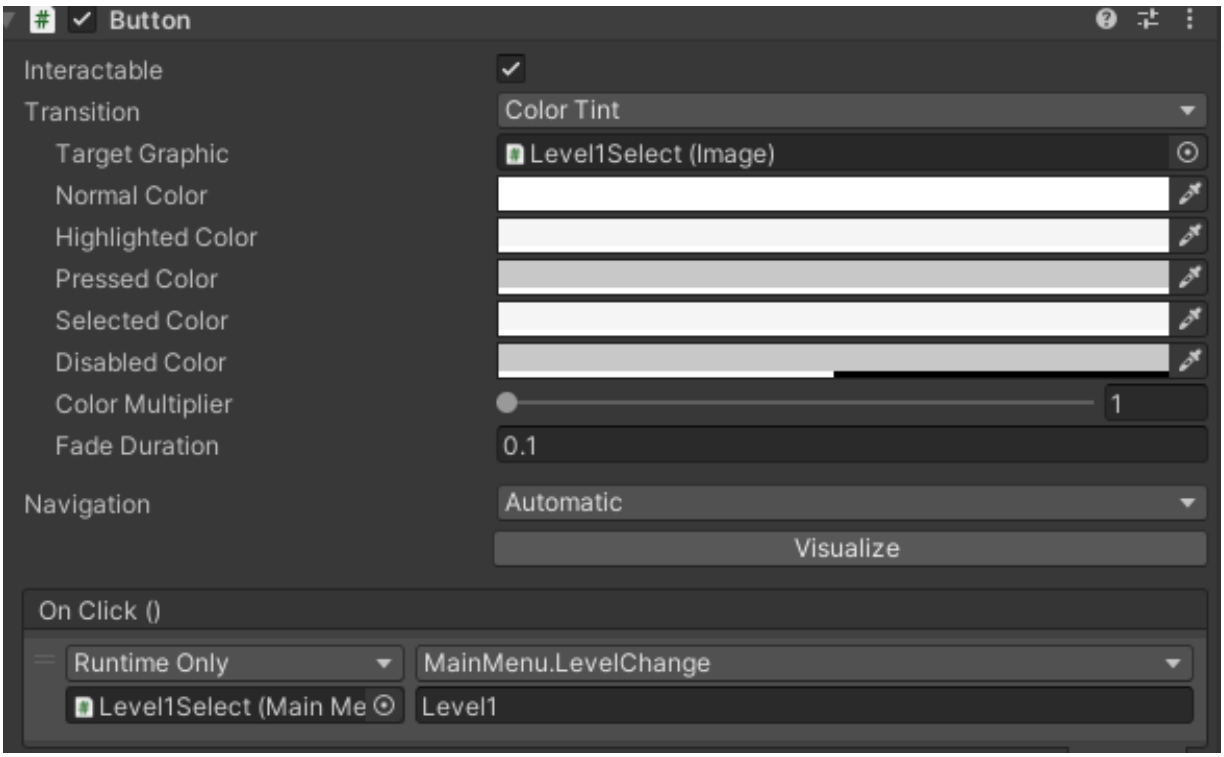
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class GameManager : MonoBehaviour
{
    public GameObject popUp;
    public TextMeshProUGUI textMessage;
    public GameObject player;
    public Vector3 checkPoint = new Vector3(-4.7f, 0.6f, 0);
    public GameObject retryButton;
    public TextMeshProUGUI buttonText;
    private AudioSource buttonSound;
    public AudioClip ButtonClick;
    public AudioClip Win;

    void Start()
    {
        popUp.SetActive(false);
        buttonSound = GetComponent<AudioSource>();
    }
    private void Update()
    {
        if (Input.GetButton("Cancel"))
        {
            SceneManager.LoadScene(0);
        }
    }
    public void GameOver()
    {
        textMessage.text = "Keep Trying!";
        retryButton.GetComponent<Button>().onClick.AddListener(
                    delegate { Reset("Level1"); }
                    );
        buttonText.text = "Click to try again";
        popUp.SetActive(true);
        PlayerPrefs.DeleteKey("LIVES_LEFT");
    }
    public void TeleportOpen(string nextScene)
    {
        textMessage.text = "Good Job!";
        retryButton.GetComponent<Button>().onClick.AddListener(
                    delegate { Reset(nextScene); }
                    );
        buttonText.text = "Click to continue";
        popUp.SetActive(true);
        buttonSound.PlayOneShot(Win);
        if (nextScene == "Level1")
        {
            PlayerPrefs.DeleteKey("LIVES_LEFT");
        }
    }
```

```
    public void moveToCheckPoint()
    {
        player.transform.position = checkPoint;
    }

    public void UpdateCheckPoint(Vector3 newCheckPoint)
    {
        checkPoint = newCheckPoint;
    }

    public void Reset(string sceneName)
    {
        buttonSound.clip = ButtonClick;
        buttonSound.Play();
        StartCoroutine(buttonPress(sceneName));
    }

    IEnumerator buttonPress(string name)
    {
        yield return new WaitForSeconds(2);
        SceneManager.LoadScene(name);
    }

}
```

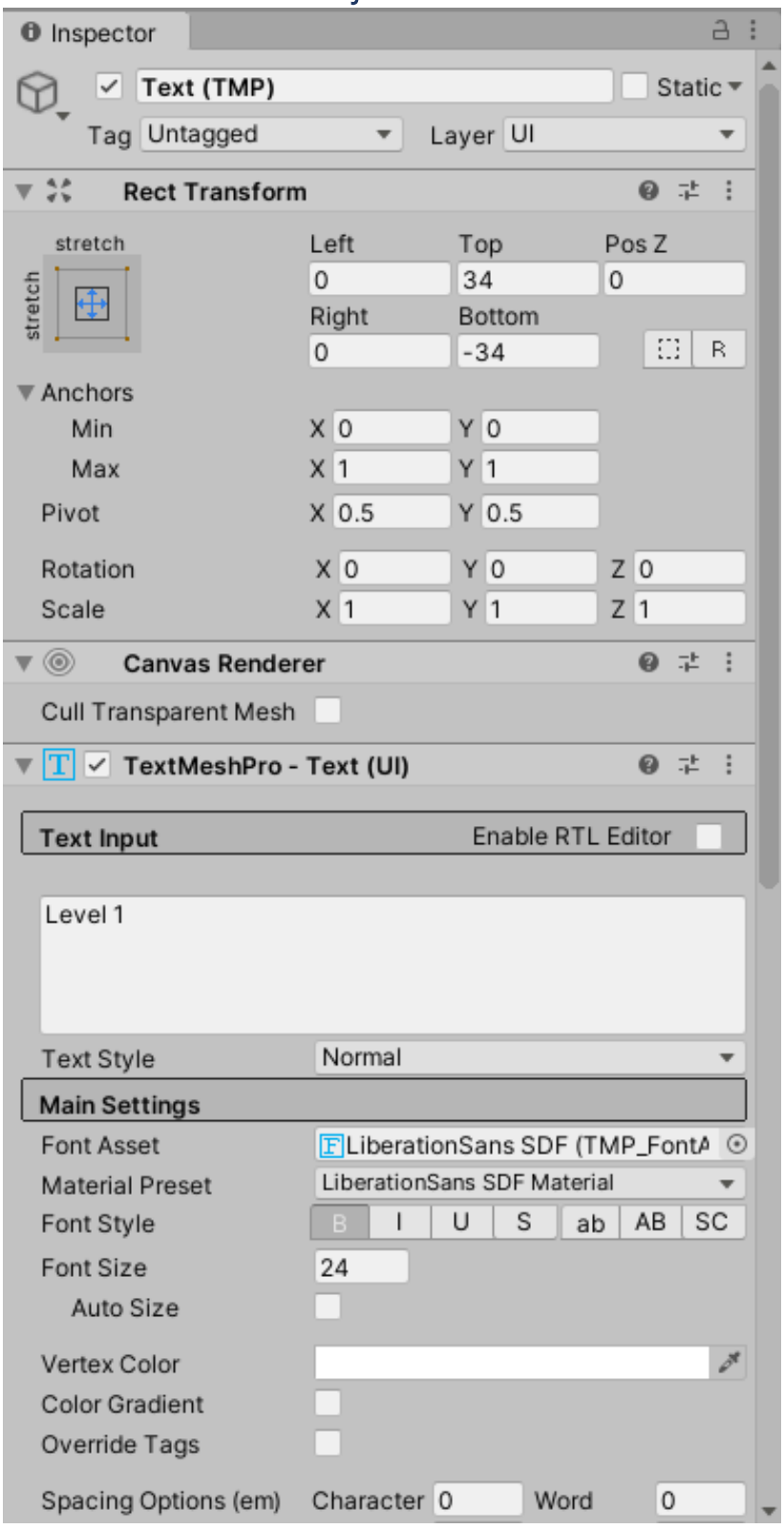
Level1Select Object



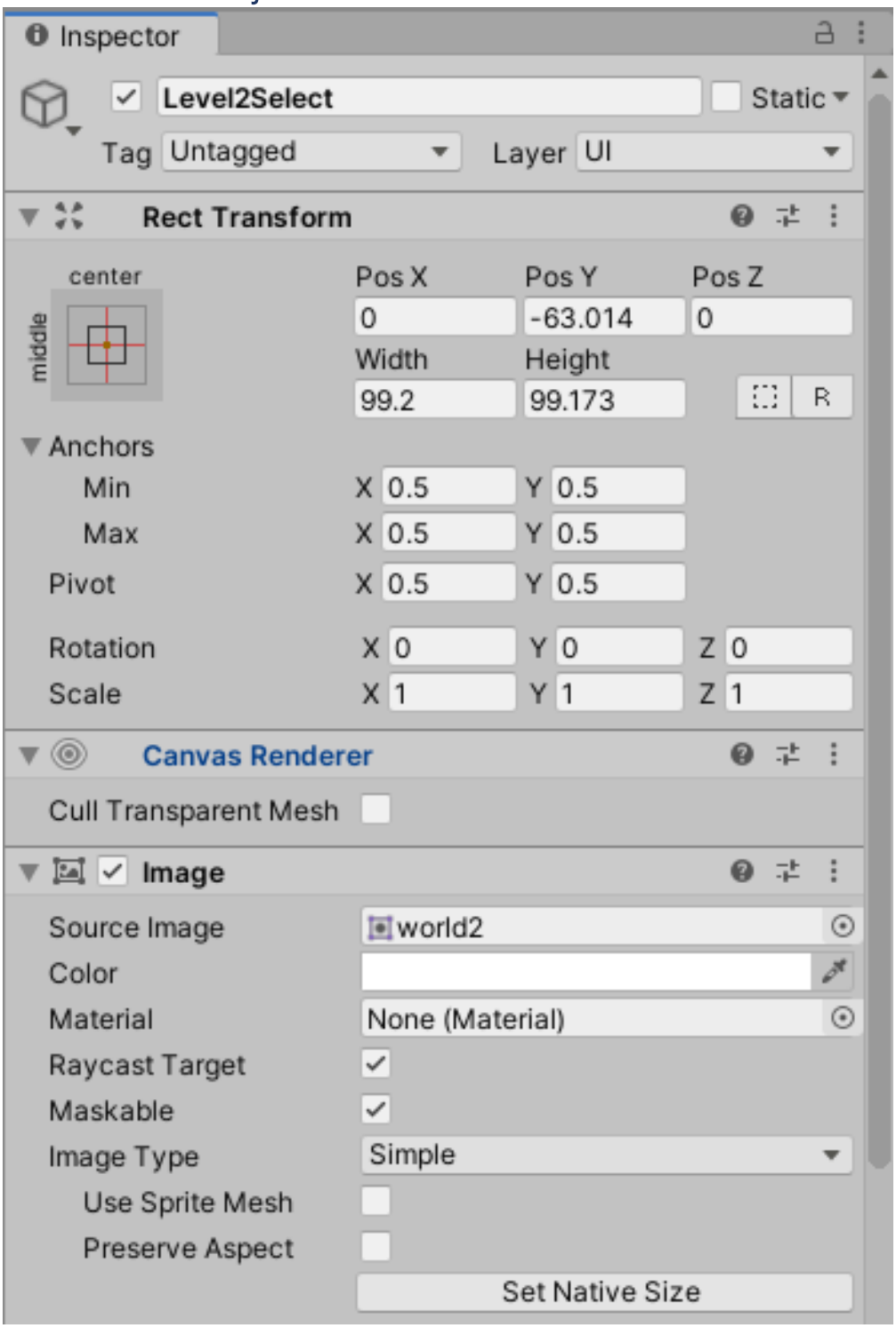
Level1Select Object Continued



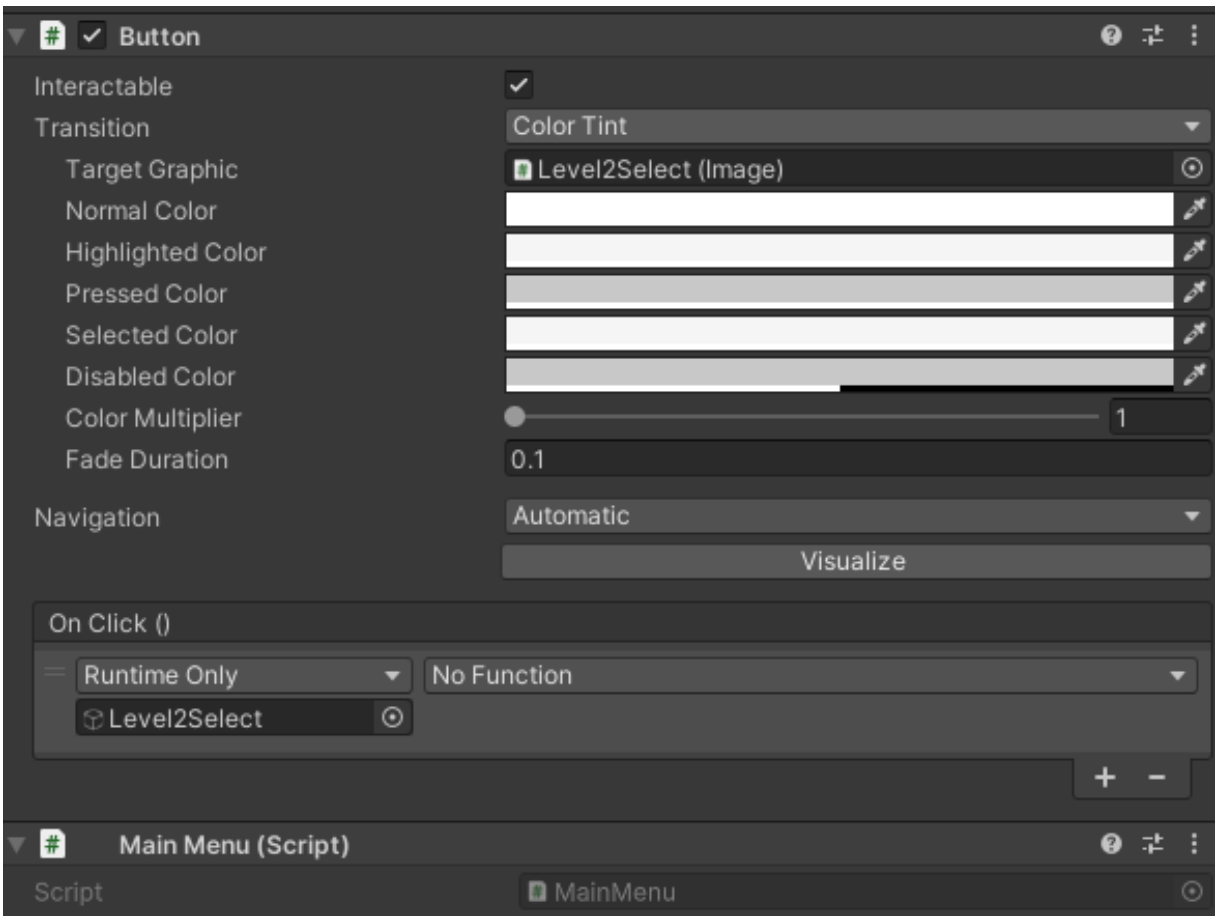
Level1Select Text Object



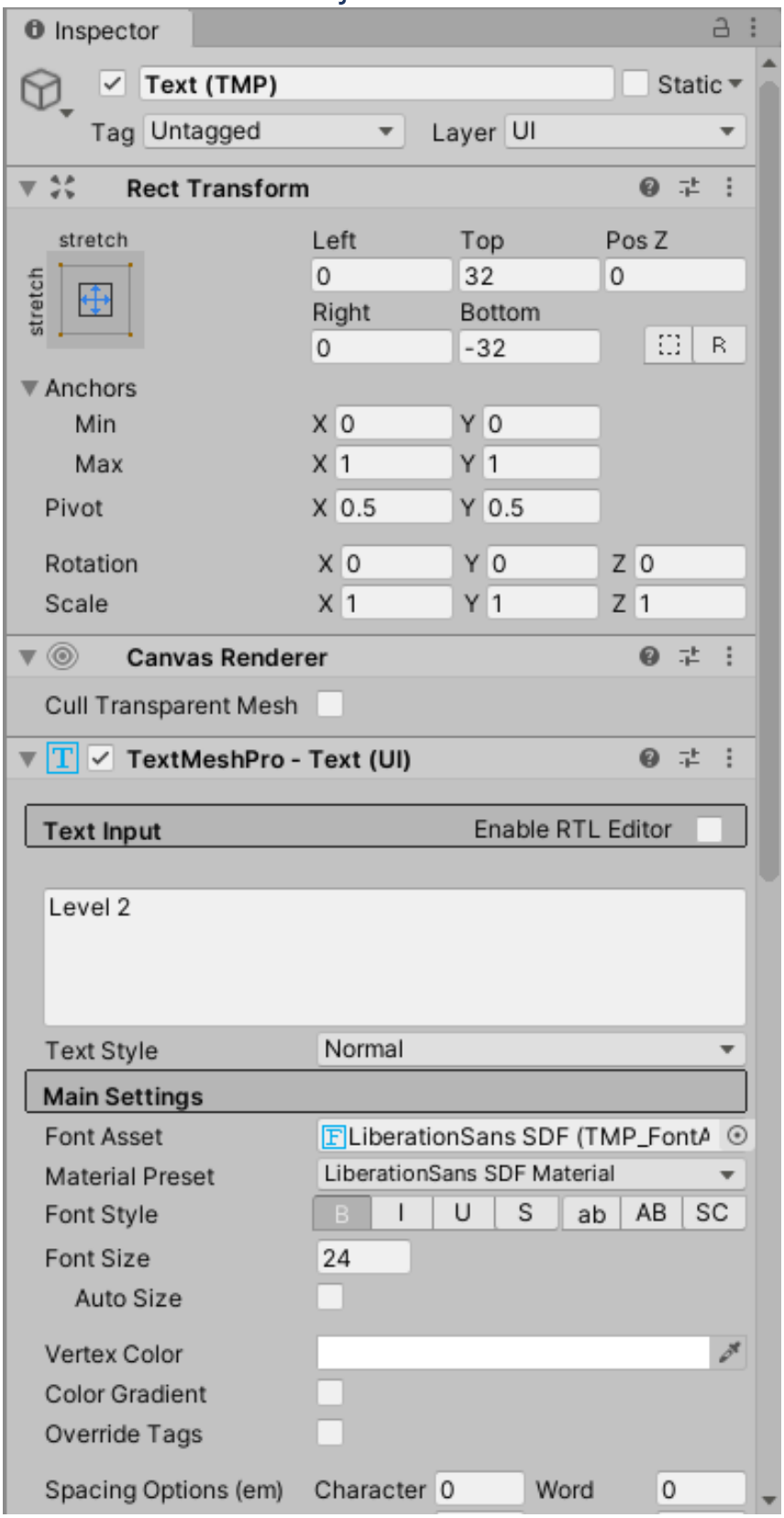
Level2Select Object



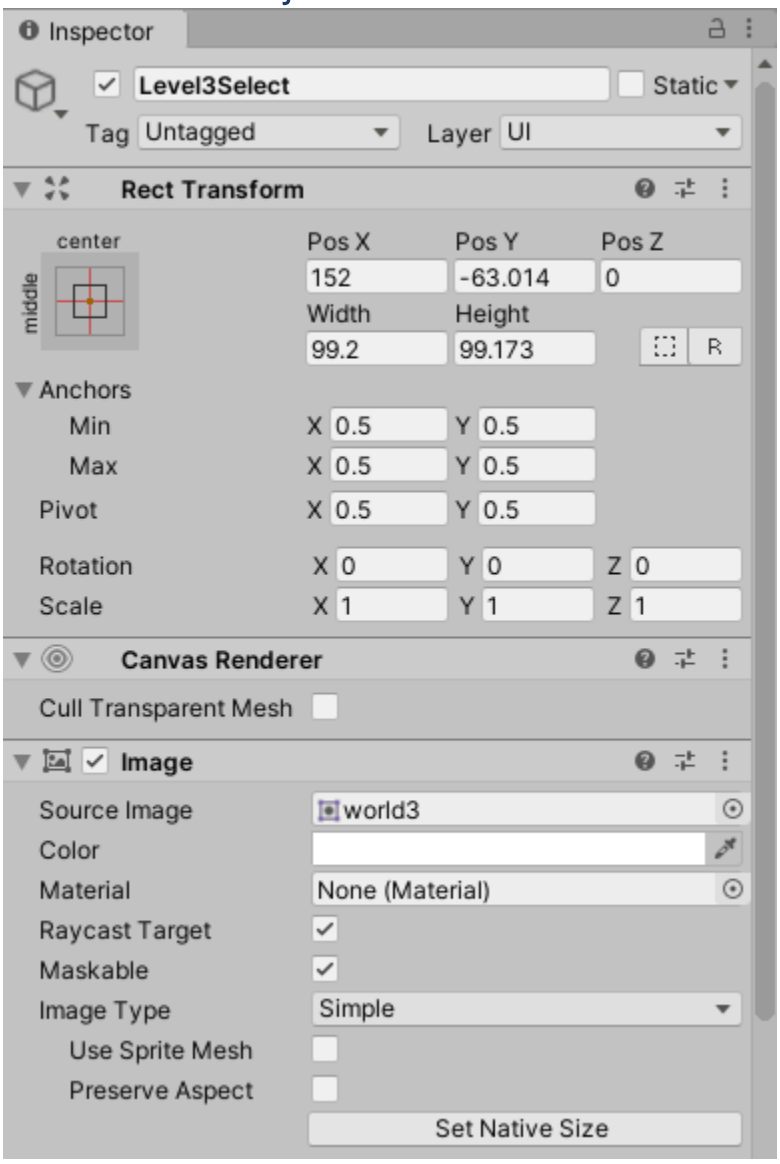
Level2Select Object Continued



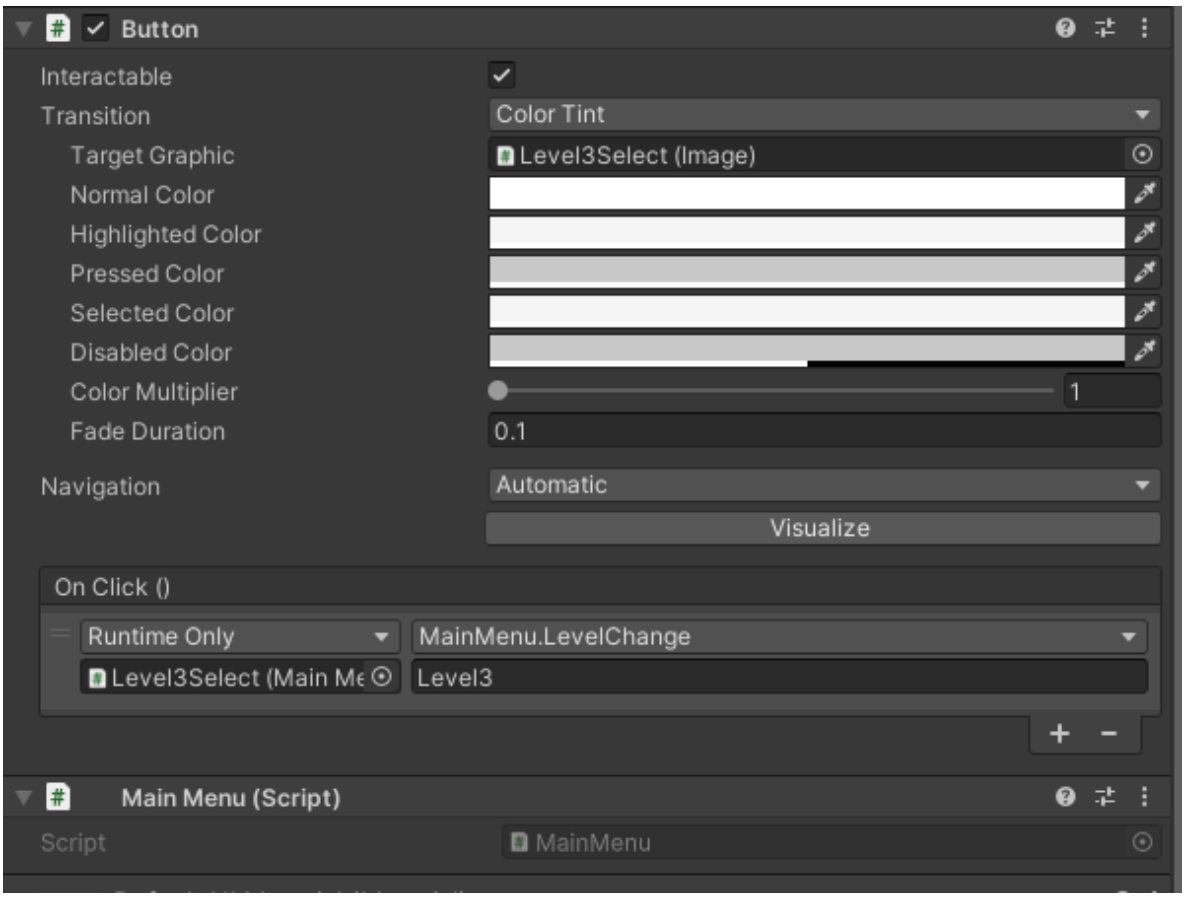
Level2Select Text Object



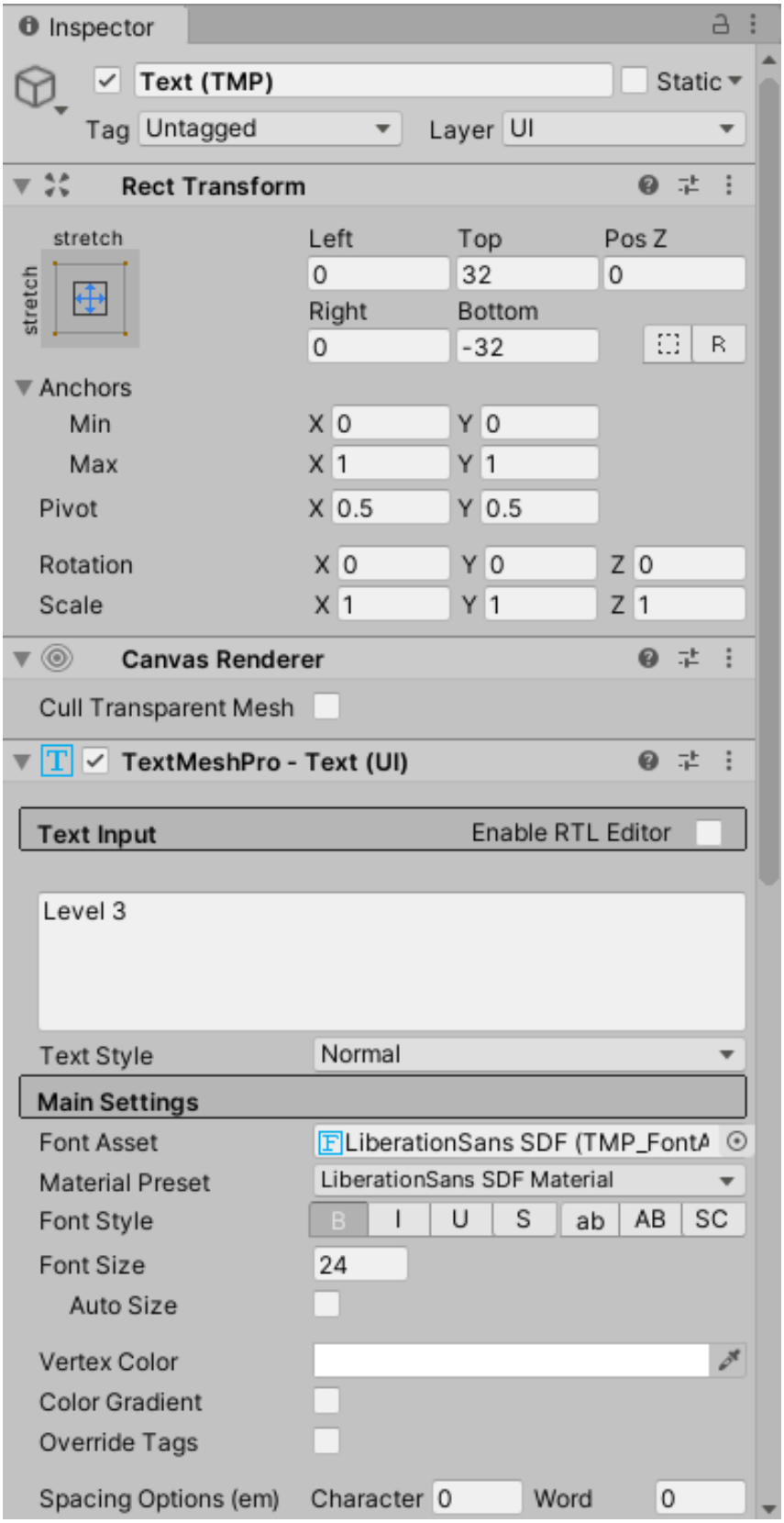
Level3Select Object



Level3Selsect Continued



Level3Select Text



MainMenu.cs Script

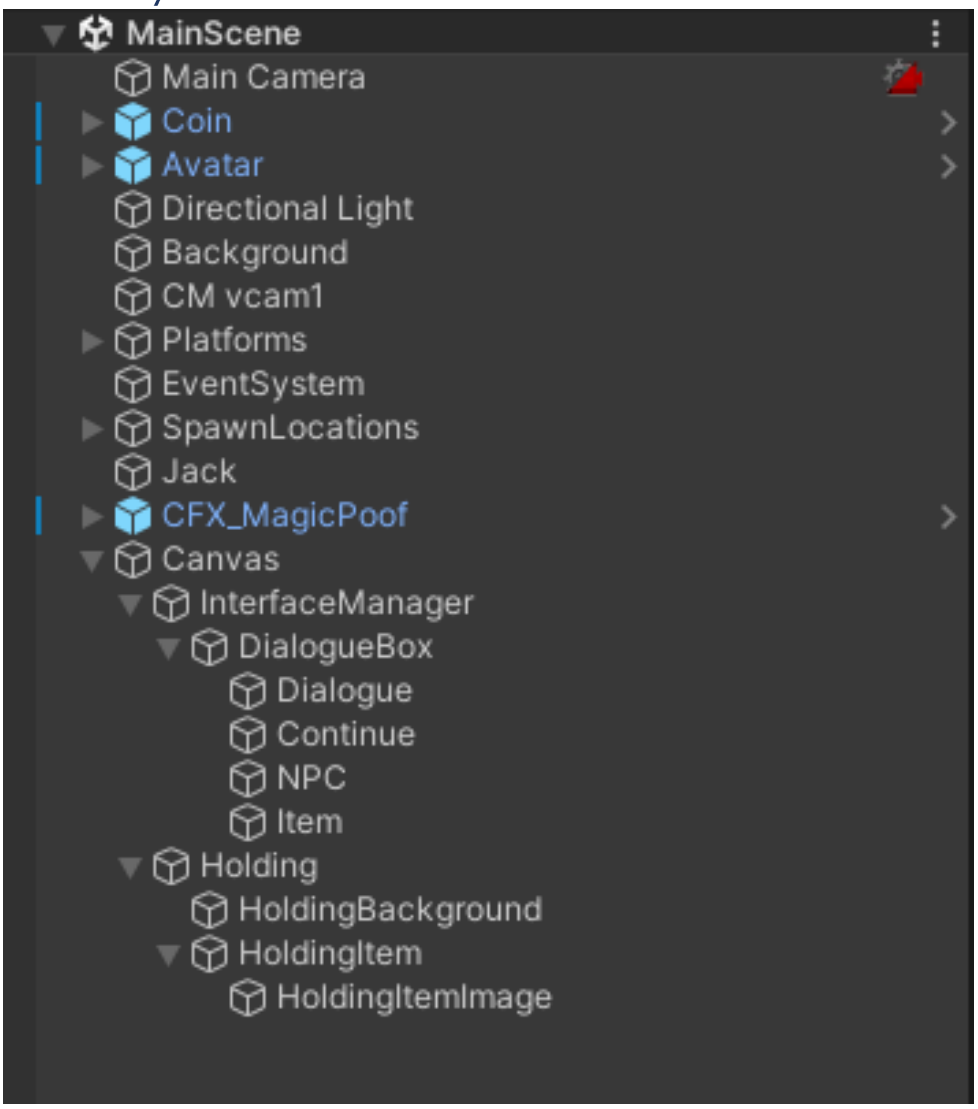
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour
{
   public void LevelChange(string level)
    {
        SceneManager.LoadScene(level);
        PlayerPrefs.DeleteKey("LIVES_LEFT");
    }

}
```

