VRTech.TermProjectReport.Li

Name: Dazhi Li

RUID: 197007456

YouTube Link: https://youtu.be/wqpJqDQJTYs
Backup YouTube Link: https://www.youtube.com/watch?v=wqpJqDQJTYs&feature=youtu.be

Google Drive Share (project file):
https://drive.google.com/open?id=138ZkWjcZoeZVpSzHB55Nh50D-K0ZaX5Q

1. Abstract

The scene I created in my final project is a FPS game. Player are assigned with mission to steal the confidential documents in a huge military base with soldiers around it. See **Figure 1** a brief view of my own designed military base. These simple models are made by myself but something like towers and soldiers are from unity asset store. The user will be respawned on a helicopter base which is near the military base. See **Figure 2** you will find out there are two enemies for user to practice shooting and understand how those intelligent objects are moving in the scene. And there is a special road I made for player to get into the military base safely without rushing in the front gate where there are many intelligent soldiers. After player get the confidential document in the military base, the mission will be changed to retreat safely from the military base. User will have a basic health point and damage system in the game. Once player is dead in my scene, the whole game will restart in the next few seconds. And when player successfully finish all the missions above, the whole game is over and a celebration image will be shown on the screen. Now, let's talk about some details.

Figure 1 Military Base

Figure 2 Heli base

2. Details

2.1 Target and mission system

Player's role in my game is a secret agent, whose job is stealing a confidential document from enemy's military weapon. Player are free to fire or just keep silent and steal it. The confidential document is hidden somewhere in the scene, I did it in a tall building with two soldiers watching it. The document is marked as a target in **Figure 3**. Once the user step into the targeted area, the mission will automatically changed and tell players they have already gotten the confidential area.

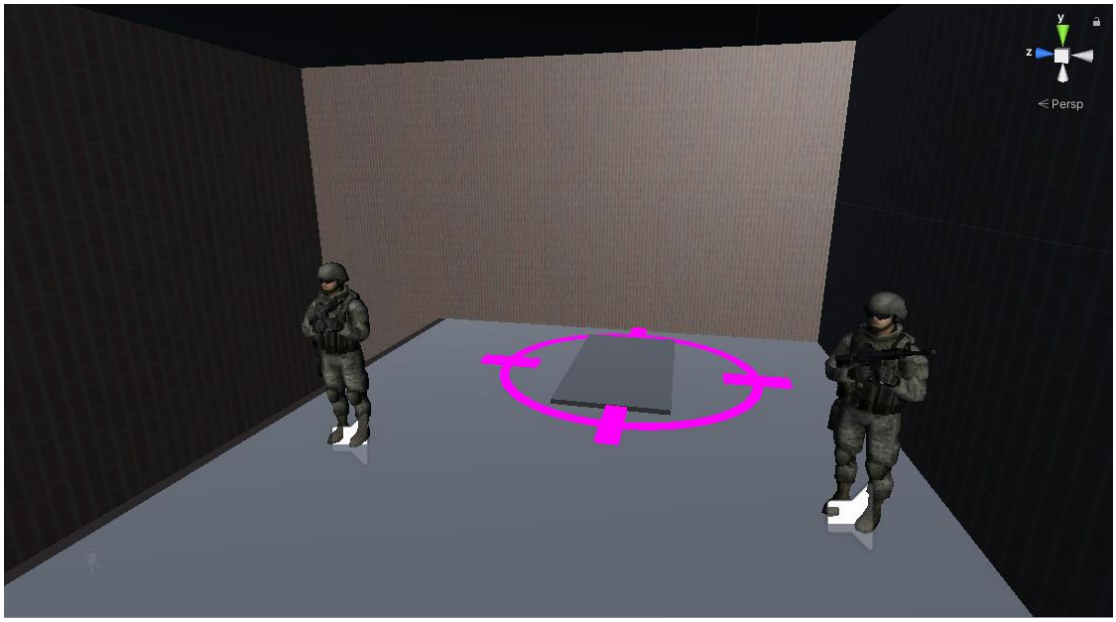

Figure 3 Confidential Document

After user get the confidential document, user's first mission is changed to retreat to the helicopter base where the player is respawned. If successfully reaching the military base, player will be shown a celebration picture on the screen like **figure 4**. After five seconds, the editor or game will automatically quit.

