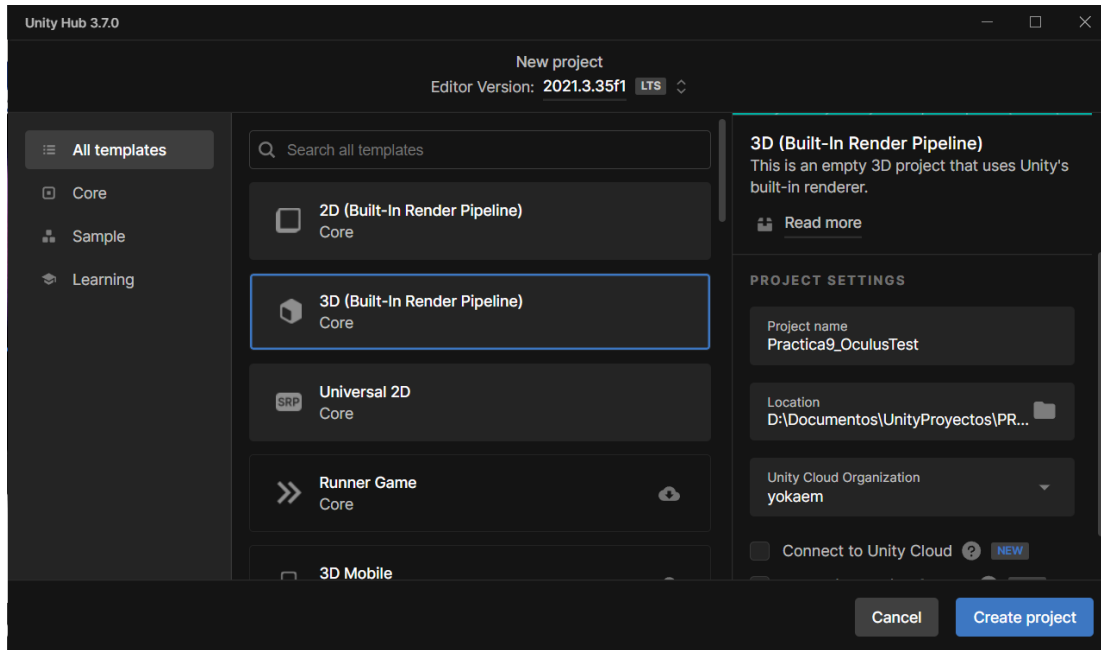
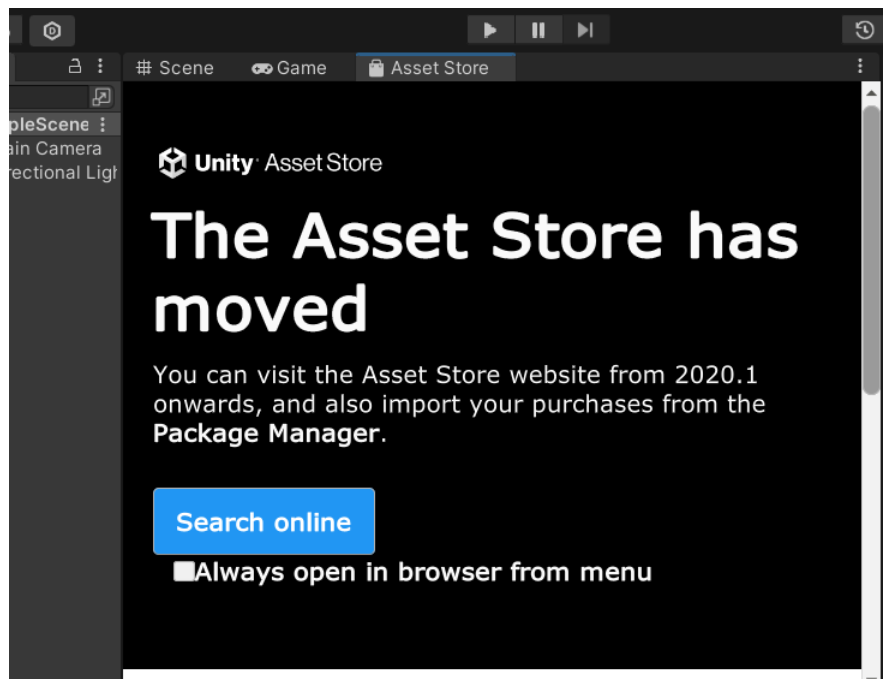


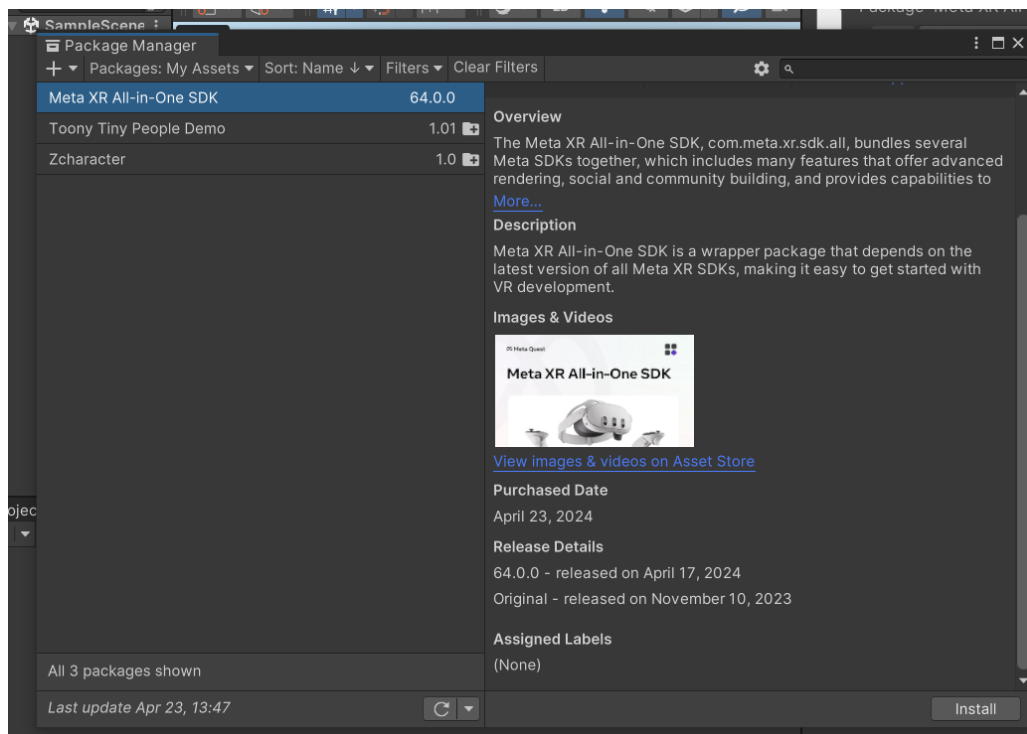
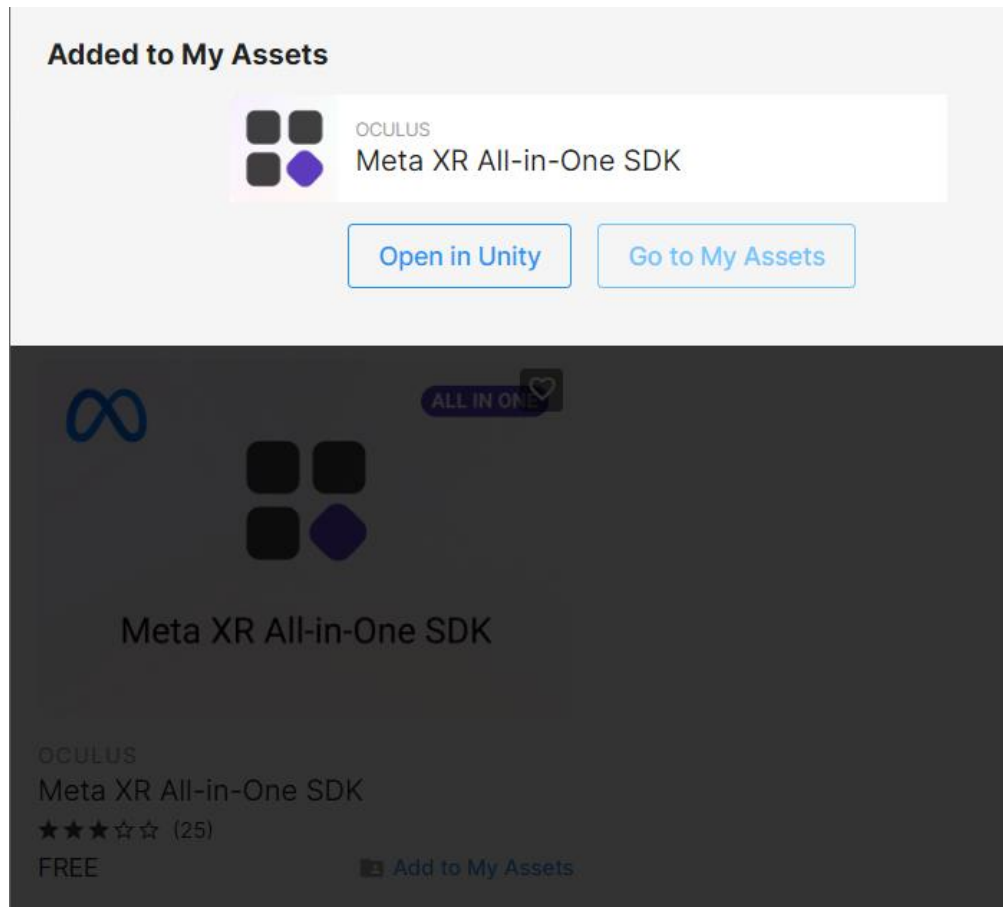
## Manual

