

# Trabajo Final MRV

## DEAD ZONE

---

CURSO: MULTIMEDIA Y REALIDAD VIRTUAL

**Pertenece:**

- |                          |                            |               |
|--------------------------|----------------------------|---------------|
| <input type="checkbox"/> | Huayhua Paco, Eleo Romario | CUI: 20150977 |
| <input type="checkbox"/> | Laura Anccasi, Raul Rene   | CUI: 20153571 |
| <input type="checkbox"/> | Laura Barrios, Erick Andy  | CUI: 20130335 |

Arequipa - 2019

## DEAD ZONE

### 1. Introducción

Se desarrolla un pequeño juego de zombies en realidad virtual usando el motor de juego Unity, usamos la store de Unity para descargar prefabs que nos ayudan en el desarrollo ágil del juego.

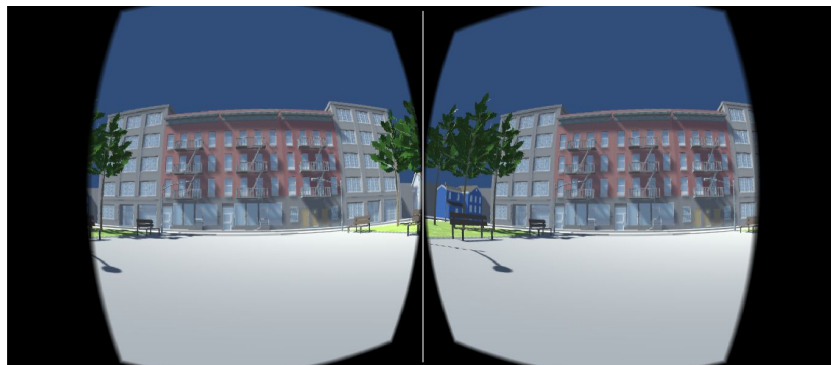
### 2. Resumen del Juego

Nuestro jugador tiene que sobrevivir el mayor tiempo posible al apocalipsis zombies.

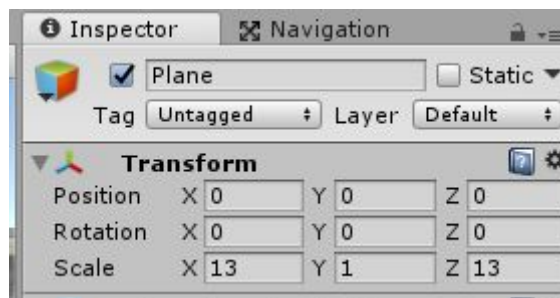
### 3. Desarrollo del Juego

Se sigue los siguientes pasos resumidos para el desarrollo del juego:

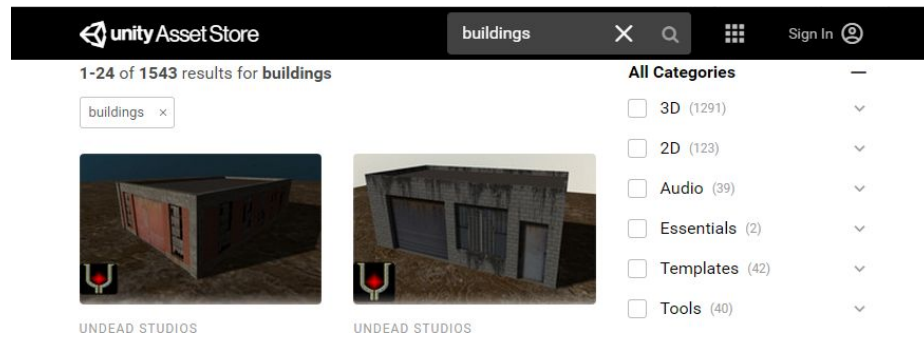
- Para desarrollar el juego primero abrimos Unity.
- Creamos una escena en este caso lo nombramos "Dead Zone"
- Usamos cardboard y lo ubicamos en el central del juego, reemplazando a la cámara de por default.