Activity Solution: Amazing Ninja Worlds Part 3 Prove Yourself

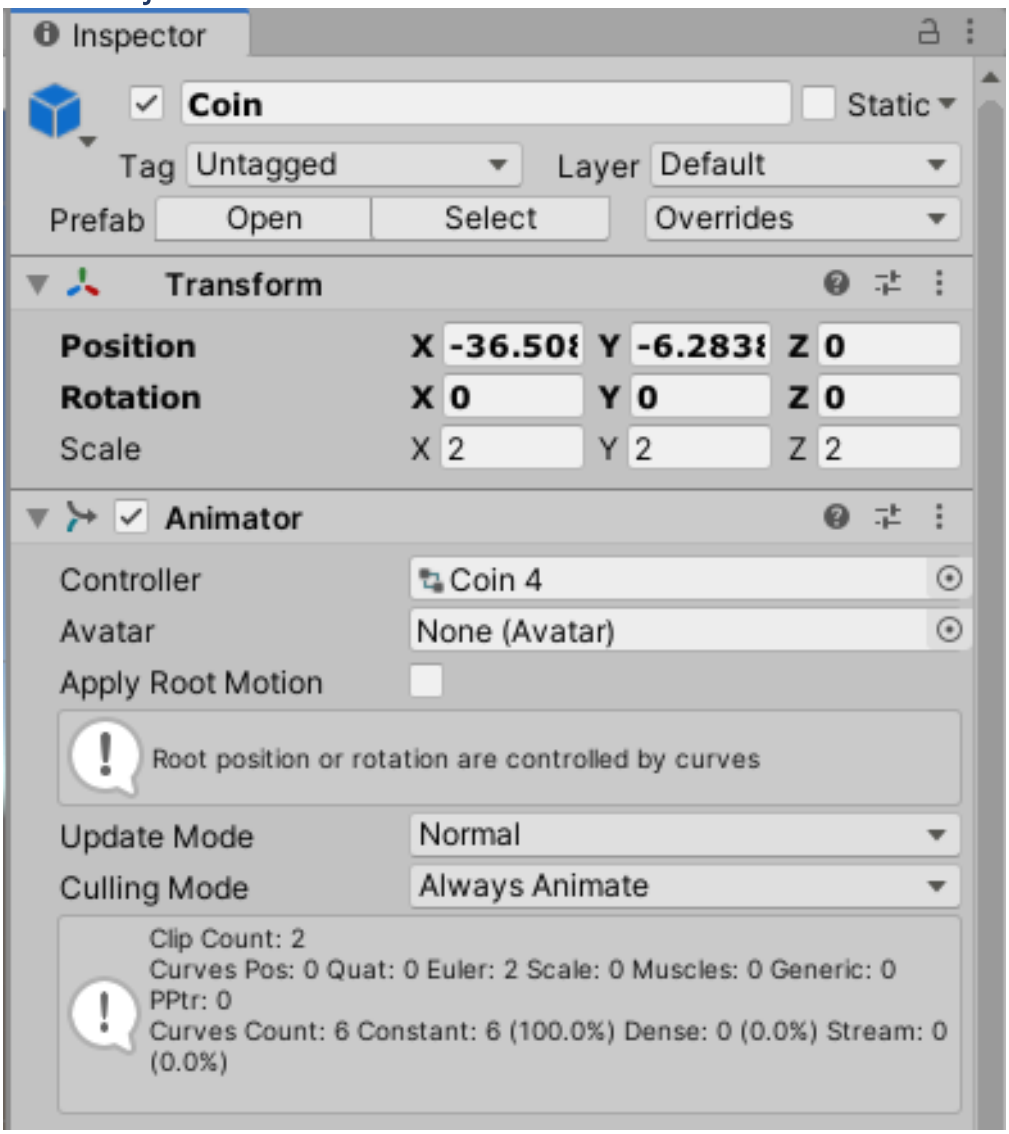
There is no provided solution for this activity. Each ninja must use what they learned to complete the tasks and create a game that has at least 3 levels.

