Sweet My Home
Description

Developer Name: Euiseok Jeong

Github: https://github.com/EuiseokJeongNZ

LinkedIn: https://www.linkedin.com/in/euiseok-jeong-965b9b310

Contents

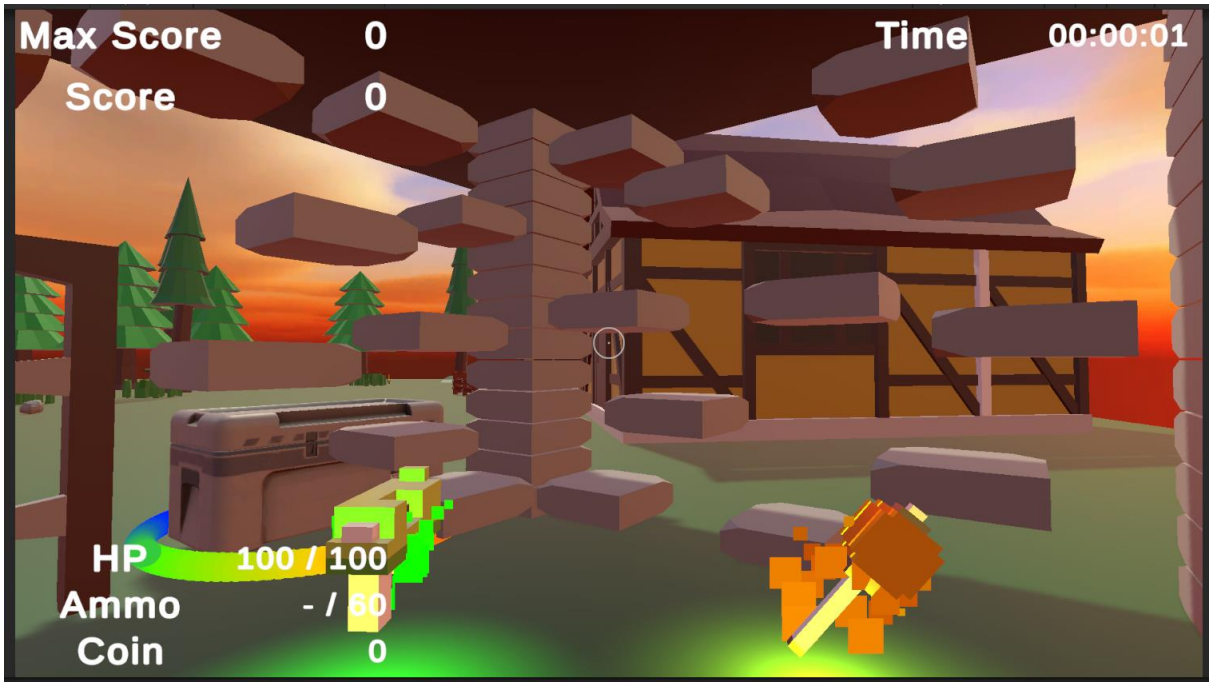
Part A

Task 1: Develop Graphical User Interface (GUI)

The main menu is the scene, including a title and two buttons. It illustrates two options to start the game and set the game option.

Once users click on the "Game Setting" button, they will be given a description of how to play the game and adjust the volume.

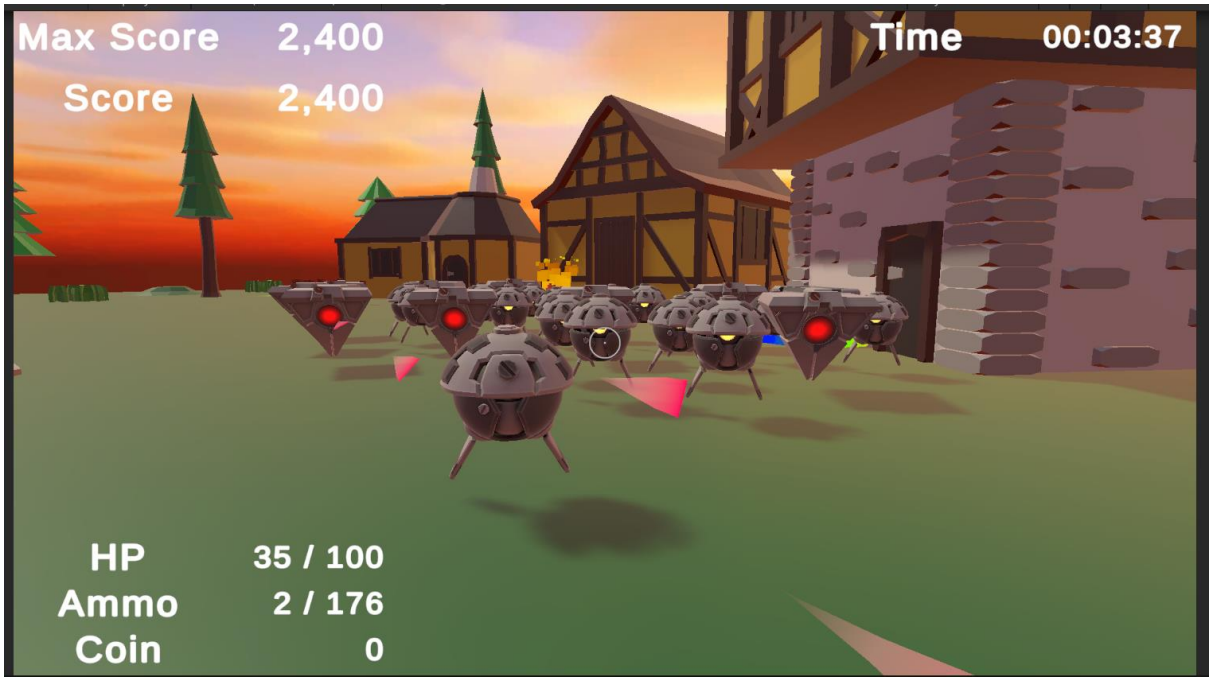
In the level 1 scene, a player is respawned at his house, showing the shop box and two weapons, those are a hammer and a handgun. Basically, his HP points are 100 and he first has 60 bullets for the handgun. The hammer and handgun damages are each 25 and 15.



When a player gets in the donut area coloured-rainbow, the shop scene will automatically appear with 3 items. A user can buy those items with coins, which enemies drop by 50 percent.

When the player goes out of the house, he will encounter a hover bot, which automatically attacks and tracks the player. It will create every 5 seconds and damage him 5 HP points as he touches them, and its HP and the score points to give the player are each 70 HP and 100 points.

It is the level 2 scene if a player reaches a score over 2,000 points, then a triangular cube-shaped robot will appear every 10 seconds. Their attack pattern is similar to the hover bot, but can launch a laser missile that damages 10 HP points the player, and its HP and the score points to give the player are each 100 HP and 200 points.



It is the last scene, which is level 3, that a PA warrior will appear every 20 seconds. The criteria to create them is the player's score is over 5,000 points. In addition, the pattern is same as the triangular cube-shaped robot, however, its missile is faster and has higher damage, 15. The shape of its missile is a sphere red-coloured. Lastly, its HP and the score points to give the player are each 150 HP and 300 points.

Finally, it is a game-over scene when a player dies, which means that the HP points under 0, and the "Main Title" will guide the main scene.

Task 2: Implement Gameplay Mechanics

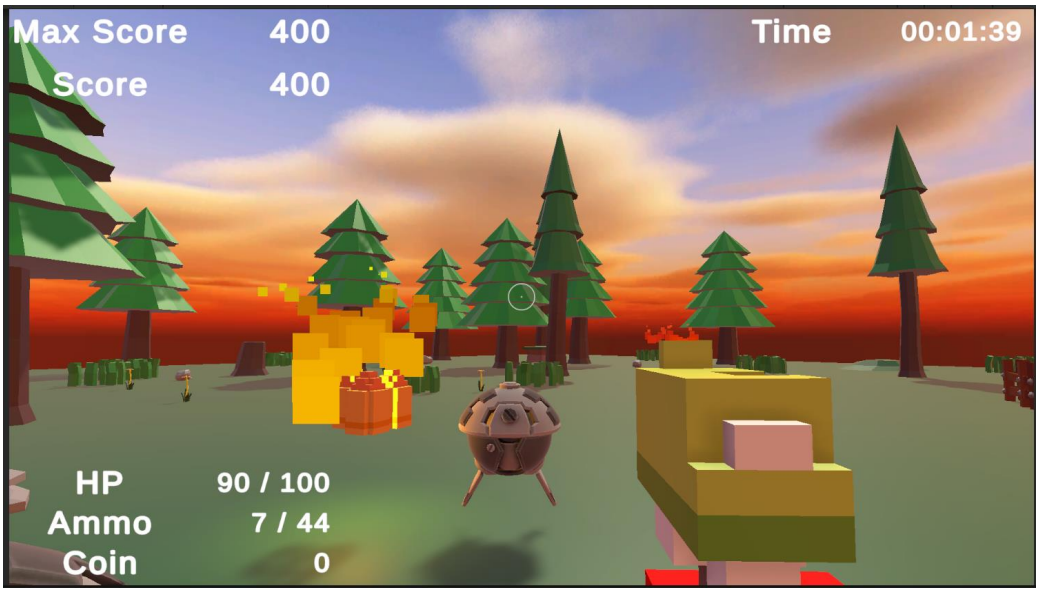
a) Game Objectives

i. Destroy Enemies Robots

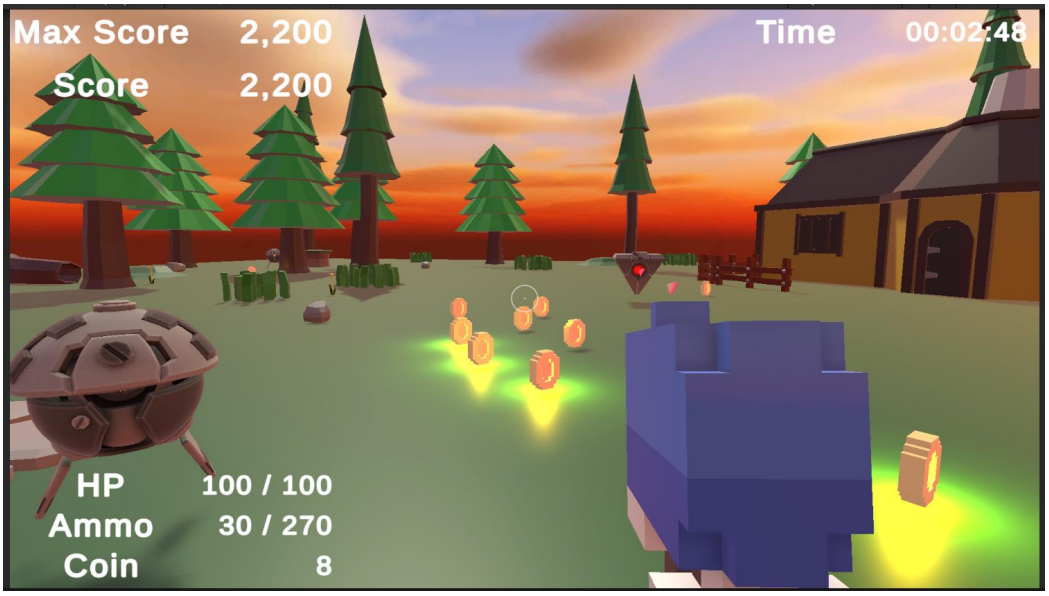
ii. Just Survive in each level 1, 2, and 3 with items

iii. Keep HP, not under 0

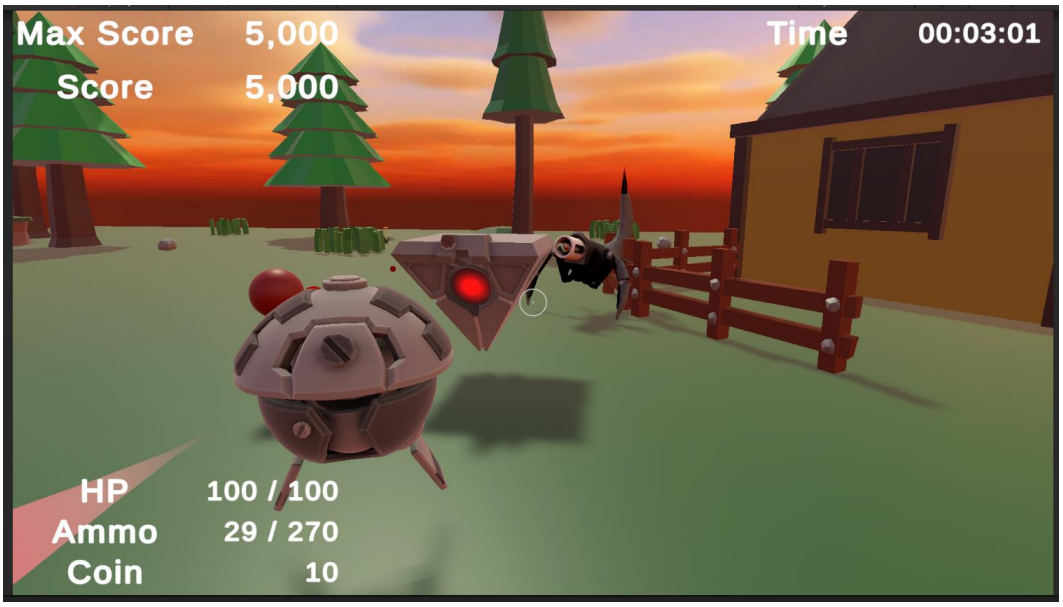
Level 1)