1.- Primero abrí Unity Hub y generé un nuevo proyecto. Llamado ExamenLaberinto.

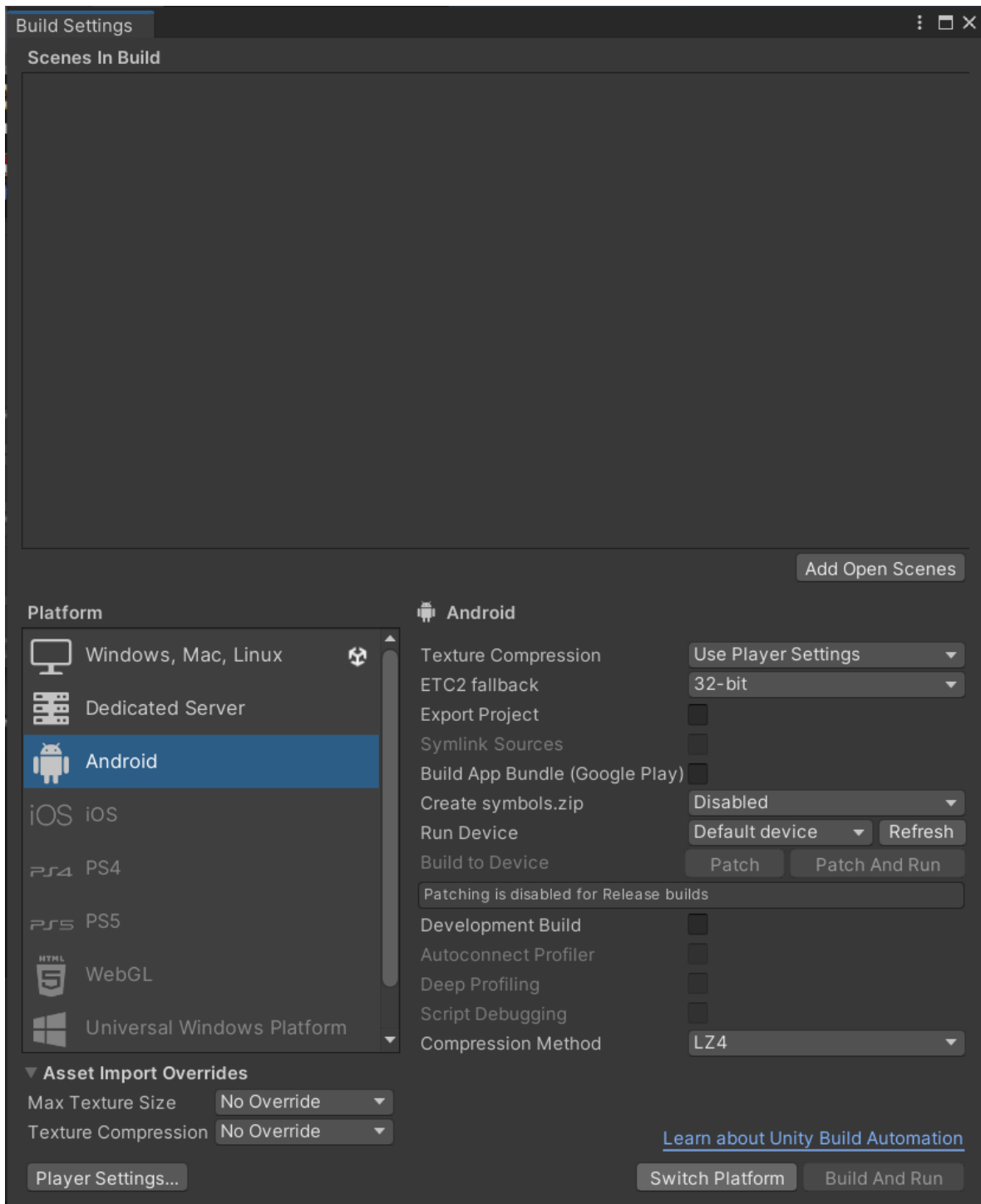


2.- Una vez en el proyecto me metí a la Asset Store de Unity y me mando a su pagina web. Luego busqué el meta xr all in one package y lo añadí a mis assets. Luego en Unity me metí al Package Manager, filtré por My Assets y ahí mismo instalé el Meta XR All In One SDK. Me pidió reiniciar así que reinicie.