Activity Solution: Scavenger Hunt Deluxe

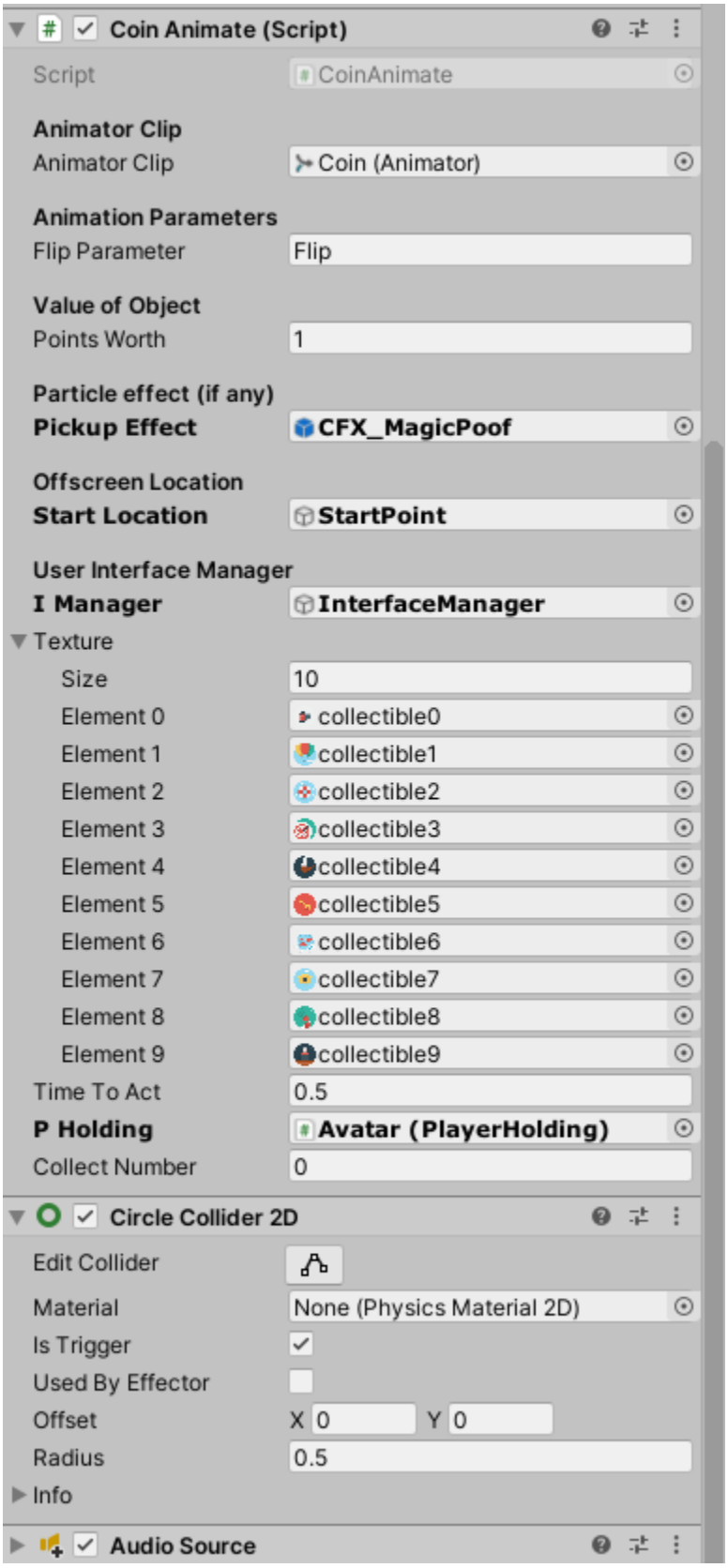
Hierarchy



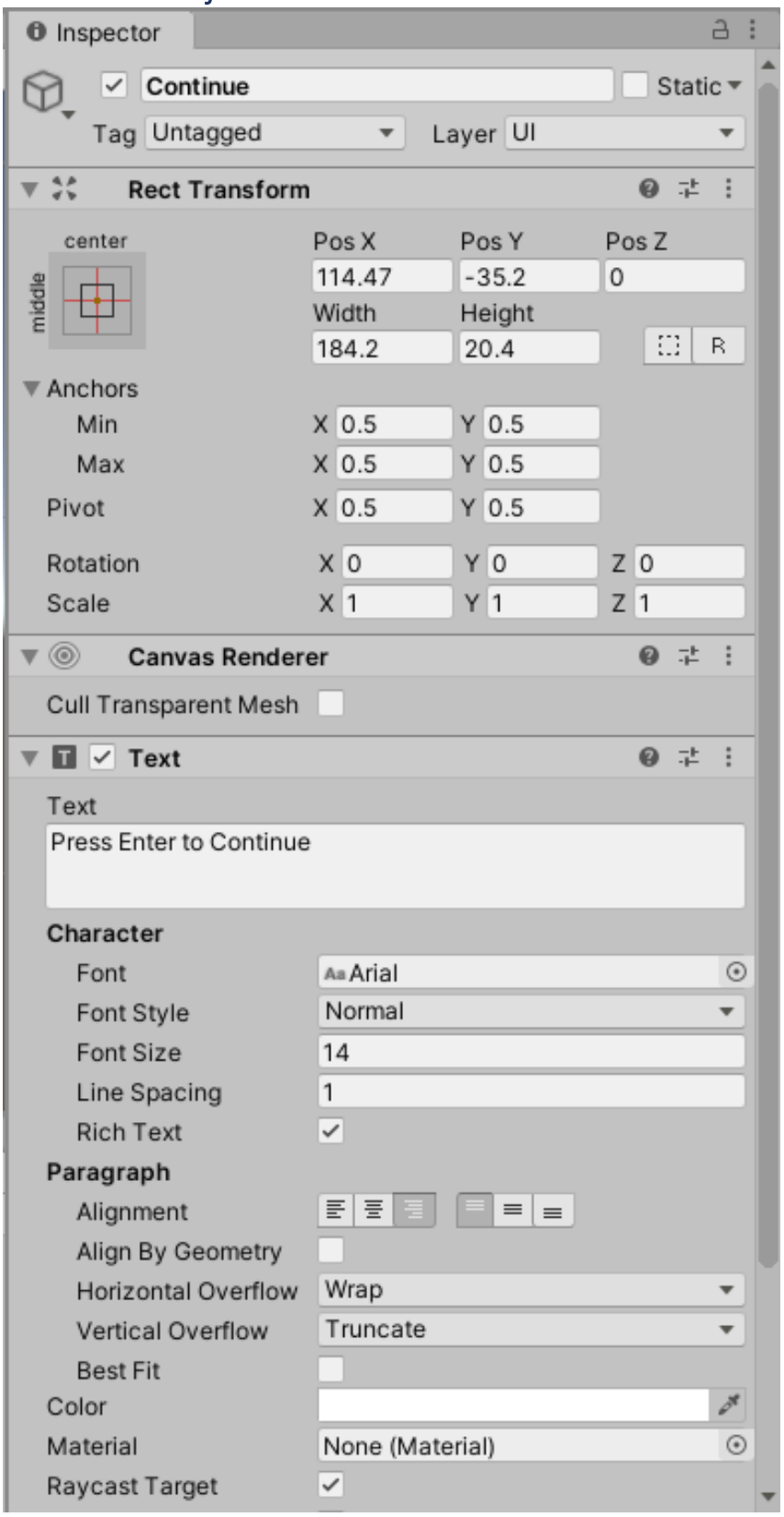
Coin Object



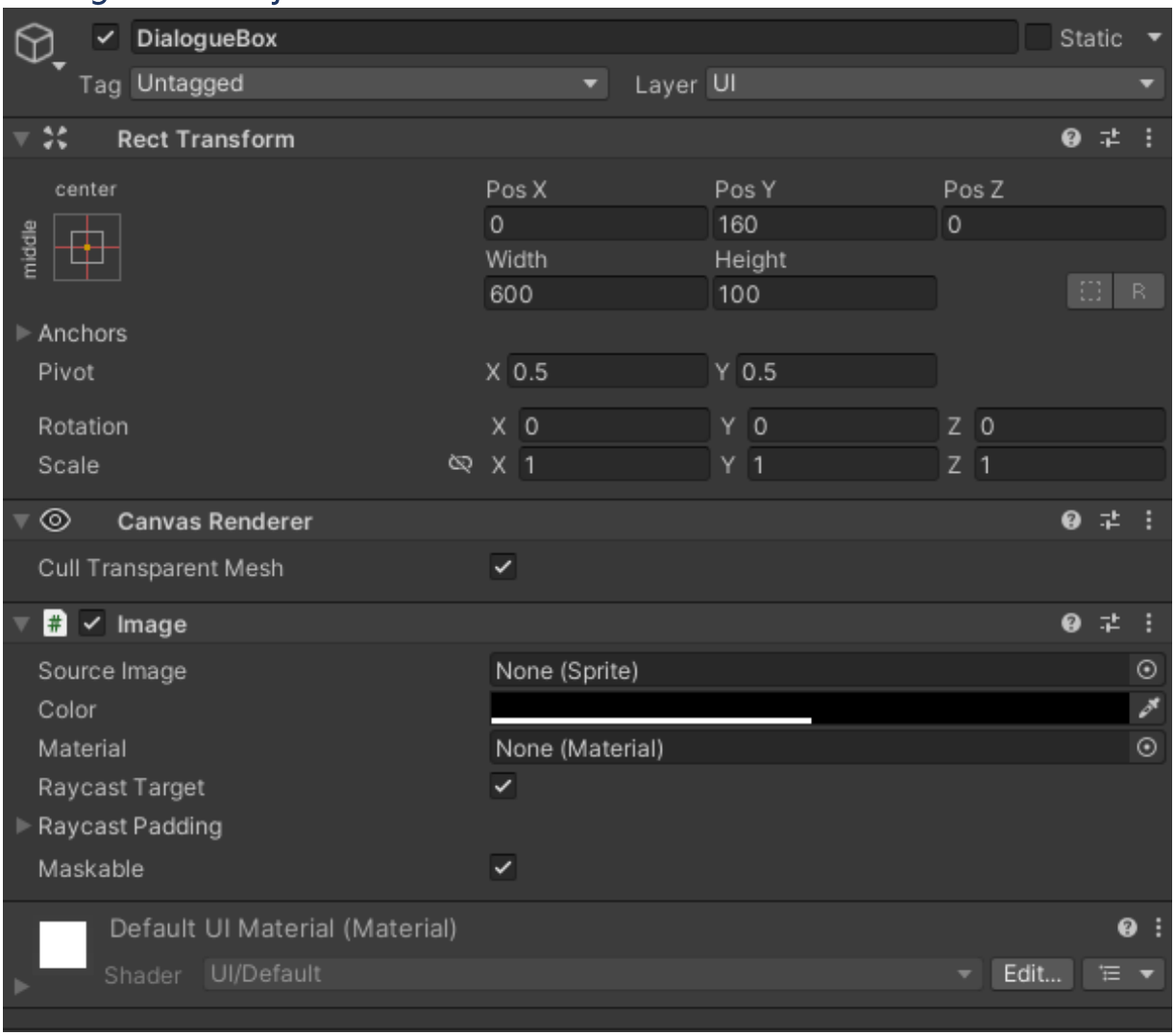
Coin Object Continued



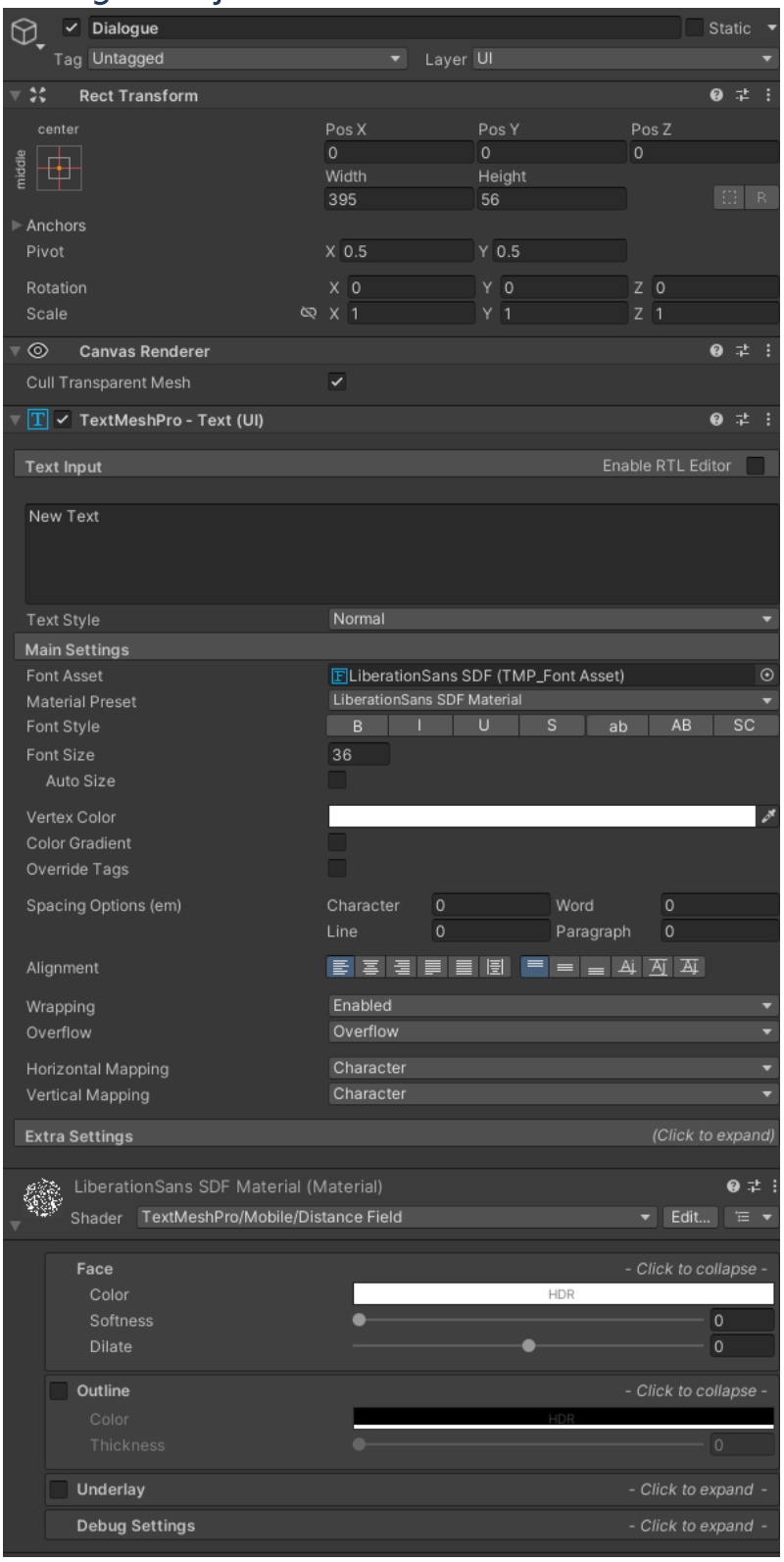
Continue Object



DialogueBox Object



Dialogue Object

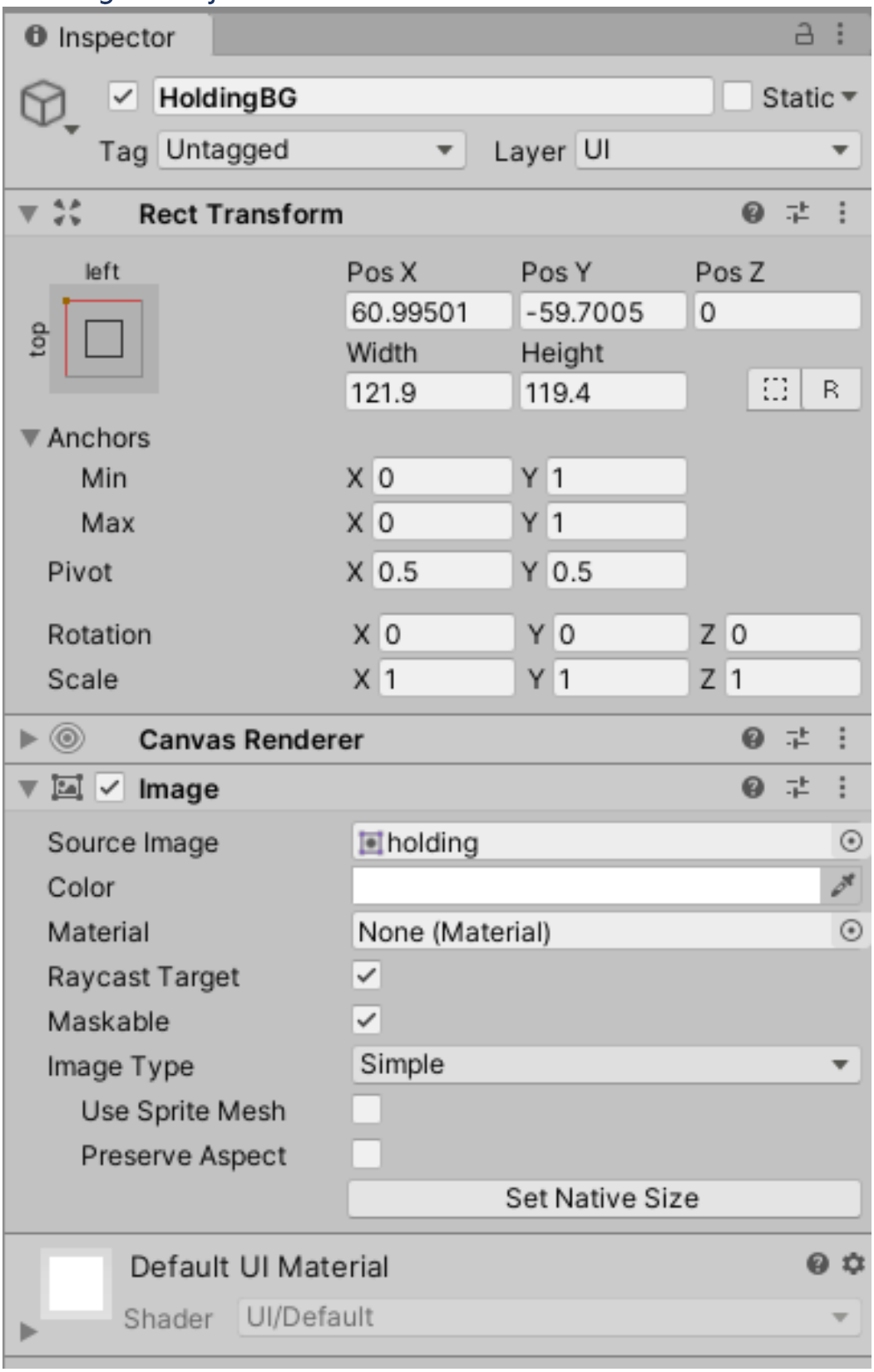


DialogueOpen.cs Script

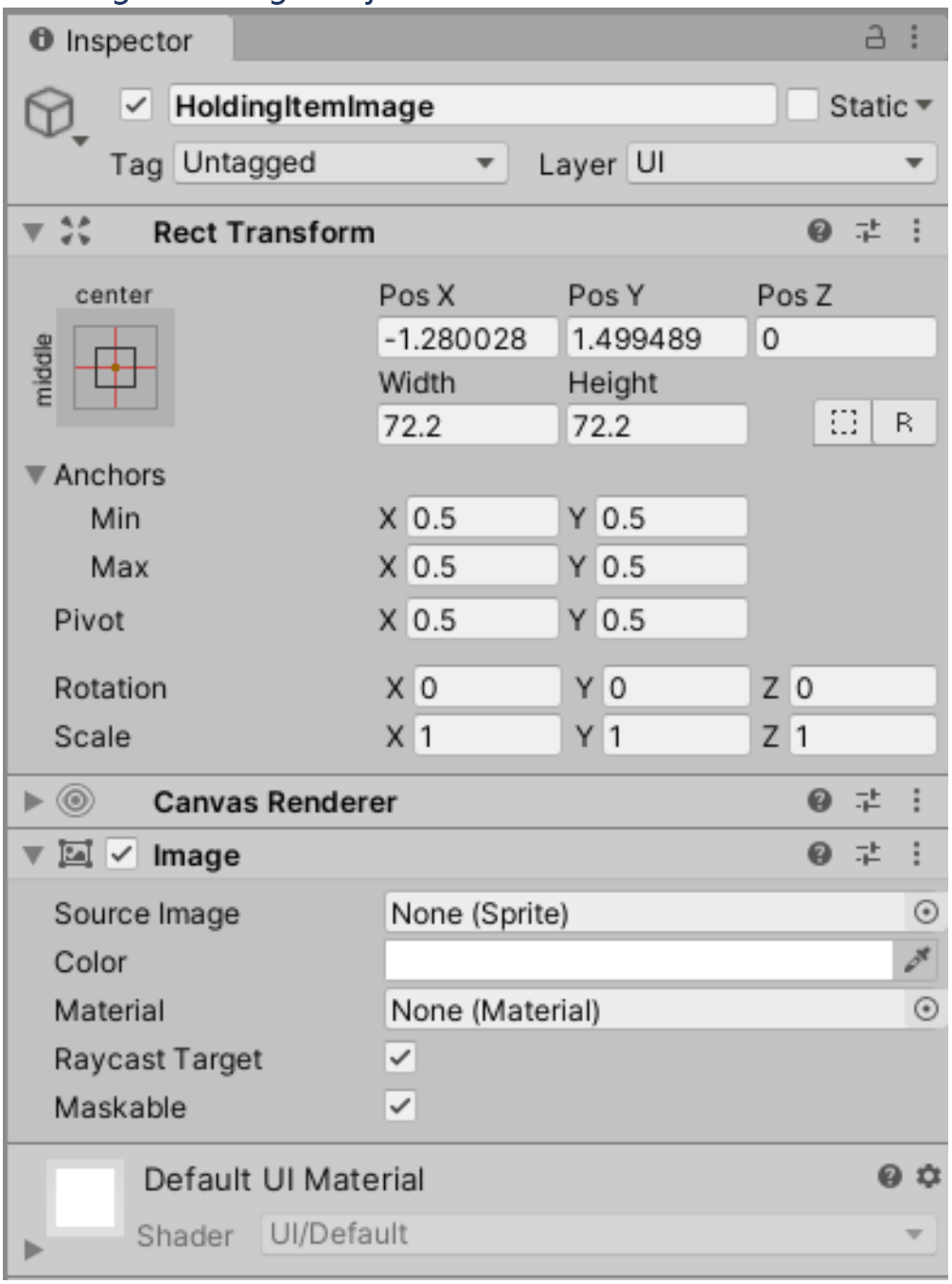
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DialogueOpen : MonoBehaviour
{

    public string dialogue;
    public GameObject interfaceManager;
    public PlayerHolding pHolding;
    public bool begin = true;
    public bool end = false;
    private string[] collectibles;
    private int clue;

    private AudioSource greeting;

    // Start is called before the first frame update
    void Start()
    {
        greeting = GetComponent<AudioSource>();
        collectibles = new string[] { "film", "balloons", "life saver", "bull's eye",
"pipe", "key", "fish", "birdhouse", "red airhorn", "magic hat" };
        createClue();
    }

    public void createClue()
    {
        clue = Random.Range(0, 9);
        searchDialogue();
    }

    public void searchDialogue()
    {
        dialogue = "Hi! Can you help me find my " + collectibles[clue] + "?";
    }

    private void OnTriggerEnter2D(Collider2D other)
    {
        if (!begin && pHolding.Verify())
        {
            checkClue();
        }
        greeting.Play(0);
        interfaceManager.GetComponent<InterfaceManager>().ShowBox(dialogue, clue);
    }

    private void checkClue()
    {
        if (pHolding.holdValue == clue)
        {
            dialogue = "You found my " + collectibles[clue] + "! Hooray!";
            end = true;
        }
        else
```

```
            {
                dialogue = "No, that's not my " + collectibles[clue] + ".";
            }
        }

        public void coinsScattered()
        {
            begin = false;
        }

    }
```

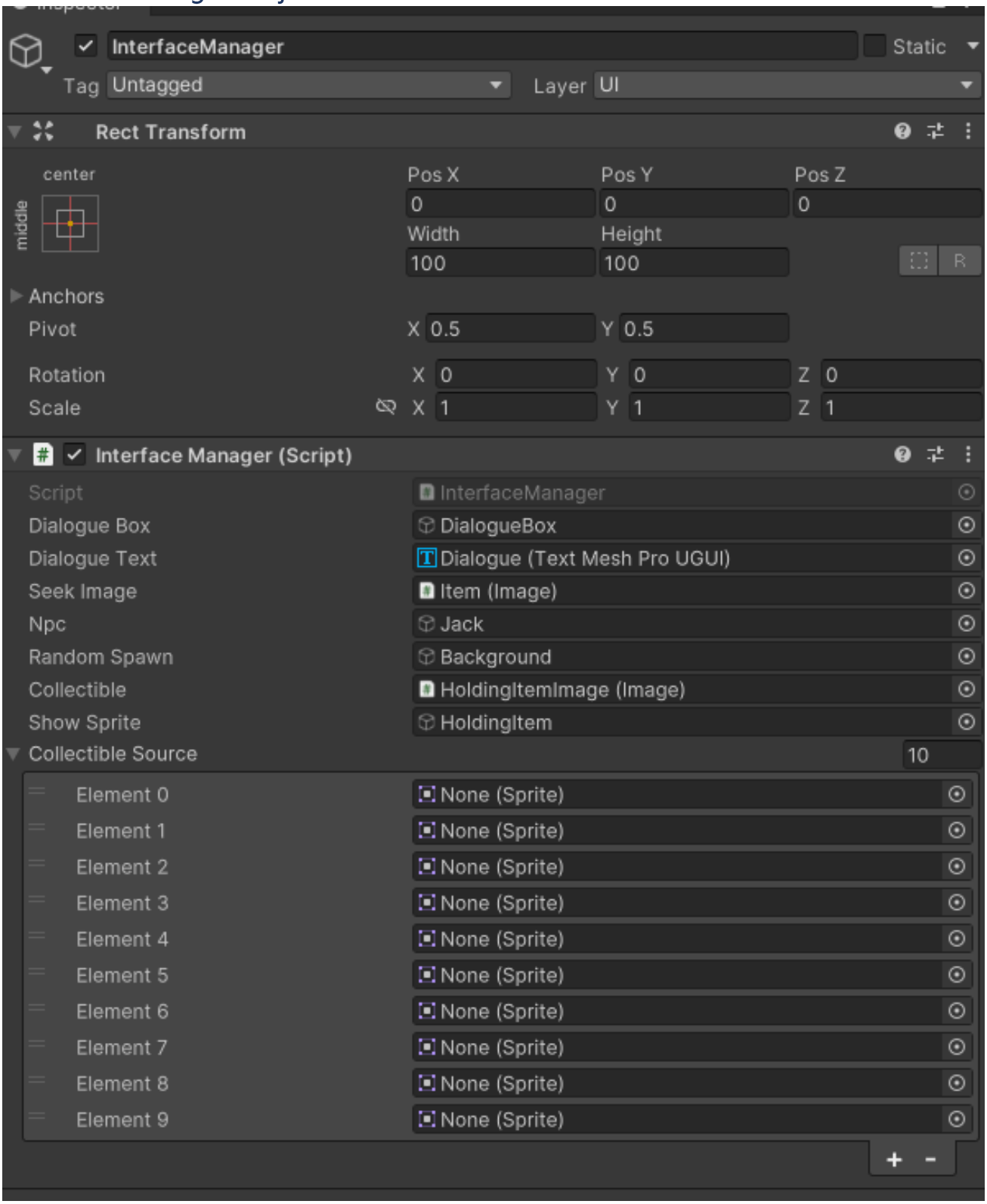
HoldingBG Object



HoldingItemImage Object



InterfaceManager Object

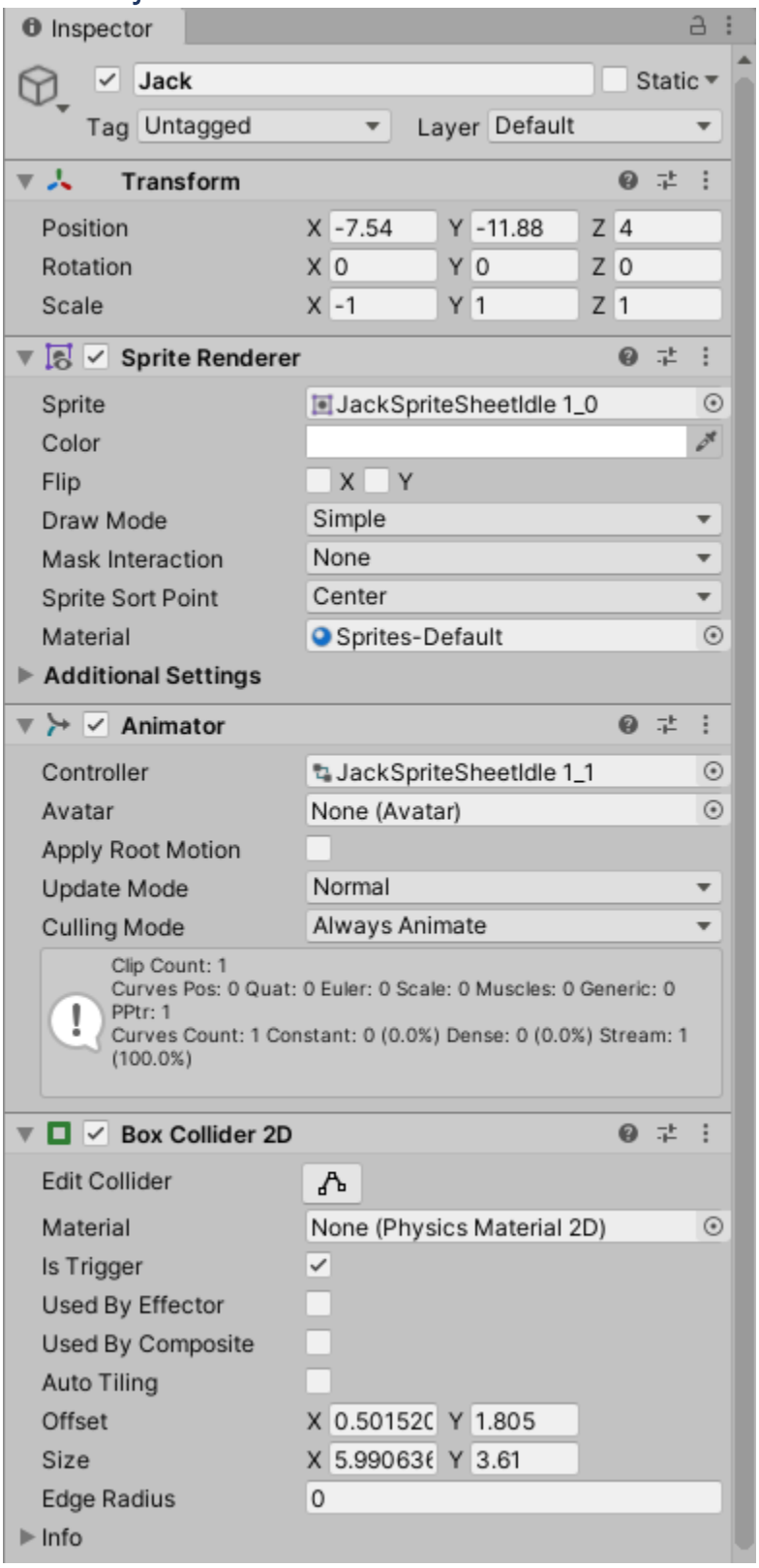


InterfaceManager.cs Script

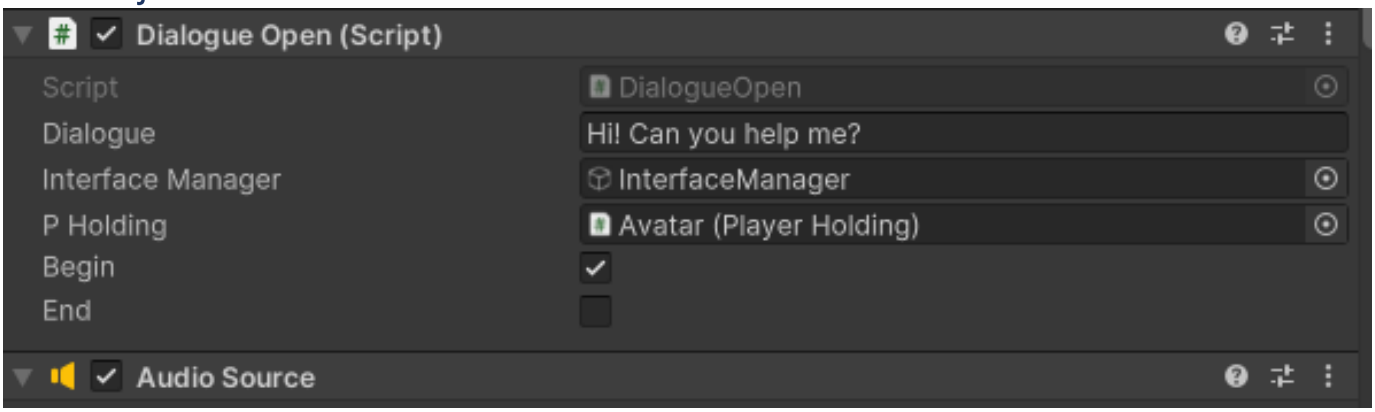
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using TMPro;

public class InterfaceManager : MonoBehaviour
{
    public GameObject dialogueBox;
    public TMP_Text dialogueText;
    public Image seekImage;
    public GameObject npc;
    public GameObject randomSpawn;

    public Image collectible;
    public GameObject showSprite;

    [SerializeField]
    private Sprite[] collectibleSource;

    // Start is called before the first frame update
    void Start()
    {
        dialogueBox.SetActive(false);
        showSprite.SetActive(false);
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetButton("Submit") && dialogueBox.activeInHierarchy)
        {
            dialogueBox.SetActive(false);

            if (npc.GetComponent<DialogueOpen>().end)
            {
                SceneManager.LoadScene(0);
            }
        }
    }

    public void CollectibleUpdate(int item)
    {
       showSprite.SetActive(true);
       collectible.GetComponent<Image>().sprite = collectibleSource[item];
    }

    public void ShowBox(string dialogue, int item)
    {
        dialogueBox.SetActive(true);
        dialogueText.text = dialogue;
        seekImage.GetComponent<Image>().sprite = collectibleSource[item];

        if (npc.GetComponent<DialogueOpen>().begin)
```

```
        {
            scatterCoins();
        }
    }

    public void scatterCoins()
    {
        randomSpawn.GetComponent<RandomSpawn>().DistributeCollectibles();
        npc.GetComponent<DialogueOpen>().coinsScattered();
    }
}
```