Level 2)



Level 3)



b) Game Logic

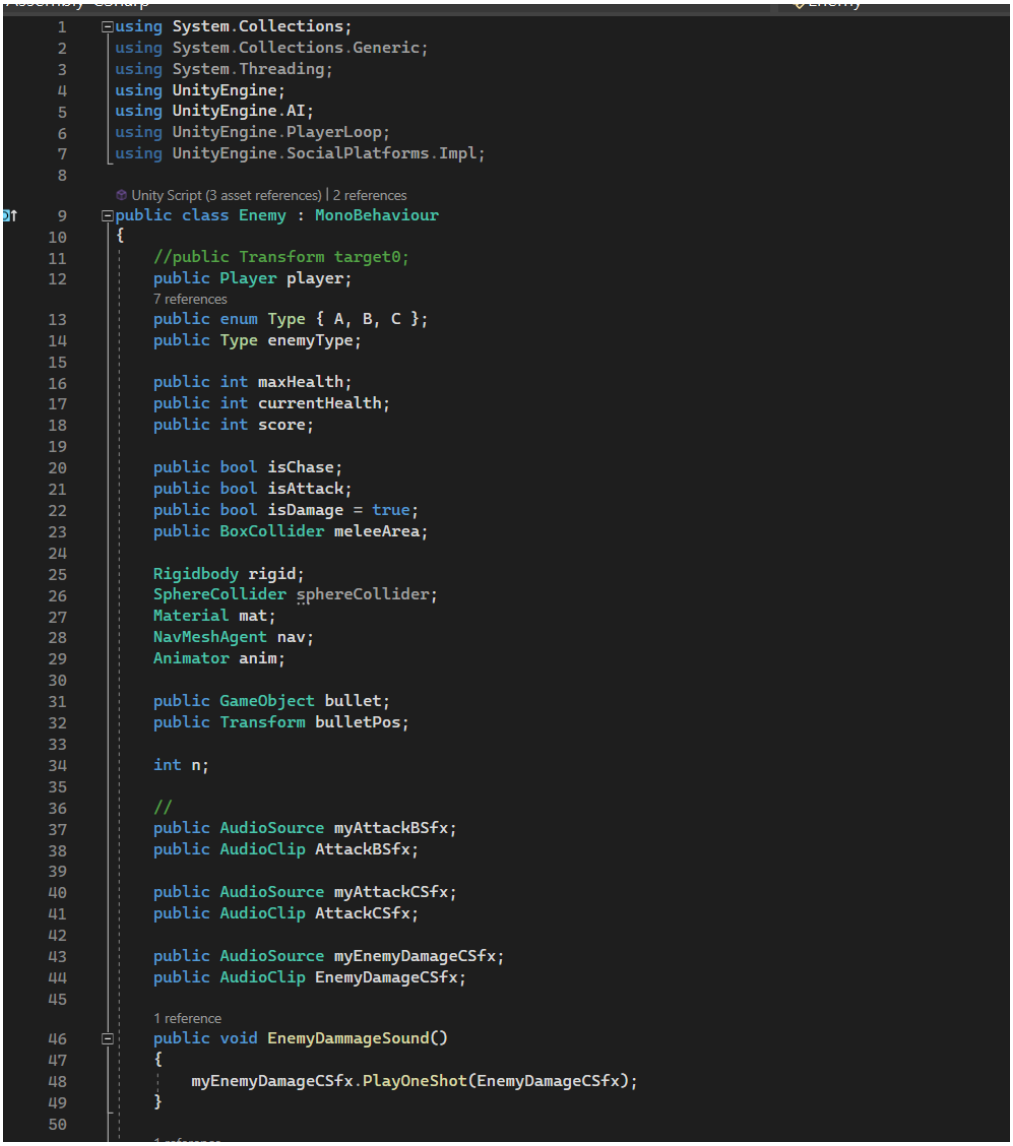
i. Enemies create every specific time, 5, 10, and 20 seconds

```
using JetBrains.Annotations;
using System.Collections;
using System.Collections.Generic;
using System.Threading;
using UnityEngine;

Unity Script (3 asset references) | 0 references
public class RandomEnemyA : MonoBehaviour
{
    public float currTime;
    public GameObject monster;

    Unity Message | 0 references
    void Update()
    {
        currTime += Time.deltaTime;

        // the enemy will automatically appear in 5 seconds
        if (currTime > 5)
        {
            // x,y,z random axis
            float newX = Random.Range(-10f, 50f), newY = Random.Range(1f, 5f), newZ = Random.Range(-35f, 60f);

            // move the object to the position
            monster.transform.position = new Vector3(newX, newY, newZ);

            // bring object where positions
            Instantiate(monster, new Vector3(newX, newY, newZ), Quaternion.identity);

            // when it reaches 10 seconds, currTime is initialised 0
            currTime = 0;
        }
    }
}
```

```
using JetBrains.Annotations;
using System.Collections;
using System.Collections.Generic;
using System.Threading;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class RandomEnemyB : MonoBehaviour
{
    public float currTime;
    public GameObject monster;

    public Player player;

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        currTime += Time.deltaTime;

        if (player.score >= 2000)
        {
            // the enemy will automatically appear in 10 seconds
            if (currTime > 10)
            {
                // x,y,z random axis
                float newX = Random.Range(-10f, 50f), newY = Random.Range(1f, 5f), newZ = Random.Range(-35f, 60f);

                // move the object to the position
                monster.transform.position = new Vector3(newX, newY, newZ);

                // bring object where positions
                Instantiate(monster, new Vector3(newX, newY, newZ), Quaternion.identity);

                // when it reaches 10 seconds, currTime is initialised 0
                currTime = 0;
            }
        }
    }
}
```

```
using JetBrains.Annotations;
using System.Collections;
using System.Collections.Generic;
using System.Threading;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class RandomEnemyC : MonoBehaviour
{
    public float currTime;
    public GameObject monster;

    public Player player;

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        currTime += Time.deltaTime;

        if (player.score >= 5000)
        {
            // the enemy will automatically appear in 20 seconds
            if (currTime > 20)
            {
                // x,y,z random axis
                float newX = Random.Range(-10f, 50f), newY = Random.Range(1f, 5f), newZ = Random.Range(-35f, 60f);

                // move the object to the position
                monster.transform.position = new Vector3(newX, newY, newZ);

                // bring object where positions
                Instantiate(monster, new Vector3(newX, newY, newZ), Quaternion.identity);

                // when it reaches 10 seconds, currTime is initialised 0
                currTime = 0;
            }
        }
    }
}
```

ii. Enemies automatically track and attack the player with touch or missiles, these will damage



```
using System.Collections;
using System.Collections.Generic;
using System.Threading;
using UnityEngine;
using UnityEngine.AI;
using UnityEngine.PlayerLoop;
using UnityEngine.SocialPlatforms.Impl;

Unity Script (3 asset references) | 2 references
public class Enemy : MonoBehaviour
{
    //public Transform target0;
    public Player player;
    7 references
    public enum Type { A, B, C };
    public Type enemyType;

    public int maxHealth;
    public int currentHealth;
    public int score;

    public bool isChase;
    public bool isAttack;
    public bool isDamage = true;
    public BoxCollider meleeArea;

    Rigidbody rigid;
    SphereCollider sphereCollider;
    Material mat;
    NavMeshAgent nav;
    Animator anim;

    public GameObject bullet;
    public Transform bulletPos;

    int n;

    //
    public AudioSource myAttackBSfx;
    public AudioClip AttackBSfx;

    public AudioSource myAttackCSfx;
    public AudioClip AttackCSfx;

    public AudioSource myEnemyDamageCSfx;
    public AudioClip EnemyDamageCSfx;

    1 reference
    public void EnemyDammageSound()
    {
        myEnemyDamageCSfx.PlayOneShot(EnemyDamageCSfx);
    }
```

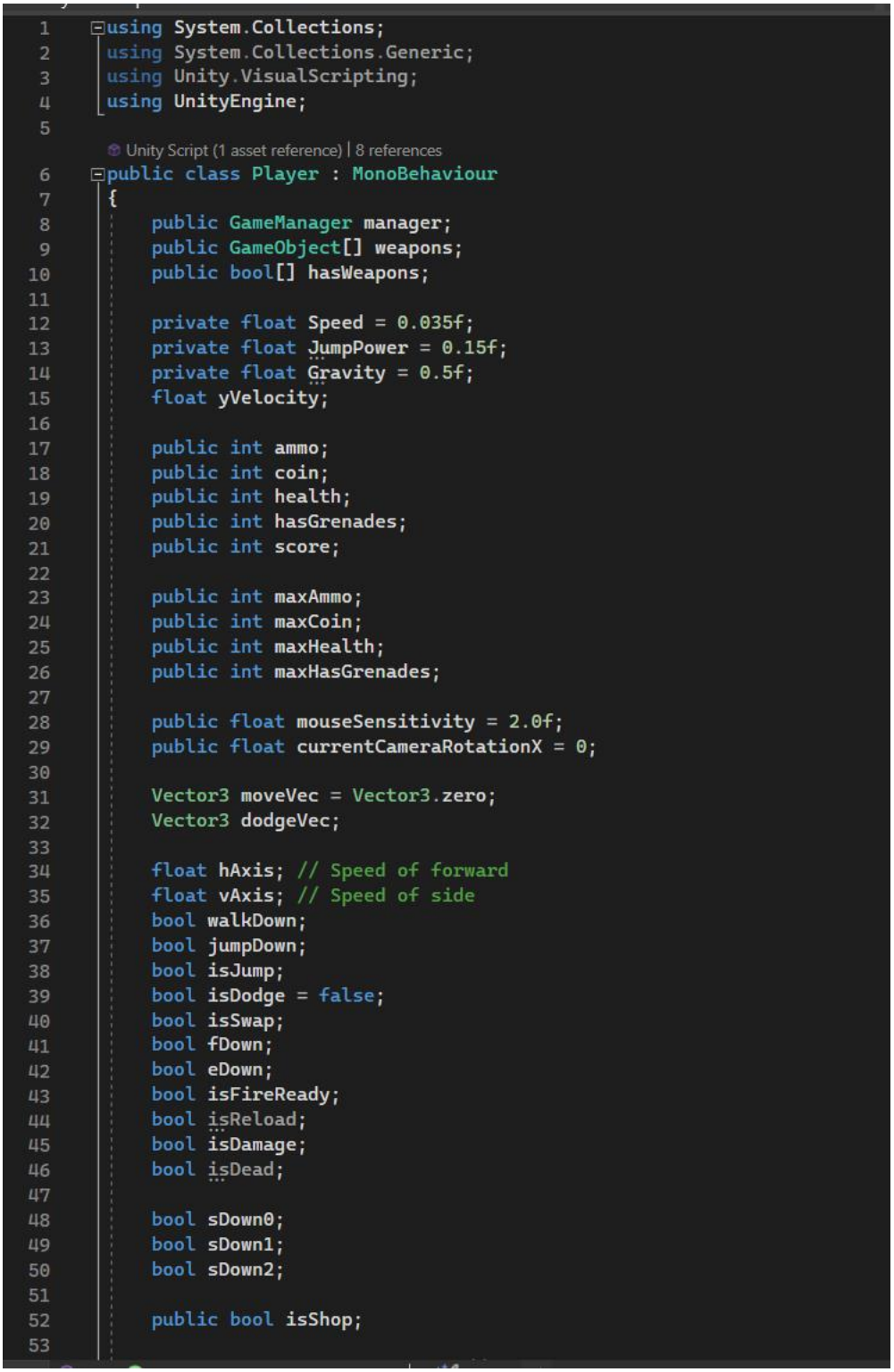
```
1 reference
public void AttackBSound()
{
    myAttackBSfx.PlayOneShot(AttackBSfx);
}
1 reference
public void AttackCSound()
{
    myAttackCSfx.PlayOneShot(AttackCSfx);
}
//

//
private GameObject getPlayer;
//

// Start is called before the first frame update
Unity Message | 0 references
void Start()
{

    rigid = GetComponent<Rigidbody>();
    sphereCollider = GetComponent<SphereCollider>();
    mat = GetComponentInChildren<SkinnedMeshRenderer>().material;
    nav = GetComponent<NavMeshAgent>();
    anim = GetComponentInChildren<Animator>();

    Invoke("ChaseStart", 2);

    //
    getPlayer = GameObject.Find("Player");
    player = GameObject.FindWithTag("Player").GetComponent<Player>();
    //


}
0 references
void ChaseStart()
{
    isChase = true;
    anim.SetBool("isWalk", true);
}
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    Vector3 playerPosition = getPlayer.transform.position;

    if (nav.enabled)
    {
        nav.SetDestination(playerPosition);
        nav.isStopped = !isChase;
    }
}
```

```
100     public GameObject itemPrefab; // Prefab for dropping item
101
        Unity Message | 0 references
102     private void OnDestroy()
103     {
104         n = Random.Range(1, 2); // Percentage of Item to drop is 25 %
105         if (itemPrefab != null && n ==1)
106         {
107             Instantiate(itemPrefab, transform.position, Quaternion.identity);
108         }
109     }
110
        Unity Message | 0 references
111     void OnTriggerEnter(Collider other)
112     {
113         if(other.tag == "Melee")
114         {
115             EnemyDammageSound();
116             Weapon weapon = other.GetComponent<Weapon>();
117             currentHealth -= weapon.damage;
118             Vector3 reactVec = transform.position - other.transform.position;
119             StartCoroutine(OnDamage(reactVec));
120         }
121         else if (other.tag == "Bullet")
122         {
123             Bullet bullet = other.GetComponent<Bullet>();
124             currentHealth -= bullet.damage;
125             Vector3 reactVec = transform.position - other.transform.position;
126             StartCoroutine(OnDamage(reactVec));
127         }
128     }
129
        2 references
130     IEnumerator OnDamage(Vector3 reactVec)
131     {
132         mat.color = Color.red;
133         yield return new WaitForSeconds(0.1f);
134
135         if(currentHealth > 0)
136         {
137             mat.color = Color.white;
138         }
139         else
140         {
141             if (isDamage)
142             {
143                 mat.color = Color.gray;
144                 gameObject.layer = 13;
145                 isChase = false;
146                 isAttack = false;
147                 nav.enabled = false;
148
149                 anim.SetTrigger("doDie");
150
```

