# CS 512 Assignment 1

First-person Shooter `(FPS)` 'Serious Sam' clone based on `Unity3d`

## Group Members:

`Yi Ren` (002269013)  `Wentao Lu` (002276355)

---

## Dependency

- Microsoft .NET Framework
- Unity3d
- Jetbrains Rider

---

## Video demo

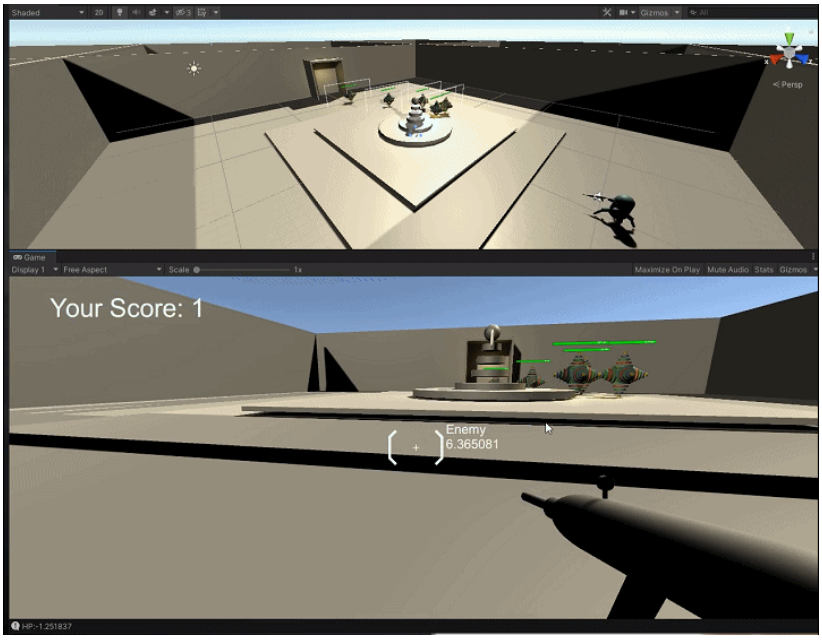
Please check Demo - Assignment1 for more information.

---

## Basic functions

### Part 1 - Basic camera control for movement and look.

In this part, we invoked `Input.GetAxis()` function to get mouse movement and keyboard input, therefore, we can implement camera rotation and player movements according to system input.




- **a. mouse look – able to look around the scene in XY directions using mouse**

▼ Code

```
private void Update()
        {
            var mouseX = Input.GetAxis("Mouse X") * Time.deltaTime *
MouseSensitive;
            var mouseY = Input.GetAxis("Mouse Y") * Time.deltaTime *
MouseSensitive;
            player.Rotate(Vector3.up *  mouseX);
            _xRotation -= mouseY;
            transform.localRotation = Quaternion.Euler(_xRotation, 0f, 0f);
        }
```

- **b. 15% WASD keyboard move – able to move in XZ plane using keypad**

▼ Code

```
 private void Update()
        {
            _movement = transform.right * Input.GetAxis("Horizontal") +
transform.forward * Input.GetAxis("Vertical");
            _velocity.y += gravity * Time.deltaTime;

            if (Input.GetButton("Jump"))
            {
                _velocity.y = playerSpeed;
            }
            controller.Move(_velocity * Time.deltaTime + _movement * playerSpeed
/ 100);
        }
```

- **c. 5% Spacebar jump (with gravity) – pressing spacebar gives small boost in y-direction, gravity pulls back**