Figure 4 Mission System

2.2 FPS controller & Guns

In this project I mainly use FPS controller provided by Easy FPS assets, guns are provided by the same assets, but player are free to add their own guns and set the parameters to the gun. Each gun has its own fire sound clip and player has reload clip, jump clip, running clip etc. The gun will be instantiated from prefabs when the player is respawned. Guns are divided into two kinds, semi or auto. You can watch the YouTube video to see how semi and auto guns work in this project. The gun has its own fire animation which shows the fire muzzle. I made gun fire to the enemy by setting a place where muzzles respawned. Then I instantiate a muzzle there and play a fire sound clip to make it realistic.

Figure 5 Fire Muzzles

2.3 Enemy or solid object recognition

Since we know when a bullet hit a human or a solid object like walls, the performance or the feedback is totally different. So I made tags to those objects. By different tags like “level part” will indicate that the user is shooting a solid object. Then a bullet hole prefab will be instantiated at the bullet hit place. If it is a enemy, a bleeding effect will be instantiated. Those are written in my scripts and shown in **figure 6**.

Figure 6 Feedback system

There is also something very important. It’s how we eject our bullets. Actually I use the center of our main camera and did a raycast hit. The main idea is suppose there is a ray ejected from the center of the main camera and this ray will hit a game object when player shoot a bullet. Then I did a decision that if the hit object is enemy or solid objetcs.

2.4 Damage and health system

As we all know the most important part in a FPS game is how do we calculate damage and give player enough freedom to make some mistakes by damage system. The player can kill any enemy directly by one shot. But the player is given a health system which has a maximum health point (hp) of 100. And players are allowed to set it if they just want to more challenges or want to be a superhero. The health point will decree immediately when enemy's bullet hit player. But player can recover their hp by rest in time. There is hyperparameter I set to user is how fast the recover speed is. Player will get one hp per second in my game. Then let us talk about something about the damage system. Enemy could give 34 damage (adjustable) to player when directly hit the user. And user will be shown a bleeding effect on the screen. The bleeding effect is just the same as I have shown for the soldier bleeding effect. And there is a real time hp reader in the game shown in **Figure 7**.

Figure 7 Health Point and bleeding effect

2.5 Intelligent Agents

Developing intelligent agent is the hardest part in this project. When we want some interactivity in a FPS game, adding a bot enemy must the best choice. But how can we move our enemy with a brain in their body? That's where the problem is and I will explain it by two different kinds of intelligent agents with different behaviors in my term project.

Static guardian:

Static guardian is kind of an intelligent agents would not change their position in the game. Even player are no longer in their sight of view, they will not chasing player and continue shooting player. This kind of soldier are usually standing on a tower of watching a gate. They are shown in **figure 8**.

Figure 8 Static Guardians

Then I am going introduce how does a enemy see our players. A soldier has one capsule collider which is small to receive damage from player's bullet and another sphere collider in its child class to percept players. This is shown in **Figure 9**.

Figure 9 Enemy Perception Range

After we add a sphere collider to the enemy, we should also take it as a trigger. So when player in this trigger we can make first decision in our scripts. Also, we need to add the game object which is the enemy's mesh to ignore ray cast layer or player's bullet will shoot on them. Then we need to define a field of view (FOV) to the enemy. The main idea is shown in figure 10.

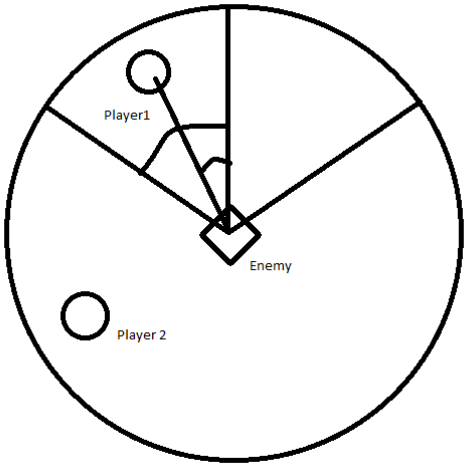

Figure 10 Enemy sight of view

Let us suppose that the diamond is enemy, the little circle is player's position. There are two kinds of player's position. Since we just talked that the big circle is enemy's perception range. We calculate the angle between player and the enemy's forward line. If the angle is less than half of enemy's FOV, then our player is sight of the enemy and enemy will see our user. If player is at position two which means the angle is larger than half of the FOV, enemy will not see our player. If we want to make it more realistic, we still need to add ray cast test. We suppose there is a ray ejected from the enemy's eyes, and this ray cast hit something. If the ray cast hit the user, we can deduce that the player is absolutely seen by the enemy. I did this just because I need to prevent there is something between the player and enemy but the enemy see the player which is not realistic.