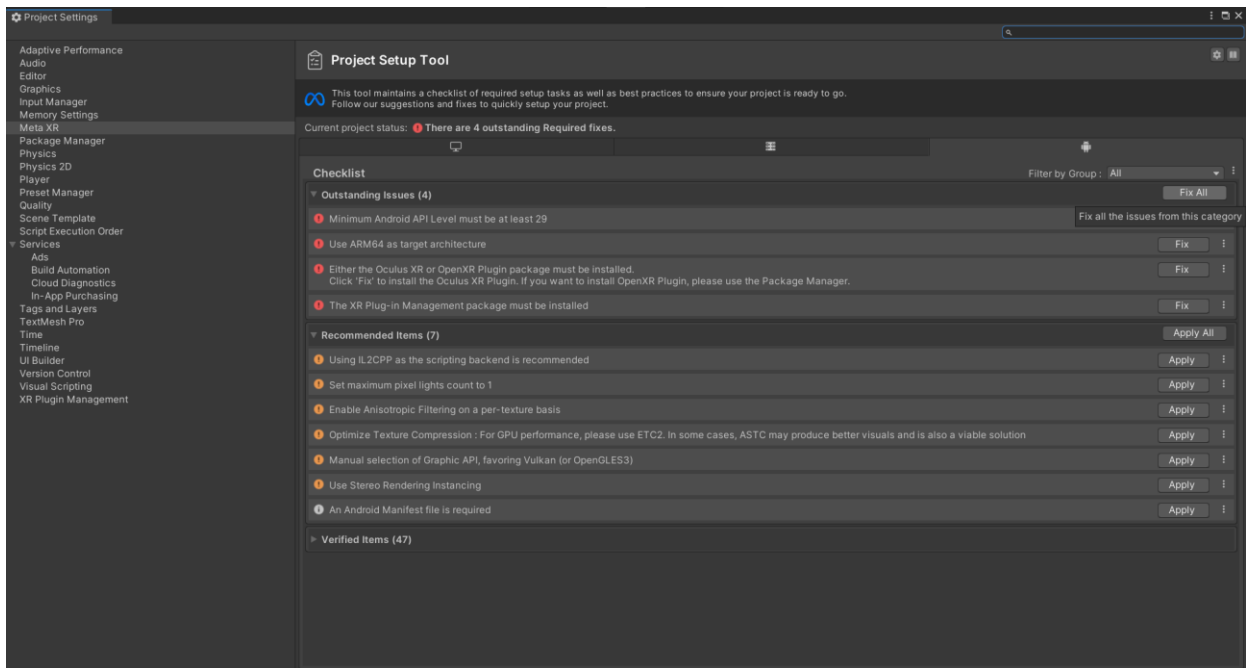
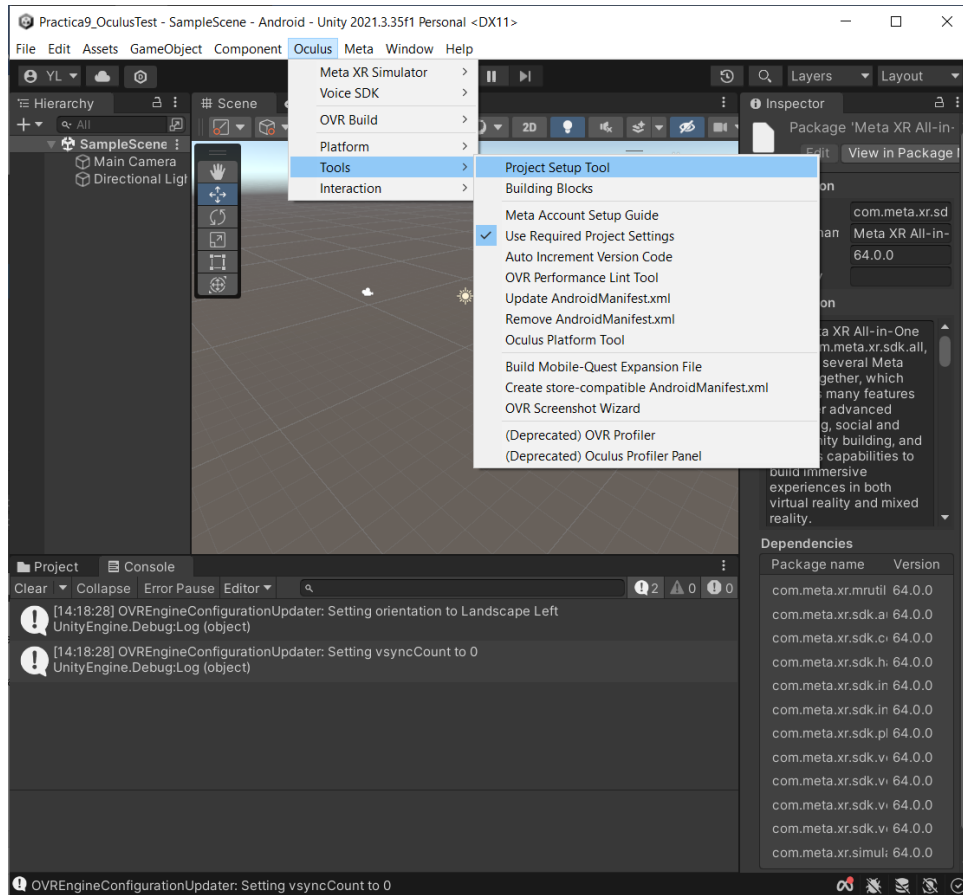


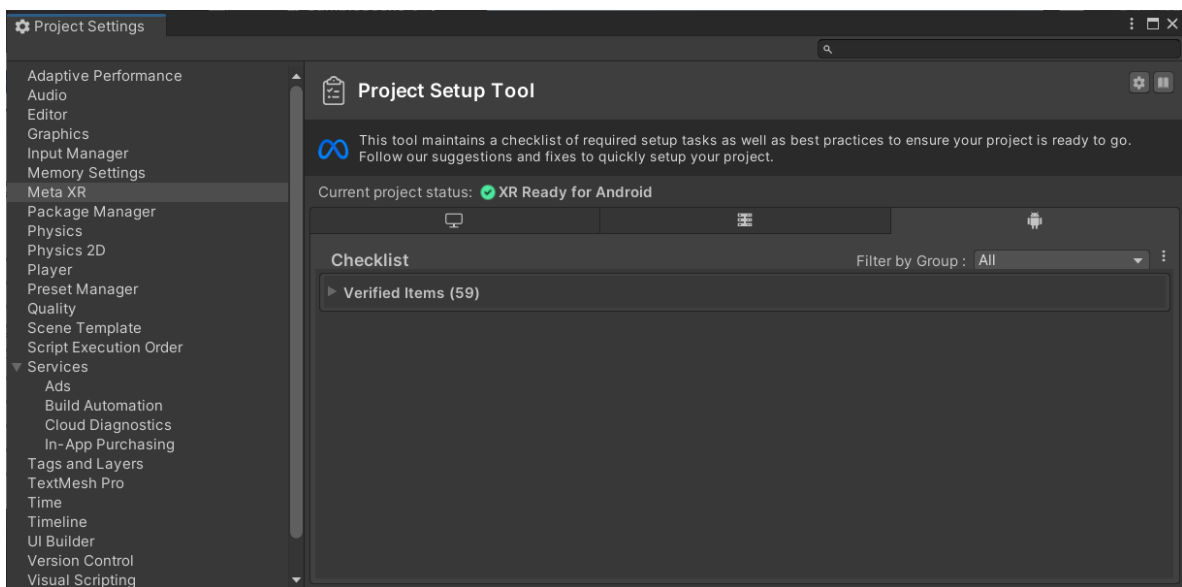
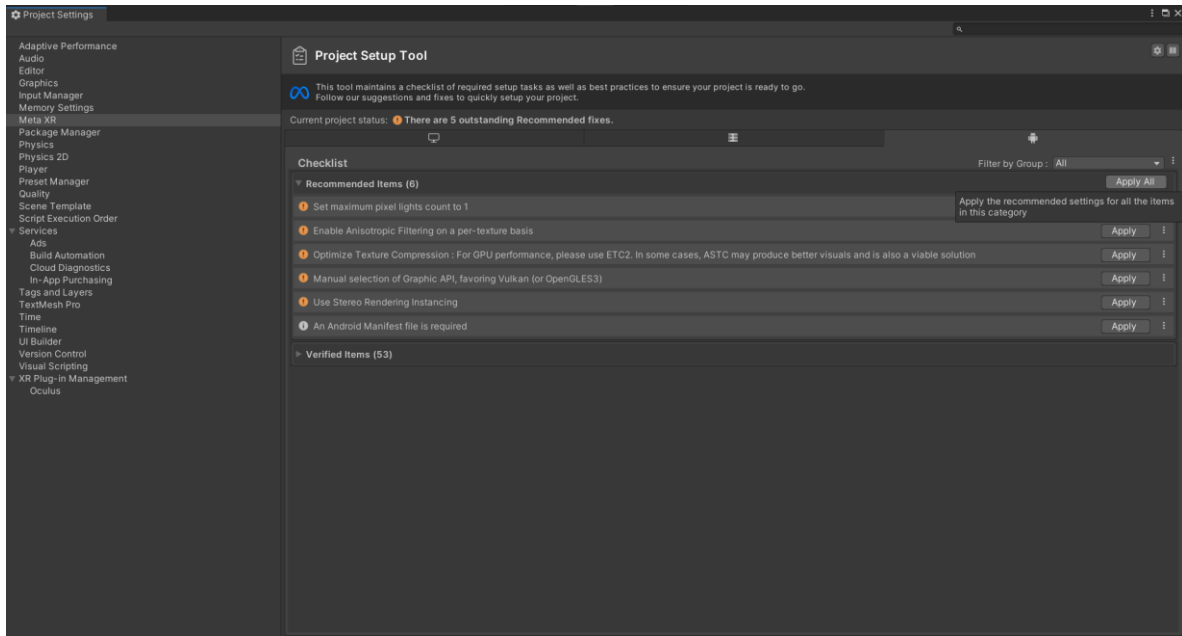


3.- Posteriormente abrí Build Settings y cambié la plataforma a Android.