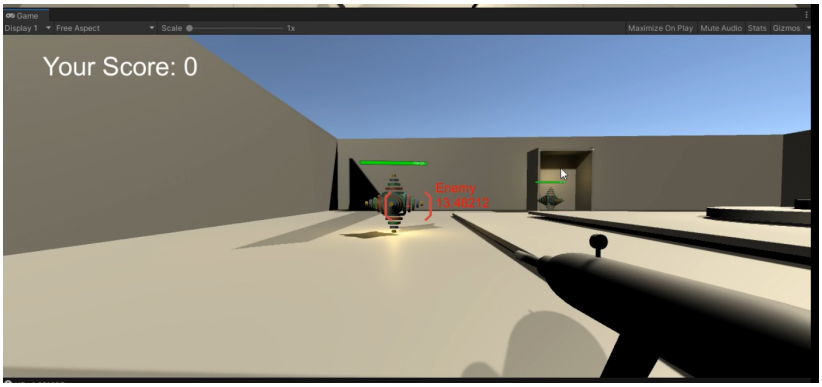
▼ Code

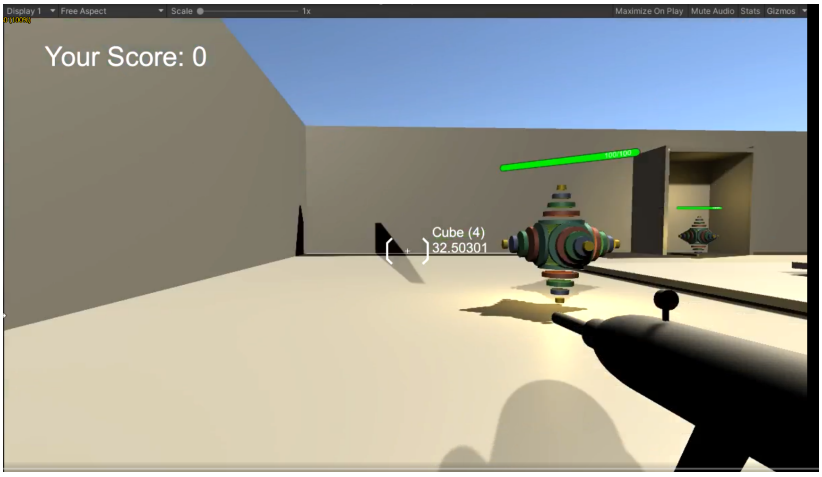
```
 private void Update()
        {
            _movement = transform.right * Input.GetAxis("Horizontal") +
transform.forward * Input.GetAxis("Vertical");
            _velocity.y += gravity * Time.deltaTime;

            if (Input.GetButton("Jump"))
            {
                _velocity.y = playerSpeed;
            }
            controller.Move(_velocity * Time.deltaTime + _movement * playerSpeed
/ 100);
        }
```

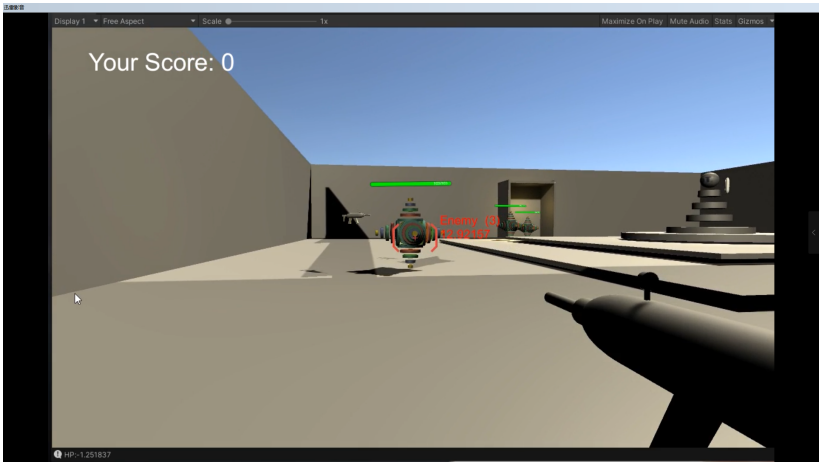
---

## Part 2 - crosshair

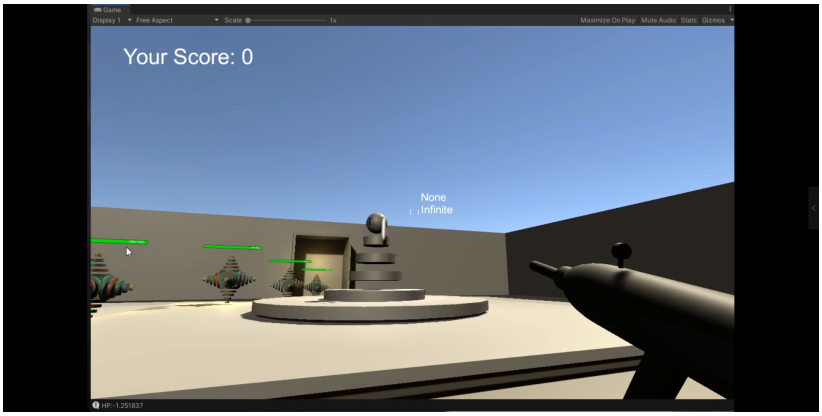
In this part, we draw the crosshair at the center of screen with 2d `textures(Sprite)`, then `Physics.Raycast()` was used to aquire the intersection point information(collide name and distance). In this way, we can draw crosshair of different size and color based on the distance and object type it points to.