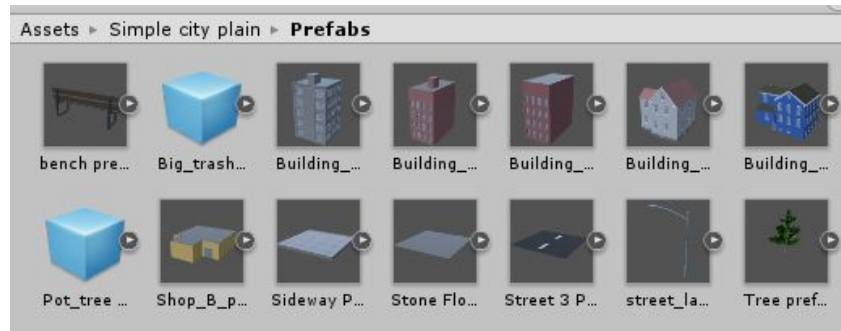
- Creamos un plano, lo que nos ayuda para simular la superficie del juego



- Descargamos prefabs del Asset Store de Unity.



- Importamos los prefabs y modelos a nuestra escena



- De acuerdo a estos modelos se va añadiendo a nuestra escena, creando un escenario óptimo para nuestro juego.



- Se coloca paredes alrededor del escenario para simular un area de cuarentena.

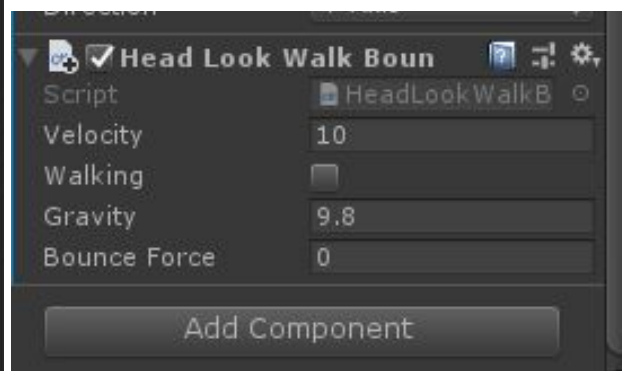


### Movimiento Camara (usuario).

Para mover al usuario principal usamos un script *HeadLookBound*. En este script creamos las variables de velocidad de movimiento, la gravedad, y el controlador e indicando