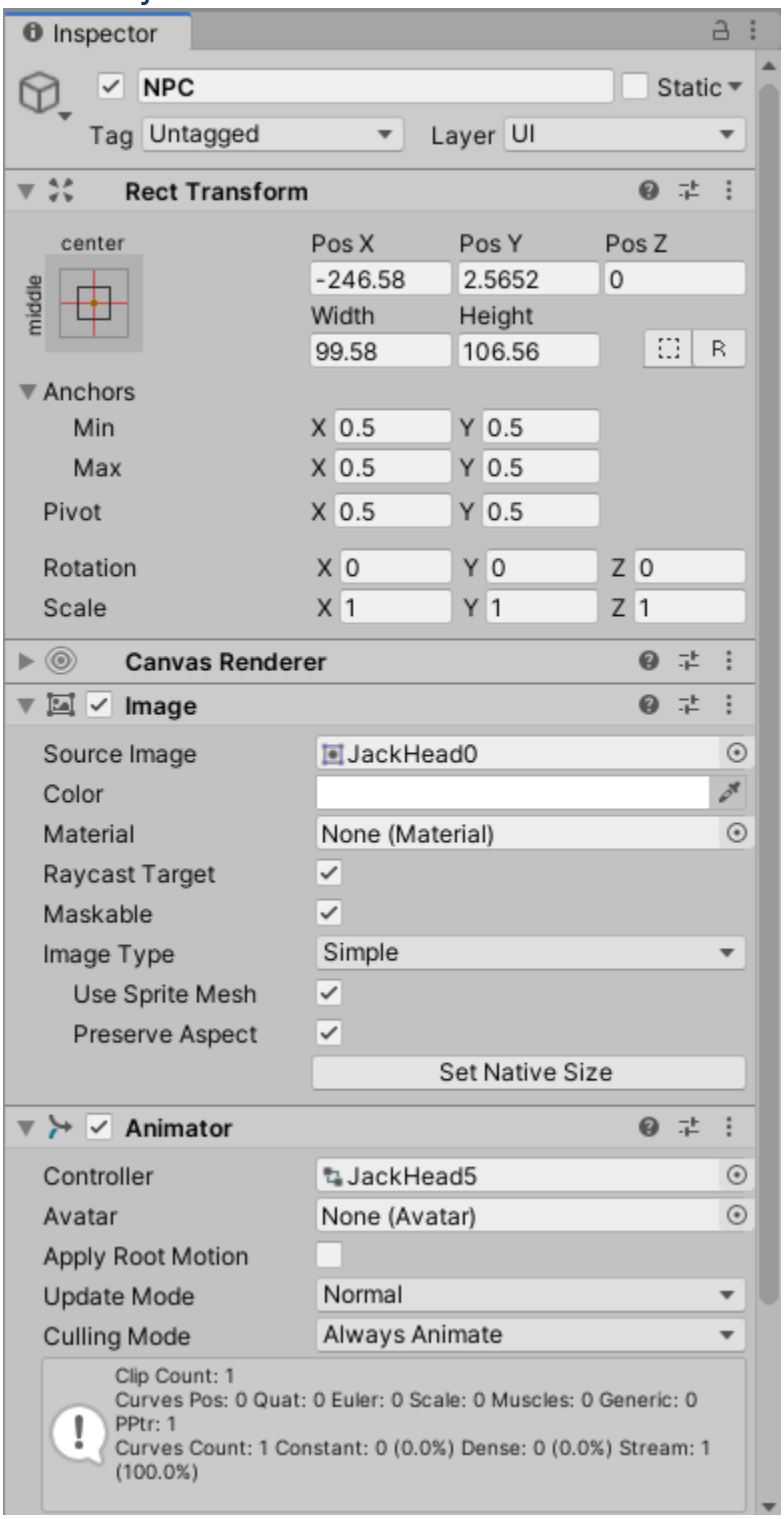
Jack Object



Jack Object Continued

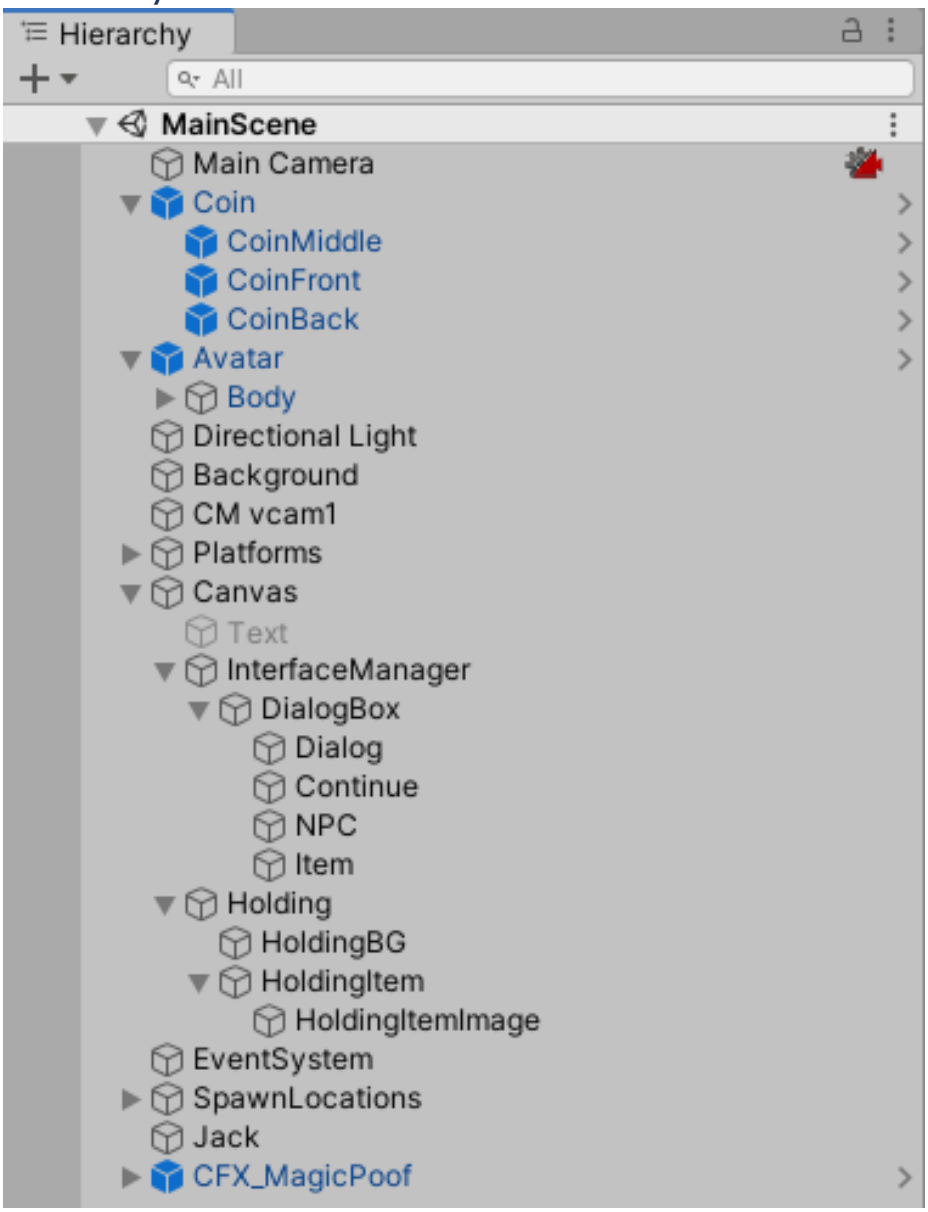


NPC Object



Activity Solution: Scavenger Hunt Deluxe Prove Yourself

Hierarchy



DialogOpen.cs Script

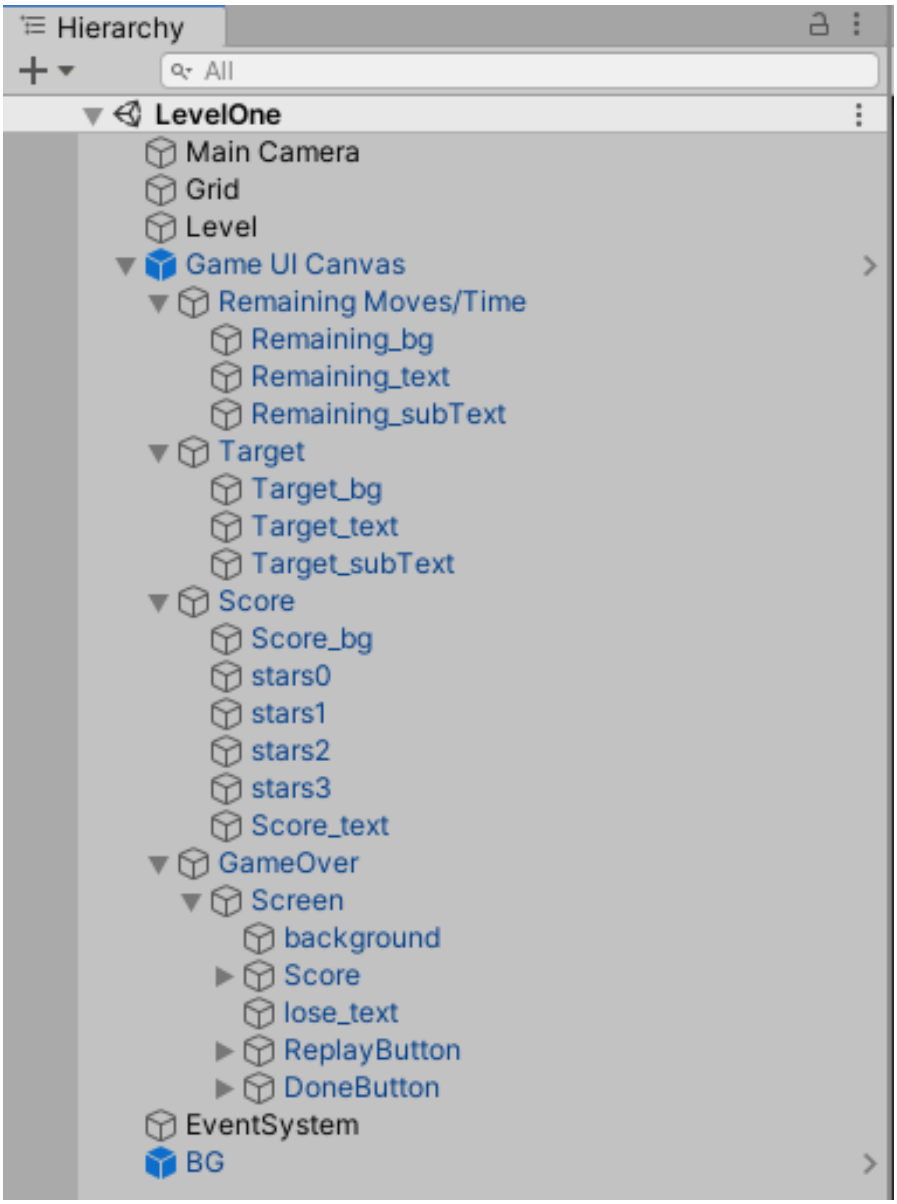
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DialogOpen : MonoBehaviour
{

    public string dialog;
    public GameObject interfaceManager;
    public PlayerHolding pHolding;
    private string[] collectibles;
    private int clue;
    private AudioSource greeting;
    public bool begin = true;
    public bool end = false;

    void Start()
    {
        greeting = GetComponent<AudioSource>();
        collectibles = new string[] { "film", "balloons",
            "life saver", "bull's eye", "pipe", "key",
            "fish", "birdhouse", "red airhorn", "magic hat"
        };
        createClue();
        begin = true;
    }

    public void createClue()
    {
        clue = Random.Range(0, 9);
        searchDialog();
    }

    private void OnTriggerEnter2D(Collider2D other)
    {
        greeting.Play(0);

        if (!begin && pHolding.Verify())
        {
            checkClue();
        }

        interfaceManager.GetComponent<InterfaceManager>().ShowBox(dialog, clue);
    }

    public void searchDialog()
    {
        dialog = "Hi! Can you help me find my " + collectibles[clue] + "?";
        begin = false;
    }
```