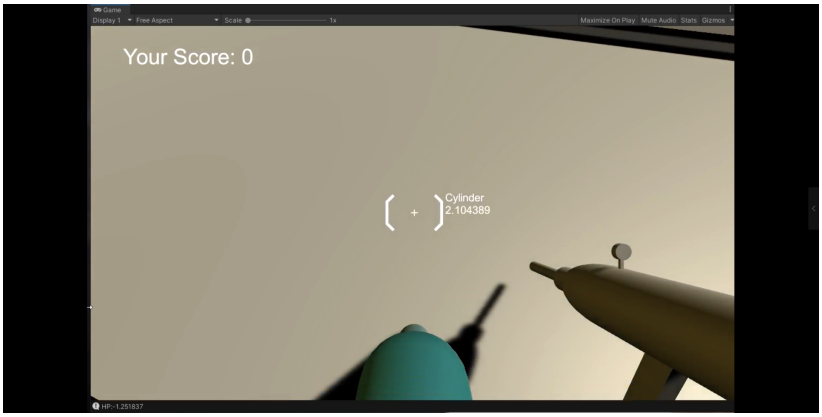


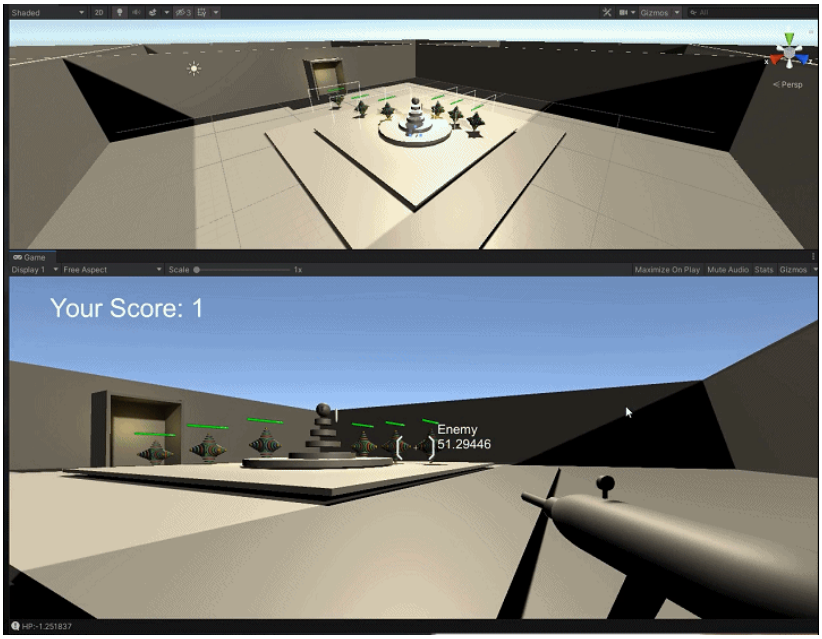




Your Score: 0
Cube (4)
46.56611


Your Score: 0
None
Infinite


Your Score: 0
Cylinder
2.104389




- **a. visible at all time (does not get occluded by objects)**

▼ Code

```
// Draw crosshair (center of screen)
 GUI.DrawTexture(new Rect((Screen.width - crossTexture.width * _crossHairScale) /
2f, (Screen.height - crossTexture.height * _crossHairScale) / 2f,
crossTexture.width * _crossHairScale, crossTexture.height * _crossHairScale),
crossTexture);
// Draw object name label (to right of crosshair)
GUI.Label(new Rect((Screen.width + crossTexture.width * _crossHairScale + 10) /
2f, Screen.height * 0.5f - 40, 100, 80), "Enemy" + "\n" + _distance , styleHint);
// Draw score label (top left of screen)
GUI.Label(new Rect(Screen.width / 20f, Screen.height / 20f, Screen.width / 10f,
Screen.width / 10f),"Your Score: " + _score, styleScore);
```