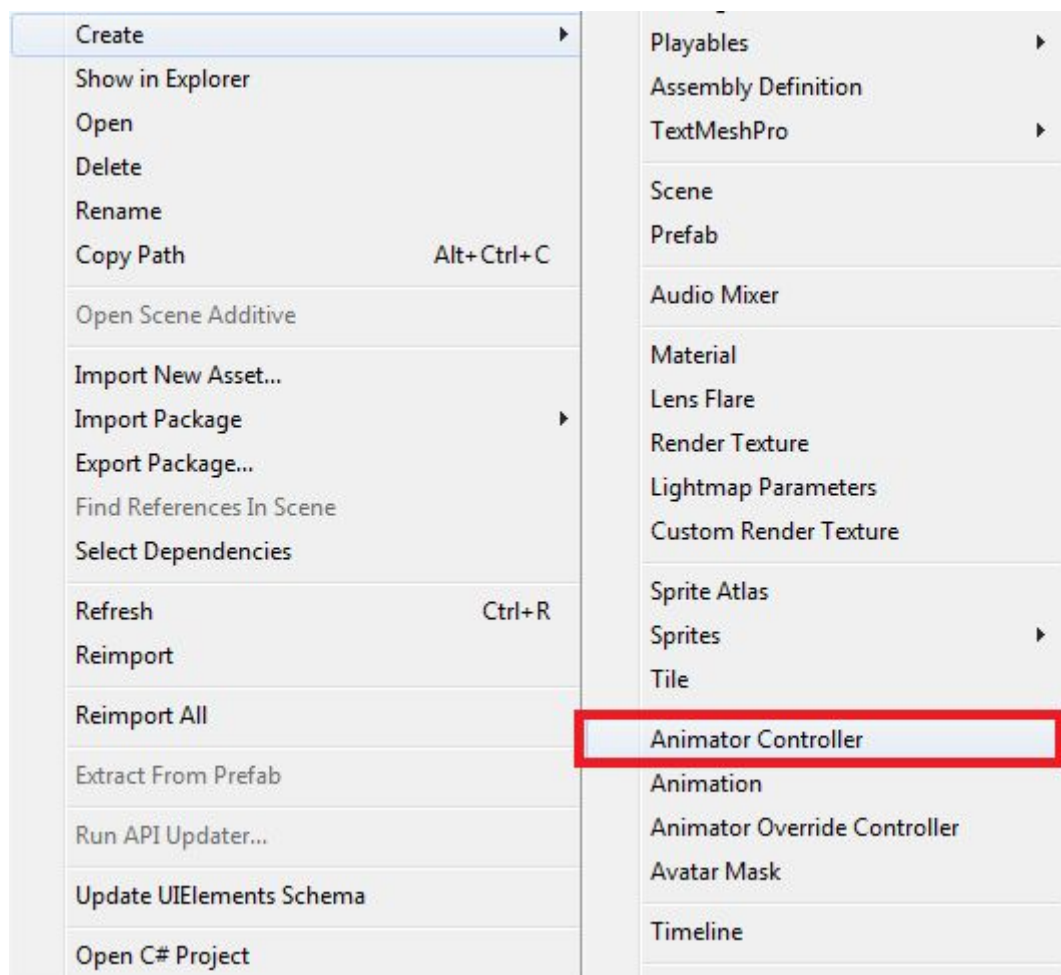
que al hacer click o presionar una tecla la camara (usuario) se desliza haciendo un seguimiento en la dirección que el puntero del mouse indique.

```
// Update is called once per frame
void Update () {
    if (clicker.clicked ()) {
        walking = !walking;
    }
    if (walking) {
        moveDirection = Camera.main.transform.forward * velocity;
    } else {
        moveDirection = Vector3.zero;
    }
    if (controller.isGrounded) {
        verticalVelocity = 0.0f;
    }
    if (bounceForce != 0.0f) {
        verticalVelocity = bounceForce * 0.02f;
        bounceForce = 0.0f;
    }
    moveDirection.y = verticalVelocity;
    verticalVelocity -= gravity * Time.deltaTime;
    controller.Move (moveDirection * Time.deltaTime);
}
```



### Movimiento de Zombie

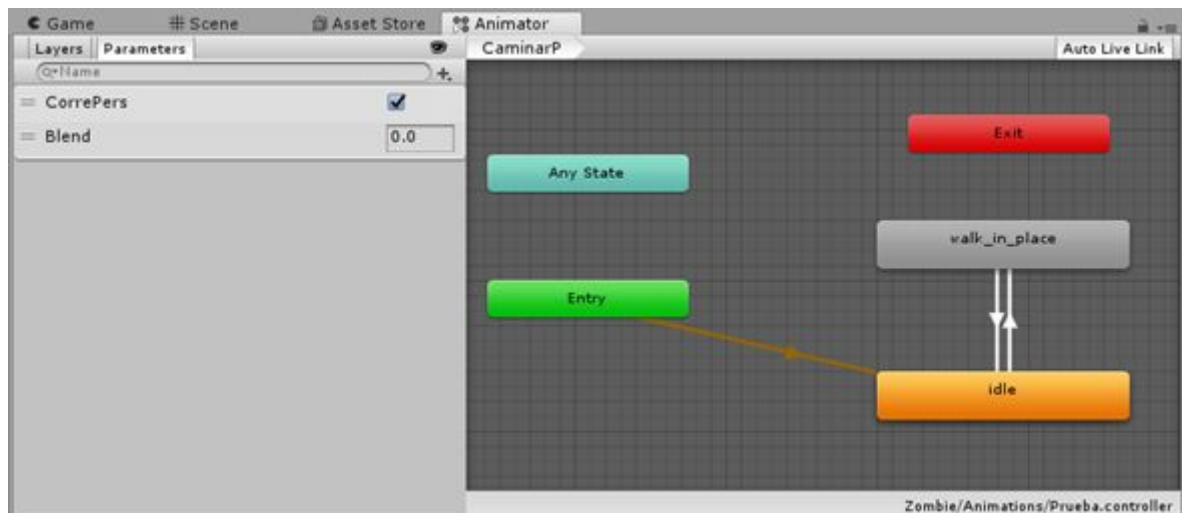
Para el movimiento del zombi se usó un componente que se encuentra en el inspector llamado **animator** para lo cual necesitamos un **animator controller**