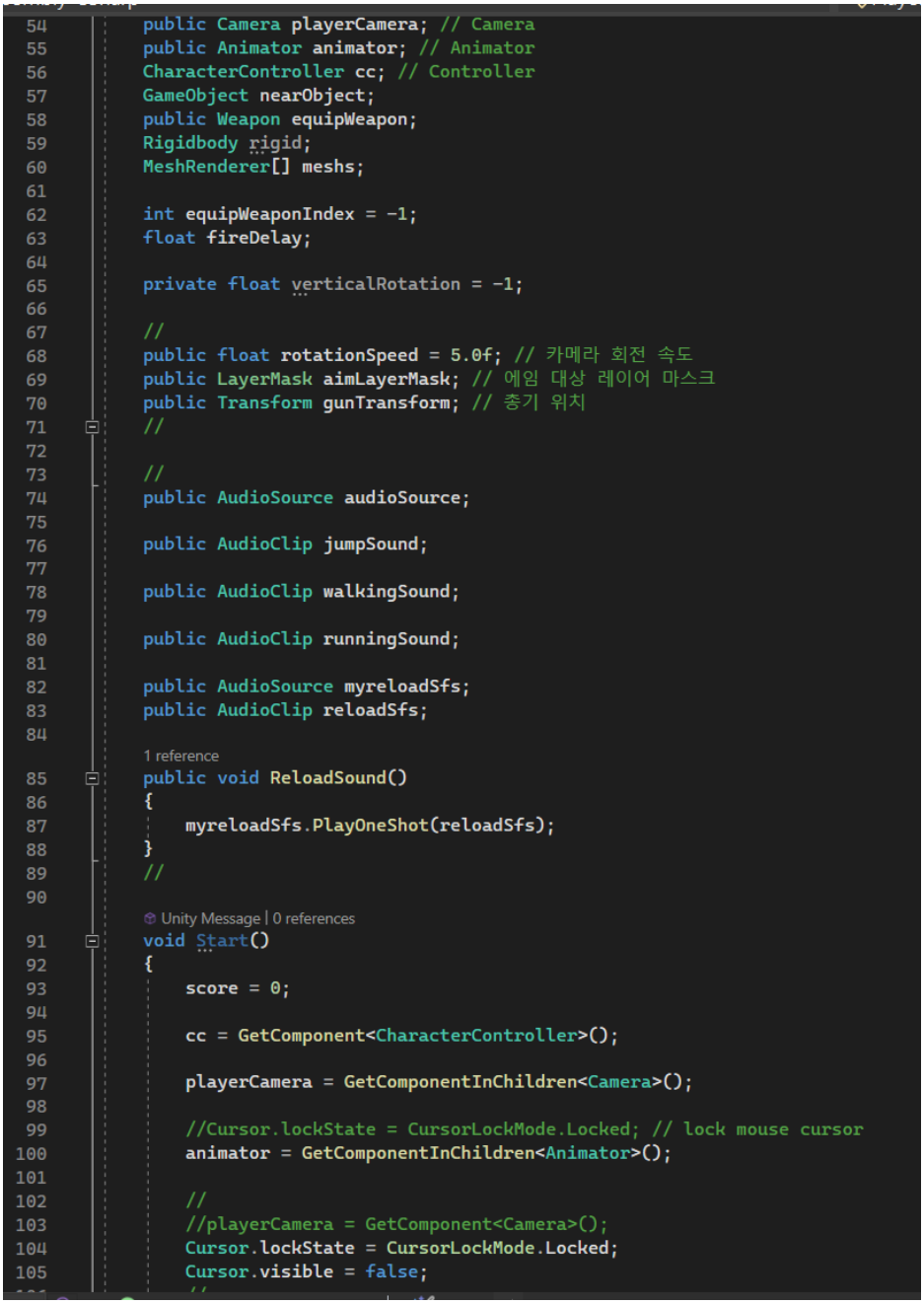
```
150
151                 reactVec = reactVec.normalized;
152                 reactVec += Vector3.up;
153
154                 rigid.AddForce(reactVec * 4f, ForceMode.Impulse);
155
156                 player.score += score;
157
158                 Destroy(gameObject, 2f);
159                 isDamage = false;
160             }
161         }
162     }
        1 reference
163     void FreezeVelocity()
164     {
165         if (isChase)
166         {
167             rigid.velocity = Vector3.zero;
168             rigid.angularVelocity = Vector3.zero;
169         }
170     }
        1 reference
171     void Targetting()
172     {
173         float targetRadius = 0f;
174         float targetRange = 0f;
175
176         switch (enemyType)
177         {
178             case Type.A:
179                 targetRadius = 1.5f;
180                 targetRange = 3f;
181                 break;
182             case Type.B:
183                 targetRadius = 2f;
184                 targetRange = 20f;
185                 break;
186             case Type.C:
187                 targetRadius = 1.5f;
188                 targetRange = 30f;
189                 break;
190         }
191
192         RaycastHit[] rayHits =
193             Physics.SphereCastAll(transform.position, targetRadius,
194                                   Vector3.forward, targetRange,
195                                   LayerMask.GetMask("Player"));
196
197         if (rayHits.Length > 0 && !isAttack)
198         {
199             StartCoroutine(Attack());
200             Debug.Log("Doing Attack Method!");
201         }
```

```
        }
    }
    1 reference
    IEnumerator Attack()
    {
        isChase = false;
        isAttack = true;
        anim.SetBool("isAttack", true);

        switch (enemyType)
        {
            case Type.A:
                yield return new WaitForSeconds(0.2f);
                meleeArea.enabled = true;

                yield return new WaitForSeconds(1f);
                meleeArea.enabled = false;

                yield return new WaitForSeconds(1f);

                break;

            case Type.B:
                yield return new WaitForSeconds(0.4f);
                GameObject instantBulletB = Instantiate(bullet, transform.position +
                    new Vector3(0f, 1.2f, 0f), transform.rotation); //transform.position
                Rigidbody rigidBulletB = instantBulletB.GetComponent<Rigidbody>();
                rigidBulletB.velocity = transform.forward * 15f;
                AttackBSound();

                break;

            case Type.C:
                yield return new WaitForSeconds(1f);
                GameObject instantBullet = Instantiate(bullet, transform.position +
                    new Vector3(0f, 1.2f, 0f), transform.rotation); //transform.position
                Rigidbody rigidBullet = instantBullet.GetComponent<Rigidbody>();
                rigidBullet.velocity = transform.forward * 20f;
                AttackCSound();
                break;
        }
        isChase = true;
        isAttack = false;
        anim.SetBool("isAttack", false);
    }
    Unity Message | 0 references
    void FixedUpdate()
    {
        Targetting();
        FreezeVelocity();
    }
}
```

iii. Players can pick up items and use items and 3 weapons to attack as well



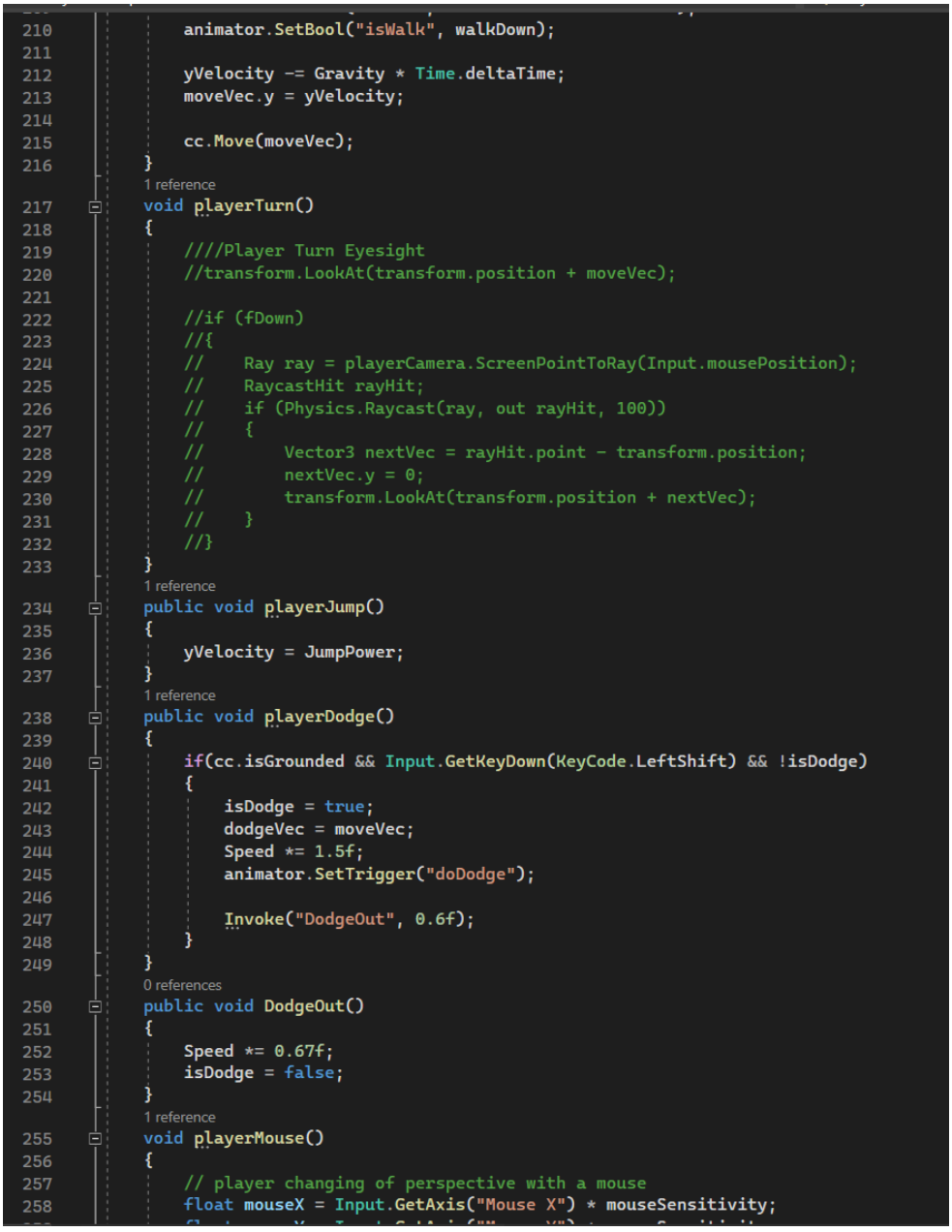
```
using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;

Unity Script (1 asset reference) | 8 references
public class Player : MonoBehaviour
{
    public GameManager manager;
    public GameObject[] weapons;
    public bool[] hasWeapons;

    private float Speed = 0.035f;
    private float JumpPower = 0.15f;
    private float Gravity = 0.5f;
    float yVelocity;

    public int ammo;
    public int coin;
    public int health;
    public int hasGrenades;
    public int score;

    public int maxAmmo;
    public int maxCoin;
    public int maxHealth;
    public int maxHasGrenades;

    public float mouseSensitivity = 2.0f;
    public float currentCameraRotationX = 0;

    Vector3 moveVec = Vector3.zero;
    Vector3 dodgeVec;

    float hAxis; // Speed of forward
    float vAxis; // Speed of side
    bool walkDown;
    bool jumpDown;
    bool isJump;
    bool isDodge = false;
    bool isSwap;
    bool fDown;
    bool eDown;
    bool isFireReady;
    bool isReload;
    bool isDamage;
    bool isDead;

    bool sDown0;
    bool sDown1;
    bool sDown2;

    public bool isShop;