```
    public void checkClue()
    {
        if (pHolding.holdValue == clue)
        {
            if (collectibles[clue] == "film")
            {
                dialog = "You found my film! Now I can take pictures!";
            }
            if (collectibles[clue] == "balloons")
            {
                dialog = "I've been looking for my balloons!";
            }
            // write an if statement for each of the items
            end = true;
        }
        else
        {
            dialog = "No, that's not my " + collectibles[clue] + ".";
        }
    }
}
```

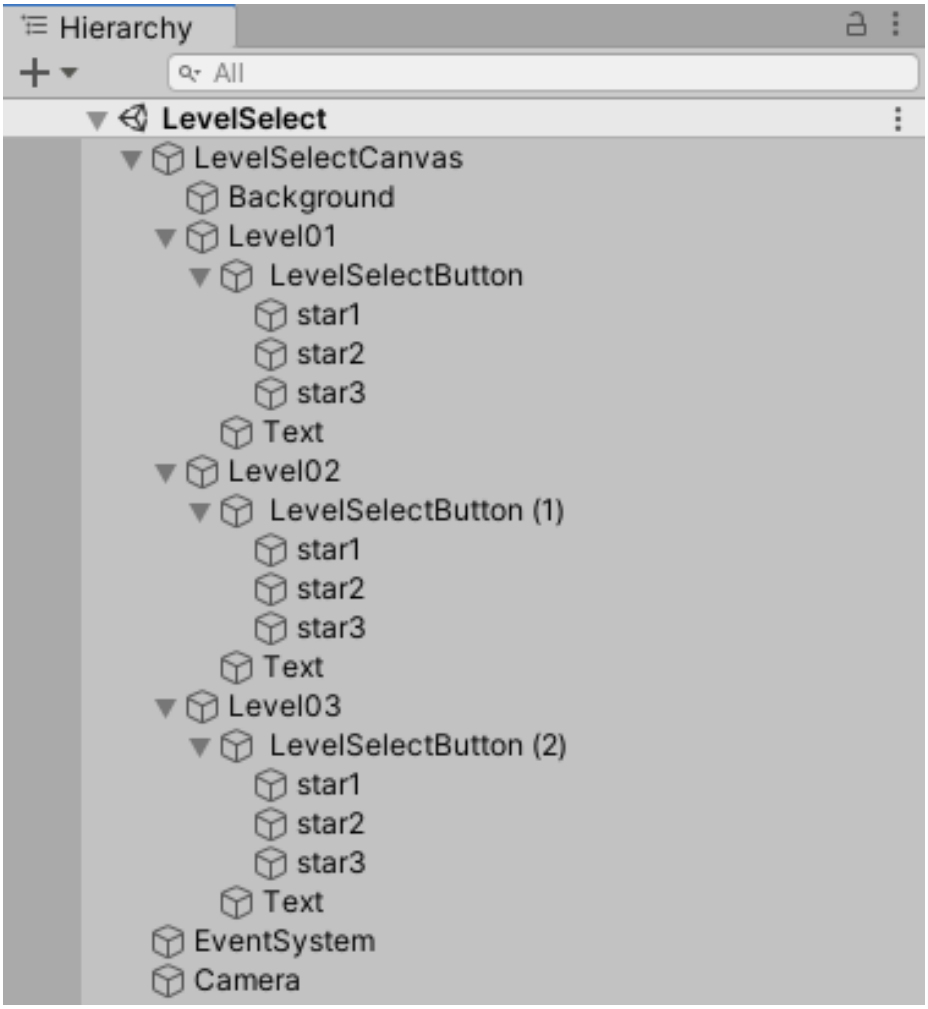
Activity Solution: Food Frenzy

Hierarchy

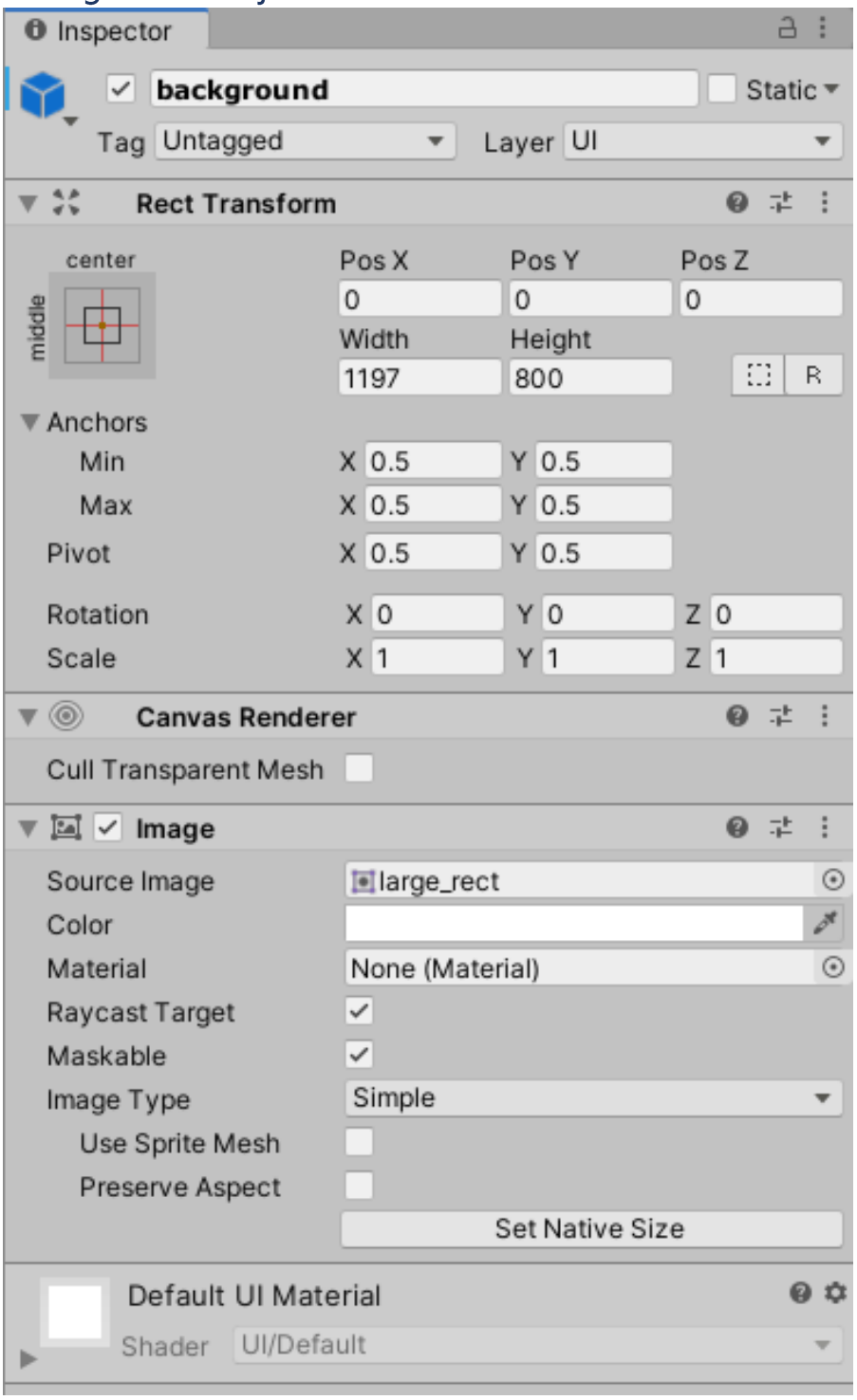
Level Scenes



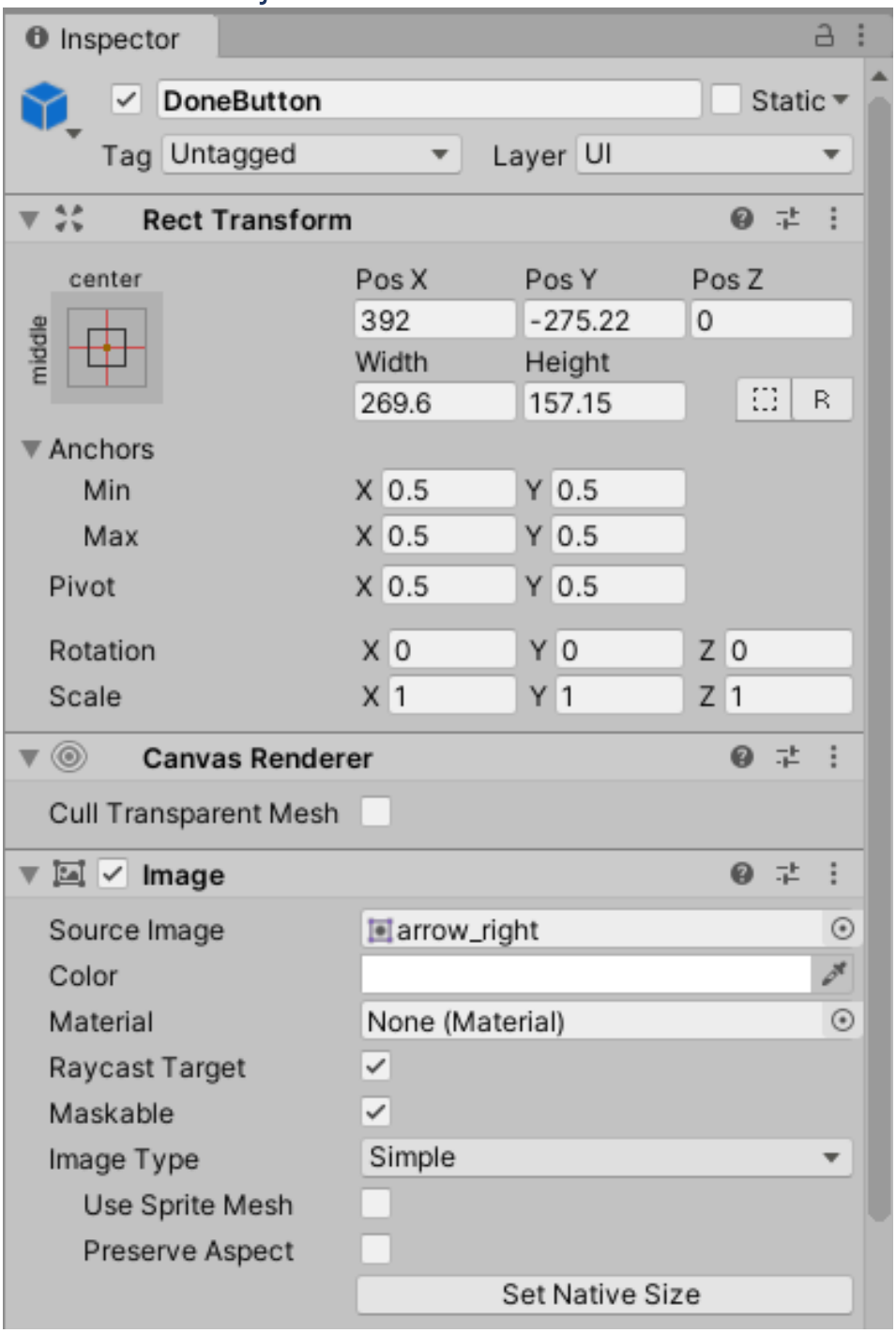
Level Select Scene



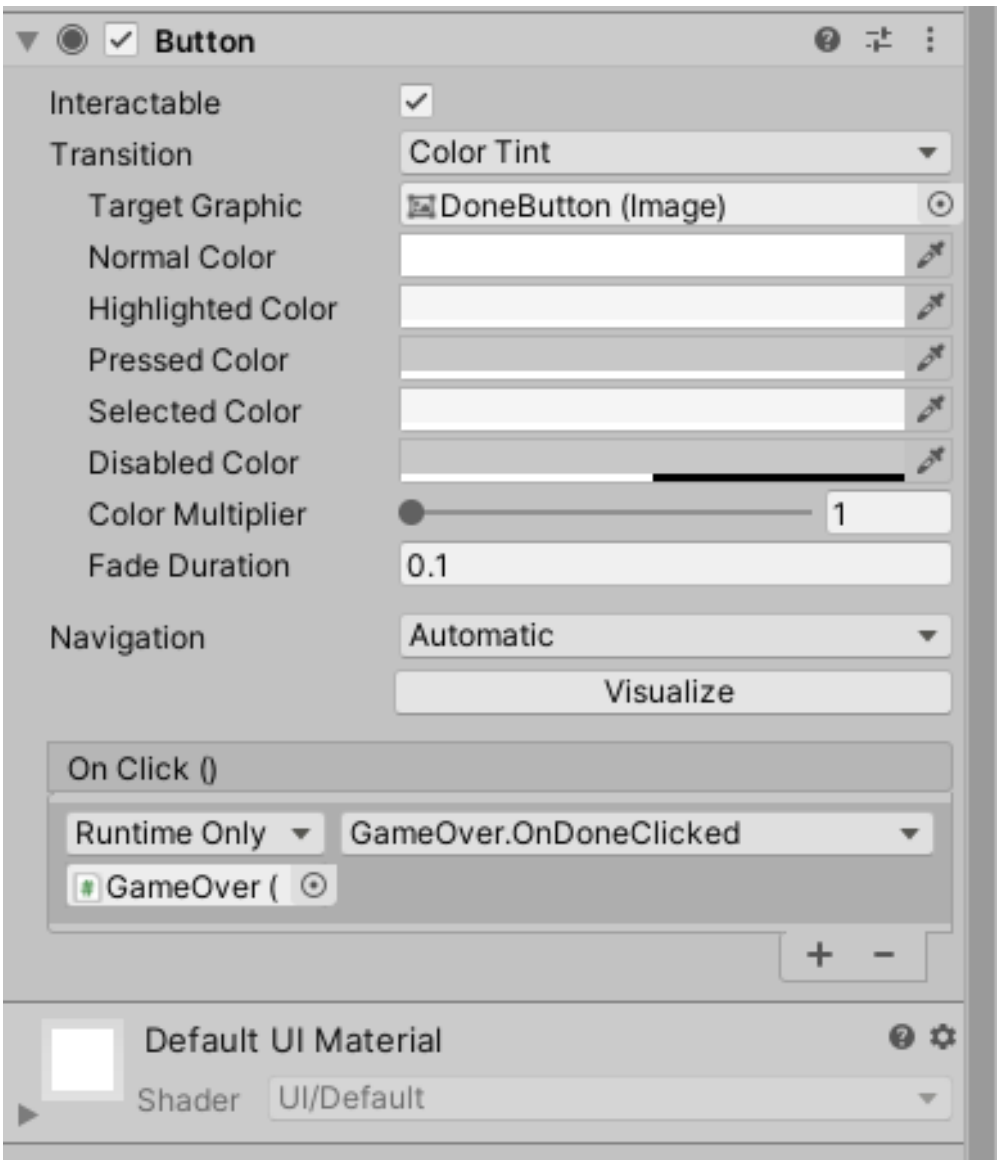
Background Object



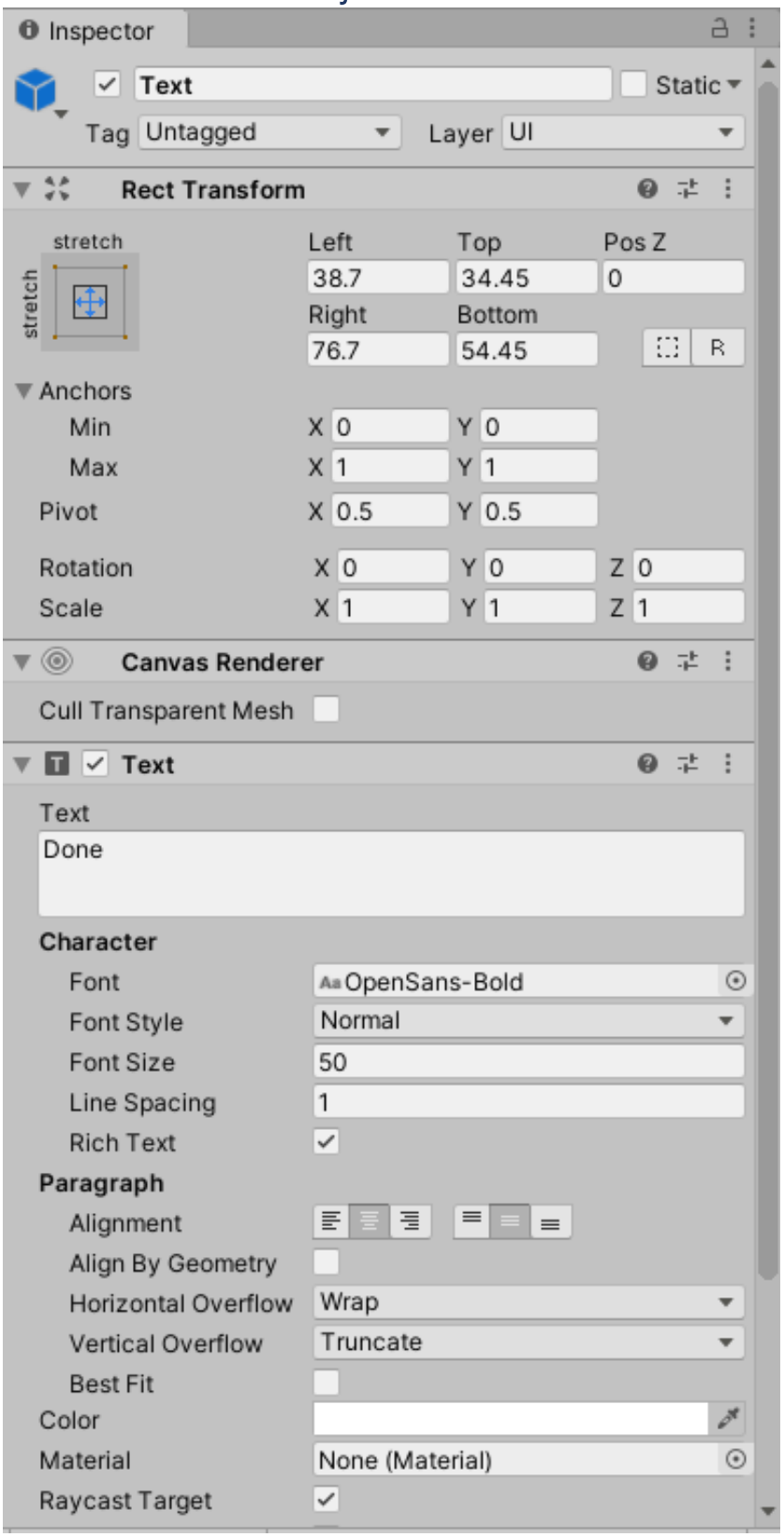
DoneButton Object



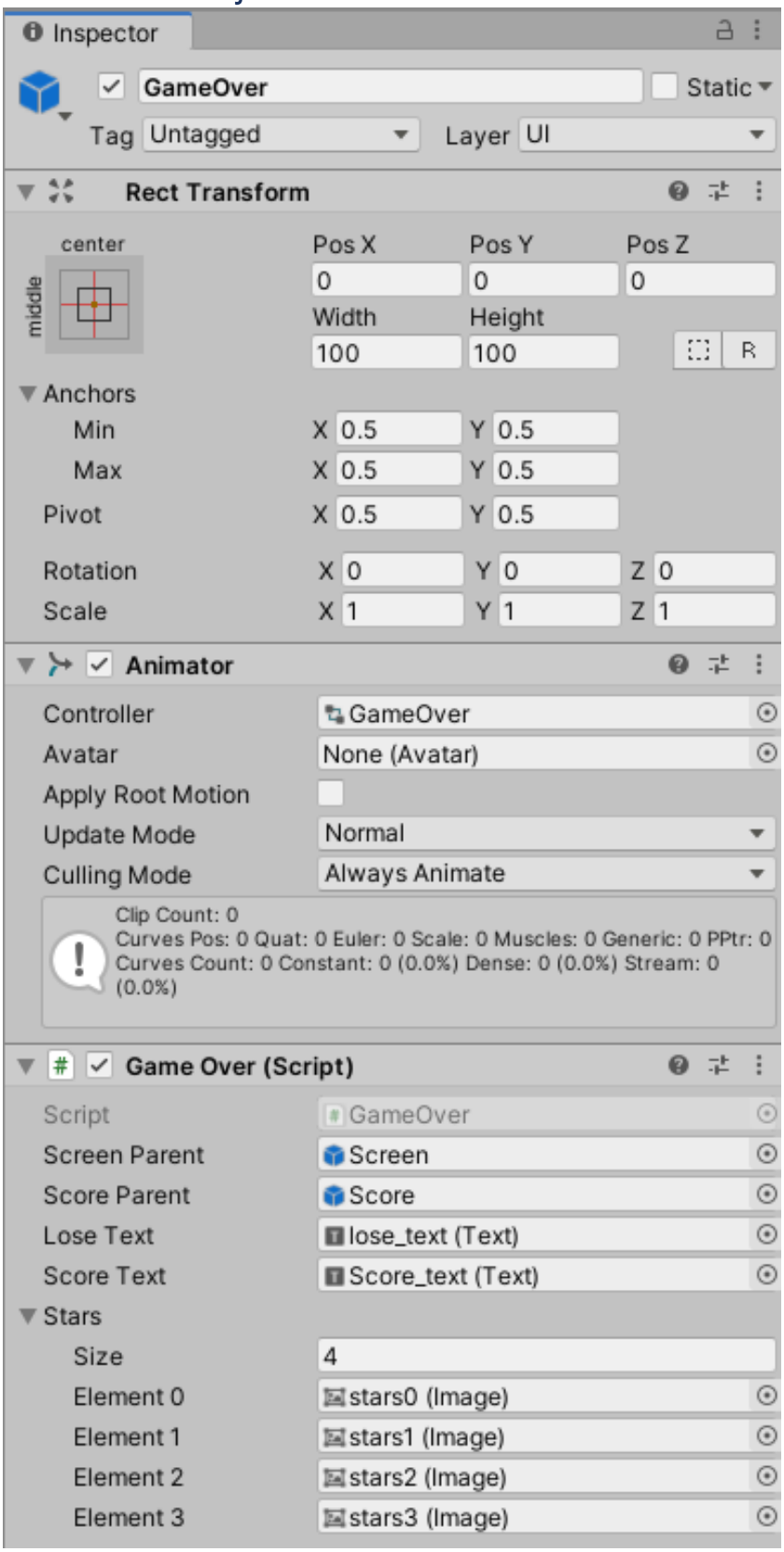
DoneButton Object Continued



DoneButton Text Object



GameOver Object

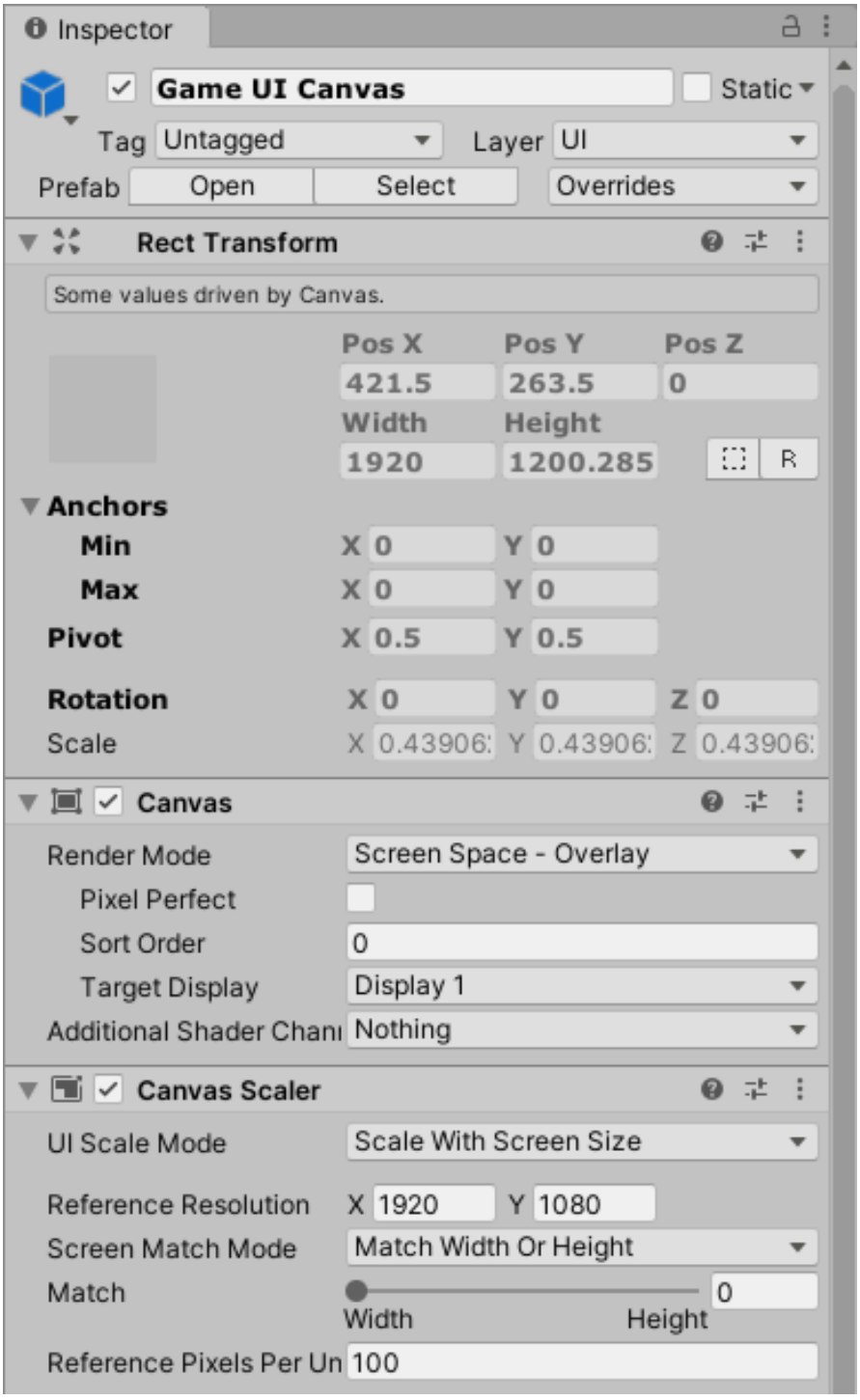


GameOver.cs Script

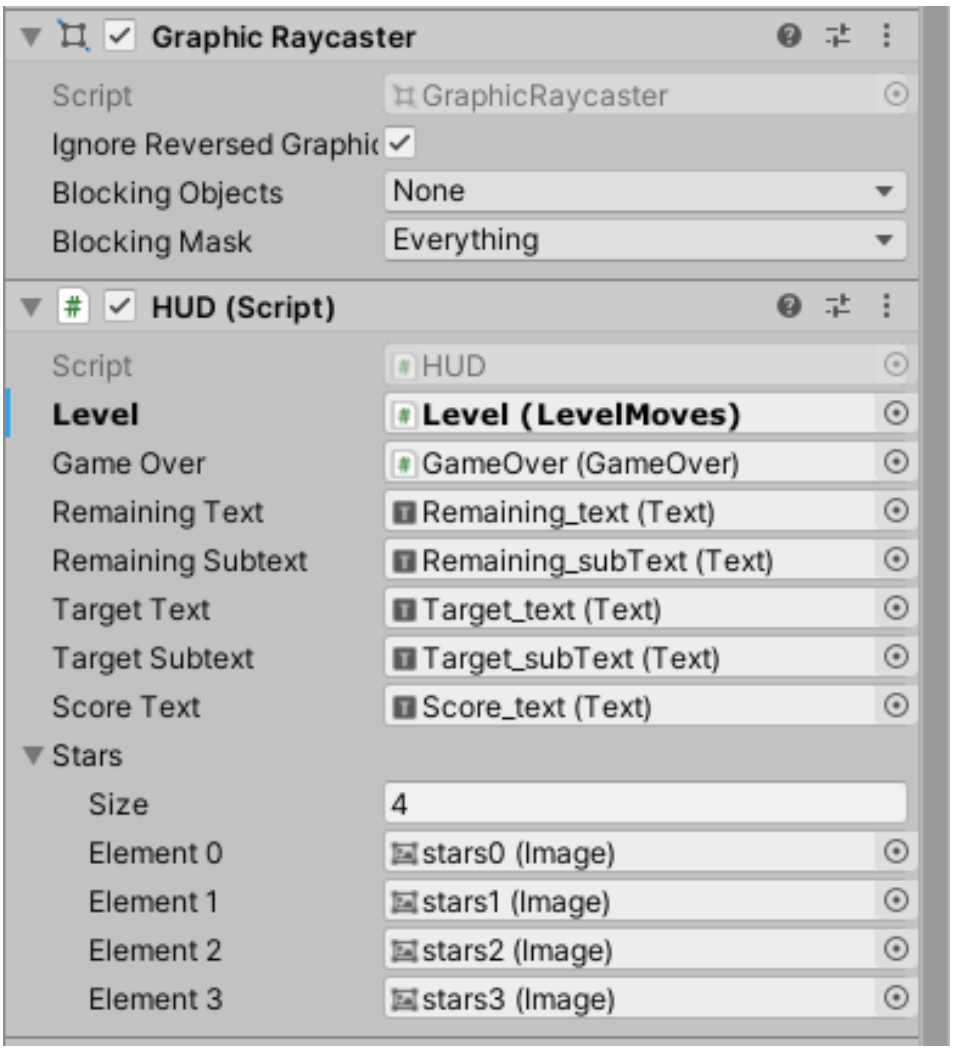
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GameOver : MonoBehaviour
{
    public GameObject screenParent;
    public GameObject scoreParent;
    public Text loseText;
    public Text scoreText;
    public Image[] stars;
    private Animator animator;

    void Start()
    {
        screenParent.SetActive(false);
        for (int i = 0; i < stars.Length; i++)
        {
            stars[i].enabled = false;
        }
        animator = GetComponent<Animator>();
    }

    public void ShowLose()
    {
        screenParent.SetActive(true);
        scoreParent.SetActive(false);
        loseText.enabled = true;
        if (animator)
        {
            animator.Play("GameOverShow");
        }
    }

    public void ShowWin(int score, int starCount)
    {
        screenParent.SetActive(true);
        scoreParent.SetActive(true);
        loseText.enabled = false;
        scoreText.text = score.ToString();
        scoreText.enabled = false;
        if (animator)
        {
            animator.Play("GameOverShow");
        }
        StartCoroutine(ShowWinCoroutine(starCount));
    }
```

```
    private IEnumerator ShowWinCoroutine(int starCount)
    {
        yield return new WaitForSeconds(0.5f);
        if (starCount < stars.Length)
        {
            for (int i = 0; i <= starCount; i++)
            {
                stars[i].enabled = true;
                if (i > 0)
                {
                    stars[i - 1].enabled = false;
                }
                yield return new WaitForSeconds(0.5f);
            }
        }
        scoreText.enabled = true;
    }

    public void OnReplayClicked()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }

    public void OnDoneClicked()
    {
        SceneManager.LoadScene("LevelSelect");
    }
}
```

Game UI Canvas



Game UI Canvas Object Continued

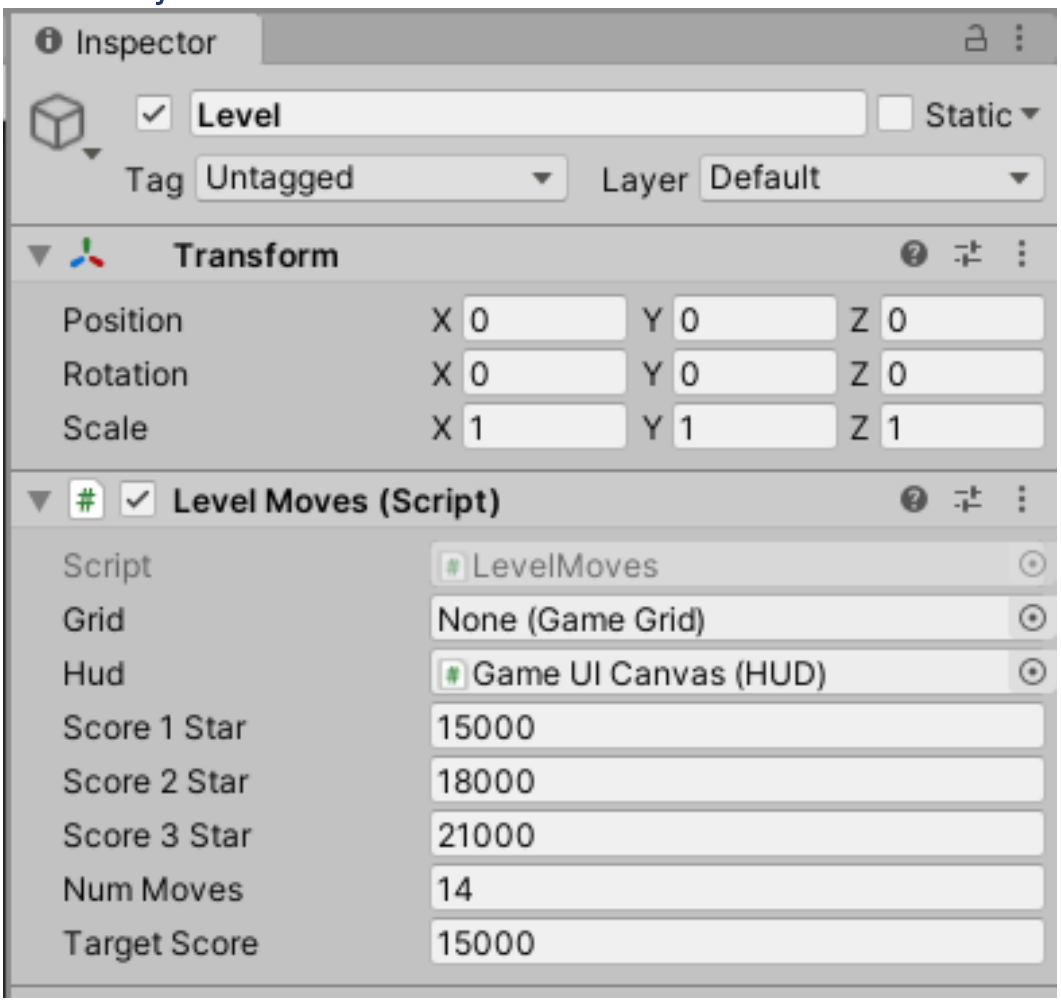


HUD.cs Script

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

public class HUD : MonoBehaviour
{
    public Level level;
    public GameOver gameOver;
    public Text remainingText;
    public Text remainingSubtext;
    public Text targetText;
    public Text targetSubtext;
    public Text scoreText;
    public Image[] stars;
    private int starIndex;
    private bool isGameOver = false;

    void Start()
    {
        for (int i = 0; i < stars.Length; i++)
        {
            if (i == starIndex)
            {
                stars[i].enabled = true;
            }
            else
            {
                stars[i].enabled = false;
            }
        }
    }
    public void SetScore(int score)
    {
        scoreText.text = score.ToString();
        int visibleStar = 0;

        if (score >= level.score1Star && score < level.score2Star)
        {
            visibleStar = 1;
        }
        else if (score >= level.score2Star && score < level.score3Star)
        {
            visibleStar = 2;
        }
        else if (score >= level.score3Star)
        {
            visibleStar = 3;
        }
        for (int i = 0; i < stars.Length; i++)
        {
            if (i == visibleStar)
            {
                stars[i].enabled = true;
```

```
            }
            else
            {
                stars[i].enabled = false;
            }
        }
        starIndex = visibleStar;
    }

    public void SetTarget(int target)
    {
        targetText.text = target.ToString();
    }

    public void SetRemaining(int remaining)
    {
        remainingText.text = remaining.ToString();
    }

    public void SetRemaining(string remaining)
    {
        remainingText.text = remaining;
    }
    public void SetLevelType(Level.LevelType type)
    {
        switch (type)
        {
            case Level.LevelType.MOVES:
                remainingSubtext.text = "moves remaining";
                targetSubtext.text = "target score";
                break;
            case Level.LevelType.OBSTACLE:
                remainingSubtext.text = "moves remaining";
                targetSubtext.text = "dishes remaining";
                break;
            case Level.LevelType.TIMER:
                remainingSubtext.text = "time remaining";
                targetSubtext.text = "target score";
                break;
        }
    }
    public void OnGameWin(int score)
    {
        gameOver.ShowWin(score, starIndex);
        if (starIndex > PlayerPrefs.GetInt(SceneManager.GetActiveScene().name, 0))
        {
            PlayerPrefs.SetInt(SceneManager.GetActiveScene().name, starIndex);
        }
    }

    public void OnGameLose()
    {
        gameOver.ShowLose();
    }
}
```

Level Object



Level.cs Script

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

public class Level : MonoBehaviour
{
    public enum LevelType
    {
        TIMER,
        OBSTACLE,
        MOVES,
    };

    public GameGrid grid;
    public HUD hud;
    public int score1Star;
    public int score2Star;
    public int score3Star;
    protected LevelType type;
    public LevelType Type
    {
        get { return type; }
    }
    protected int currentScore;
    protected bool didWin;

    void Start()
    {
        hud.SetScore(currentScore);
    }

    public virtual void GameWin()
    {
        didWin = true;
        grid.GameOver();
        StartCoroutine(WaitForGridFill());
    }

    public virtual void GameLose()
    {
        didWin = false;
        grid.GameOver();
        StartCoroutine(WaitForGridFill());
    }

    public virtual void OnMove()
    {

    }

    public virtual void OnPieceCleared(GamePiece piece)
    {
        currentScore += piece.score;
        hud.SetScore(currentScore);
    }
```

```
    protected virtual IEnumerator WaitForGridFill()
    {
        while (grid.IsFilling)
        {
            yield return 0;
        }

        if (didWin && !grid.IsFilling)
        {
            hud.OnGameWin(currentScore);
        } else
        {
            hud.OnGameLose();
        }
    }
}
```