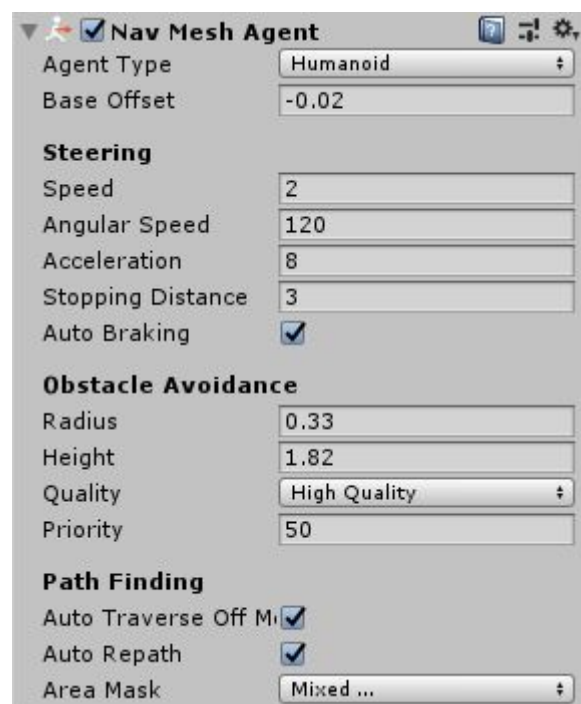
Nos muestra un archivo vacío pero nosotros le damos la siguiente forma usando las animaciones que se encuentran con el paquete prefabricado que se descargó para este caso solo usamos el de **walk\_in\_place**.







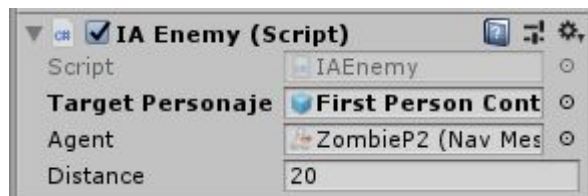
Existe 2 tipos de **zombies** que creamos uno es el **ZombieP1** y el otro es el **zombieP2** ambos hacen uso de **animator** que se encuentra en el inspector pero para cada zombie se usa scripts diferentes también tiene que tener el componente “**Nav Mesh Agent**”