```



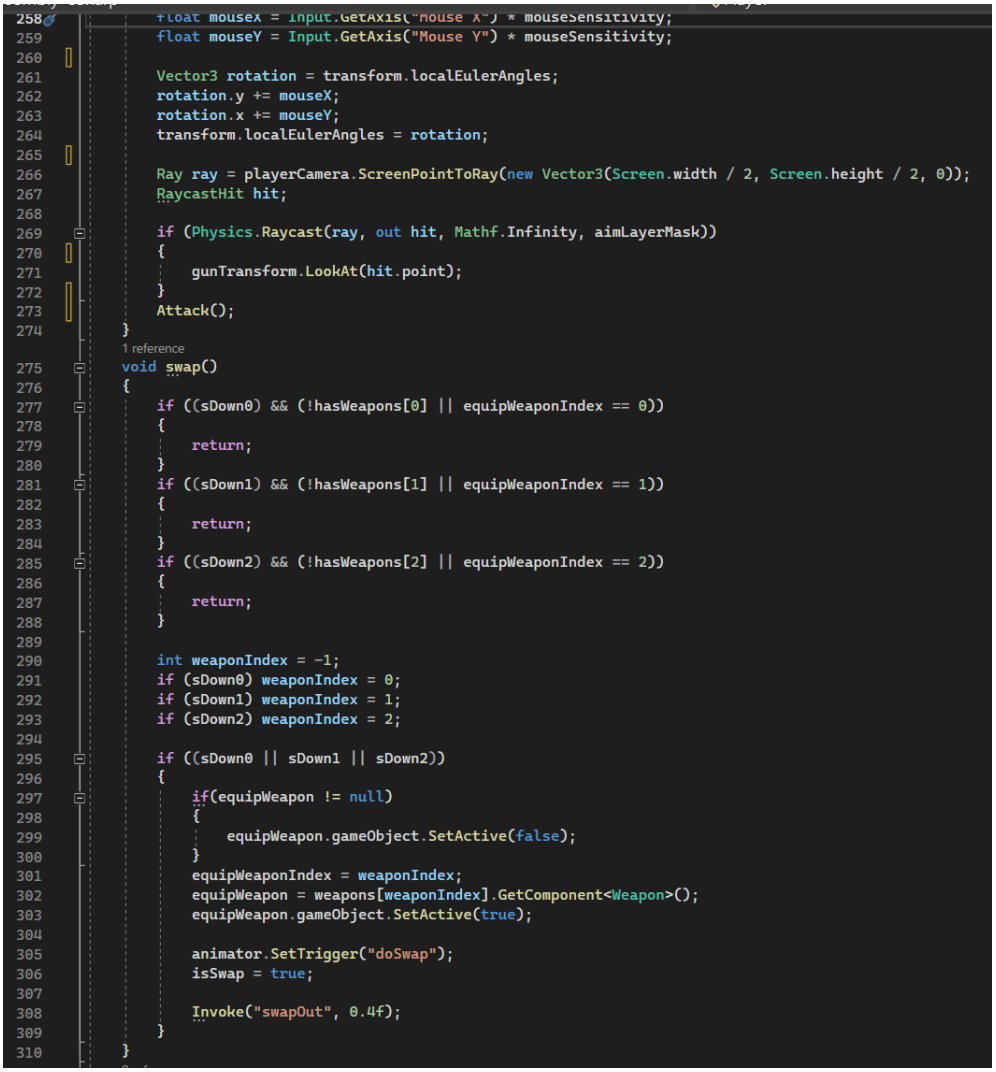
```
54      public Camera playerCamera; // Camera
55      public Animator animator; // Animator
56      CharacterController cc; // Controller
57      GameObject nearObject;
58      public Weapon equipWeapon;
59      Rigidbody rigid;
60      MeshRenderer[] meshs;
61
62      int equipWeaponIndex = -1;
63      float fireDelay;
64
65      private float verticalRotation = -1;
66
67      //
68      public float rotationSpeed = 5.0f; // 카메라 회전 속도
69      public LayerMask aimLayerMask; // 에임 대상 레이어 마스크
70      public Transform gunTransform; // 총기 위치
71      //
72
73      //
74      public AudioSource audioSource;
75
76      public AudioClip jumpSound;
77
78      public AudioClip walkingSound;
79
80      public AudioClip runningSound;
81
82      public AudioSource myreloadSfs;
83      public AudioClip reloadSfs;
84
        1 reference
85      public void ReloadSound()
86      {
87          myreloadSfs.PlayOneShot(reloadSfs);
88      }
89      //
90
        Unity Message | 0 references
91      void Start()
92      {
93          score = 0;
94
95          cc = GetComponent<CharacterController>();
96
97          playerCamera = GetComponentInChildren<Camera>();
98
99          //Cursor.lockState = CursorLockMode.Locked; // lock mouse cursor
100         animator = GetComponentInChildren<Animator>();
101
102         //
103         //playerCamera = GetComponent<Camera>();
104         Cursor.lockState = CursorLockMode.Locked;
105         Cursor.visible = false;
```

```
105         Cursor.visible = false;
106         //
107
108         meshs = GetComponentsInChildren<MeshRenderer>();
109
110         //
111         audioSource = GetComponent<AudioSource>();
112         //
113     }
        Unity Message | 0 references
114     void Update() // Activate 60 times per a second
115     {
116         rigid = GetComponent<Rigidbody>();
117         GetInput();
118
119         if (health > 0)
120         {
121             if (cc.isGrounded && jumpDown)
122             {
123                 playerJump();
124                 animator.SetBool("isJump", !isJump);
125                 animator.SetTrigger("doJump");
126                 isJump = true;
127
128
129                 if (cc.isGrounded)
130                 {
131                     isJump = false;
132                     animator.SetBool("isJump", isJump);
133                 }
134             }
135
136             if (cc.isGrounded)
137             {
138                 if (cc.velocity.magnitude > 0)
139                 {
140                     if (!audioSource.isPlaying)
141                     {
142                         audioSource.clip = !walkDown ? runningSound : walkingSound;
143                         audioSource.Play();
144                     }
145                 }
146                 else
147                 {
148                     audioSource.Stop();
149                 }
150
151                 if (Input.GetButtonDown("Jump"))
152                 {
153                     audioSource.clip = jumpSound;
154                     audioSource.Play();
155                 }
156             }
157             else
```

```
158                 {
159                     audioSource.Stop();
160                 }
161 
162                 playerMove();
163                 playerMouse();
164                 playerDodge();
165                 playerTurn();
166                 interaction();
167                 swap();
168                 //Attack();
169                 Reload();
170                 //playerTurn();
171             }
172             else
173             {
174                 isDead = true;
175                 OnDie();
176 
177                 Cursor.lockState = CursorLockMode.None;
178                 Cursor.visible = true;
179             }
180         }
            1 reference
181         void GetInput()
182         {
183             hAxis = Input.GetAxisRaw("Horizontal");
184             vAxis = Input.GetAxisRaw("Vertical");
185             walkDown = Input.GetKey(KeyCode.LeftControl);
186             jumpDown = Input.GetKeyDown(KeyCode.Space);
187             sDown0 = Input.GetKeyDown(KeyCode.Alpha1);
188             sDown1 = Input.GetKeyDown(KeyCode.Alpha2);
189             sDown2 = Input.GetKeyDown(KeyCode.Alpha3);
190             fDown = Input.GetKey(KeyCode.Mouse0);
191             eDown = Input.GetKeyDown(KeyCode.E);
192         }
193 
            1 reference
194         void playerMove()
195         {
196             moveVec.x = hAxis;
197             moveVec.y = 0;
198             moveVec.z = vAxis;
199 
200             moveVec = transform.TransformDirection(moveVec);
201             moveVec *= Speed * (walkDown ? 0.3f : 1f);
202 
203             if (Input.GetKeyDown(KeyCode.LeftShift))
204             {
205                 moveVec = dodgeVec;
206             }
207 
208             // Walk and Run
```

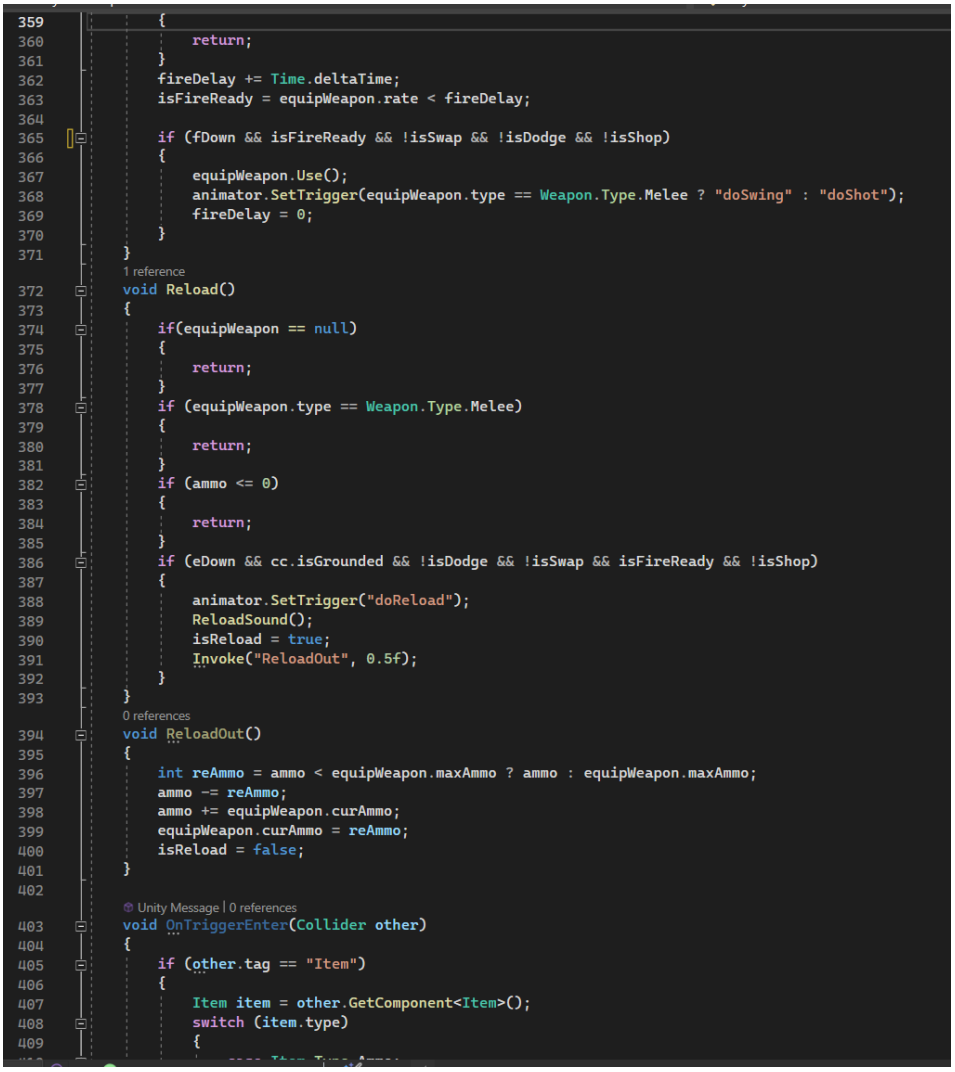


```
210         animator.SetBool("isWalk", walkDown);
211
212         yVelocity -= Gravity * Time.deltaTime;
213         moveVec.y = yVelocity;
214
215         cc.Move(moveVec);
216     }
        1 reference
217     void playerTurn()
218     {
219         ////Player Turn Eyesight
220         //transform.LookAt(transform.position + moveVec);
221
222         //if (fDown)
223         //{
224         //    Ray ray = playerCamera.ScreenPointToRay(Input.mousePosition);
225         //    RaycastHit rayHit;
226         //    if (Physics.Raycast(ray, out rayHit, 100))
227         //    {
228         //        Vector3 nextVec = rayHit.point - transform.position;
229         //        nextVec.y = 0;
230         //        transform.LookAt(transform.position + nextVec);
231         //    }
232         //}
233     }
        1 reference
234     public void playerJump()
235     {
236         yVelocity = JumpPower;
237     }
        1 reference
238     public void playerDodge()
239     {
240         if(cc.isGrounded && Input.GetKeyDown(KeyCode.LeftShift) && !isDodge)
241         {
242             isDodge = true;
243             dodgeVec = moveVec;
244             Speed *= 1.5f;
245             animator.SetTrigger("doDodge");
246
247             Invoke("DodgeOut", 0.6f);
248         }
249     }
        0 references
250     public void DodgeOut()
251     {
252         Speed *= 0.67f;
253         isDodge = false;
254     }
        1 reference
255     void playerMouse()
256     {
257         // player changing of perspective with a mouse
258         float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity;
```