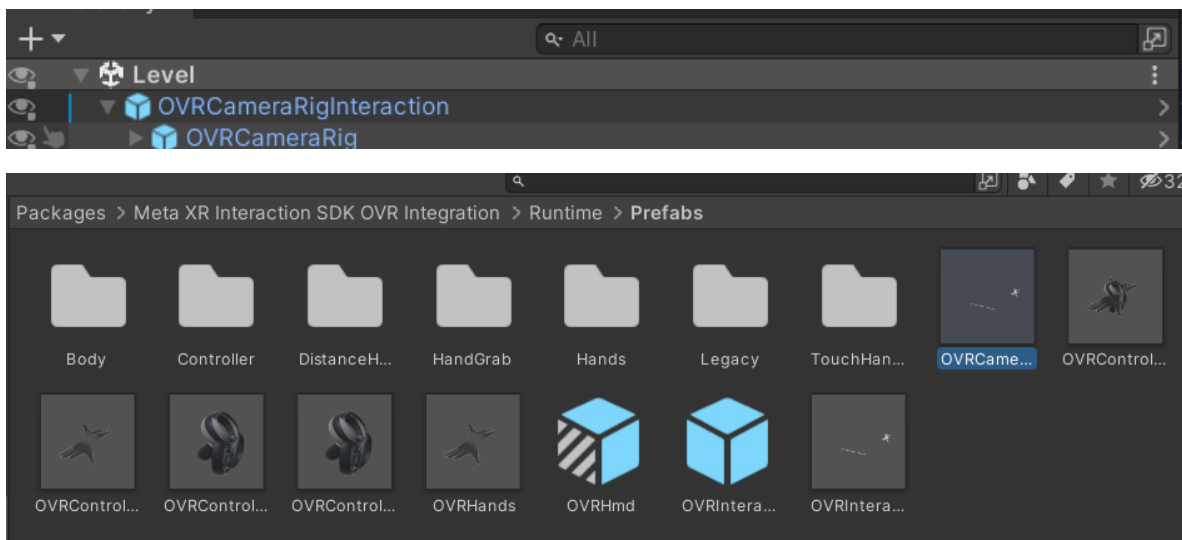
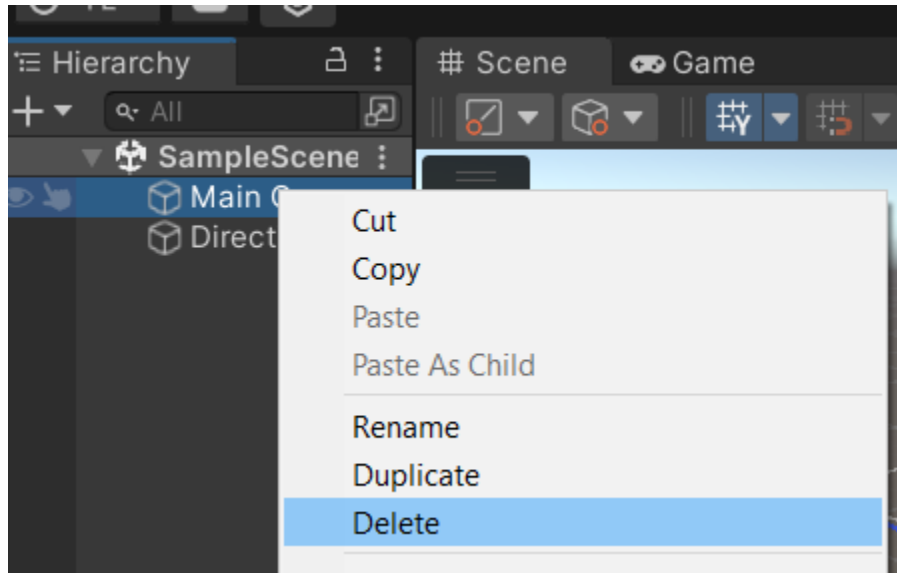


4.- Luego fui a Oculus -> Tools -> Project Setup Tool. Di a Fix All dos veces hasta que desaparezcan los errores y posteriormente di a Apply All.






5.- Luego eliminé la MainCamera por default y agregué en su lugar el prefab de OVRCameraRigInteraction dentro de las carpetas de Packages, Meta XR Core SDK y Prefabs.





### PARA MOVIMIENTO DEL PERSONAJE

6.- Después agregué en el componente padre de OVRCameraRigInteraction un Character Controller y los scripts OVR Scene Sampler Controller y OVR Player Controller. Esto basándome en un prefab del Meta SDK llamado OVR Player Controller también.

☒

Character Controller



Slope Limit

45

Step Offset

0.3

Skin Width

0.08

Min Move Distance

0.001

Center


X 0 Y 1.07 Z 0

Radius


0.5


Height

1.74

☐

Nav Mesh Agent





☒

OVR Scene Sample Controller (Script)



Script

 OVRSceneSampleController 

Quit Key

Escape

Fade In Texture

None (Texture) 

Speed Rotation Increme

0.05

Layer Name


Default

☒

OVR Player Controller (Script)



Script

 OVRPlayerController 

Acceleration

0.1

Damping

0.3

Back And Side Dampen

0.5

Jump Force

0.3

Rotation Amount

1.5

Rotation Ratchet

45

Snap Rotation

☒

Rotate Around Guardian

☐

Fixed Speed Steps

0

Hmd Resets Y

☒

Hmd Rotates Y

☒

Gravity Modifier

0.379

Use Profile Data

☒

Enable Linear Movemer

☒

Enable Rotation

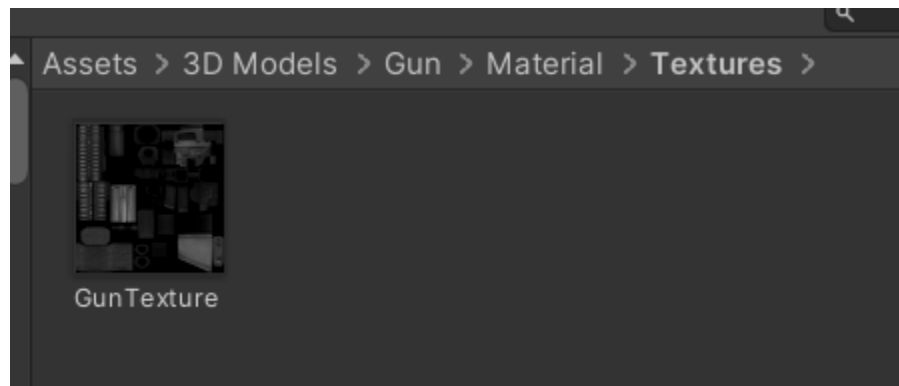
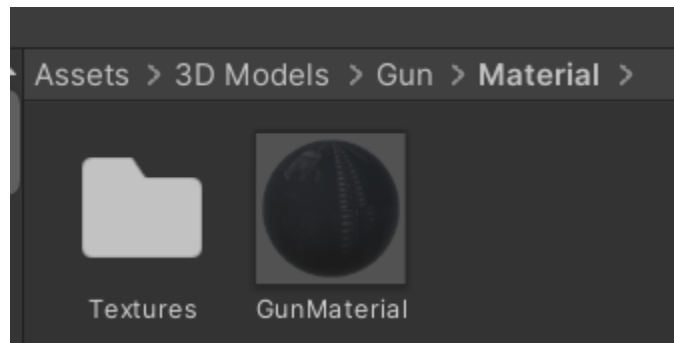
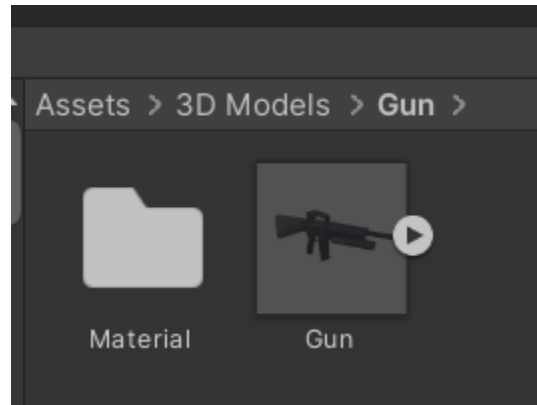
☒