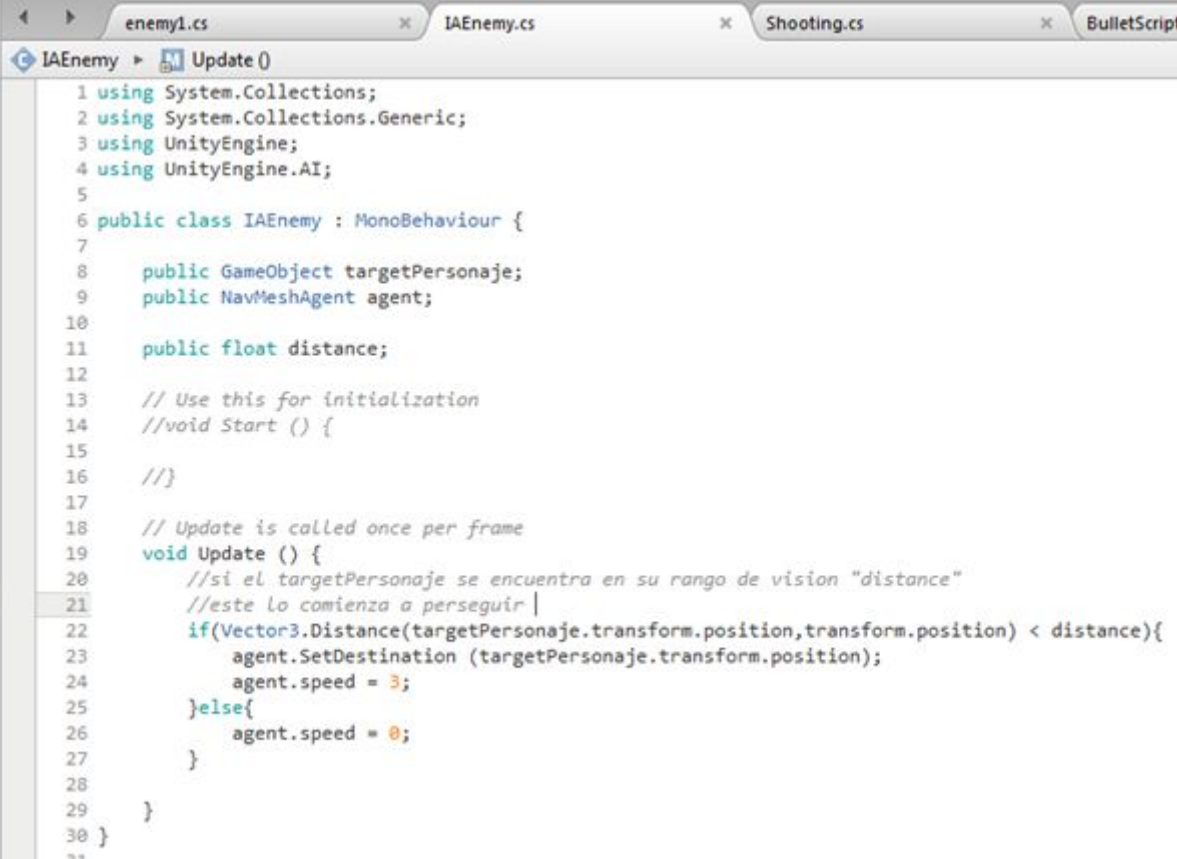


**zombieP1**: este zombi persigue al personaje desde que inicia el juego este donde este

```
enemy1.cs | IAEnemy.cs | Shooting.cs | Bullet.cs
enemy1 ▶ Start ()
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.AI;
5
6 public class enemy1 : MonoBehaviour {
7
8     public Transform player;
9     public NavMeshAgent nav;
10
11     // Use this for initialization
12     void Start () {
13         //esta parte del codigo es para que el zombie persiga a nuestro personaje
14         //con el tag usuario
15         player = GameObject.FindWithTag ("usuario").transform;
16         //usamos el navmesh agent para que el zombie se pueda desplazar por todo el terreno
17         nav = GetComponent<NavMeshAgent> ();
18
19
20
21     }
22
23     // Update is called once per frame
24     void Update () {
25         nav.SetDestination (player.position);
26     }
27 }
28
```

**zombieP2:** este zombi persigue al personaje siempre y cuando esté en su rango de visión que puede ser modificado de acuerdo al script



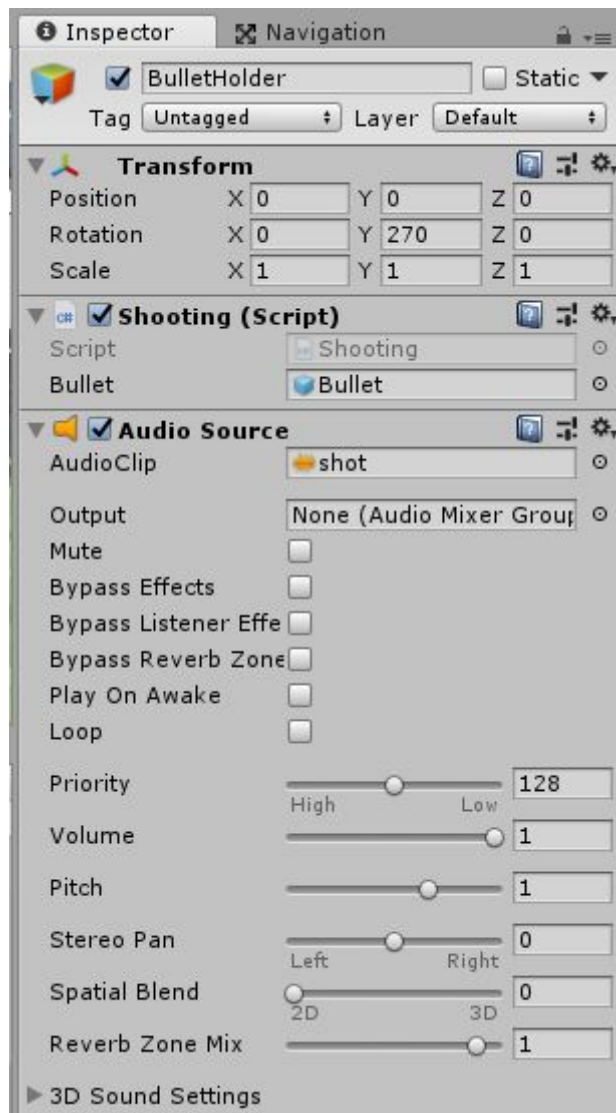


```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.AI;
5
6 public class IAEnemy : MonoBehaviour {
7
8     public GameObject targetPersonaje;
9     public NavMeshAgent agent;
10
11     public float distance;
12
13     // Use this for initialization
14     //void Start () {
15
16     //}
17
18     // Update is called once per frame
19     void Update () {
20         //si el targetPersonaje se encuentra en su rango de vision "distance"
21         //este lo comienza a perseguir |
22         if(Vector3.Distance(targetPersonaje.transform.position,transform.position) < distance){
23             agent.SetDestination (targetPersonaje.transform.position);
24             agent.speed = 3;
25         }else{
26             agent.speed = 0;
27         }
28     }
29 }
30
31
```