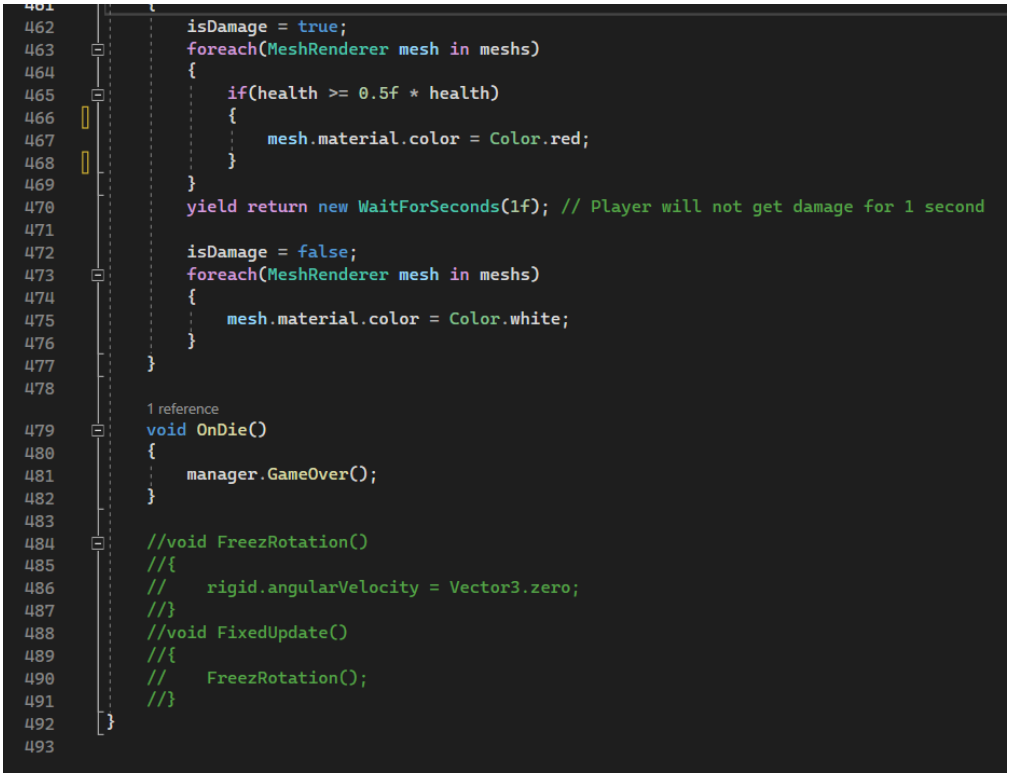
```
        float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity;
        float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity;

        Vector3 rotation = transform.localEulerAngles;
        rotation.y += mouseX;
        rotation.x += mouseY;
        transform.localEulerAngles = rotation;

        Ray ray = playerCamera.ScreenPointToRay(new Vector3(Screen.width / 2, Screen.height / 2, 0));
        RaycastHit hit;

        if (Physics.Raycast(ray, out hit, Mathf.Infinity, aimLayerMask))
        {
            gunTransform.LookAt(hit.point);
        }
        Attack();
    }
    1 reference
    void swap()
    {
        if ((sDown0) && (!hasWeapons[0] || equipWeaponIndex == 0))
        {
            return;
        }
        if ((sDown1) && (!hasWeapons[1] || equipWeaponIndex == 1))
        {
            return;
        }
        if ((sDown2) && (!hasWeapons[2] || equipWeaponIndex == 2))
        {
            return;
        }

        int weaponIndex = -1;
        if (sDown0) weaponIndex = 0;
        if (sDown1) weaponIndex = 1;
        if (sDown2) weaponIndex = 2;

        if ((sDown0 || sDown1 || sDown2))
        {
            if(equipWeapon != null)
            {
                equipWeapon.gameObject.SetActive(false);
            }
            equipWeaponIndex = weaponIndex;
            equipWeapon = weapons[weaponIndex].GetComponent<Weapon>();
            equipWeapon.gameObject.SetActive(true);

            animator.SetTrigger("doSwap");
            isSwap = true;

            Invoke("swapOut", 0.4f);
        }
    }
```

```
}
0 references
void swapOut()
{
    isSwap = false;
}
Unity Message | 0 references
private void OnTriggerStay(Collider other)
{
    if (other.tag == "Weapon" || other.tag == "Shop")
    {
        nearObject = other.gameObject;
    }
}
Unity Message | 0 references
private void OnTriggerExit(Collider other)
{
    if (other.tag == "Weapon" )
    {
        nearObject = null;
    }
    else if (other.tag == "Shop")
    {
        Shop shop = nearObject.GetComponent<Shop>();
        shop.Exit();
        isShop = false;
        nearObject = null;
    }
}
1 reference
void interaction()
{
    if (nearObject != null)
    {
        if(nearObject.tag == "Weapon")
        {
            Item item = nearObject.GetComponent<Item>();
            int weaponIndex = item.value;
            hasWeapons[weaponIndex] = true;

            Destroy(nearObject);
        }
        else if (nearObject.tag == "Shop")
        {
            Shop shop = nearObject.GetComponent<Shop>();
            shop.Enter(this);
            isShop = true;
        }
    }
}
1 reference
void Attack()
{
    if (equipWeapon == null)
    {
```



```
359         {
360             return;
361         }
362         fireDelay += Time.deltaTime;
363         isFireReady = equipWeapon.rate < fireDelay;
364 
365         if (fDown && isFireReady && !isSwap && !isDodge && !isShop)
366         {
367             equipWeapon.Use();
368             animator.SetTrigger(equipWeapon.type == Weapon.Type.Melee ? "doSwing" : "doShot");
369             fireDelay = 0;
370         }
371     }
        1 reference
372     void Reload()
373     {
374         if(equipWeapon == null)
375         {
376             return;
377         }
378         if (equipWeapon.type == Weapon.Type.Melee)
379         {
380             return;
381         }
382         if (ammo <= 0)
383         {
384             return;
385         }
386         if (eDown && cc.isGrounded && !isDodge && !isSwap && isFireReady && !isShop)
387         {
388             animator.SetTrigger("doReload");
389             ReloadSound();
390             isReload = true;
391             Invoke("ReloadOut", 0.5f);
392         }
393     }
        0 references
394     void ReloadOut()
395     {
396         int reAmmo = ammo < equipWeapon.maxAmmo ? ammo : equipWeapon.maxAmmo;
397         ammo -= reAmmo;
398         ammo += equipWeapon.curAmmo;
399         equipWeapon.curAmmo = reAmmo;
400         isReload = false;
401     }
402 
        Unity Message | 0 references
403     void OnTriggerEnter(Collider other)
404     {
405         if (other.tag == "Item")
406         {
407             Item item = other.GetComponent<Item>();
408             switch (item.type)
409             {
```

```
409             {
410                 case Item.Type.Ammo:
411                     {
412                         ammo += item.value;
413                         if (ammo > maxAmmo)
414                         {
415                             ammo = maxAmmo;
416                         }
417                         break;
418                     }
419                 case Item.Type.Coin:
420                     {
421                         coin += item.value;
422                         if (coin > maxCoin)
423                         {
424                             coin = maxCoin;
425                         }
426                         break;
427                     }
428                 case Item.Type.Heart:
429                     {
430                         health += item.value;
431                         if (health > maxHealth)
432                         {
433                             health = maxHealth;
434                         }
435                         break;
436                     }
437                 case Item.Type.Grenade:
438                     {
439                         hasGrenades += item.value;
440                         if (hasGrenades > maxHasGrenades)
441                         {
442                             hasGrenades = maxHasGrenades;
443                         }
444                         break;
445                     }
446             }
447             Destroy(other.gameObject);
448         }
449         else if (other.tag == "EnemyBullet")
450         {
451             if (!isDamage)
452             {
453                 Bullet enemyBullet = other.GetComponent<Bullet>();
454                 health -= enemyBullet.damage;
455 
456                 StartCoroutine(OnDamage());
457             }
458         }
459     }
        1 reference
460     IEnumerator OnDamage()
461     {
            isDamage = true;
```

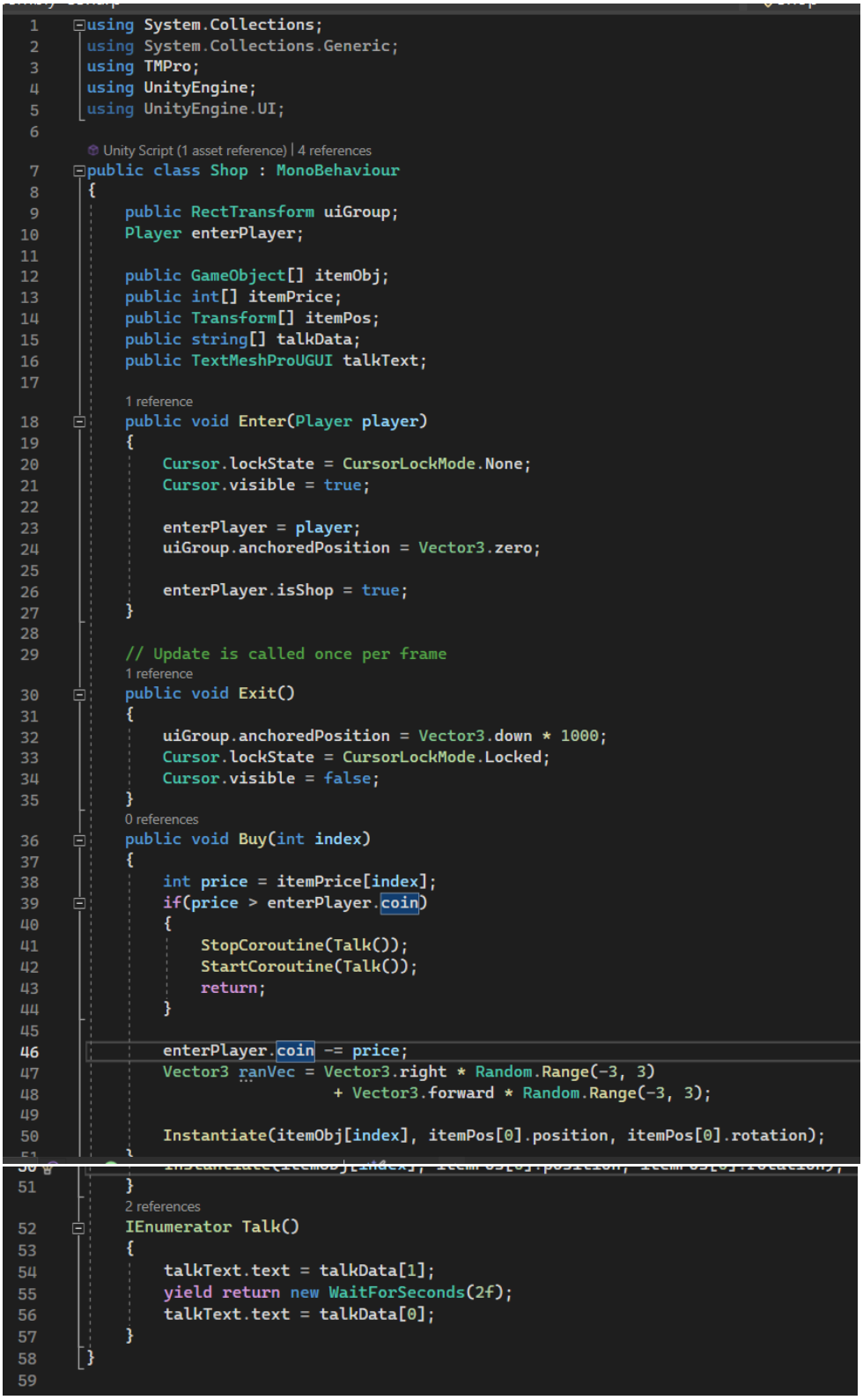


```
462         isDamage = true;
463         foreach(MeshRenderer mesh in meshs)
464         {
465             if(health >= 0.5f * health)
466             {
467                 mesh.material.color = Color.red;
468             }
469         }
470         yield return new WaitForSeconds(1f); // Player will not get damage for 1 second
471
472         isDamage = false;
473         foreach(MeshRenderer mesh in meshs)
474         {
475             mesh.material.color = Color.white;
476         }
477     }
478
        1 reference
479     void OnDie()
480     {
481         manager.GameOver();
482     }
483
484     //void FreezRotation()
485     //{
486     //    rigid.angularVelocity = Vector3.zero;
487     //}
488     //void FixedUpdate()
489     //{
490     //    FreezRotation();
491     //}
492 }
493
```

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

Unity Script (6 asset references) | 7 references
public class Weapon : MonoBehaviour
{
    6 references
    public enum Type { Melee, Range};
    public Type type;
    public int damage;
    public float rate;
    public BoxCollider meleeArea;
    public TrailRenderer trailEffect;

    public Transform bulletPos;
    public GameObject bullet;
    public Transform bulletCasePos;
    public GameObject bulletCase;

    public int maxAmmo;
    public int curAmmo;

    public AudioSource myHammerSfs;
    public AudioClip HammerSfs;

    public AudioSource myGunSfs;
    public AudioClip GunSfs;

    Player player;

    0 references
    public void HammerSound()
    {
        myHammerSfs.PlayOneShot(HammerSfs);
    }

    1 reference
    public void GunSound()
    {
        myGunSfs.PlayOneShot(GunSfs);
    }

    1 reference
    public void Use()
    {
        if(type == Type.Melee)
        {
            StopCoroutine("Swing");
            StartCoroutine("Swing");
        }
        else if(type == Type.Range && curAmmo > 0)
        {
            curAmmo--;
            StartCoroutine("Shot");
```

```
            StartCoroutine("Shot");
        }
    }
    0 references
    IEnumerator Swing()
    {
        // 1
        yield return new WaitForSeconds(0.1f);
        meleeArea.enabled = true;
        trailEffect.enabled = true;

        // 2
        yield return new WaitForSeconds(0.3f);
        meleeArea.enabled = false;