LevelMoves.cs Script

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

public class LevelMoves : Level
{
    public int numMoves;
    public int targetScore;
    private int movesUsed = 0;

    void Start()
    {
        type = LevelType.MOVES;
        hud.SetLevelType(type);
        hud.SetScore(currentScore);
        hud.SetTarget(targetScore);
        hud.SetRemaining(numMoves);
    }

    public override void OnMove()
    {
        base.OnMove();
        movesUsed++;
        hud.SetRemaining(numMoves - movesUsed);
        if (numMoves - movesUsed == 0)
        {
            if (currentScore >= targetScore)
            {
                GameWin();
            }
            else
            {
                GameLose();
            }
        }
    }
}
```

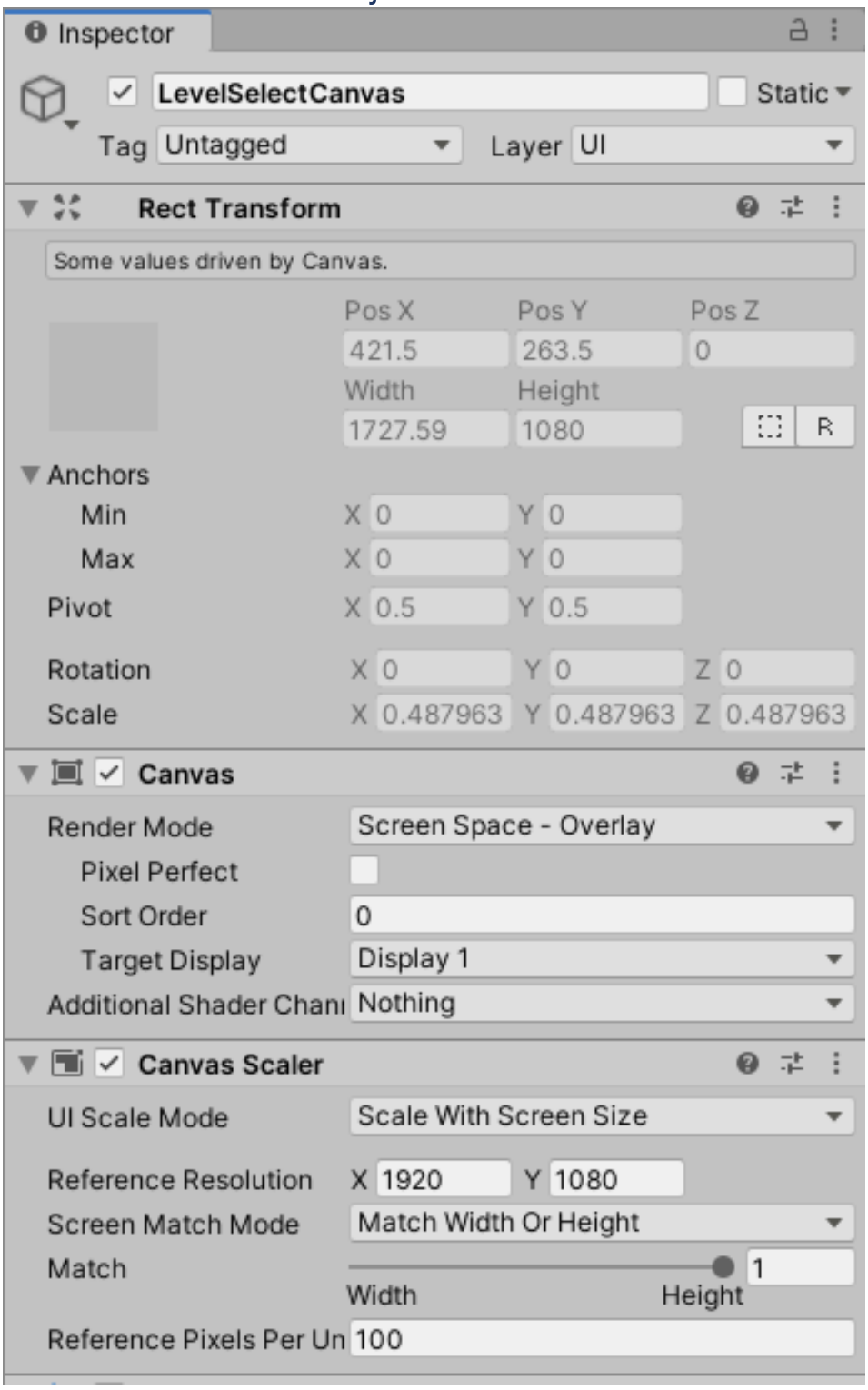
LevelObstacles.cs Script

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

public class LevelObstacles : Level
{
    public int numMoves;
    public GameGrid.PieceType[] obstacleTypes;
    private int movesUsed = 0;
    private int numObstaclesLeft;

    void Start()
    {
        type = LevelType.OBSTACLE;
        for (int i = 0; i < obstacleTypes.Length; i++)
        {
            numObstaclesLeft += grid.GetPiecesOfType(obstacleTypes[i]).Count;
        }
        hud.SetLevelType(type);
        hud.SetScore(currentScore);
        hud.SetTarget(numObstaclesLeft);
        hud.SetRemaining(numMoves);
    }

    public override void OnMove()
    {
        base.OnMove();
        movesUsed++;
        hud.SetRemaining(numMoves - movesUsed);
        if (numMoves - movesUsed == 0 && numObstaclesLeft > 0)
        {
            GameLose();
        }
    }

    public override void OnPieceCleared(GamePiece piece)
    {
        base.OnPieceCleared(piece);
        for (int i = 0; i < obstacleTypes.Length; i++)
        {
            if (obstacleTypes[i] == piece.Type)
            {
                numObstaclesLeft--;
                hud.SetTarget(numObstaclesLeft);
                if (numObstaclesLeft == 0)
                {
                    currentScore += 1000 * (numMoves - movesUsed);
                    hud.SetScore(currentScore);
                    GameWin();
                }
            }
        }
    }
}
```

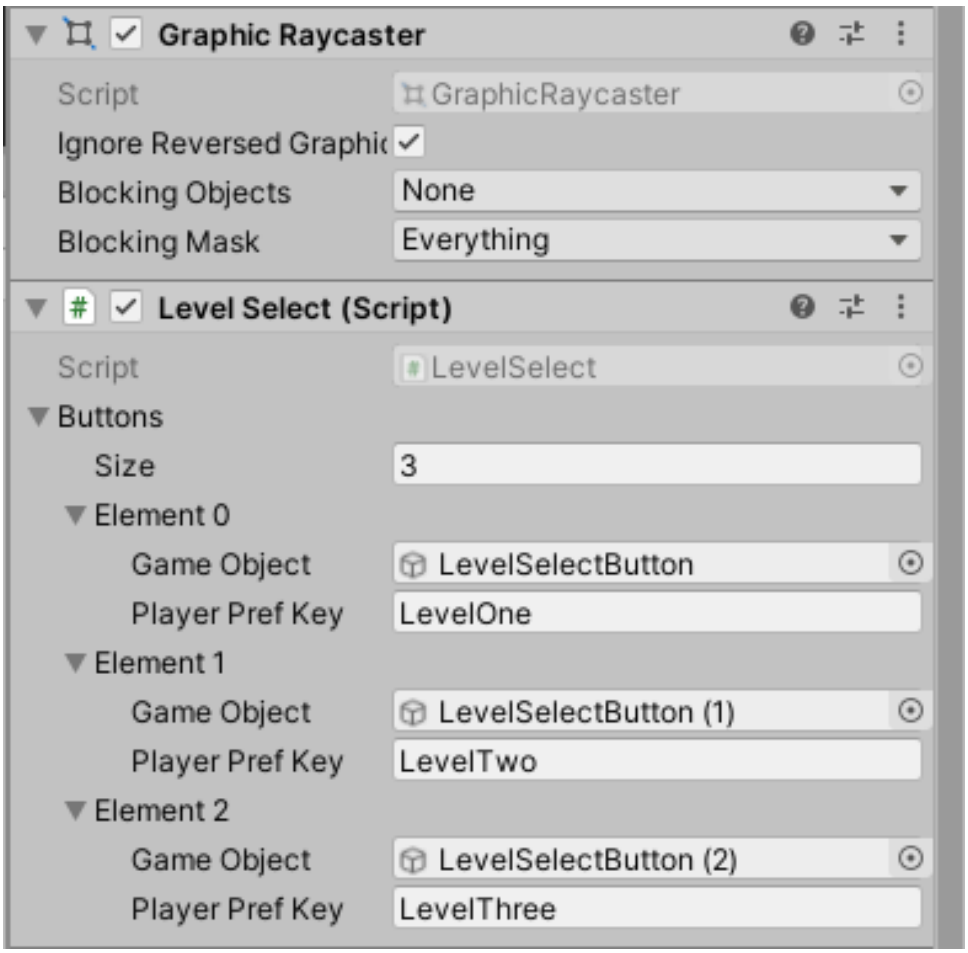
LevelSelect.cs Script

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

public class LevelSelect : MonoBehaviour
{
    [System.Serializable]
    public struct ButtonPlayerPrefs
    {
        public GameObject gameObject;
        public string playerPrefKey;
    }
    public ButtonPlayerPrefs[] buttons;

    void Start()
    {
        for (int i = 0; i < buttons.Length; i++)
        {
            int score = PlayerPrefs.GetInt(buttons[i].playerPrefKey, 0);
            for (int starIndex = 1; starIndex <= 3; starIndex++)
            {
                Transform star = buttons[i].gameObject.transform.Find(
                            "star" + starIndex);
                if (starIndex <= score)
                {
                    star.gameObject.SetActive(true);
                }
                else
                {
                    star.gameObject.SetActive(false);
                }
            }
        }
    }

    public void OnButtonPress(string levelName)
    {
        SceneManager.LoadScene(levelName);
    }
}
```

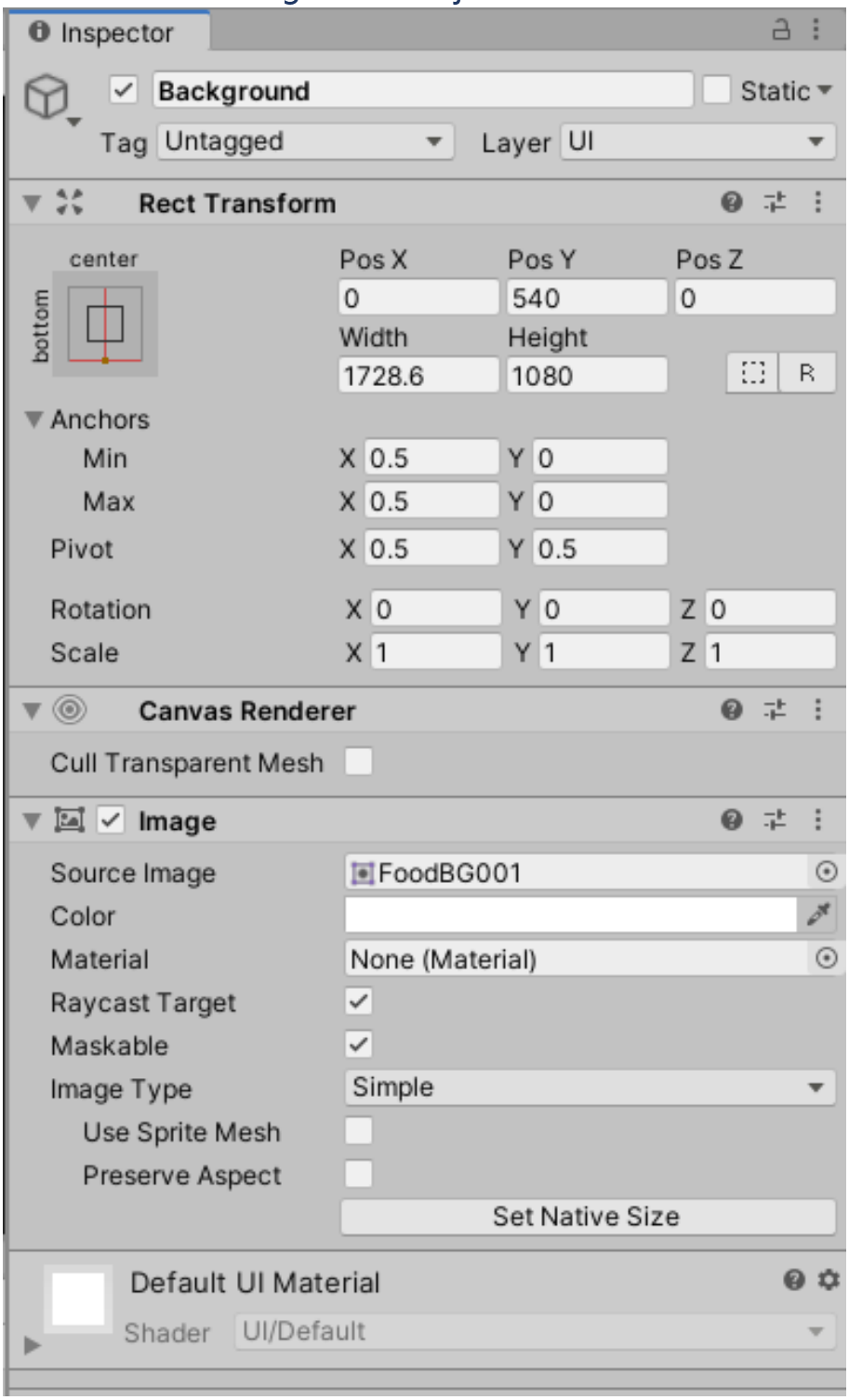
Level Select Canvas Object



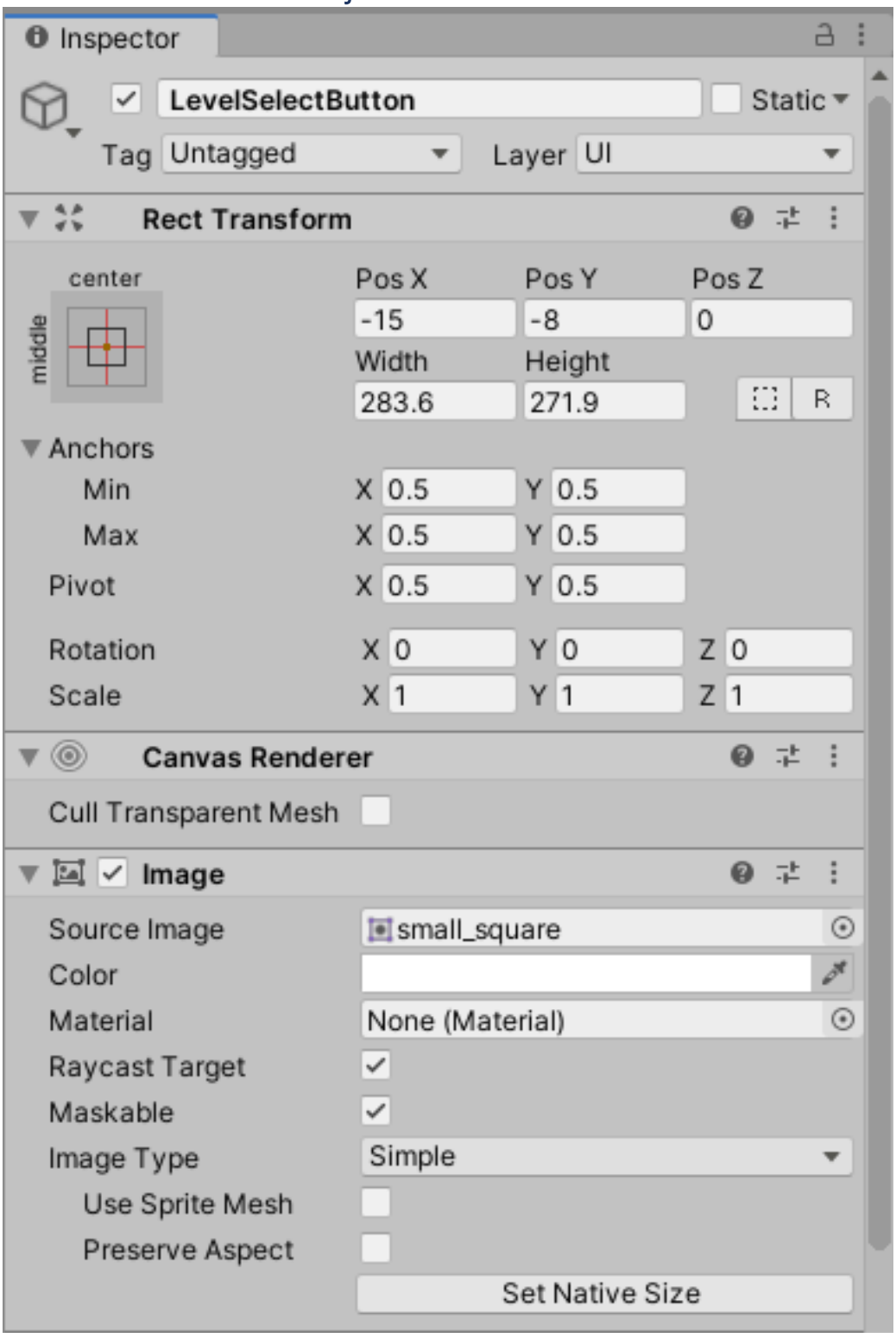
Level Select Canvas Object Continued



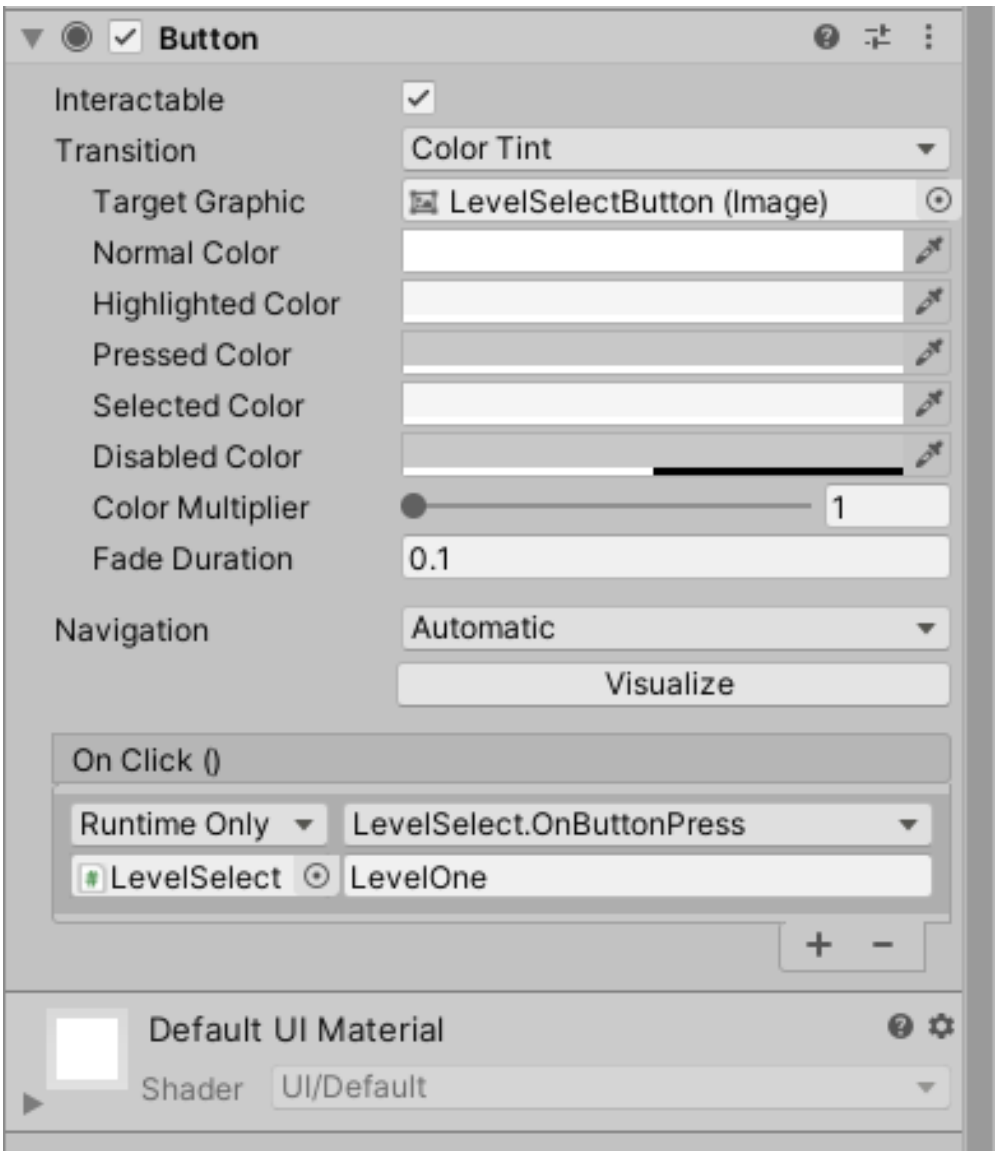
Level Select Background Object



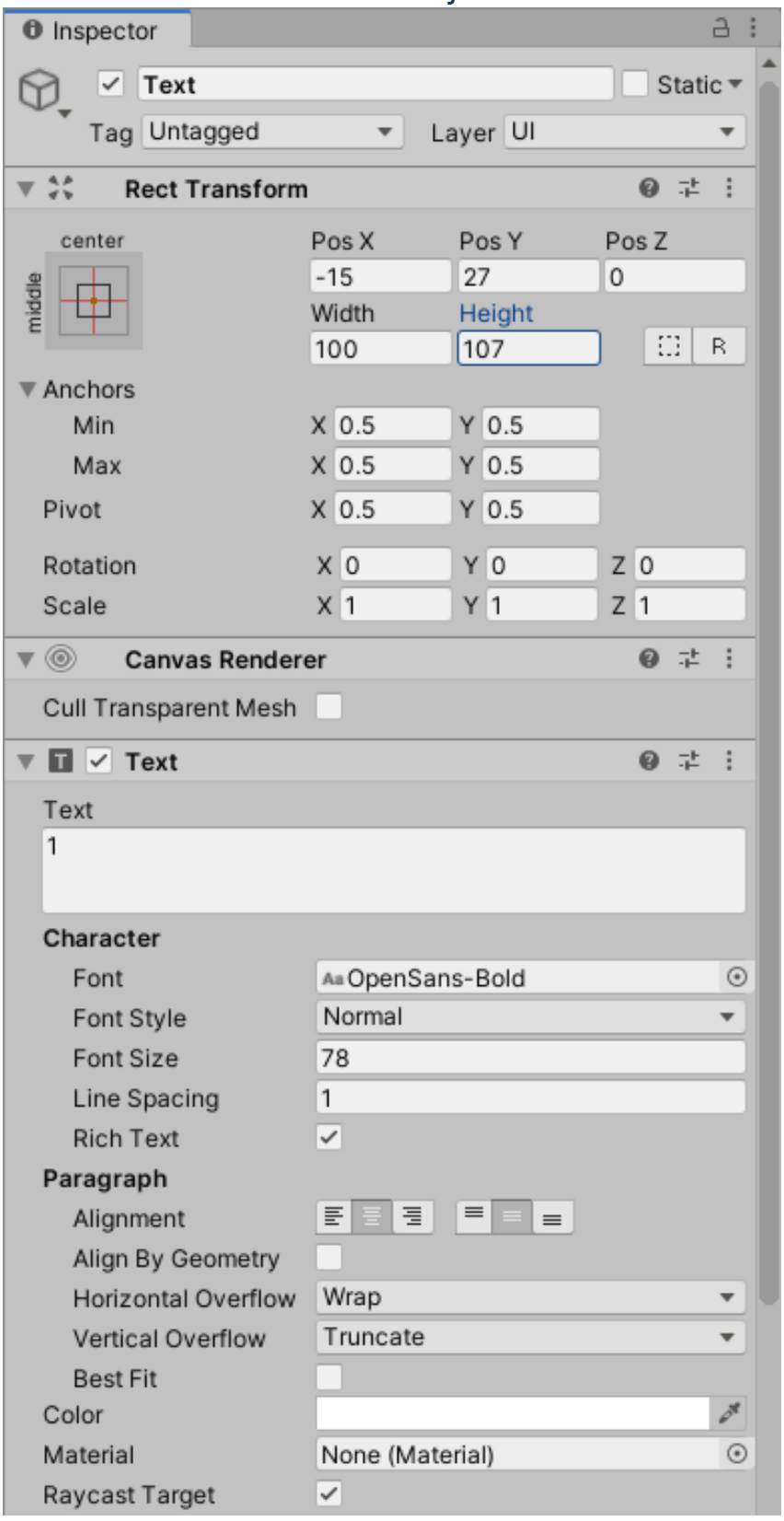
Level Select Button Objects



Level Select Buttons Continued



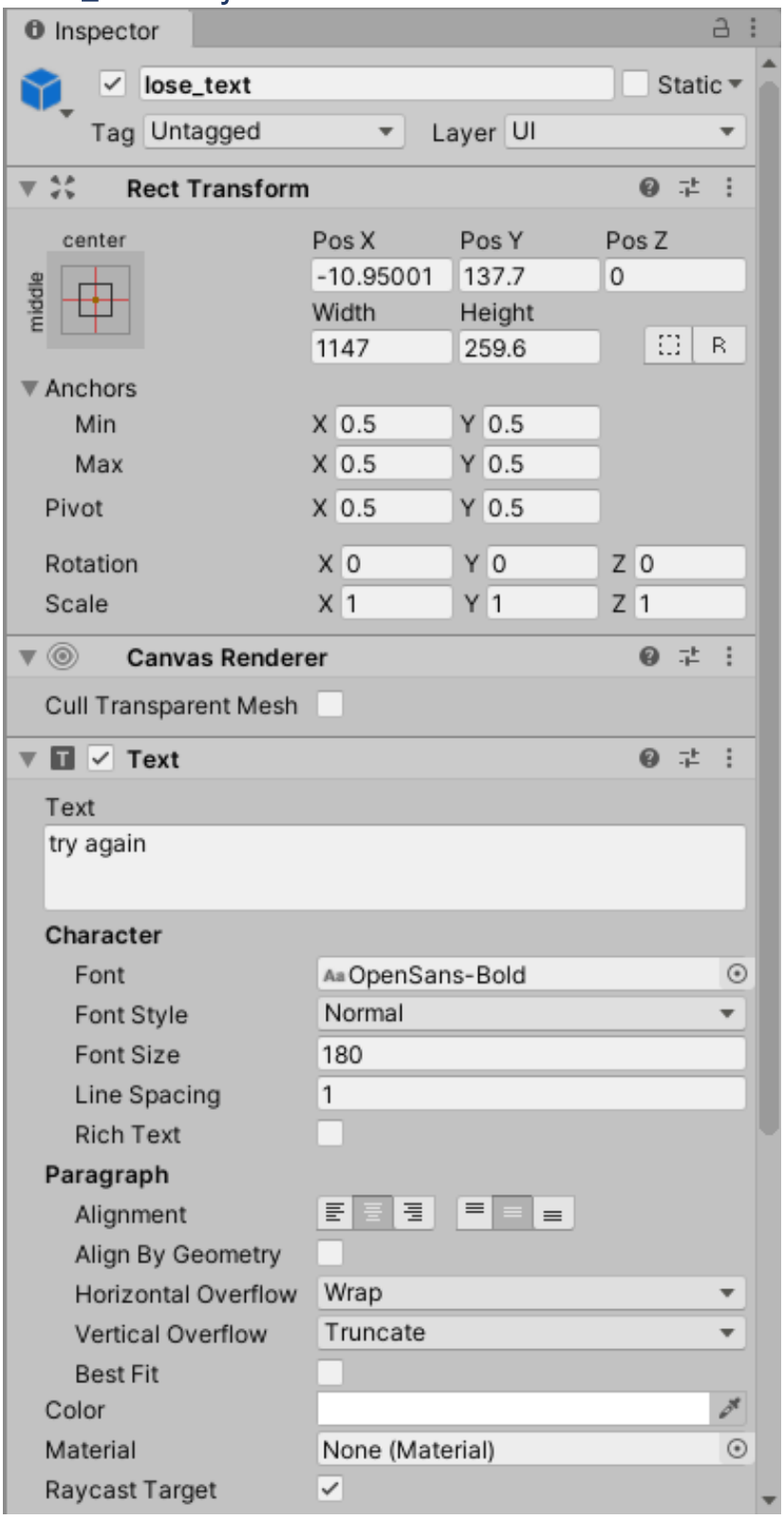
Level Select Button Text Objects



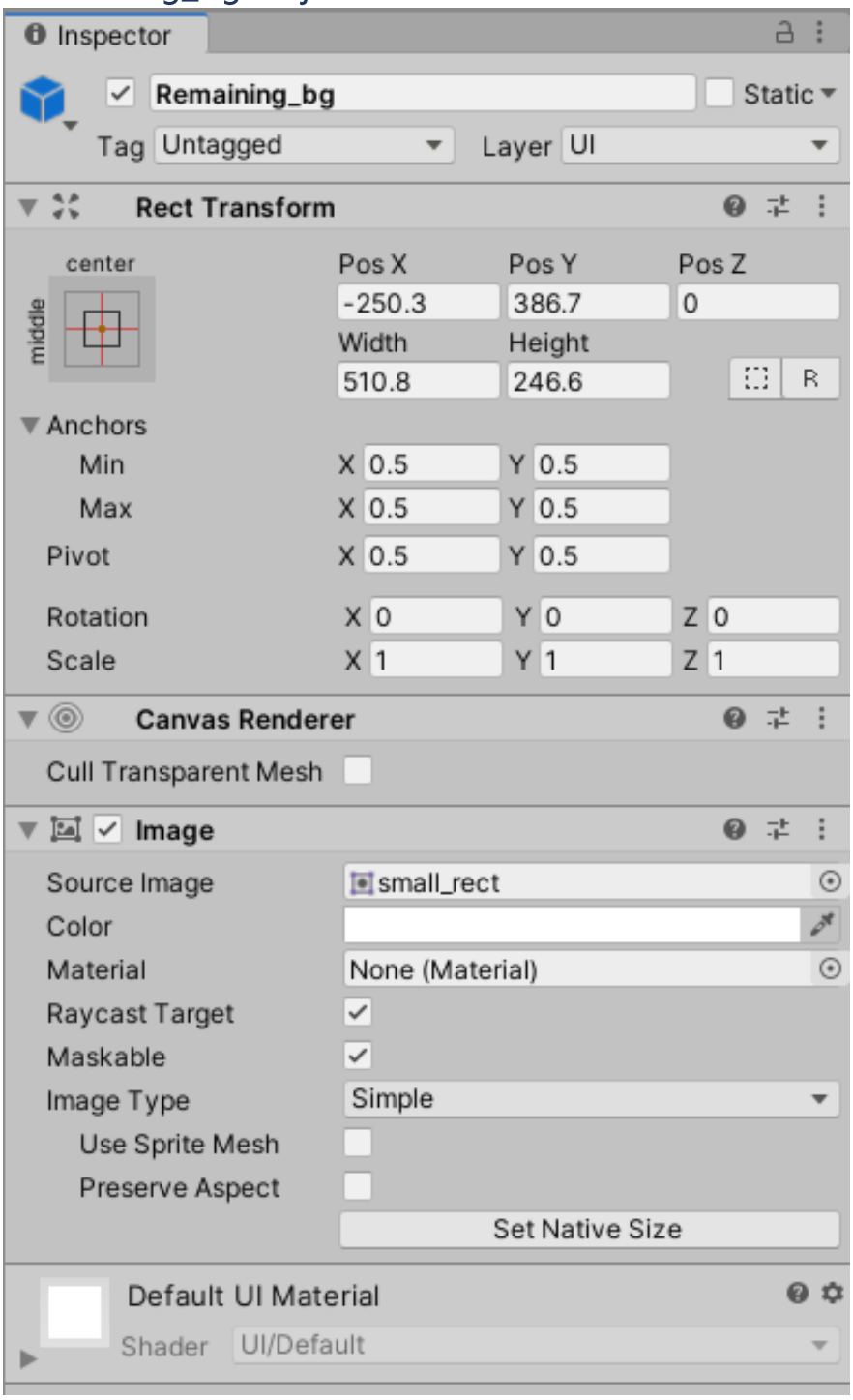
LevelTimer.cs Script

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

public class LevelTimer : Level
{
    public int timeInSeconds;
    public int targetScore;
    private float timer;
    private bool timeOut = false;

    void Start()
    {
        type = LevelType.TIMER;
        hud.SetLevelType(type);
        hud.SetScore(currentScore);
        hud.SetTarget(targetScore);
        hud.SetRemaining(string.Format("{0}:{1:00}",
            timeInSeconds / 60, timeInSeconds % 60)
            );
    }

    void Update()
    {
        if (!timeOut)
        {
            timer += Time.deltaTime;
            hud.SetRemaining(string.Format("{0}:{1:00}",
                (int)Mathf.Max((timeInSeconds - timer) / 60, 0),
                (int)Mathf.Max((timeInSeconds - timer) % 60, 0))
                );
            if (timeInSeconds - timer <= 0)
            {
                if (currentScore >= targetScore)
                {
                    GameWin();
                }
                else
                {
                    GameLose();
                }
                timeOut = true;
            }
        }
    }
}
```