- **b. cross-shaped (see image above)**

▼ Code

```
public Texture2D crossNormal;
public Texture2D crossAimed;
```

- **c. changes color when over enemy**

▼ Code

```
            if (_colliderName.Contains("Enemy"))
            {
                styleHint.normal.textColor = Color.red;
                crossTexture = crossAimed;
            }else {
```

```
                styleHint.normal.textColor = Color.white;
                crossTexture = crossNormal;
            }
```

- **d. changes distance from camera based on where object it points to is located (see video)**

▼ Code

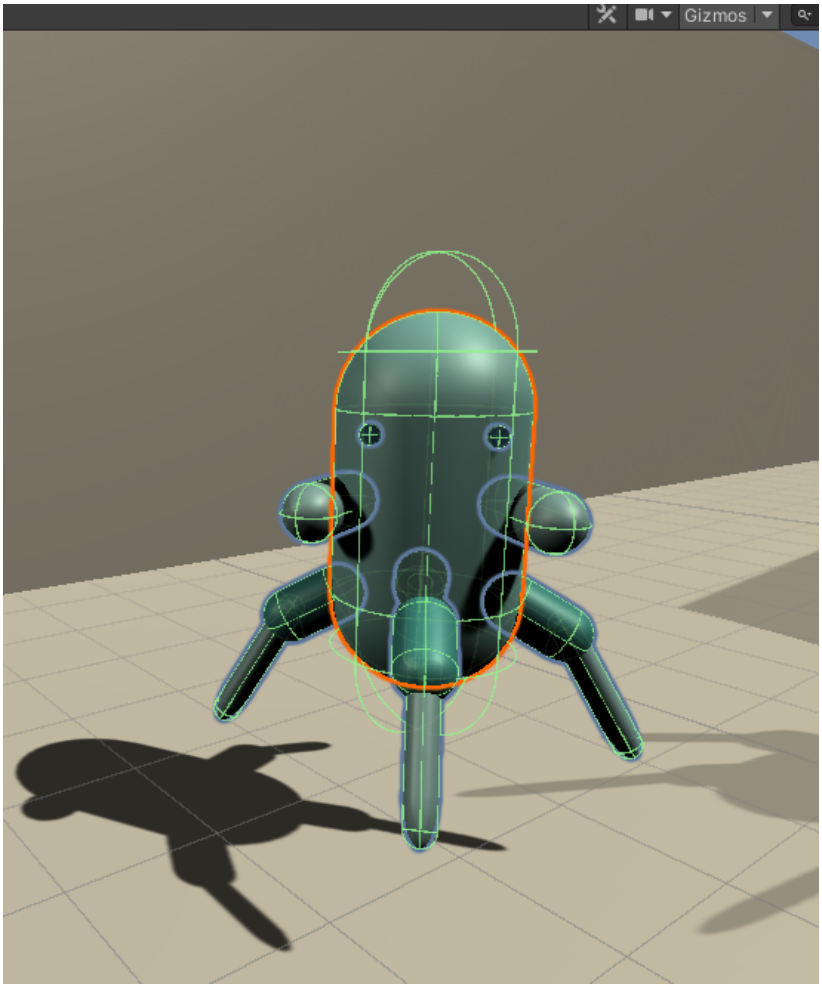
```
        private RaycastHit  GetIntersectionDistance()
        {
            var position = _transform.position;
            var ray = new Ray(position, _transform.forward);
            Physics.Raycast(ray, out _raycastHit, Mathf.Infinity);
            return _raycastHit;
        }

        private void OnGUI()
        {
            _raycastHit = GetIntersectionDistance();
            _crossHairScale = _raycastHit.collider == null ? 0.1f :
(float)MyUtils.GetZoomScaleByDistance(_raycastHit.distance);
            _distance = _raycastHit.collider == null ? "Infinite" :
_raycastHit.distance.ToString();
            _colliderName = _raycastHit.collider == null ? "None" :
_raycastHit.collider.name;
            // Draw crosshair (center of screen)
            GUI.DrawTexture(new Rect((Screen.width - crossTexture.width *
_crossHairScale) / 2f, (Screen.height - crossTexture.height * _crossHairScale) /
2f, crossTexture.width * _crossHairScale, crossTexture.height * _crossHairScale),
crossTexture);
        }
```