        //3
        yield return new WaitForSeconds(0.3f);
        trailEffect.enabled = false;
    }
    0 references
    IEnumerator Shot()
    {
        // 1 Start to shot
        GameObject instantBullet = Instantiate(bullet, bulletPos.position, bulletPos.rotation);
        Rigidbody bulletRigid = instantBullet.GetComponent<Rigidbody>();

        if (Input.GetKey(KeyCode.Mouse0))
        {
            GunSound();
        }

        bulletRigid.velocity = bulletPos.forward * 50; // 50 is bullet speed

        yield return null;

        // 2 Discharge bullets
        GameObject instantCase = Instantiate(bulletCase, bulletCasePos.position, bulletCasePos.rotation);
        Rigidbody caseRigid = instantCase.GetComponent<Rigidbody>();
        Vector3 caseVec = bulletCasePos.forward * Random.Range(-3, -2) + Vector3.up * Random.Range(2, 3);
        caseRigid.AddForce(caseVec, ForceMode.Impulse);
        caseRigid.AddTorque(Vector3.up * 10, ForceMode.Impulse);
    }
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

    }
}
```

iv. Players can buy items



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

Unity Script (1 asset reference) | 4 references
public class Shop : MonoBehaviour
{
    public RectTransform uiGroup;
    Player enterPlayer;

    public GameObject[] itemObj;
    public int[] itemPrice;
    public Transform[] itemPos;
    public string[] talkData;
    public TextMeshProUGUI talkText;

    1 reference
    public void Enter(Player player)
    {
        Cursor.lockState = CursorLockMode.None;
        Cursor.visible = true;

        enterPlayer = player;
        uiGroup.anchoredPosition = Vector3.zero;

        enterPlayer.isShop = true;
    }

    // Update is called once per frame
    1 reference
    public void Exit()
    {
        uiGroup.anchoredPosition = Vector3.down * 1000;
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
    }
    0 references
    public void Buy(int index)
    {
        int price = itemPrice[index];
        if(price > enterPlayer.coin)
        {
            StopCoroutine(Talk());
            StartCoroutine(Talk());
            return;
        }

        enterPlayer.coin -= price;
        Vector3 ranVec = Vector3.right * Random.Range(-3, 3)
                         + Vector3.forward * Random.Range(-3, 3);

        Instantiate(itemObj[index], itemPos[0].position, itemPos[0].rotation);
    }
    2 references
    IEnumerator Talk()
    {
        talkText.text = talkData[1];
        yield return new WaitForSeconds(2f);
        talkText.text = talkData[0];
    }
}
```

v. Bullets for enemies and players will be destroyed when they touch a floor or out of a map

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

Unity Script (6 asset references) | 4 references
public class Bullet : MonoBehaviour
{
    public int damage;

    Unity Message | 0 references
    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.tag == "Floor")
        {
            Destroy(gameObject, 3);
        }
    }
    Unity Message | 0 references
    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.tag == "Wall")
        {
            Destroy(gameObject);
        }
    }

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

    }
}
```

vi. Items are rotating and creates randomly on the map

sembly-CSharp

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

Unity Script (7 asset references) | 10 references
public class Item : MonoBehaviour
{
    5 references
    public enum Type { Ammo, Coin, Grenade, Heart, Weapon};
    public Type type;
    public int value;
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        transform.Rotate(Vector3.up * 40 * Time.deltaTime);
    }
}
```

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

Unity Script (2 asset references) | 0 references
public class RandomItemForAmmo : MonoBehaviour
{
    public float currTime;
    public GameObject ammo;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        // make time to go
        currTime += Time.deltaTime;


        // the enemy will automatically appear in 5 seconds
        if (currTime > 30)
        {
            // x,y,z random axis
            float newX = Random.Range(-10f, 40f), newY = Random.Range(4.5f, 5f), newZ = Random.Range(-30f, 50f);

            // move the object to the position
            ammo.transform.position = new Vector3(newX, newY, newZ);

            // bring object where positions
            Instantiate(ammo, new Vector3(newX, newY, newZ), Quaternion.identity);

            // when it reaches 10 seconds, currTime is initialised 0
            currTime = 0;
            Debug.Log("Ammo Created!");
        }
    }
}
```

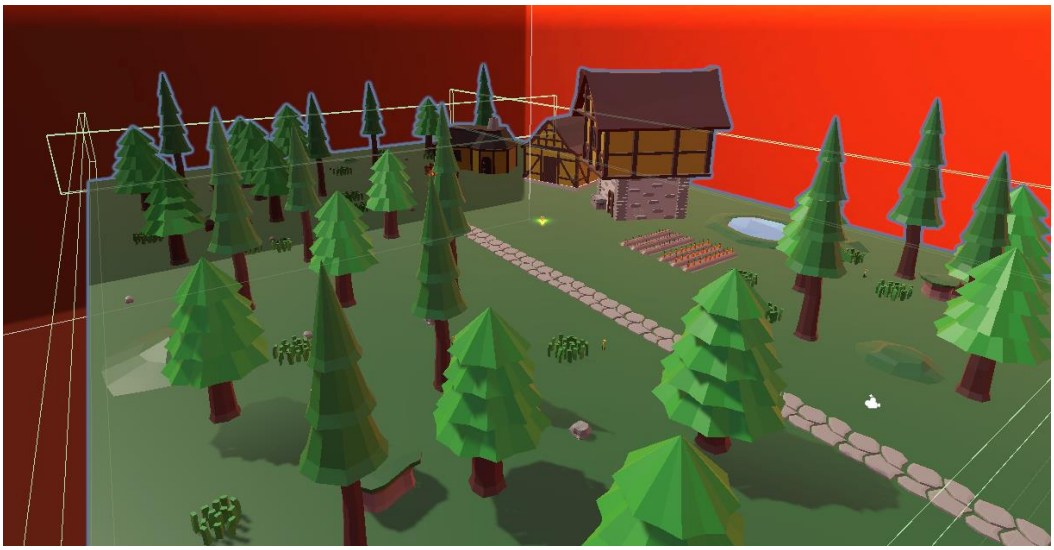
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script (2 asset references) | 0 references
public class RandomItemForHeart : MonoBehaviour
{
    public float currTime;
    public GameObject heart;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        // make time to go
        currTime += Time.deltaTime;


        // the enemy will automatically appear in 5 seconds
        if (currTime > 60)
        {
            // x,y,z random axis
            float newX = Random.Range(-10f, 40f), newY = Random.Range(4.5f, 5f), newZ = Random.Range(-30f, 50f);

            // move the object to the position
            heart.transform.position = new Vector3(newX, newY, newZ);

            // bring object where positions
            Instantiate(heart, new Vector3(newX, newY, newZ), Quaternion.identity);

            // when it reaches 10 seconds, currTime is initialised 0
            currTime = 0;
            Debug.Log("Heart Created!");
        }
    }
}
```

c) Game Progression: The progression is Main – Settings(optional) – Game – Game Over

Main)

Settings)

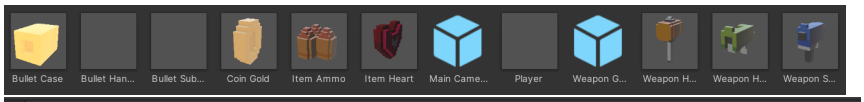
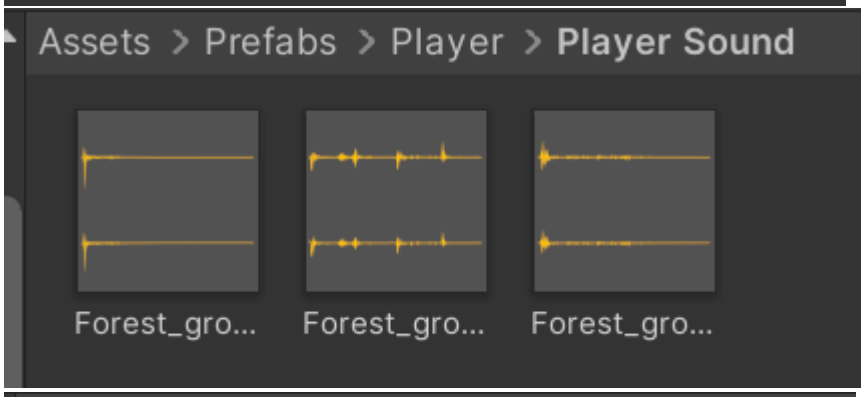
Game) Including 3 Levels



Game Over)

d) Game Resources

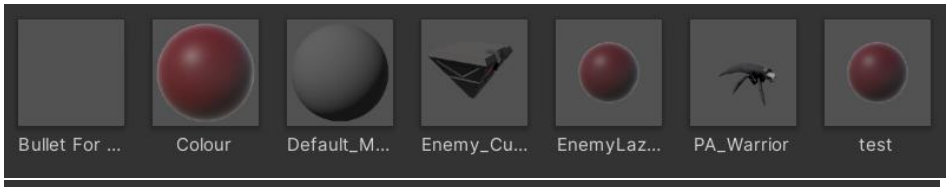
i. Player Prefebs

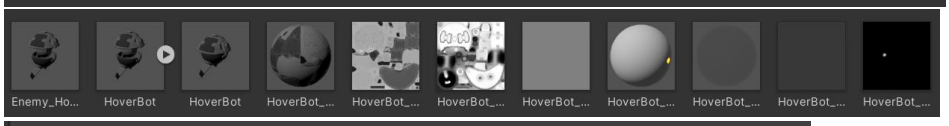
- Player
- Bullt Case
- Bullet Handgun
- Bullet Machinegun
- Coin Gold
- Item Ammo
- Item Heart
- Main Camera
- Weapon Hammer
- Weapon Handgun
- Weapon Machinegun
- Player's Sounds (Walking, Running, Jump, Weapons)
  - 
  - 

  - 

ii. Enemy Prefebs

- Bullets For Enemy
- Enemy Cube
- PA Warrior
- However Bot
- Enemies' Sounds (Attack)

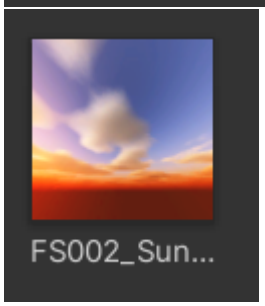
- 

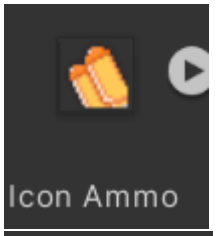
- 

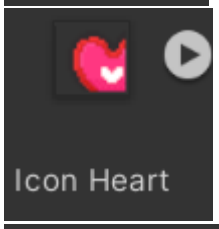
- 

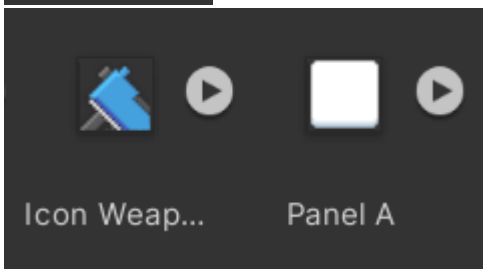
iii. Others

- BGM (Audio Mixer)
- Canvas
- Crate
- Dead Image
- Dreamy Ambient (background sound)
- Event System
- Game Manager
- Map
- Menu Camera
- Roboto Font
- FS002 Sunset Cubemap
- Icons
  - 