Rotation Either Thumbst

☐

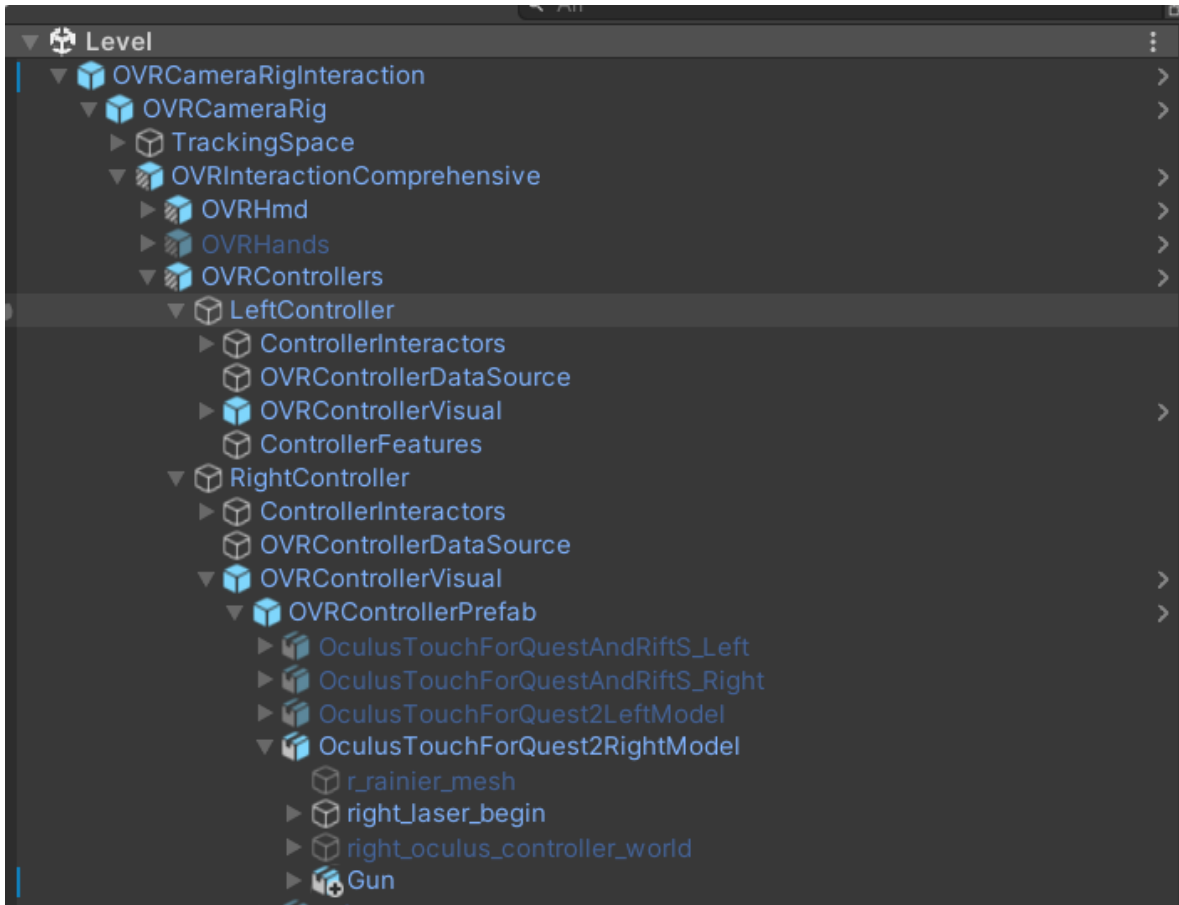
### PARA EL ARMA Y EL DISPARO

7.- Importe mi arma desde los archivos de la practica 5 junto con sus materiales y texturas.

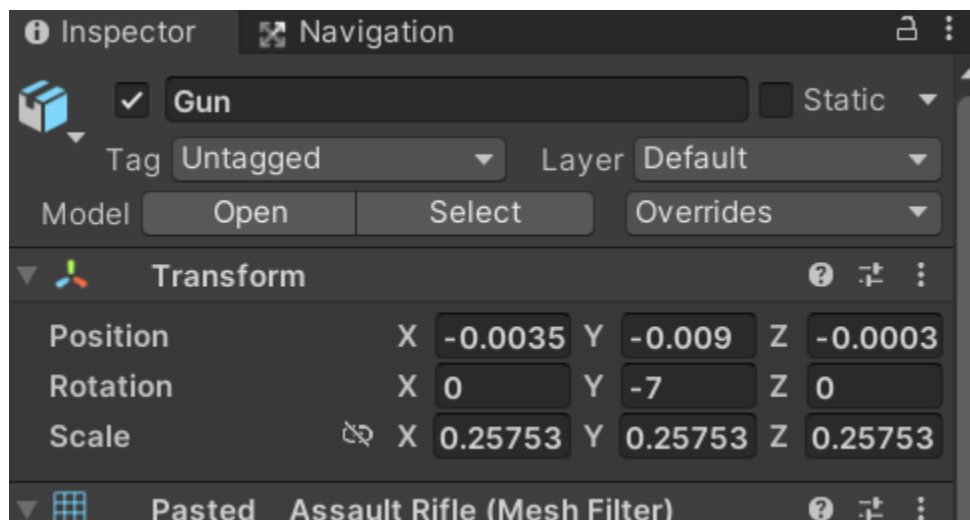


8.- Luego la agregué dentro de la jerarquía del OVRCameraRigInteraction justo donde esta la parte visual del control derecho, desactivando componentes innecesarios que pertenecían principalmente al modelo del control derecho, el control izquierdo en su totalidad y el Locomotion.