These have not solved our problem that how to add a "brain" in our intelligent objects. As we just add perception to our intelligent object, we should add some behavior to our enemy agents. They are done by a combination of animator controller and scripts. I set a lot of bool values like if the player in sight to do this. I understand the animator controller is just like state machine. When Player not in sight they are in standing state. But when player in sight they are in combat state. And when bullets kill them they are in dead state. I did the rotation for enemy looking at our player in scripts and set a fire interval to them or they will continuously firing which will make the game too difficult for player. That's basically how I implement static guardian, more information can be checked through the google drive.

Patrol Guardian:

This is a kind of advanced intelligent agents to the previous one since this kind of agent will chase our player once in sight. Patrol guardian will follow a root I set to them to do patrol. I need to set patrol way points to them and they will start from the first and walking to the last one and then come back to the first to form a loop. This can be seen in **Figure 11**.

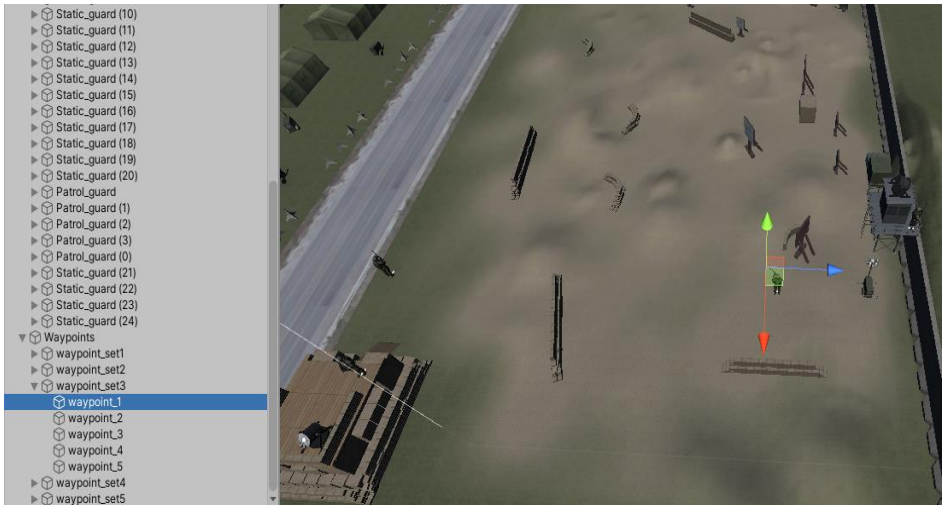

Figure 11 Patrol way points

Those way points are just empty objects but soldier will utilize their position information. And they are sored in a list for a soldier to move from one by one. The perception machine is just the same as the static guardian. But the animator controller is different. It has running state, stand shooting state and dead state. When they do not see the player they are running to patrol. And once they have seen the player, they will increase their moving speed to chase player and shooting. I set a stopping distance to them since in realistic a soldier would not running to the face of anyone with a gun. If they reach the stopping distance, they will change their state the stand shooting state. And when player running away, the state will be changed back to running state to chase our player. They will chase player until player is dead or they do. So, that's basically how I make intelligent agents in my project.

2.6 Model Management

Since it's a huge scene, we cannot render everything together with the full details. I set Level of Details (LOD) to most objects in the scene. LOD will help increase my fps at an average 100 fps. Then for modeling part, I did model something easy to model like the tall building and search light etc.

2.7 UI

UI are achieved by myself, which is a combination of raw image and text. Text shows the mission target and player hp. Picture will show if the player is dead or successfully finish the mission, shown in **figure 12**. After 5 seconds user will respawn at the helicopter base and the whole scene will be reloaded.

Figure 12 Mission failed

3. Scripts

EnemyRun

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

public class EnemyRun : MonoBehaviour
{
    public float patrolSpeed = 2f;
    public float chaseSpeed = 5f;
    public Transform[] patrolWayPoint;

    private EnemySight enemysight;
    private GameObject player;
    private PlayerHealth hp;
    private int wayPointIndex=0;
    private GameObject mover;
    private bool isdead;

    // Start is called before the first frame update
    void Start()
    {
        enemysight = this.GetComponent<EnemySight>();
        player = GameObject.FindGameObjectWithTag("Player");
        hp = player.GetComponent<PlayerHealth>();
        mover = transform.parent.gameObject;
    }