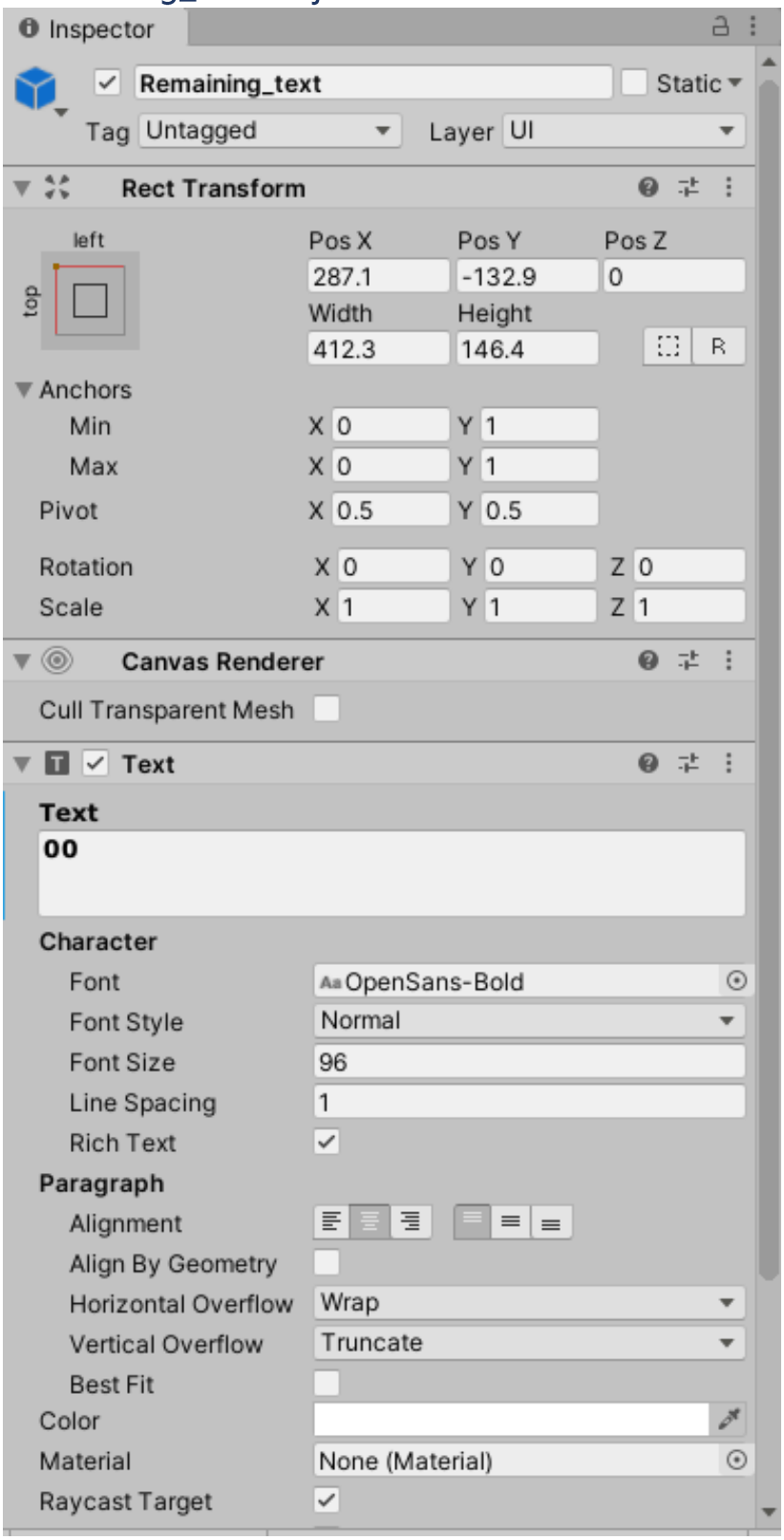
Lose_text Object



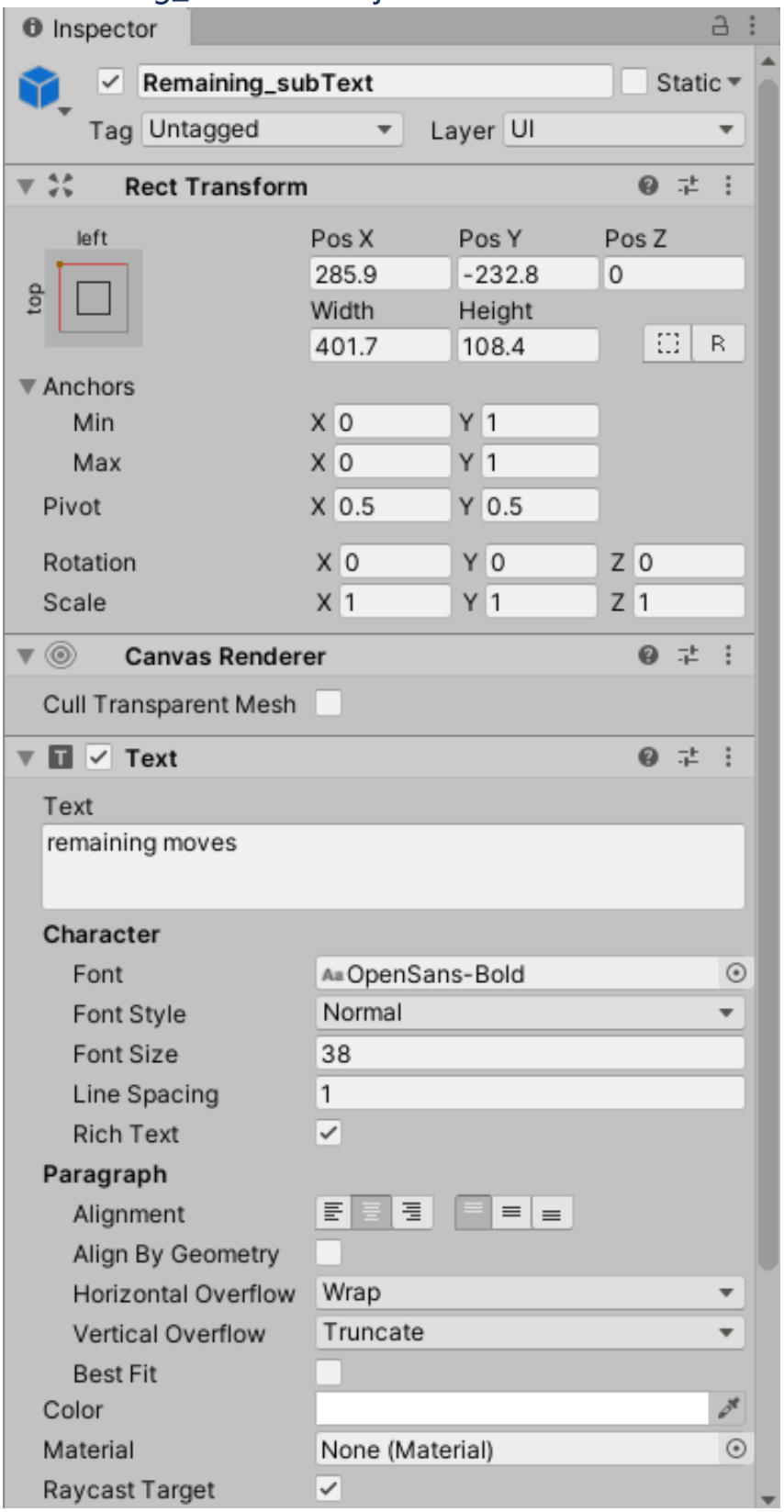
Remaining_bg Object



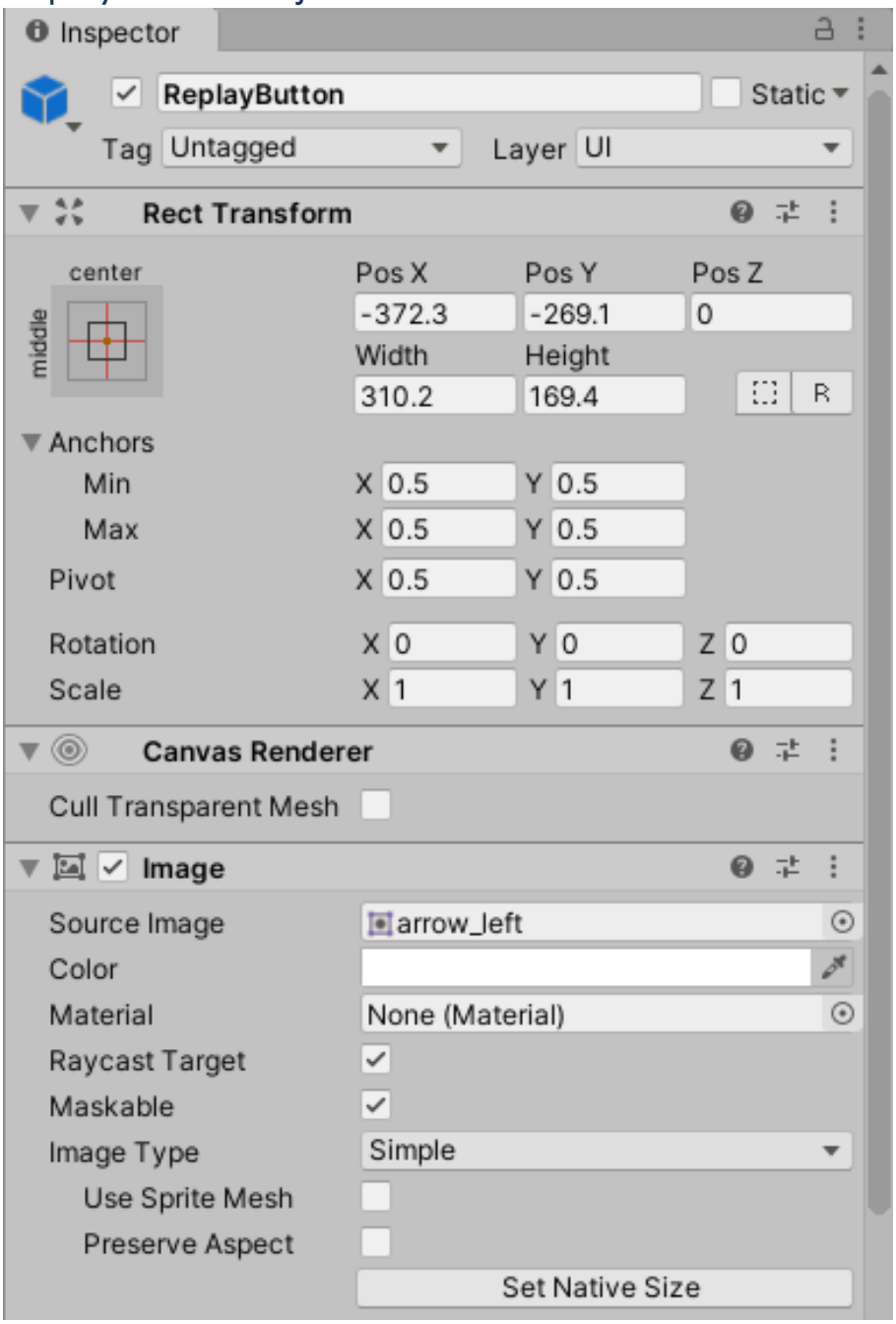
Remaining_text object



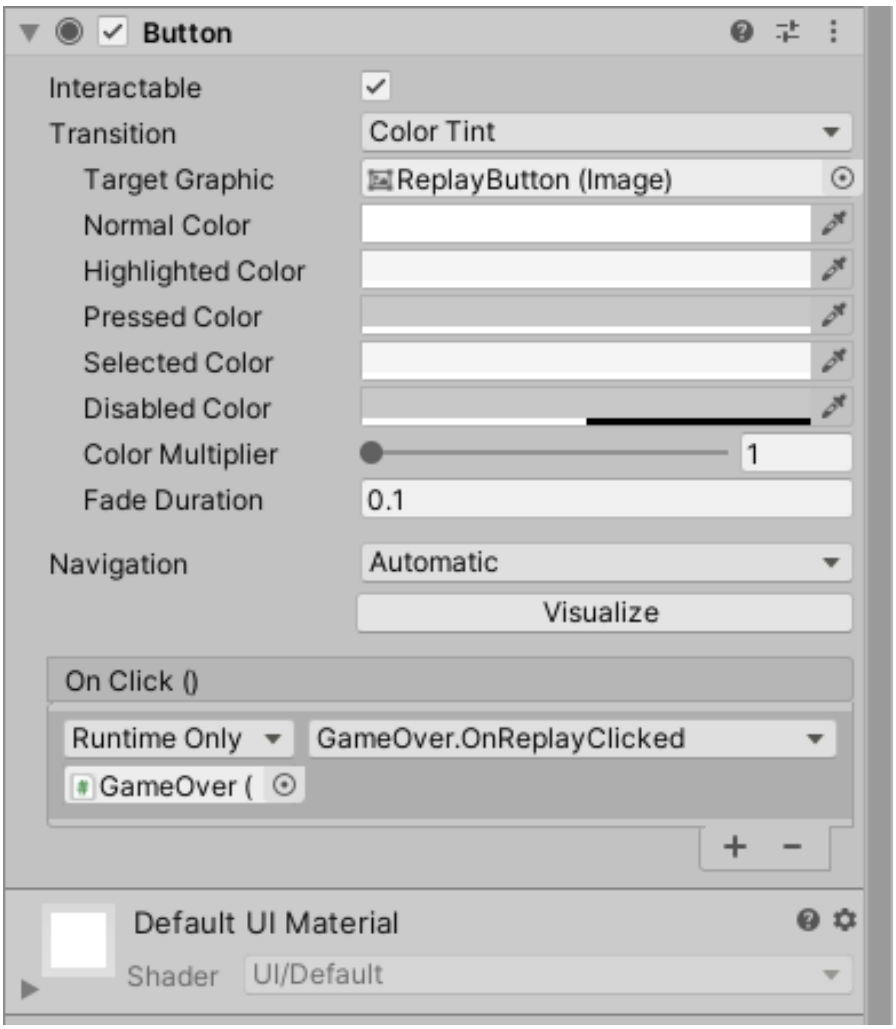
Remaining_subText Object



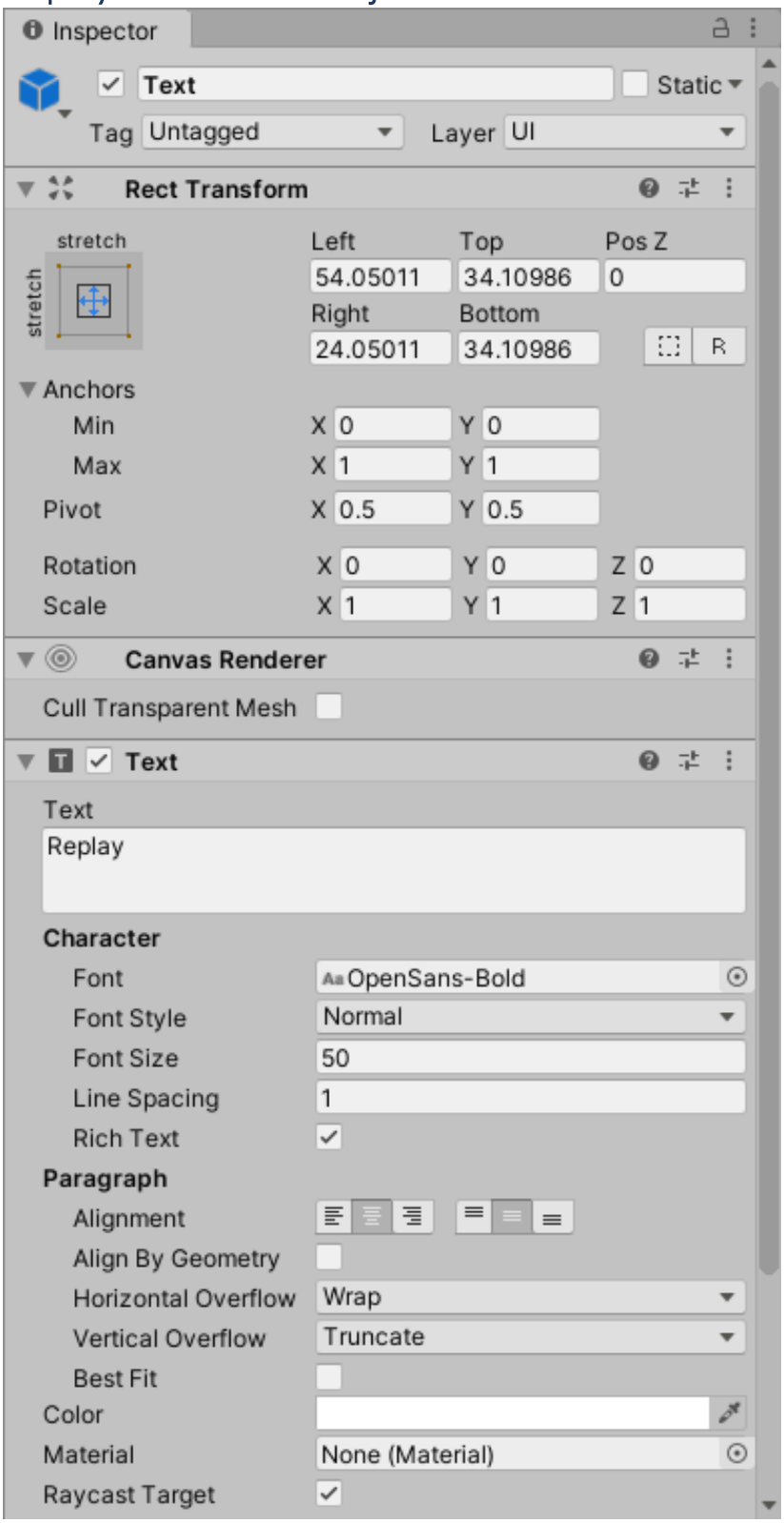
ReplayButton Object



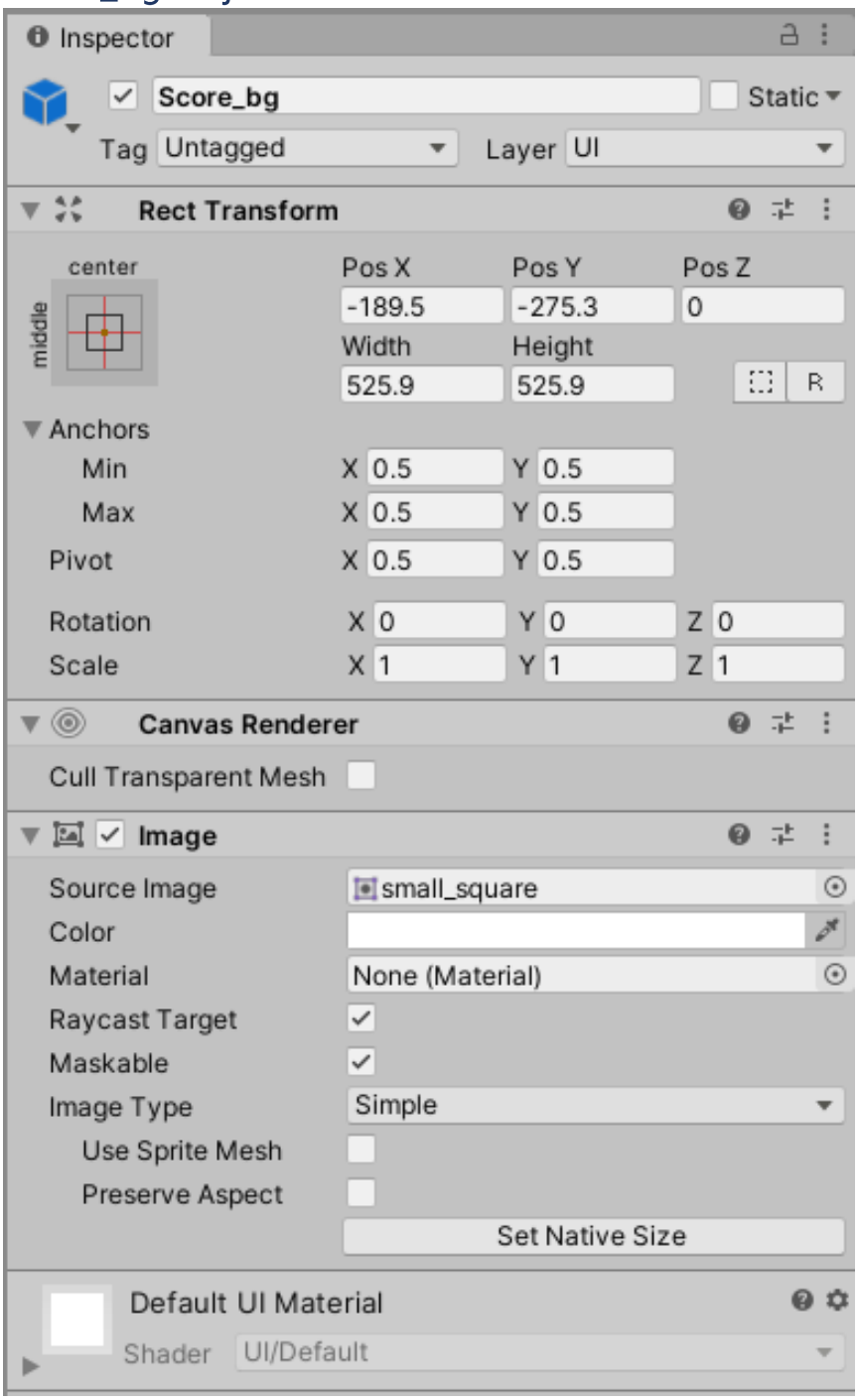