---

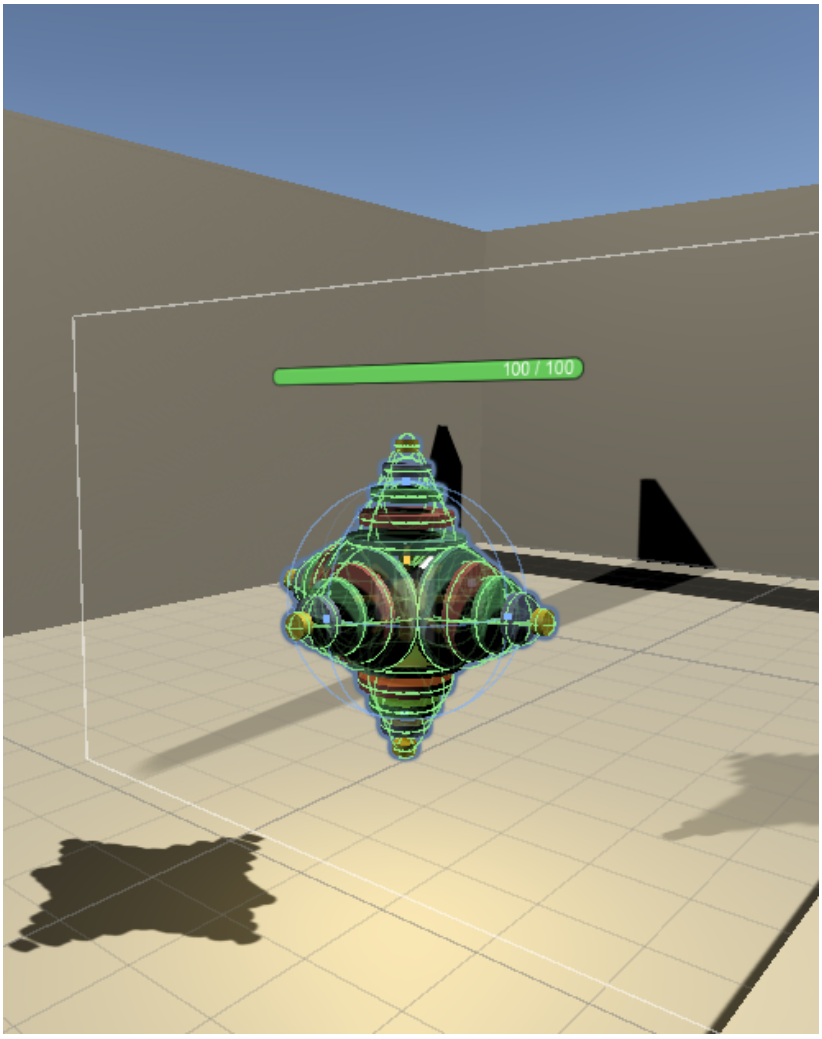
## Part 3 - Single level with gameplay (must be able to shoot enemies in some type of environment)

In this part, we implemented a simple AI for enemy. For every 5 secs, a new enemy will be initialized at the rebirth point. When attacked, HP of enemy decreases according to the damage of bullets. Also, enemy would be destroyed if HP equal or less than 0. To make player feel more challenging, we made enemy move to players after initialized, so player will be surrounded by enemies unless he/she kills enemies efficiently. For the gun part, player is allowed to use both left and right button of mouse. When left button clicked, the gun will fire. When player hold the right button, he/she is allowed to zoom in or out the camera by scrolling the mouse wheel.

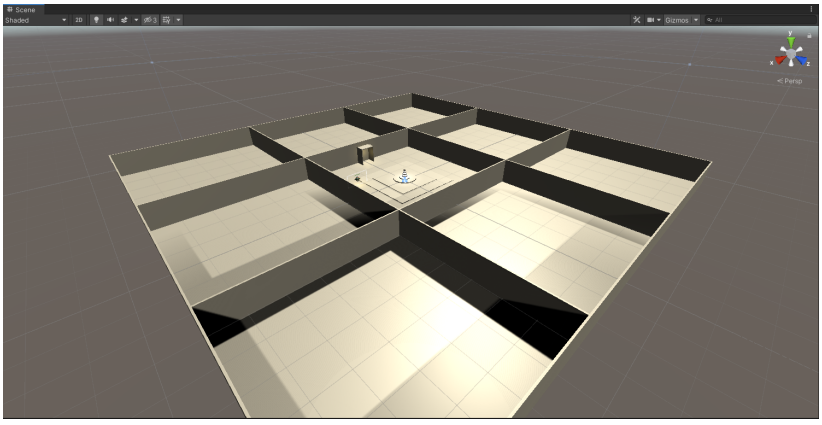
- **a. appearance (how good your environment/content looks)**
  `Player`