  - 

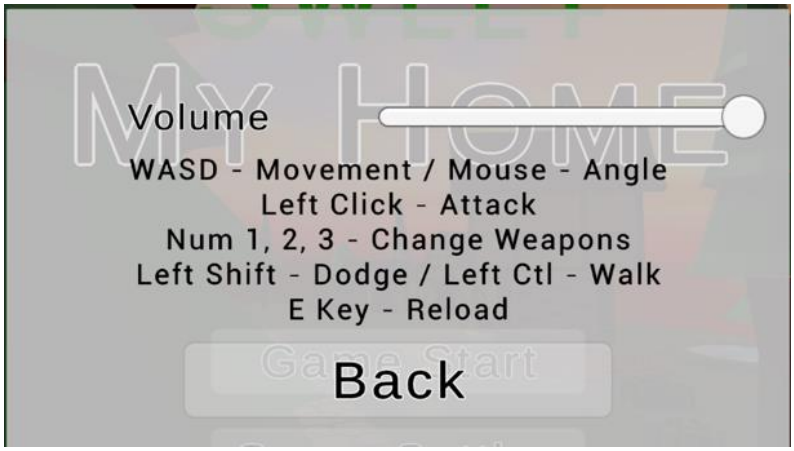
- 

- 

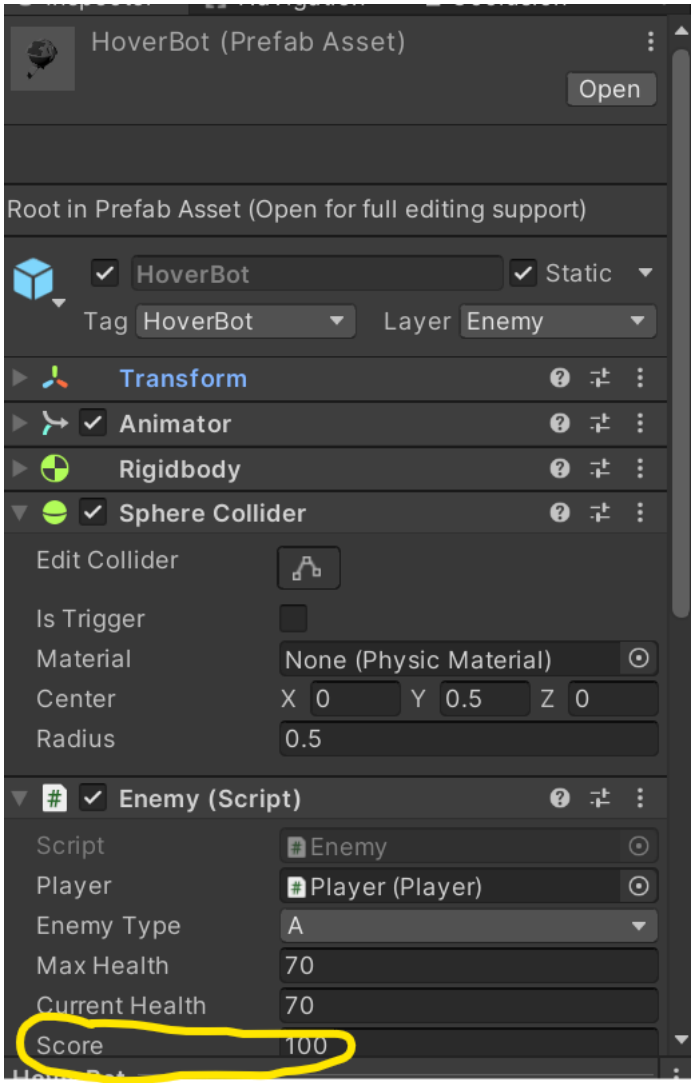
- 

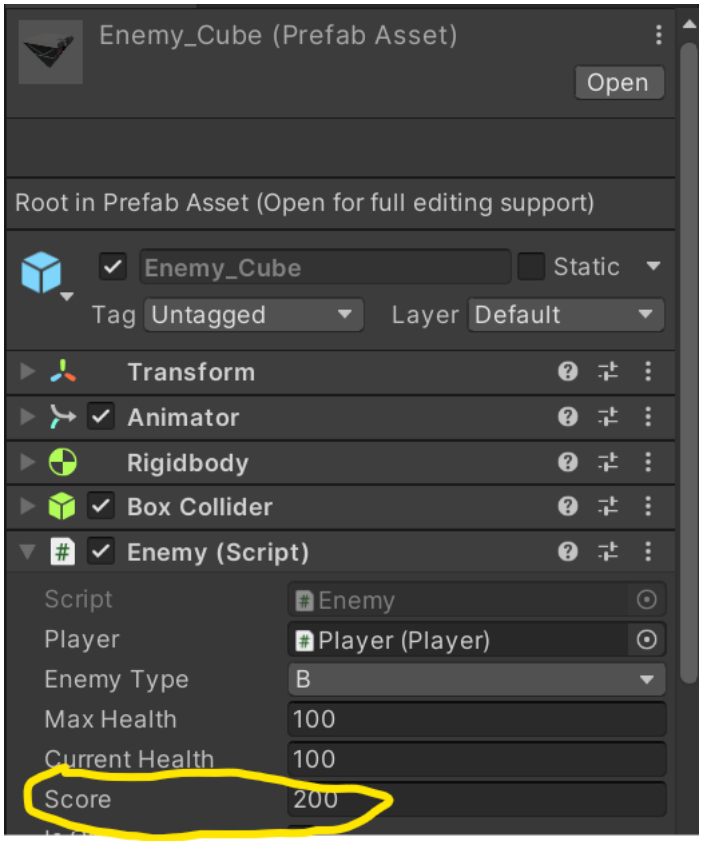
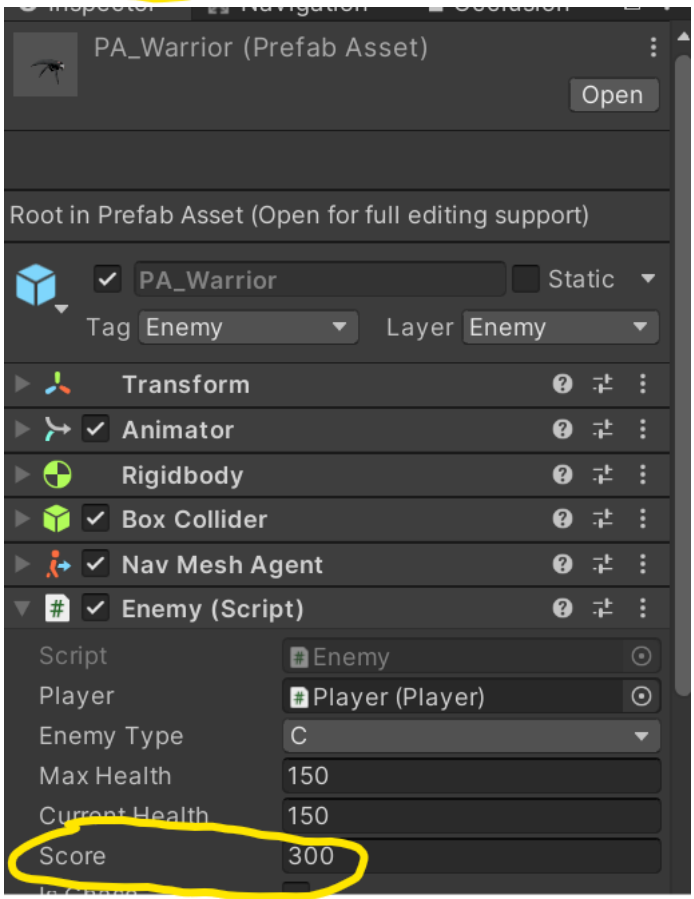
e) Game Mechanics

I. Movement

- 

II. In-Game Info: The 3 enemies will give a player rewards as different scores of 100, 200, and 300 in each 1, 2, and 3 level.

- 
HoverBot (Prefab Asset)
Open
Root in Prefab Asset (Open for full editing support)
HoverBot
Static
Tag HoverBot
Layer Enemy
Transform
Animator
Rigidbody
Sphere Collider
Edit Collider
Is Trigger
Material None (Physic Material)
Center X 0 Y 0.5 Z 0
Radius 0.5
Enemy (Script)
Script Enemy
Player Player (Player)
Enemy Type A
Max Health 70
Current Health 70
Score 100

- 
Enemy_Cube (Prefab Asset)
Open
Root in Prefab Asset (Open for full editing support)
Enemy_Cube
Static
Tag Untagged
Layer Default
Transform
Animator
Rigidbody
Box Collider
Enemy (Script)
Script Enemy
Player Player (Player)
Enemy Type B
Max Health 100
Current Health 100
Score 200
- 
PA_Warrior (Prefab Asset)
Open
Root in Prefab Asset (Open for full editing support)
PA_Warrior
Static
Tag Enemy
Layer Enemy
Transform
Animator
Rigidbody
Box Collider
Nav Mesh Agent
Enemy (Script)
Script Enemy
Player Player (Player)
Enemy Type C
Max Health 150
Current Health 150
Score 300

-

```
2 references
IEnumerator OnDamage(Vector3 reactVec)
{
    mat.color = Color.red;
    yield return new WaitForSeconds(0.1f);

    if(currentHealth > 0)
    {
        mat.color = Color.white;
    }
    else
    {
        if (isDamage)
        {
            mat.color = Color.gray;
            gameObject.layer = 13;
            isChase = false;
            isAttack = false;
            nav.enabled = false;

            anim.SetTrigger("doDie");

            reactVec = reactVec.normalized;
            reactVec += Vector3.up;

            rigid.AddForce(reactVec * 4f, ForceMode.Impulse);

            player.score += score;

            Destroy(gameObject, 2f);
            isDamage = false;
        }
    }
}
```

Task 3: Source Code Implementation

- The script files are attached in the file named as Scripts.

Part B

Task 1: Test Plan and Development

i. **Test 1**
   - Test Objective: To check whether the "Game Setting" button in the main functions appropriately
   - Test Description: The button will be executed by the tester
   - Test Condition: See Test Environment
   - Expected Results: The "Setting Panel" will appear

ii. **Test 2**
   - Test Objective: To check whether the "Back" button in the "Setting Panel" functions appropriately
   - Test Description: The button will be executed by the tester
   - Test Condition: See Test Environment
   - Expected Results: The tester will see the main

iii. **Test 3**
   - Test Objective: To check whether the "Game Start" button in the main functions appropriately
   - Test Description: The button will be executed by the tester
   - Test Condition: See Test Environment
   - Expected Results: The "Menu Camera" will be set off, while the "Main Camera" in "Player" will be set on, then the game will begin.

iv. **Test 4**
   - Test Objective: To check whether the tester controls the "Player" character's movement in the game appropriately
   - Test Description: For the test, the tester will command with W, S, A, D, Space Bar, Left Shift, Left Ctrl, and Mouse
   - Test Condition: See Test Environment
   - Expected Results: The character will move properly

v. **Test 5**
   - Test Objective: To check whether the "Player" character will be damaged, and its HP points will be lost when robots attack
   - Test Description: For the test, the tester will let the character get damaged by robots
   - Test Condition See Test Environment

- Expected Results: The character will get damaged and lose its HP becoming its skin red

vi. Test 6

- Test Objective: To check whether the "Player" character can get items, and they work properly
- Test Description: For the test, the tester will touch the items including weapons, bullets, and hearts on the ground and the items will disappear
- Test Condition: See Test Environment
- Expected Results: the player can attack with weapons and its HP and Ammo can be filled with items

vii. Test 7

- Test Objective: To check whether the "Player" character can damage the enemy robots with weapons
- Test Description: For the test, the tester will attack with the 3 weapons and check that the robots will be damaged by them
- Test Condition: See Test Environment
- Expected Results: When the robots' HP runs out, they are destroyed, and their colour becomes red when they get damaged

viii. Test 8

- Test Objective: To check whether the "Player" character can buy items in the shop shaped like a box
- Test Description: For the test, the tester will purchase 3 different items with coins, however, he cannot buy the items if does not have enough coins
- Test Condition: See Test Environment
- Expected Results: The tester can get items with coins in the shop, but cannot purchase when does not have enough coins

ix. Test 9

- Test Objective: To check whether the enemy robots drop the coin item when die
- Test Description: For the test, the tester will kill robots with weapons and check they will drop the coin, which the percentage is 50
- Test Condition: See Test Environment
- Expected Results: The coin item to rotate will appear when robots are killed

x. Test 10

- Test Objective: To check whether the "Main Title" button in the game functions appropriately
- Test Description: The button will be executed by the tester
- Test Condition: See Test Environment

- Expected Results: The main scene will appear

Task 2: Text Execution

a) Execute all ten (10) test cases and record the results

i. Test 1

- Action Performed: Click the "Game Setting" button in the main
- Tester: Euisoek Jeong
- Environment: Windows 10
- Date: 4/10/2023
- Conclusion: Pass

ii. Test 2

- Action Performed: Click the "Back" button in the "Setting Panel"
- Tester: Euisoek Jeong
- Environment: Windows 10
- Date: 4/10/2023
- Conclusion: Pass

iii. Test 3

- Action Performed: Click the "Game Start" button in the main
- Tester: Euisoek Jeong
- Environment: Windows 10
- Date: 4/10/2023
- Conclusion: Pass

iv. Test 4

- Action Performed: Use the keyboard button and mouse
- Tester: Euisoek Jeong
- Environment: Windows 10
- Date: 4/10/2023
- Conclusion: Pass

v. Test 5

- Action Performed: use the keyboard button and mouse to get damaged
- Tester: Euisoek Jeong
- Environment: Windows 10
- Date: 4/10/2023
- Conclusion: Pass

vi. **Test 6**

- Action Performed: use the keyboard button and mouse to get items on the ground, and use it properly
- Tester: Euisoek Jeong
- Environment: Windows 10
- Date: 4/10/2023
- Conclusion: Pass

vii. **Test 7**

- Action Performed: use the keyboard button and mouse to damage robot enemies
- Tester: Euisoek Jeong
- Environment: Windows 10
- Date: 4/10/2023
- Conclusion: Pass

viii. **Test 8**

- Action Performed: use the keyboard button and mouse to buy items with coins in the shop
- Tester: Euisoek Jeong
- Environment: Windows 10
- Date: 4/10/2023
- Conclusion: Pass

ix. **Test 9**

- Action Performed: use the keyboard button and mouse to damage robot enemies dropping coins
- Tester: Euisoek Jeong
- Environment: Windows 10
- Date: 4/10/2023
- Conclusion: Pass

x. **Test 10**

- Action Performed: Click the "Main Title" button in the game
- Tester: Euisoek Jeong
- Environment: Windows 10
- Date: 4/10/2023
- Conclusion: Pass

b) Description of Bugs and Errors

I have encountered some bugs and errors while developing this game, for instance, one of them is the slide bar for BGM. I had done to make an audio mix including

background and effect sounds, because I did not import UnityEngine.UI embedded in Unity. On the other hand, I had another issue similar to the above one. It was a text issue that could not read letters. The reason for the error was that I should have used TextMeshProUGUI, not Text which was an old version.

c) Priority of identified bugs or errors in terms of importance and urgency

i. Slide-Bar: High Priority – The reason why it was given high priority is making it creates an occurrence of errors not enabling the game start. It should be immediately fixed first for users to play.

ii. Text: low Priority – This issue is less necessary to play the game, but it is true that players are interested in the sound quality of the game, which makes the game more concentrated. Poor sound parts in the game can reduce their satisfaction.