    // Update is called once per frame
```

```
        void Update()
        {
            isdead = this.GetComponent<Animator>().GetBool("isDead");
            if (enemysight.PlayerPosition != enemysight.resetPostion && !isdead)
            {
                Chasing();
            }
            else if(enemysight.PlayerPosition == enemysight.resetPostion && !isdead)
            {
                Patrol();
            }
        }

        private void Chasing()
        {
            Vector3 toward_player = player.transform.position - mover.transform.position;
            if(toward_player.sqrMagnitude>49f)
            {
                this.GetComponent<Animator>().SetBool("isStand", false);
                mover.transform.position = Vector3.MoveTowards(mover.transform.position, player.transform.position+transform.up, 0.01f *
chaseSpeed);
            }
            else
            {
                this.GetComponent<Animator>().SetBool("isStand",true);
            }
        }

        private void Patrol()
        {
            if(mover.transform.position==patrolWayPoint[wayPointIndex].position && wayPointIndex==patrolWayPoint.Length-1)
            {
                wayPointIndex = 0;
            }
            else if(mover.transform.position == patrolWayPoint[wayPointIndex].position)
            {
                wayPointIndex++;
            }
            mover.transform.LookAt(patrolWayPoint[wayPointIndex].position);
            mover.transform.position = Vector3.MoveTowards(mover.transform.position, patrolWayPoint[wayPointIndex].position, 0.01f *
patrolSpeed);
        }
}
```

EnemySight

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

public class EnemySight : MonoBehaviour
{
    public float FOV = 110f;
    public bool PlayerInSight;
    public Vector3 PlayerPosition;
    public Vector3 resetPostion = Vector3.zero;
    public GameObject fire;
    public GameObject fire_spawn;
    public float fire_interval;
    public GameObject BloodScreen;

    private SphereCollider col;
    private Animator anim;
    private GameObject player;
    private GameObject cube;
    private bool isdead;
    private AudioSource gun_sound;
    private float timebeforenextfire;
    private PlayerHealth hp;
    private bool first_time;
    private float last_time;
    private GameObject cam;

    void Start()
    {
        col = GetComponentInChildren<SphereCollider>();
        anim = this.GetComponent<Animator>();
        player = GameObject.FindGameObjectWithTag("Player");
        cube = transform.parent.gameObject;
        gun_sound = this.GetComponent<AudioSource>();
        hp = player.GetComponent<PlayerHealth>();
        first_time = true;
        cam = GameObject.FindGameObjectWithTag("MainCamera");
    }

    void OnTriggerStay(Collider other)
    {
        if(other.gameObject==player)
        {
            PlayerInSight = false;
```

```
            Vector3 direction = other.transform.position - transform.position;
            float angle = Vector3.Angle(direction, transform.forward);
            if(angle<FOV*0.5f)
            {
                RaycastHit hit;
                if(Physics.Raycast(transform.position,direction.normalized,out hit,col.radius))
                {
                    if(hit.collider.gameObject==player)
                    {
                        PlayerInSight = true;
                        anim.SetBool("PlayerInSight", true);
                        PlayerPosition = player.transform.position;
                    }
                }
            }
        }
    }

    void Update()
    {
        isdead = anim.GetBool("isDead");
        if (PlayerInSight && !isdead)
        {
            cube.transform.LookAt(player.transform.position+transform.up);
            if (first_time)
            {
                last_time = Time.time;
                first_time = false;
            }
            if (Time.time>timebeforenextfire&&Time.time>last_time+1f)
            {
                GameObject clone = Instantiate(fire, fire_spawn.transform.position, fire_spawn.transform.rotation);
                clone.SetActive(true);
                gun_sound.Play();
                timebeforenextfire = Time.time + fire_interval;
                if(true)
                {
                    int damage = 36;
                    hp.hp = hp.hp - damage;
                    GameObject blood = Instantiate(BloodScreen, cam.transform.position, cam.transform.rotation);
                }
            }
            if(hp.playerisdead)
            {
```

```
                    this.GetComponent<EnemySight>().enabled=false;
                }
            }
        }
    }
```