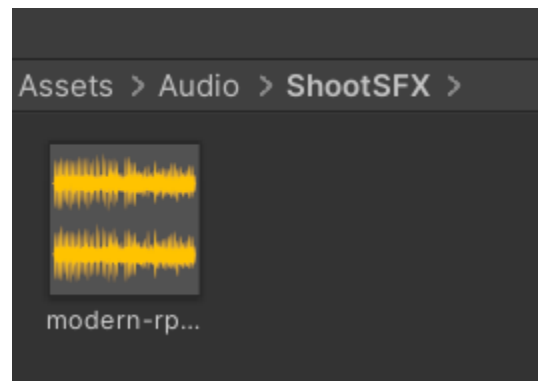
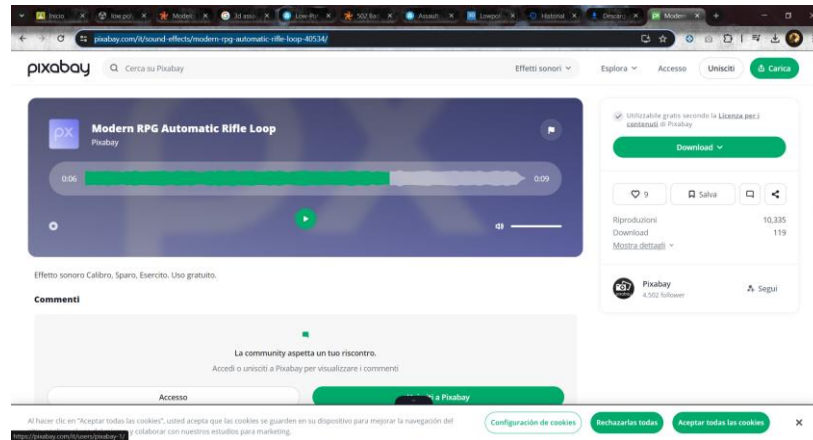




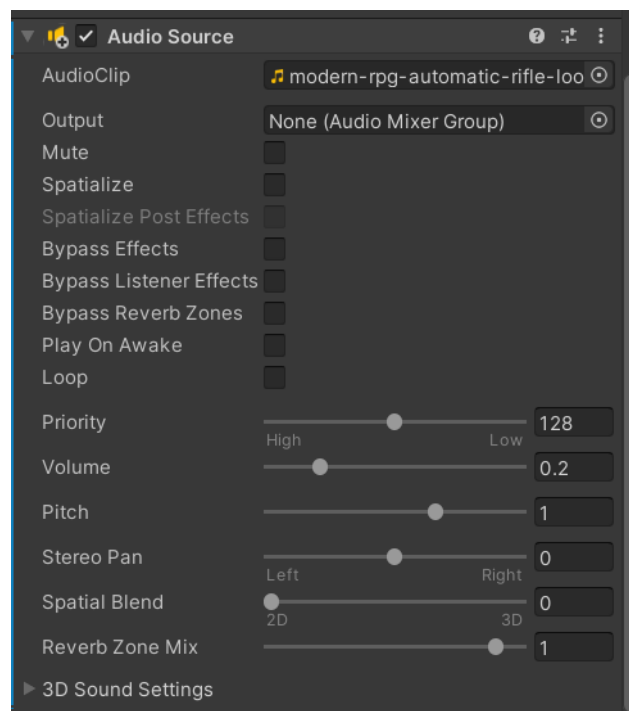
9.- Para alinear correctamente el arma al momento de correr el juego en las Meta Quest 2, tuve que aplicar rotaciones y pequeñas traslaciones.

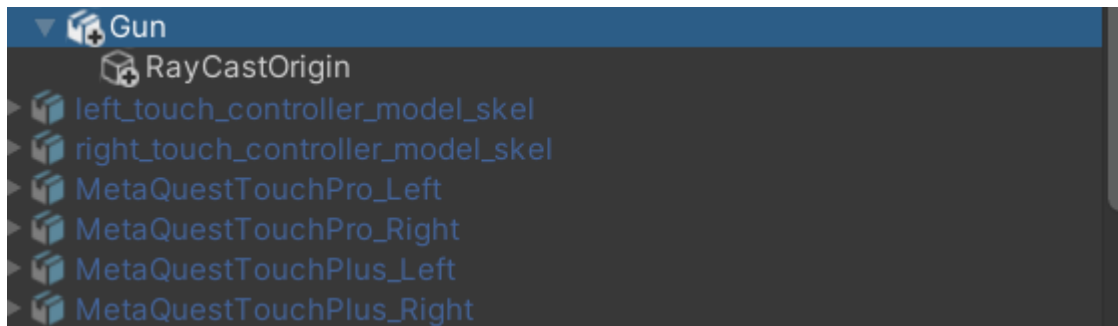


10.- Posteriormente agregué un sonido SFX de disparo de rifle de asalto que obtuve del siguiente enlace: <https://pixabay.com/it/sound-effects/modern-rpg-automatic-rifle-loop-40534/>.

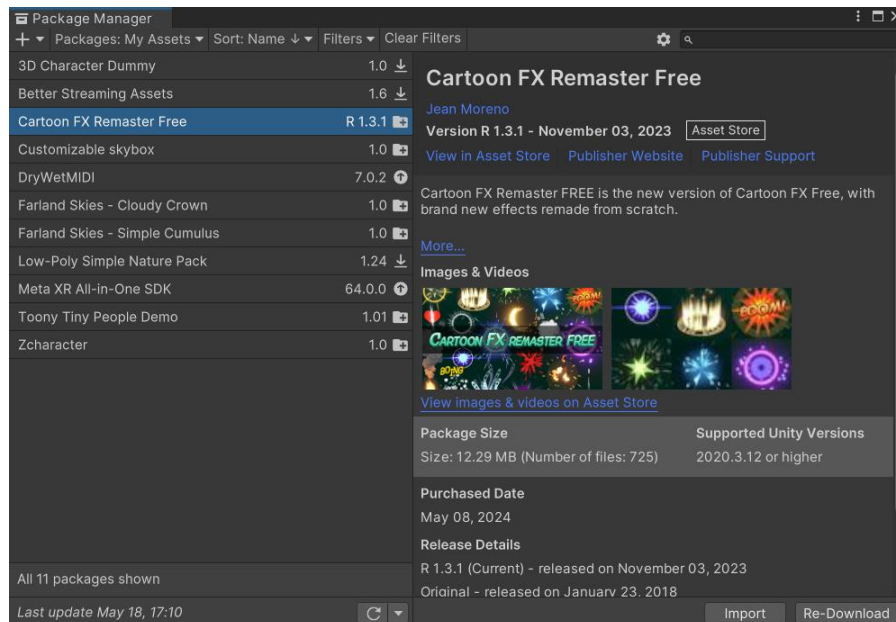
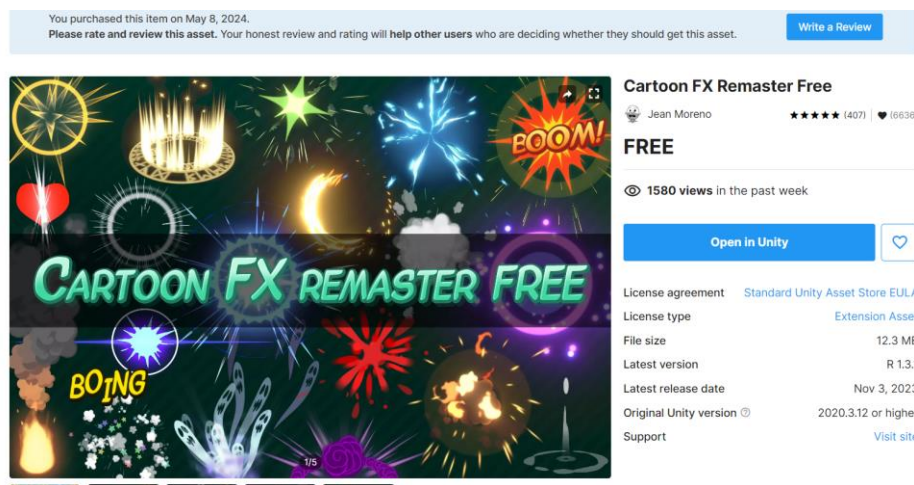


11.- Generé un Audio Source que llama a este nuevo efecto dentro del componente del arma en la escena y además generé un hijo vacío que sirve para ser el origen de mi raycast de disparo.