d) Troubleshooting and Debugging of Bugs: Here are 6 processes to troubleshoot and debug for the above issues

i. Identify the Bug: Understand and document the specific issue, including expected and actual behavior.

ii. Check Settings: Verify that Slider and Text component settings are configured correctly.

iii. Debug Logs: Use Debug.Log statements to print relevant information in your code.

iv. UI Interaction: Ensure that UI interactions and events are set up correctly.

v. Reproduce the Issue: Try to consistently reproduce the bug to understand when and how it occurs.

vi. Code Check: Ensure using a compatible Unity version for coding or import proper Engine

e) Modification of Codes and Assets for Bugs

Firstly, I identified the bug, and began comprehensively understanding the bug what errors caused in the codes. In addition, I examined the scripts and UI components associated with both the slider bar and text with utilising the internet search for proper and current syntax for Unity and debugging tools like "Debug.Log" in proper breakpoints, which could track where the bugs were caused. Lastly, once I finished to code, I applied the slide bar and text into the component in order to check for the final step.

f) Test for the Modified Codes and Assets

During the test planning steps, the code enhancements were implemented to improve the game quality. These changes facilitated the detection and resolution of potential

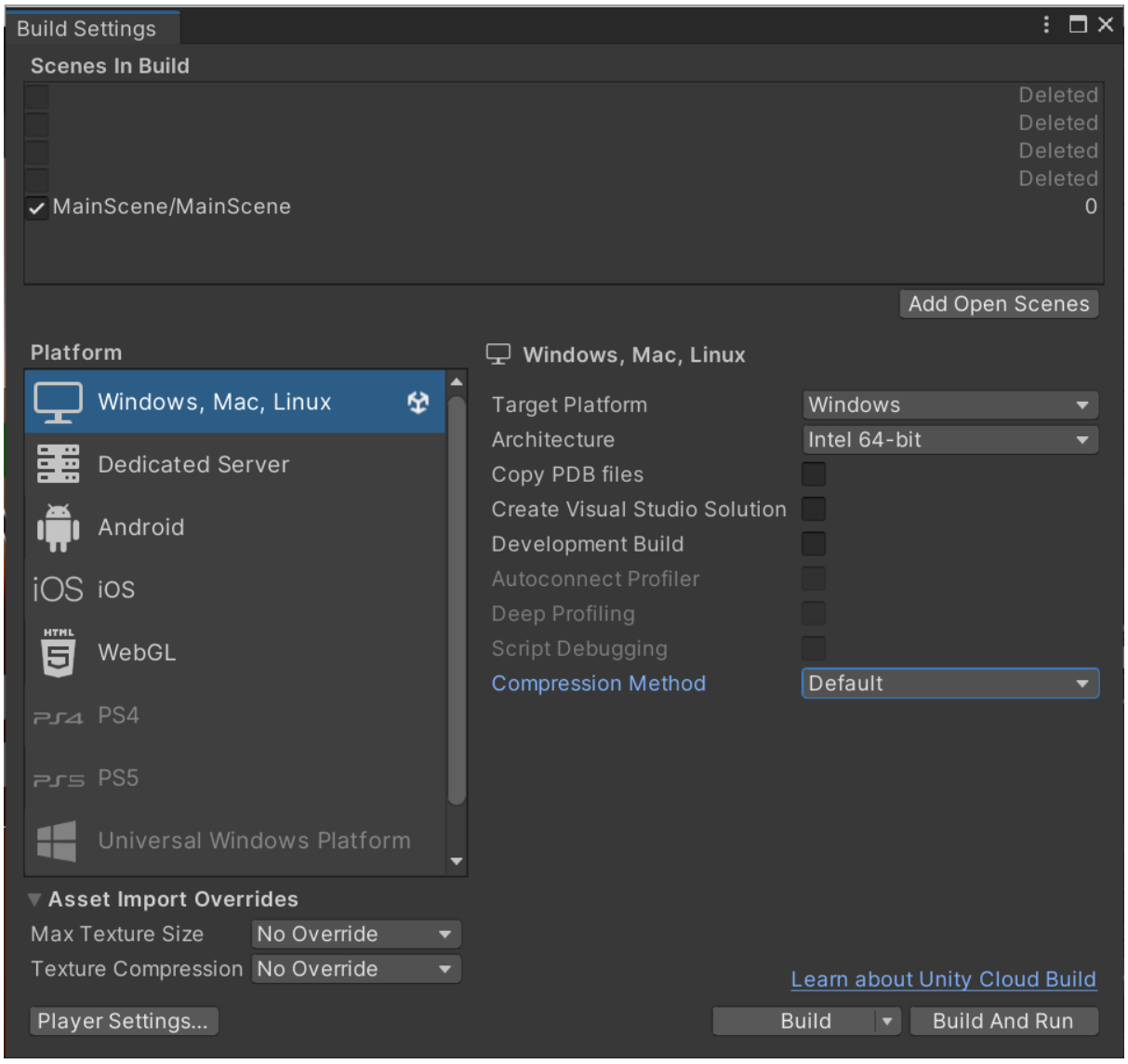
issues that might affect the UX, making sure that the game adhered to the anticipated criteria for functionality and user-friendliness.

Part C

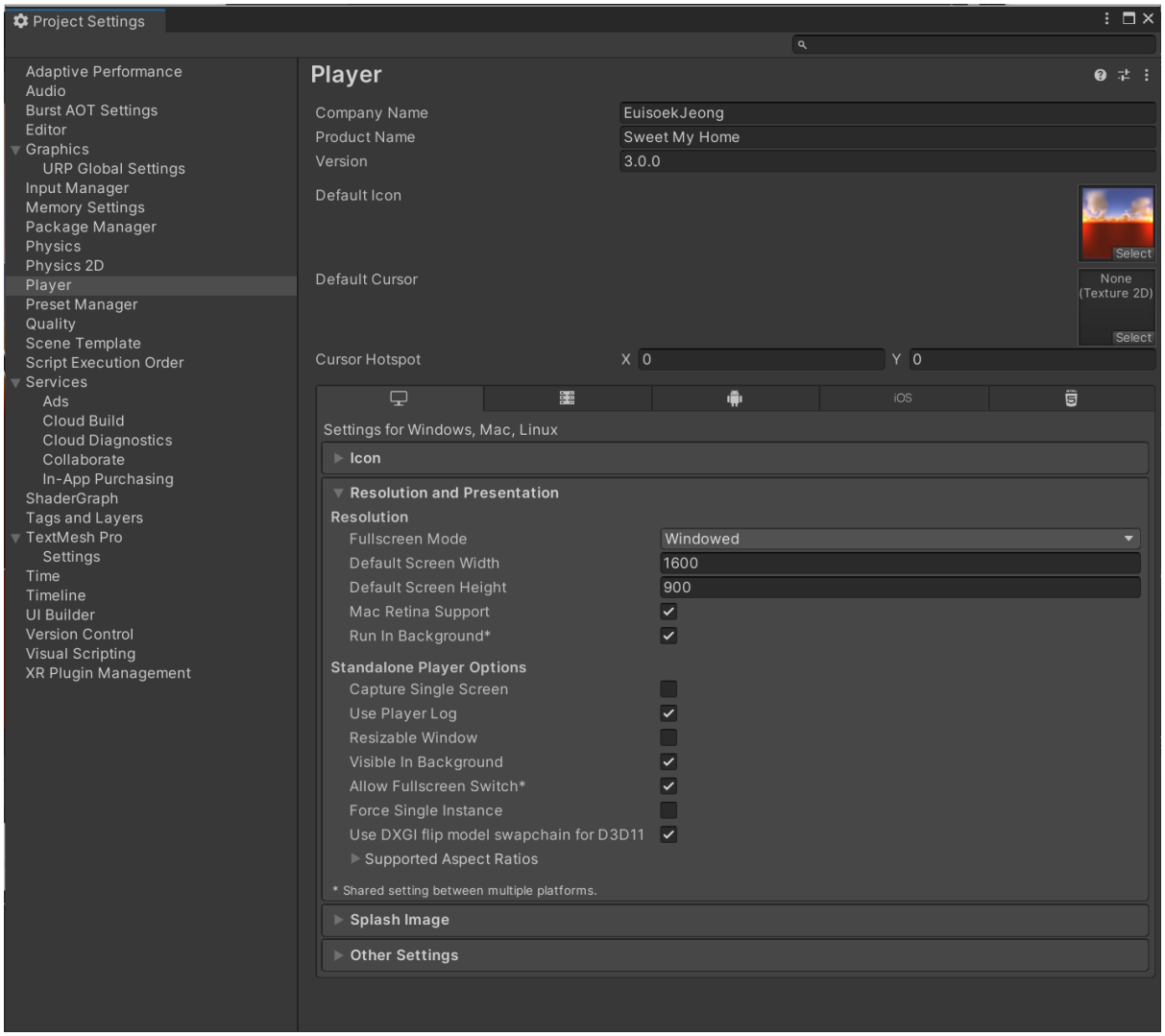
Task 1: Game Build

Window)

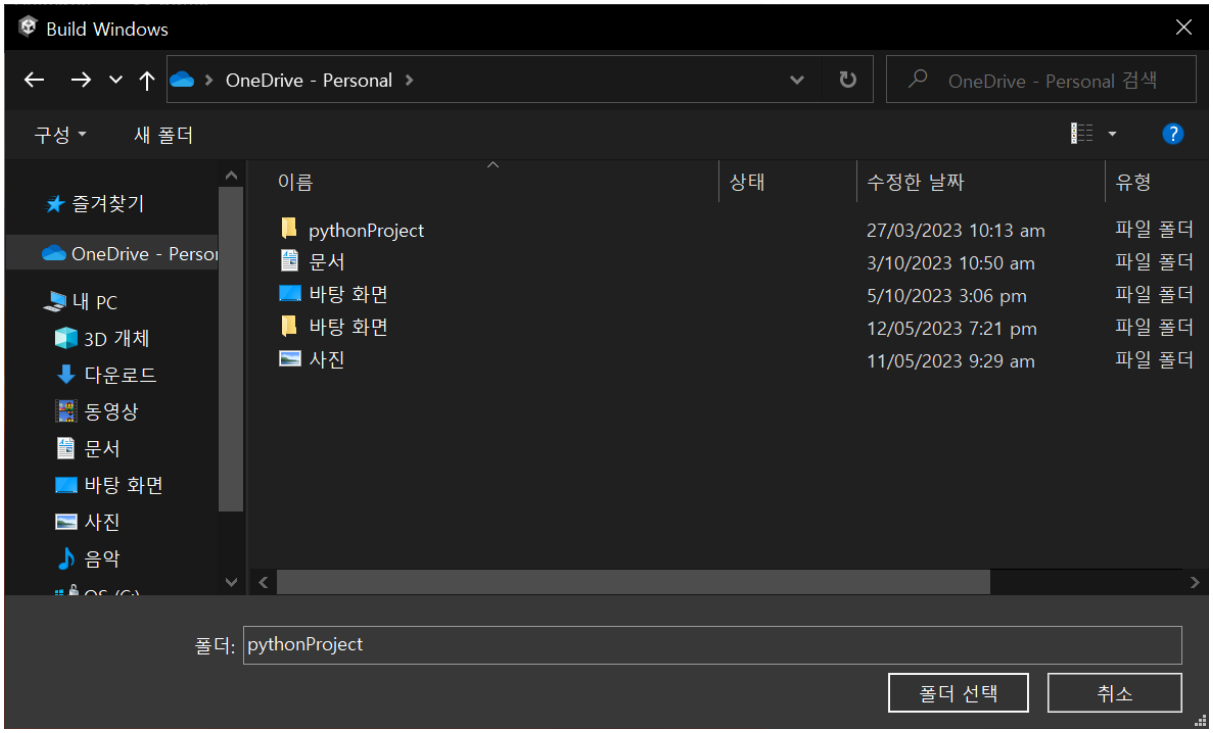
Firstly, open the project and select Build Settings for the File menu. For platform settings, select Windows, Mac, Linux in the Build Settings, then select Windows as Target Platform.

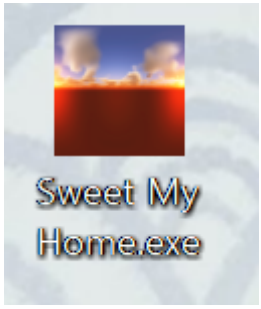


For the next step, go to Player Settings to edit game settings such as icon, name, and version. In addition, it is available to choose the size of the game.



Finally, in the Build Settings again, click the Build button to select the folder and file name to build. When the process is done, it is possible to find the exe file in where you choose to download it.


Build Windows
OneDrive - Personal
OneDrive - Personal 검색
구성
새 폴더
즐겨찾기
OneDrive - Perso
내 PC
3D 개체
다운로드
동영상
문서
바탕 화면
사진
음악
이름
상태
수정한 날짜
유형
pythonProject
27/03/2023 10:13 am
파일 폴더
문서
3/10/2023 10:50 am
파일 폴더
바탕 화면
5/10/2023 3:06 pm
파일 폴더
바탕 화면
12/05/2023 7:21 pm
파일 폴더
사진
11/05/2023 9:29 am
파일 폴더
폴더: pythonProject
폴더 선택
취소


Sweet My
Home.exe