GetDocuments

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

    public class GetDocuments : MonoBehaviour
    {
        public bool GetDocument;
        public GameObject text_obj;

        private GameObject player;

        void Start()
        {
            player = GameObject.FindGameObjectWithTag("Player");
            GetDocument = false;
        }

        void OnTriggerEnter(Collider other)
        {
            if(other.gameObject==player)
            {
                GetDocument = true;
                text_obj.GetComponent<Text>().text = "File checked, retreat from our heli base";
            }
        }
}
```

PlayerHealth

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

    public class PlayerHealth : MonoBehaviour
    {
        public float hp;
        public float recover_speed;
```

```
public bool playerisdead;
public float resetAfterDeathTime = 5f;
public GameObject pic_obj;
public Texture mission_failed;
public GameObject HP_status;

private float max_hp = 100f;
private float timer = 0;
private GameObject player;
private RawImage img;
//private FadeInOut fader;
// Start is called before the first frame update

void hard_restart()
{
    SceneManager.LoadScene(SceneManager.GetActiveScene().name);
}

void Start()
{
    hp = max_hp;
    playerisdead = false;
    player = GameObject.FindGameObjectWithTag("Player");
    img = pic_obj.GetComponent<RawImage>();
}

void Update()
{
    if(hp<100&&hp!=0)
    {
        hp = hp + Time.deltaTime * recover_speed;
    }
    if(hp<=0)
    {
        hp = 0f;
        playerisdead = true;
        player.GetComponent<PlayerMovementScript>().enabled=false;
        player.GetComponent<MouseLookScript>().enabled = false;
        player.GetComponent<GunInventory>().enabled = false;
        img.color = Color.red;
        img.texture=mission_failed;
        Invoke("hard_restart", resetAfterDeathTime);
    }
    if(hp>max_hp)
```

```
            {
                hp = max_hp;
            }
            string current_health = hp.ToString("F0");
            string maximum_health = max_hp.ToString("F0");
            HP_status.GetComponent<Text>().text = current_health + "/" + maximum_health;

        }
}
```

Succeed

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

public class Succeed: MonoBehaviour
{
    public GameObject pic_obj;
    public Texture Mission_Succeed;

    private GameObject player;
    private GameObject document;
    private GetDocuments doc;
    private RawImage img;

    void QuitEditor()
    {
        UnityEditor.EditorApplication.isPlaying = false;
        Application.Quit();
    }

    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        document = GameObject.FindGameObjectWithTag("confidential");
        doc = document.GetComponent<GetDocuments>();
        img = pic_obj.GetComponent<RawImage>();
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject == player && doc.GetDocument)
        {
            player.GetComponent<PlayerMovementScript>().enabled = false;
```

```
                player.GetComponent<MouseLookScript>().enabled = false;
                player.GetComponent<GunInventory>().enabled = false;
                img.color = Color.white;
                img.texture = Mission_Succeed;
                Invoke("QuitEditor", 5f);
            }
        }
}
```

BulletScript

```
using UnityEngine;
using System.Collections;

public class BulletScript : MonoBehaviour {

    [Tooltip("Furthest distance bullet will look for target")]
    public float maxDistance = 1000000;
    RaycastHit hit;
    [Tooltip("Prefab of wall damange hit. The object needs 'LevelPart' tag to create decal on it.")]
    public GameObject decalHitWall;
    [Tooltip("Decal will need to be sligtly infront of the wall so it doesnt cause rendeing problems so for best feel put from 0.01-0.1.")]
    public float floatInfrontOfWall;
    [Tooltip("Blood prefab particle this bullet will create upoon hitting enemy")]
    public GameObject bloodEffect;
    [Tooltip("Put Weapon layer and Player layer to ignore bullet raycast.")]
    public LayerMask ignoreLayer;
    private Animator animator;

    /*
    * Uppon bullet creation with this script attatched,
    * bullet creates a raycast which searches for corresponding tags.
    * If raycast finds somethig it will create a decal of corresponding tag.
    */
    void Update () {

        if(Physics.Raycast(transform.position, transform.forward,out hit, maxDistance)){
            if(decalHitWall){
                if(hit.transform.tag == "LevelPart"){
                    Instantiate(decalHitWall, hit.point + hit.normal * floatInfrontOfWall,
Quaternion.LookRotation(hit.normal));
                    Destroy(gameObject);
                }
                if(hit.transform.tag == "Dummie"){
                    animator=hit.transform.GetComponent<Animator>();
                    Instantiate(bloodEffect, hit.point, Quaternion.LookRotation(hit.normal));
```

```
                                Destroy(gameObject);
                                animator.SetBool("isDead",true);
                            }
                        }
                        Destroy(gameObject);
                    }
                    Destroy(gameObject, 0.1f);
                }


}
```

4. Conclusion

   My term project was successfully finished and make this scene interactive. I briefly understand how does programmer control intelligent agents and improve my unity programming skills. I still could make it better by adding realistic audio or adding hear channel to enemies. And I plan to set an alarm state to all enemies but it seems that there was not enough time to do so.