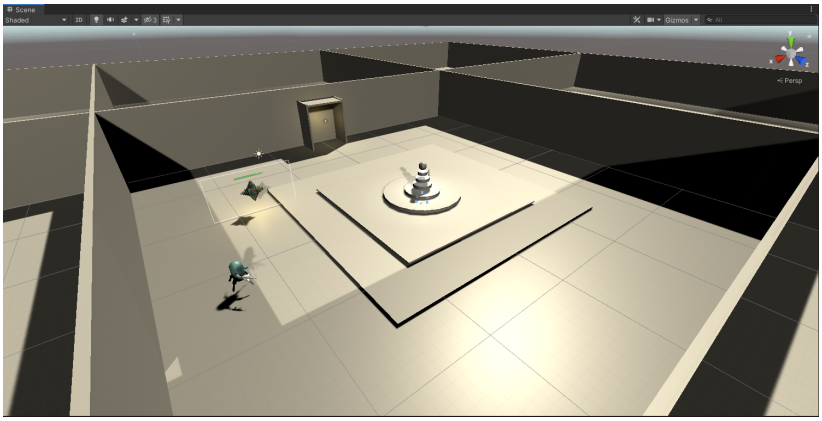



Enemy

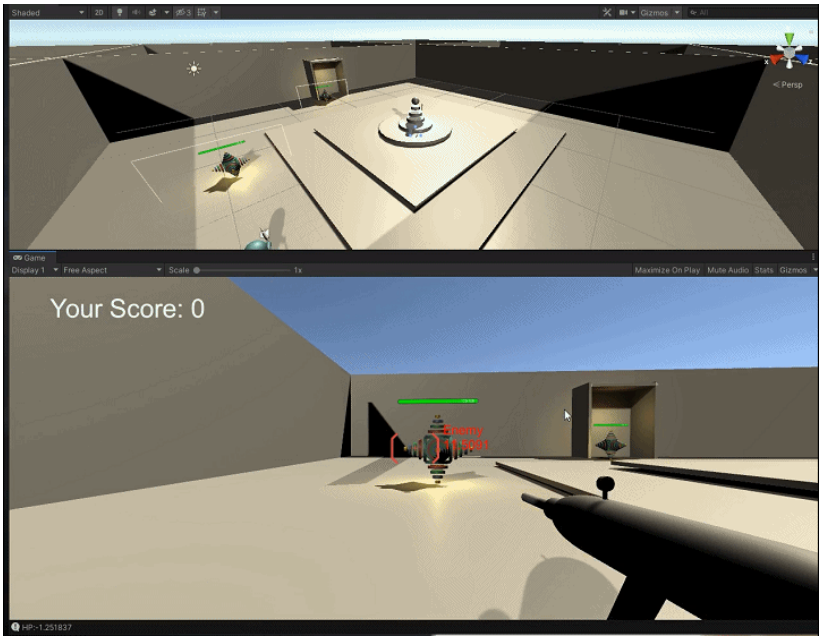


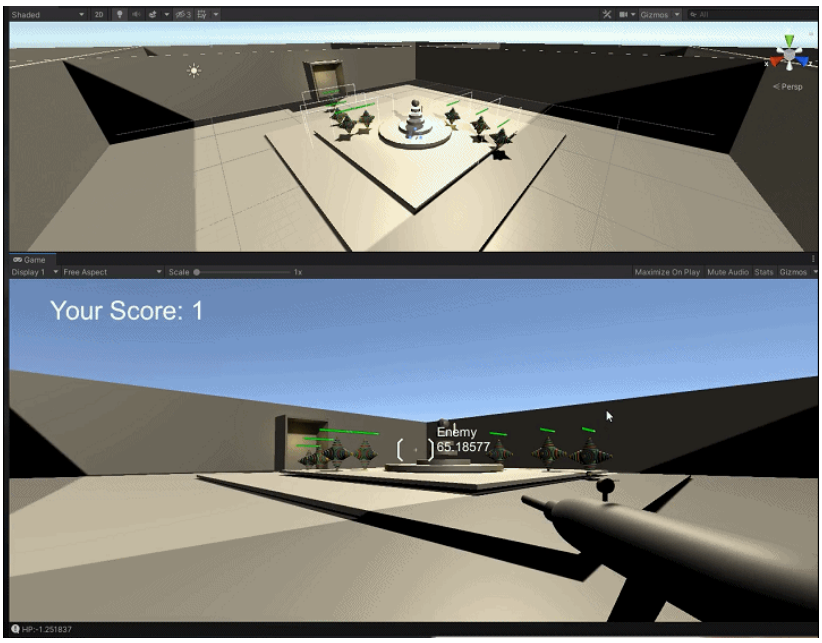

## Environment





- **b. enemy behavior (how the enemy behaves)**



▼ Code

```
        public void GetAttacked(float damage)
        {
            _isAttacked = true;
            _healthPoint -= Random.Range(damage, damage * 0.5f);
            slider.value = _healthPoint;
            textHealth.text = (int)_healthPoint + " / " + slider.maxValue;

            if (_healthPoint <= slider.maxValue * 0.3f)
                slider.gameObject.transform.Find("Fill
Area").Find("Fill").GetComponent<Image>().color = Color.red;
            if (_healthPoint <= 0)
            {
                destroyAudio.Play();
                if(gameObject.activeInHierarchy)
                    DrawCrossHair.SharedInstance.AddScore(1);
                gameObject.SetActive(false);
                // Invoke("InactiveEnemy",1f);
            }
            _rigidbody.velocity = Vector3.zero;
        }
```

▼ Code

```
        private void MoveTowardsPlayer()
        {
            var playerPosition = player.transform.position;
            var position = transform.position;
            var distance = Vector3.Distance(position, playerPosition);
            var normalizedDirection = new Vector3((playerPosition.x - position.x)
* Time.deltaTime * MoveSpeed/ distance,0f,(playerPosition.z - position.z)  *
Time.deltaTime * MoveSpeed/ distance);
            canvas.transform.Rotate(Quaternion.FromToRotation(Vector3.forward,
normalizedDirection).eulerAngles);
            transform.Translate(normalizedDirection);
            ResetCanvasRotation();
        }
```

▼ Code

```
        private void ResetCanvasRotation()
        {
            var direction = new Vector3(player.transform.position.x -