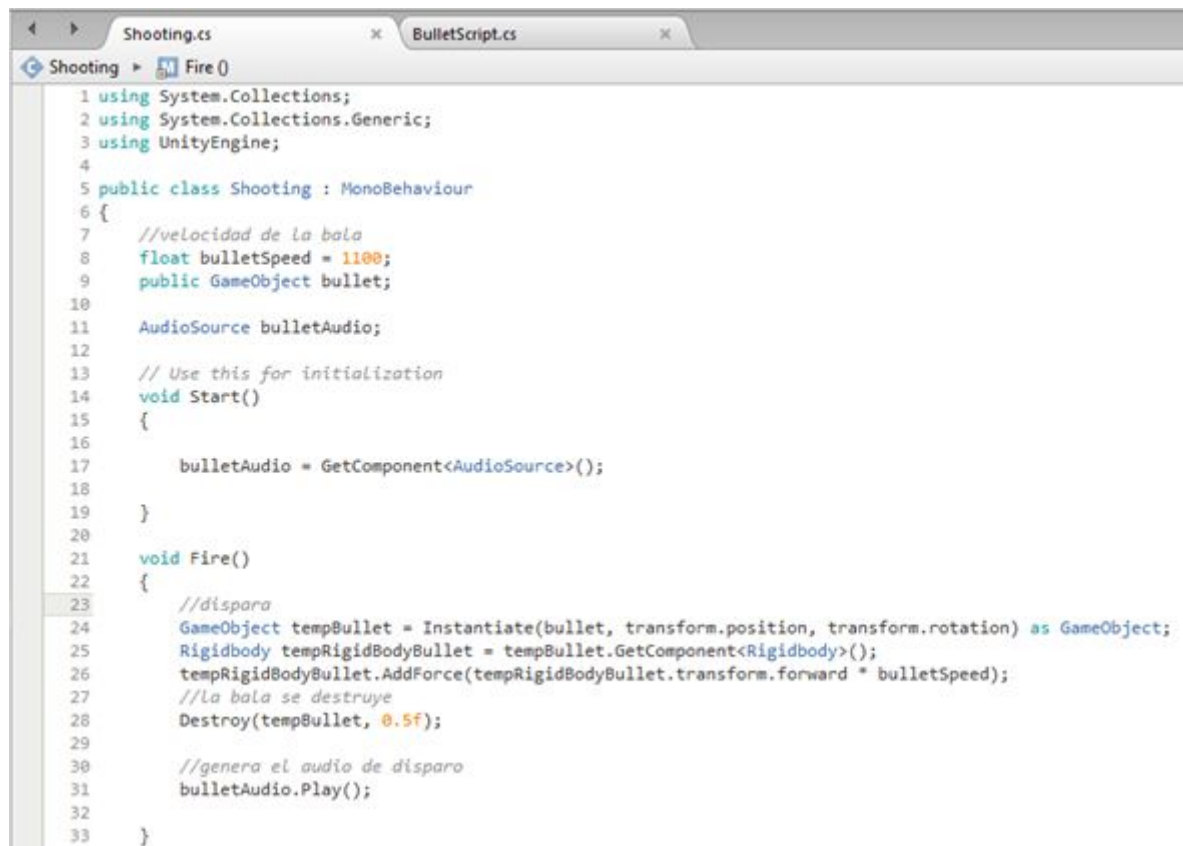
## DISPARO DEL PERSONAJE

Dentro de **CardboardMain** y **First Person Controller** al mismo nivel que MainCamera colocamos el arma y dentro de ella un cubo llamado Gun y dentro del cubo un archivo vacio llamado **BulletHorder** que tendrá un script llamado shooting y un audio que corresponde al disparo