ReplayButton Continued



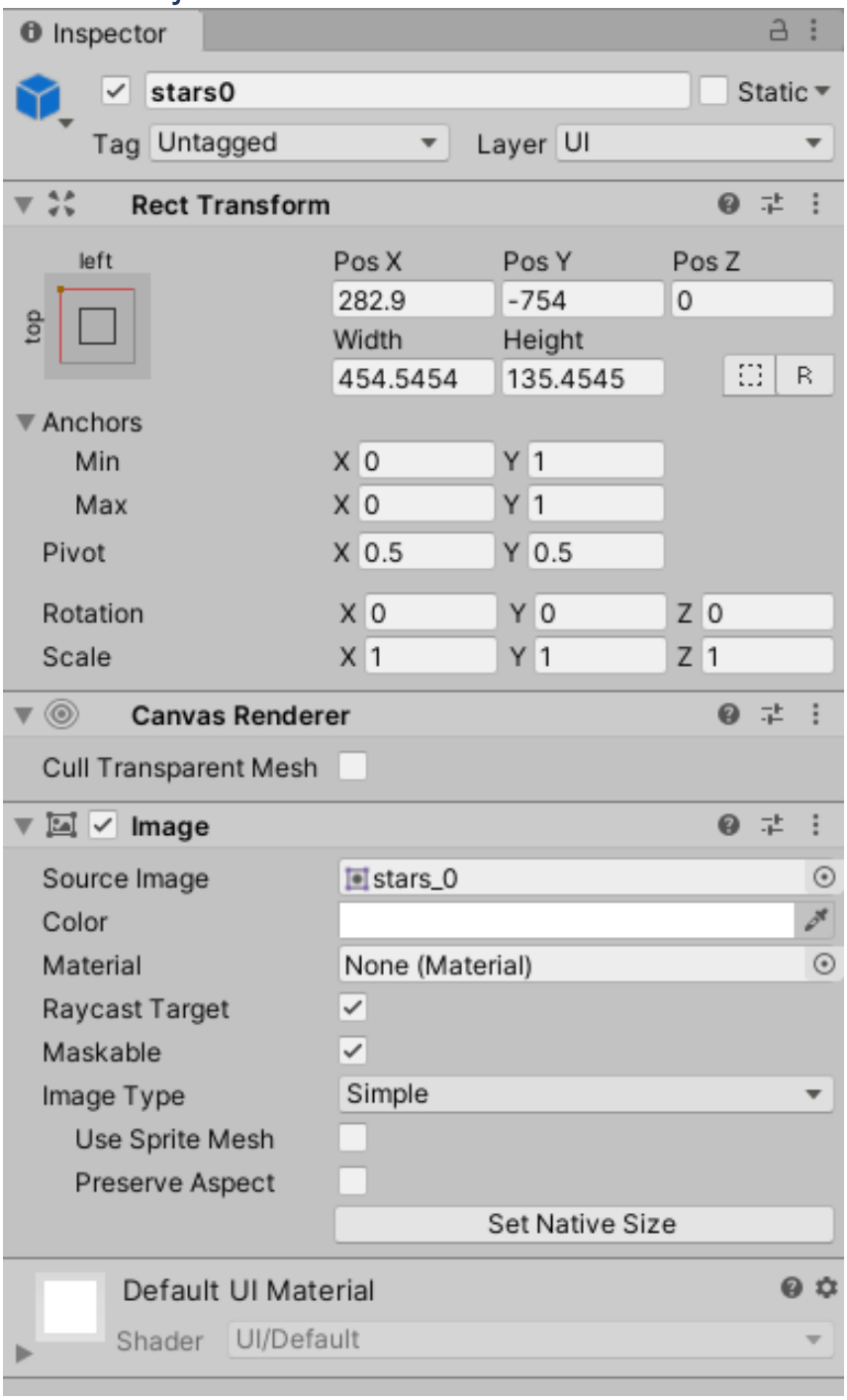
ReplayButton Text Object



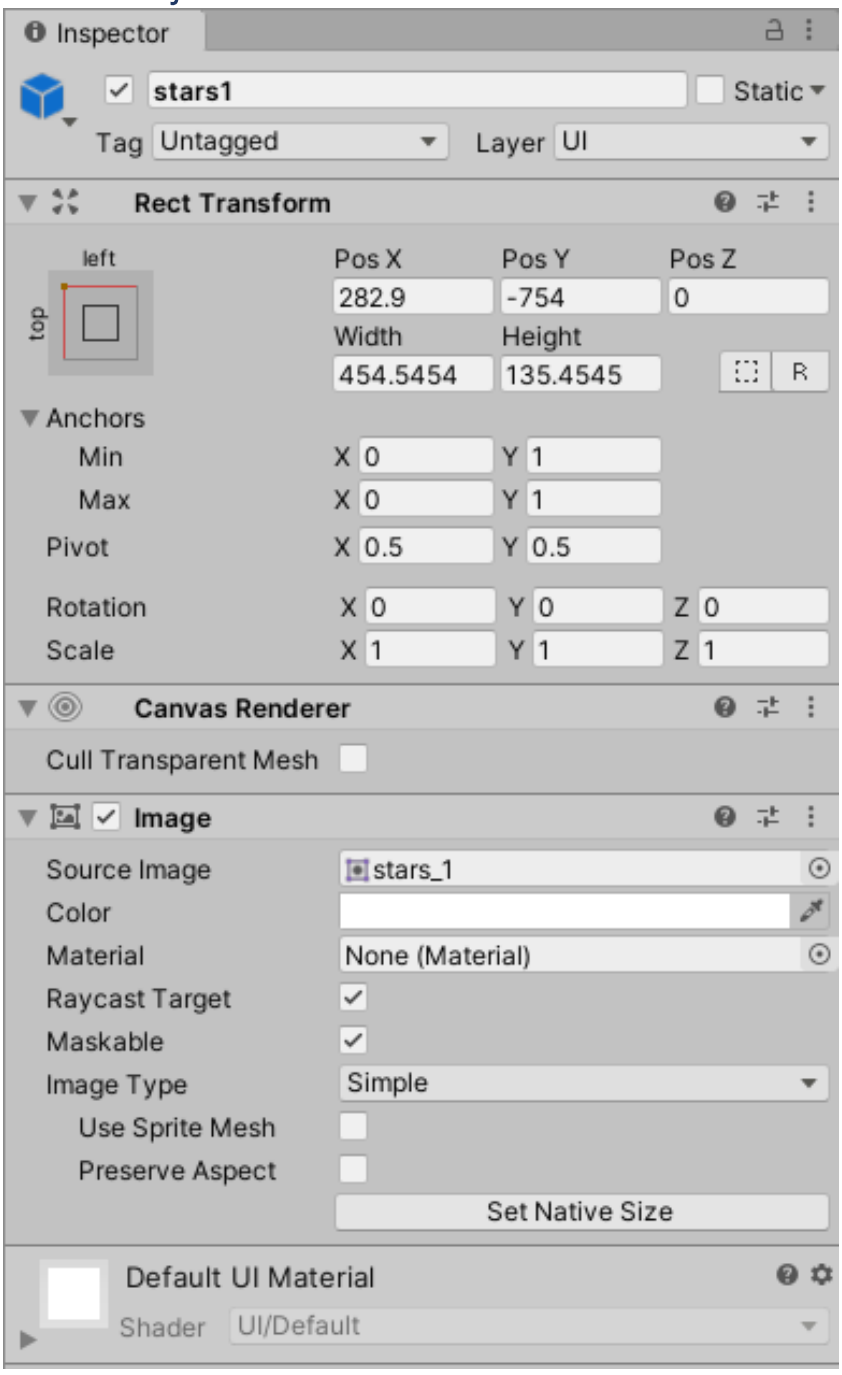
Score_bg Object



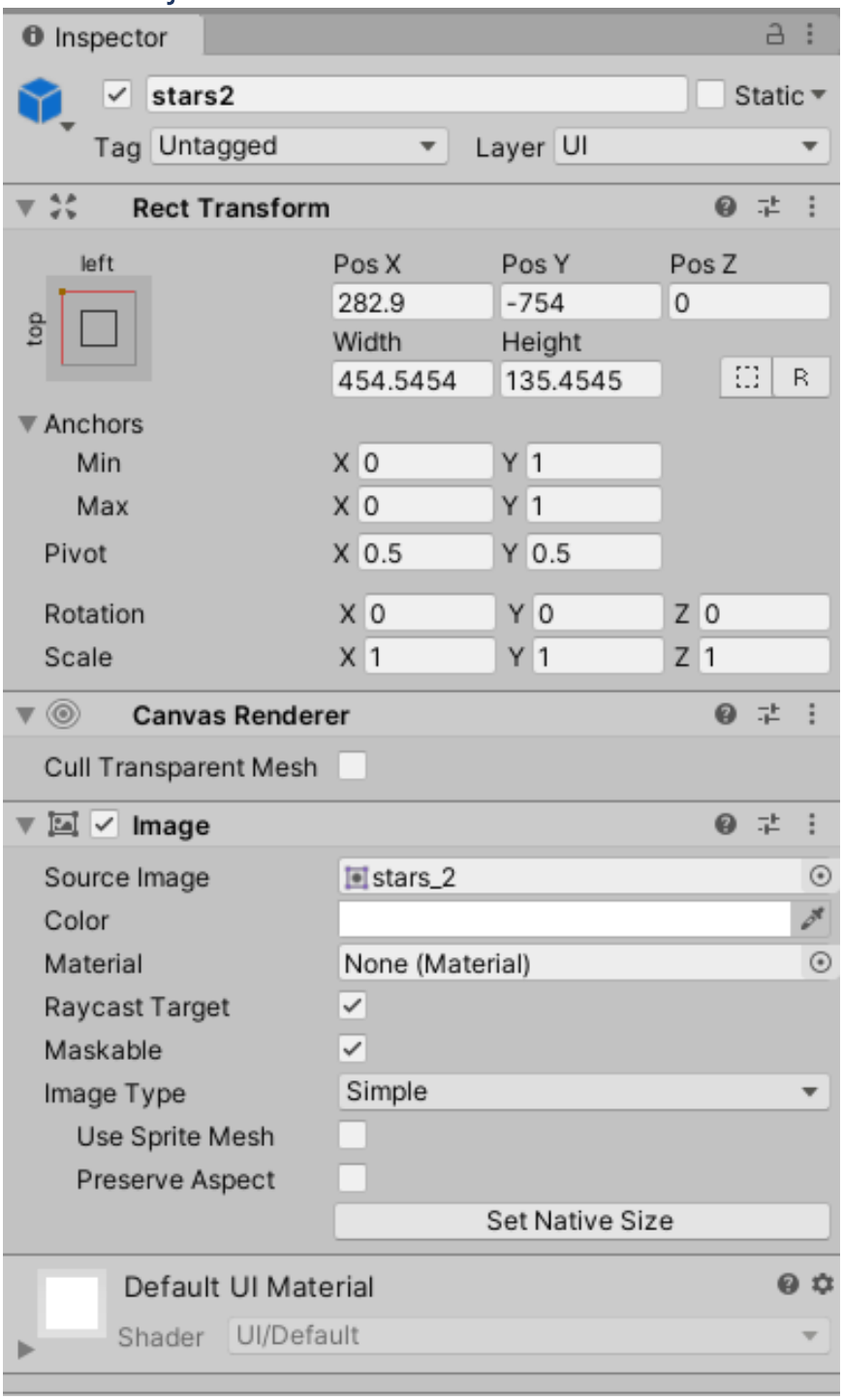
Stars0 Object



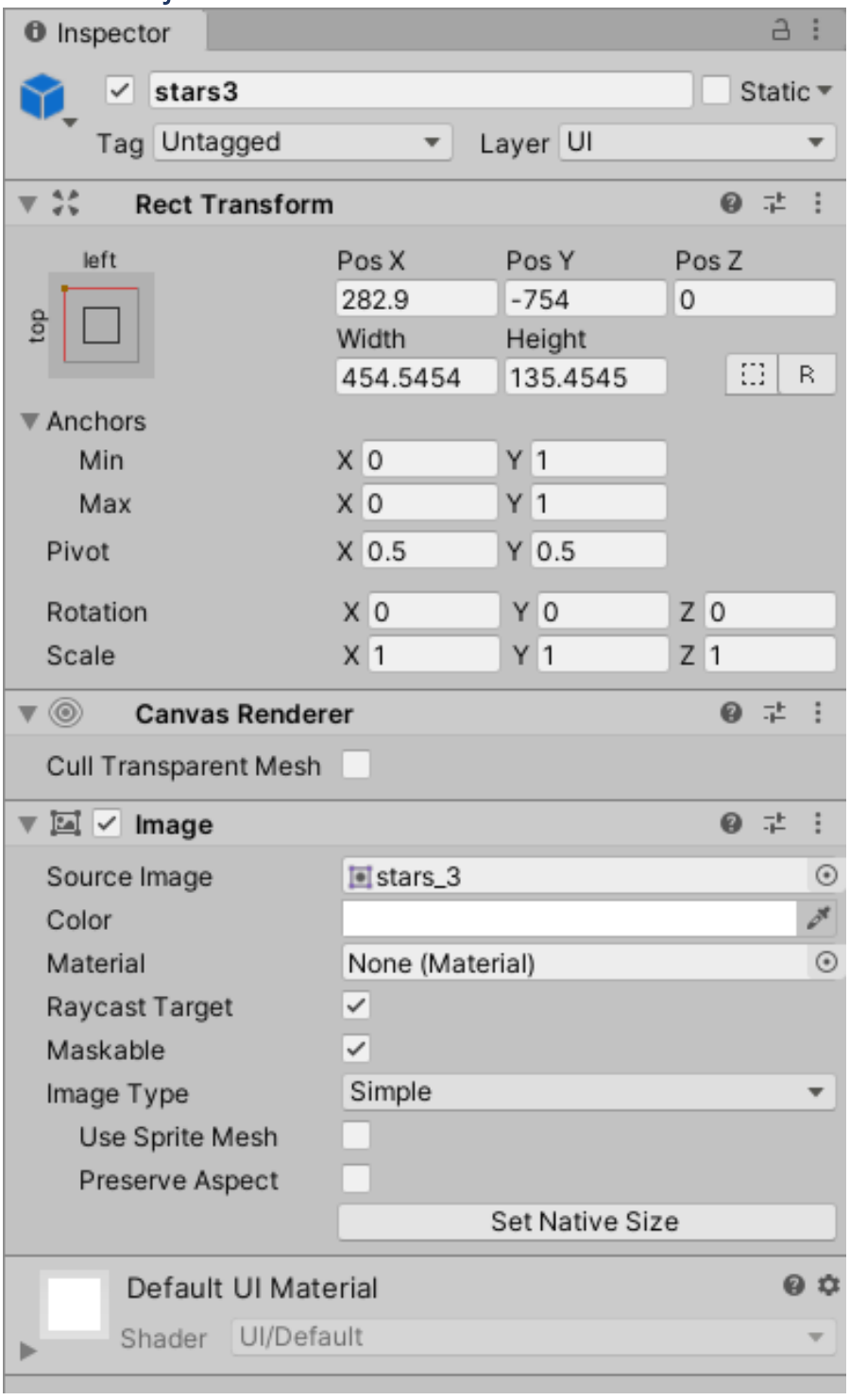
Stars1 Object



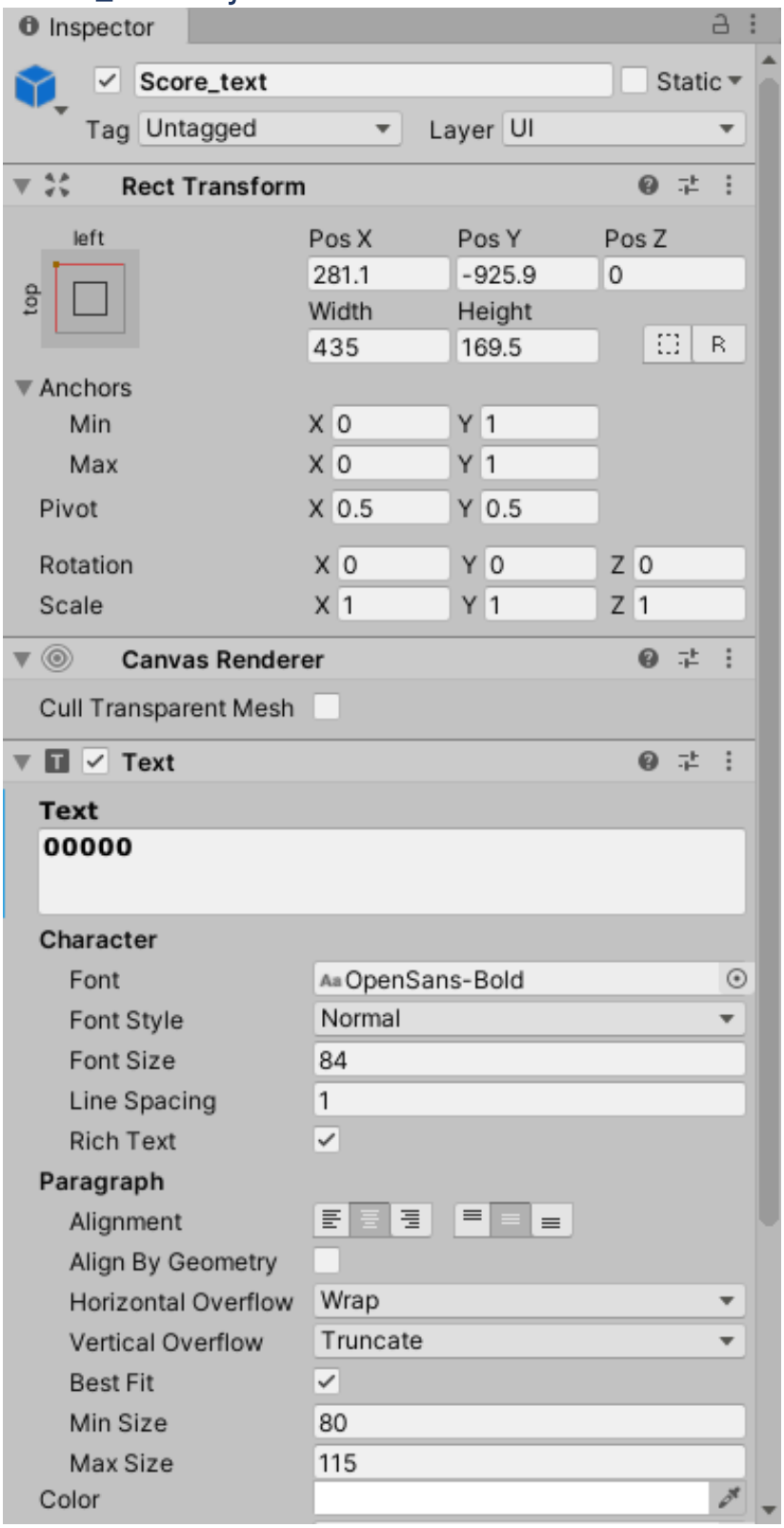
Stars2 Object



Stars3 Object



Score_text object



Target_subText Object