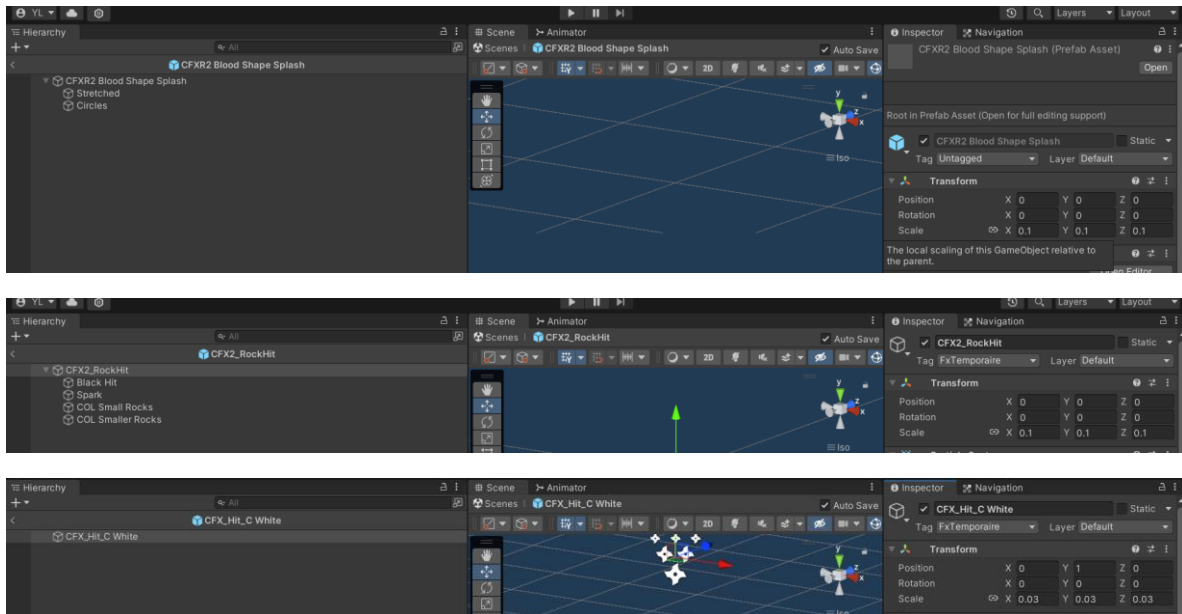




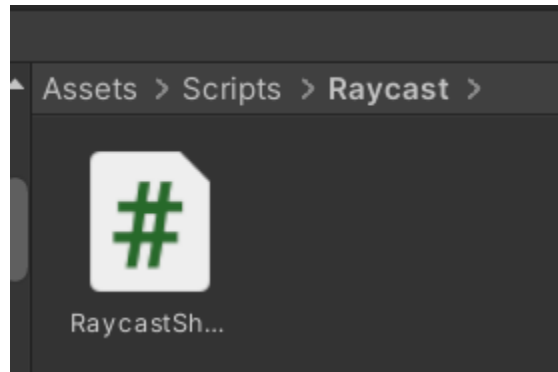
12.- Luego busqué en la Unity Asset Store un paquete de partículas y efectos. Termine encontrando el siguiente: <https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-remaster-free-109565>. Lo agregué a mi cuenta y lo añadí a mi proyecto mediante el package manager.

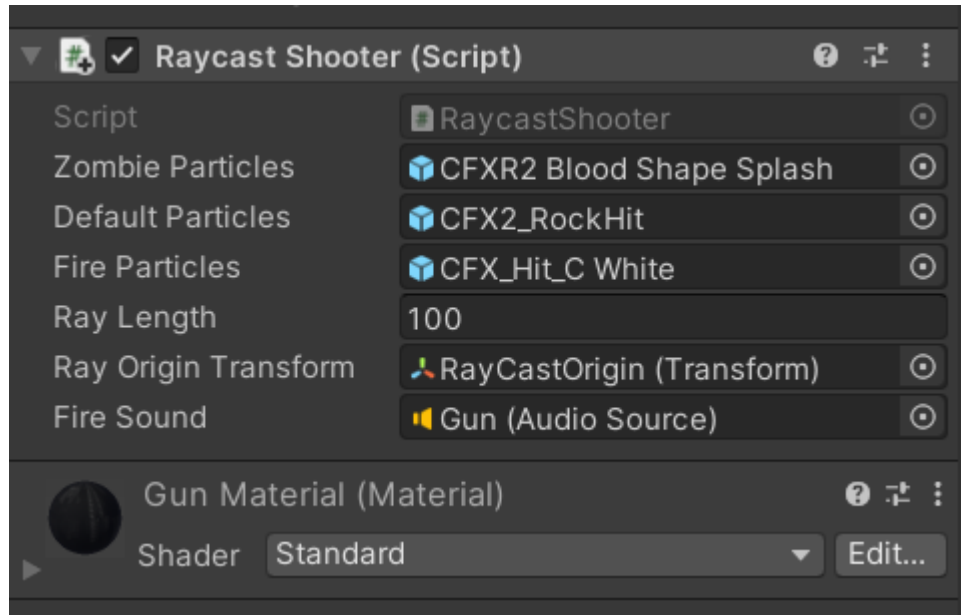


13.- Modifiqué el tamaño de las partículas que decidí utilizar. En este caso fueron dos, una de sangre cuando disparas a un zombi y una de rocas cuando disparas a cualquier otra cosa. Además de la que aparece en el arma al disparar en general.



14.- Posteriormente generé un script llamado RaycastShooter y se lo agregué también al componente Gun, pasándole los parámetros necesarios.





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

public class RaycastShooter : MonoBehaviour
{
    public GameObject zombieParticles; // Prefab de partículas para zombies
    public GameObject defaultParticles; // Prefab de partículas por defecto
    public GameObject fireParticles; // Prefab de partículas por defecto
    public float rayLength = 100f; // Longitud del raycast
    public Transform rayOriginTransform; // Asigna el transform de origen del rayo
    public AudioSource fireSound;

    private bool isFiring = false; // Indica si el botón está siendo presionado

    void Start()
    {
        if (rayOriginTransform == null)
        {
            Debug.LogError("Transform de origen del rayo no asignado.");
            this.enabled = false;
            return;
        }
    }
}
```

```
void Update()
{
    // Comprueba si el botón está siendo presionado
    isFiring = OVRInput.Get(OVRInput.Axis1D.SecondaryIndexTrigger) ==
1f;

    // Si el botón está siendo presionado, dispara y reproduce el
sonido en bucle
    if (isFiring)
    {
        ShootRaycast();
        if (!fireSound.isPlaying)
        {
            fireSound.loop = true;
            fireSound.Play();
        }
    }
    else
    {
        // Si el botón no está siendo presionado, detiene la
reproducción del sonido
        if (fireSound.isPlaying)
        {
            fireSound.loop = false;
            fireSound.Stop();
        }
    }
}

void ShootRaycast()
{
    RaycastHit hit;
    Vector3 start = rayOriginTransform.position;
    Vector3 end =
rayOriginTransform.TransformDirection(Vector3.forward) * rayLength;