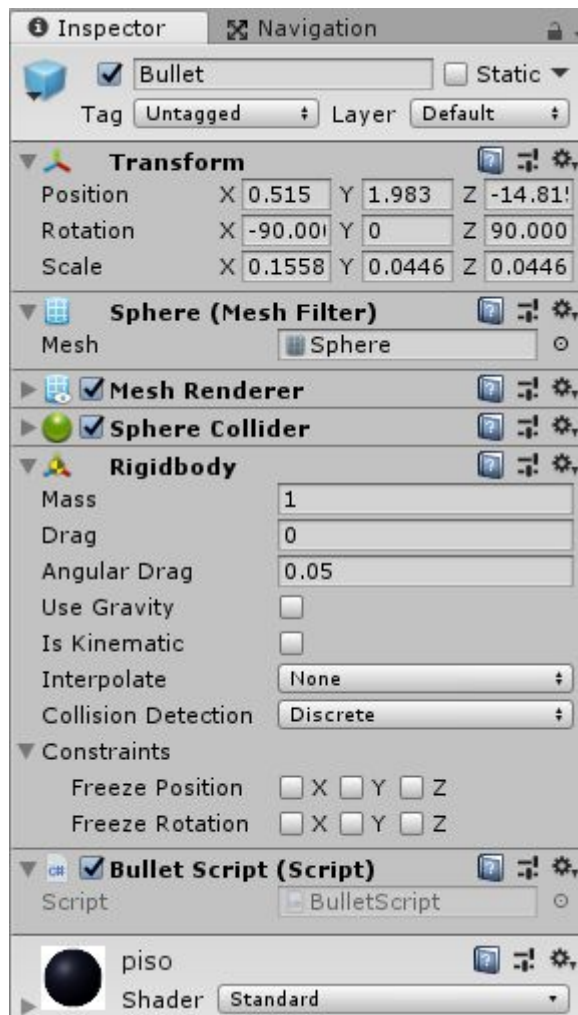


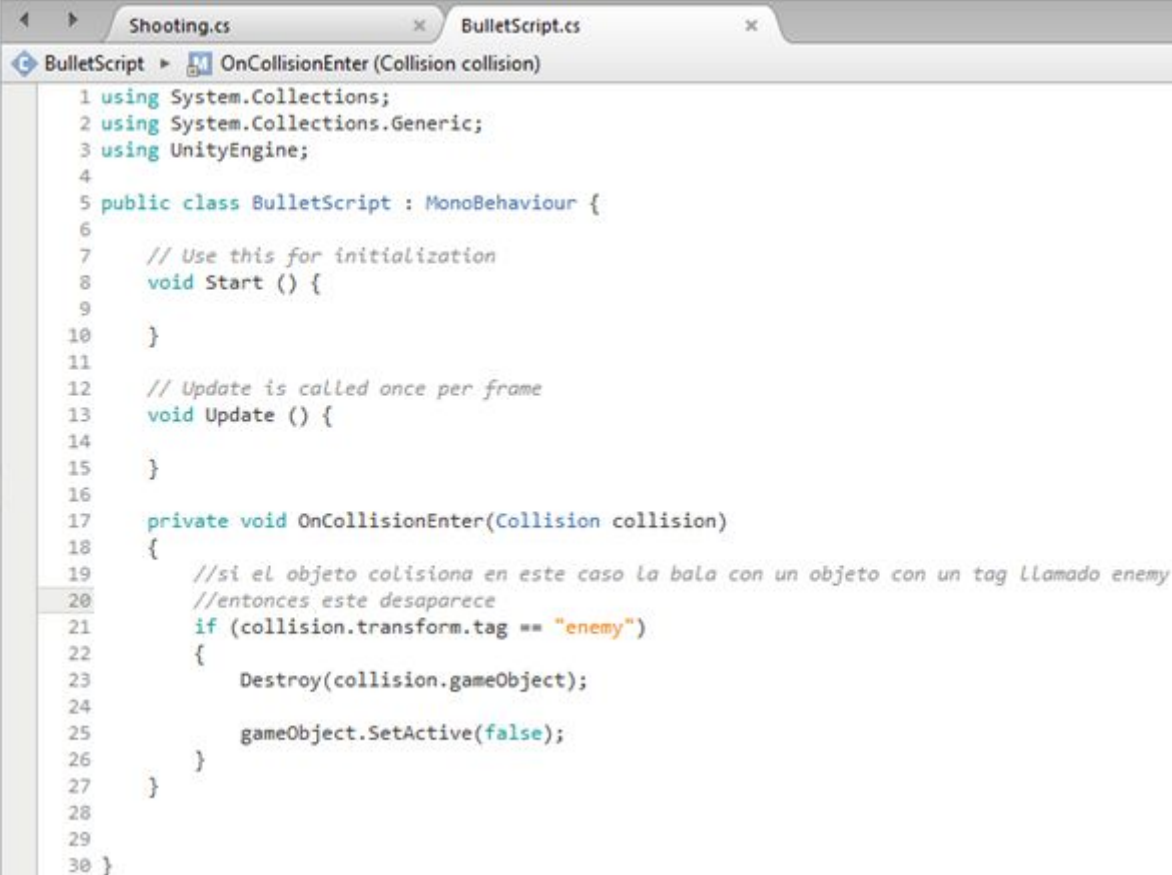
script shooting



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Shooting : MonoBehaviour
6 {
7     //velocidad de la bala
8     float bulletSpeed = 1100;
9     public GameObject bullet;
10
11     AudioSource bulletAudio;
12
13     // Use this for initialization
14     void Start()
15     {
16
17         bulletAudio = GetComponent<AudioSource>();
18
19     }
20
21     void Fire()
22     {
23         //dispara
24         GameObject tempBullet = Instantiate(bullet, transform.position, transform.rotation) as GameObject;
25         Rigidbody tempRigidBodyBullet = tempBullet.GetComponent<Rigidbody>();
26         tempRigidBodyBullet.AddForce(tempRigidBodyBullet.transform.forward * bulletSpeed);
27         //La bala se destruye
28         Destroy(tempBullet, 0.5f);
29
30         //genera el audio de disparo
31         bulletAudio.Play();
32
33     }
```

La bala en este caso nombrado **bullet** hace uso del script **BulletScript** para poder eliminar a el zombie





```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class BulletScript : MonoBehaviour {
6
7     // Use this for initialization
8     void Start () {
9
10    }
11
12    // Update is called once per frame
13    void Update () {
14
15    }
16
17    private void OnCollisionEnter(Collision collision)
18    {
19        //si el objeto colisiona en este caso la bala con un objeto con un tag llamado enemy
20        //entonces este desaparece
21        if (collision.transform.tag == "enemy")
22        {
23            Destroy(collision.gameObject);
24
25            gameObject.SetActive(false);
26        }
27    }
28
29
30 }
```

#### 4. Conclusiones

Unity VR le permite apuntar a dispositivos de realidad virtual directamente desde Unity, sin ningún complemento externo en los proyectos, esto da muchas facilidades al momento de crear un proyecto con esta tecnología.

El proyecto **DEADZONE** es juego de simulación de supervivencia frente a un evento post-apocalíptico, y al ser creado con la tecnología VR introduce al usuario a una experiencia aún más real.