transform.position.x, 0f, player.transform.position.z - transform.position.z);
            // transform.Rotate(Quaternion.FromToRotation(Vector3.forward,
direction).eulerAngles);
            canvas.transform.Rotate(Quaternion.FromToRotation(Vector3.forward,
```

```
direction).eulerAngles);
        }

        public void ResetEnemy()
        {
            var maxValue = slider.maxValue;
            _isAttacked = true;
            slider.value = maxValue;
            slider.gameObject.transform.Find("Fill
Area").Find("Fill").GetComponent<Image>().color = Color.green;
            _healthPoint = maxValue;
            _initRotation = transform.rotation;
            textHealth.text = maxValue + "/" + maxValue;
        }
```

▼ Code

```
public class EnemyManager : MonoBehaviour
    {
        // [FormerlySerializedAs("RebirthPoint")] public GameObject rebirthPoint;

        private const float RebirthTime = 5f;
        private float _countDownTime;
        public Enemy enemyX;
        // Start is called before the first frame update
        private void Start()
        {
            _countDownTime = RebirthTime;
            CreateEnemy();
        }

        // Update is called once per frame
        private void Update()
        {
            if (_countDownTime <= 0)
            {
                CreateEnemy();
                _countDownTime = RebirthTime;
            }
            _countDownTime -= Time.deltaTime;
        }

        private void CreateEnemy()
        {
            var enemy = EnemyPool.SharedInstance.GetNewEnemy();
            if (enemy.textHealth == null)
                enemy = Instantiate(enemyX);
            enemy.SetLocation(transform.position);
            enemy.SetIsActive(true);
            enemy.ResetEnemy();
        }
    }
```

- **c. weapon (gun sound effects, shooting animation, etc.)**