    if (Physics.Raycast(start, end, out hit, rayLength))
    {
        // Comprueba el tag del objeto y genera las partículas
correspondientes
        if (hit.transform.tag == "Zombie")
        {
            ZombieMovement zombie =
hit.collider.GetComponent<ZombieMovement>();
```

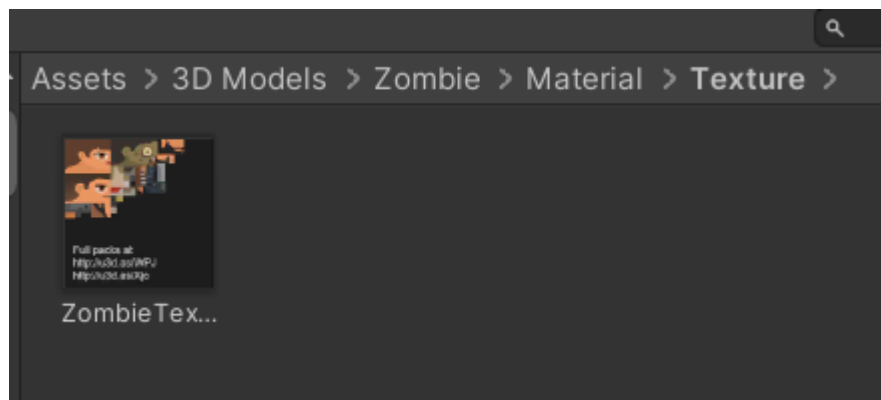
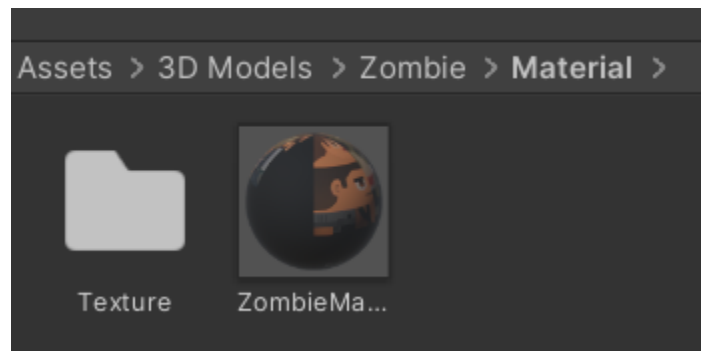
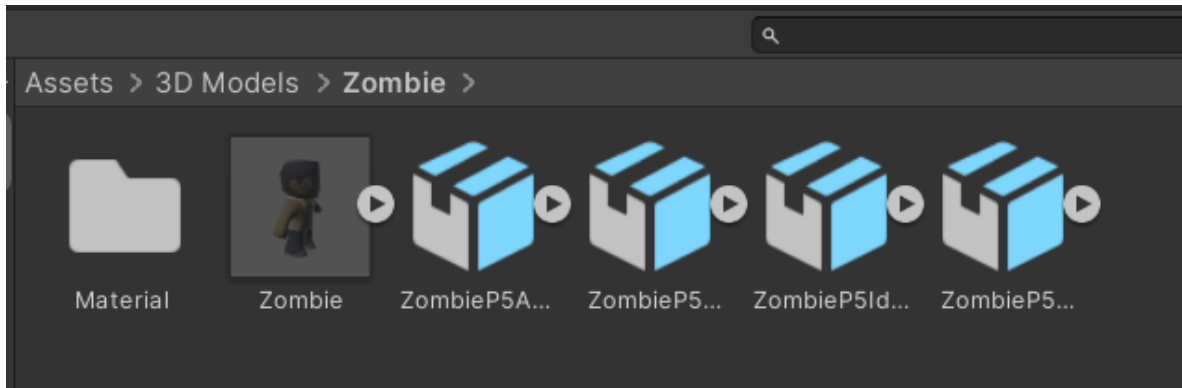
```
        GameObject b = Instantiate(zombieParticles, hit.point +
hit.normal * 0.2f, Quaternion.identity);
        b.transform.SetParent(hit.collider.transform); //
Establece el padre del objeto b al objeto golpeado
        Destroy(b, 1);
        zombie.HitByRaycast(); // Llama a la función que maneja el
impacto en el script del zombie
    }
    else
    {
        if (hit.transform.tag != "UI" && hit.transform.tag !=
"Wall" )
        {
            GameObject c = Instantiate(defaultParticles,
hit.point, Quaternion.identity);
            Destroy(c, 1);
        }
    }

    GameObject a = Instantiate(fireParticles, start,
Quaternion.identity);
    // Establece el objeto instanciado como hijo del objeto que
tiene este script
    a.transform.SetParent(this.transform);
    Destroy(a, 0.3f);

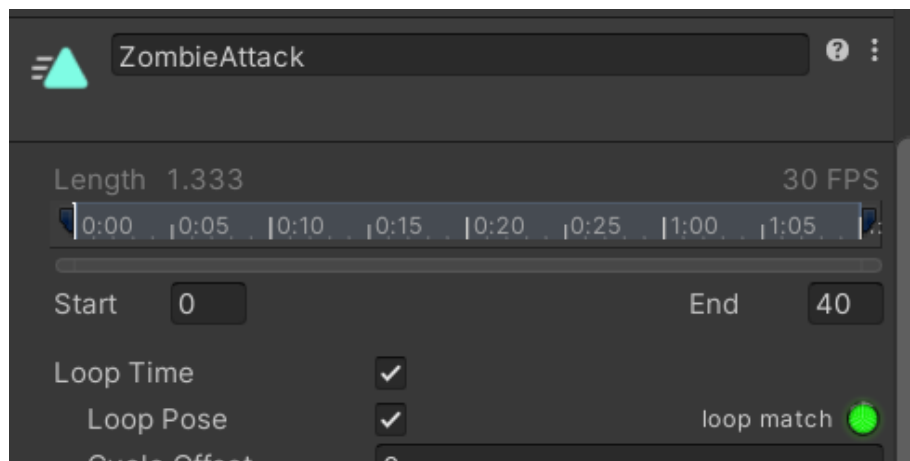
}
}
```

#### PARA EL ENEMIGO

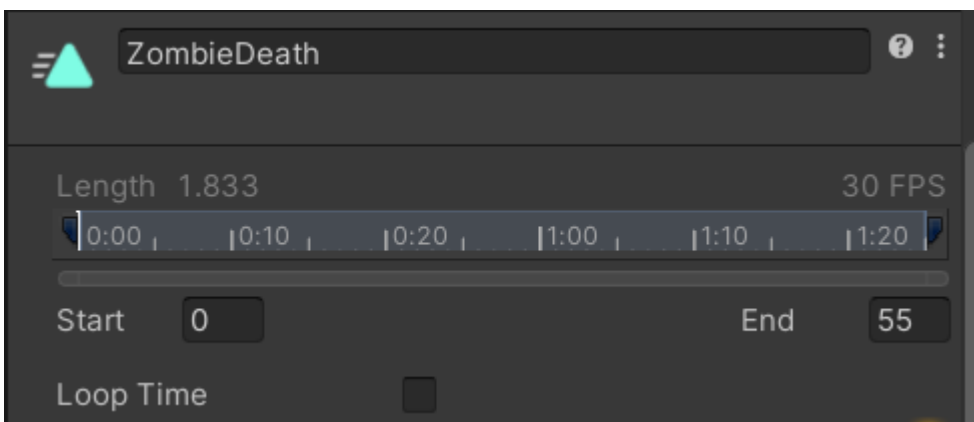
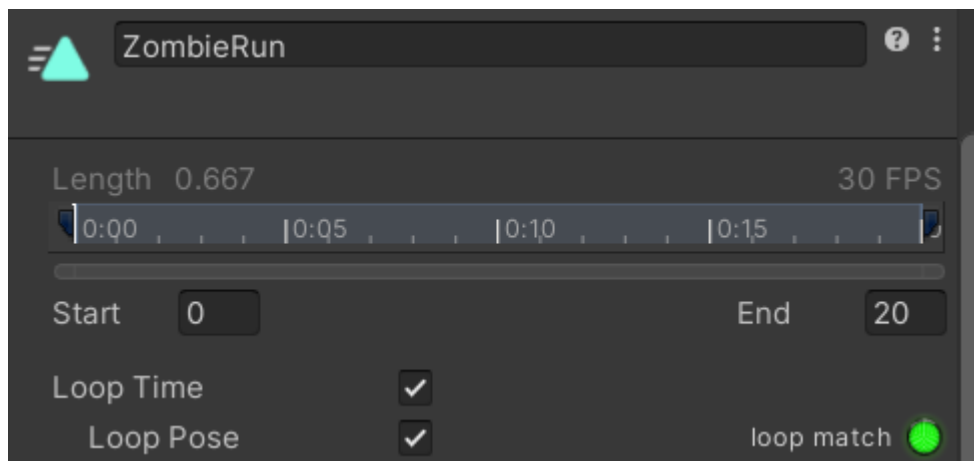
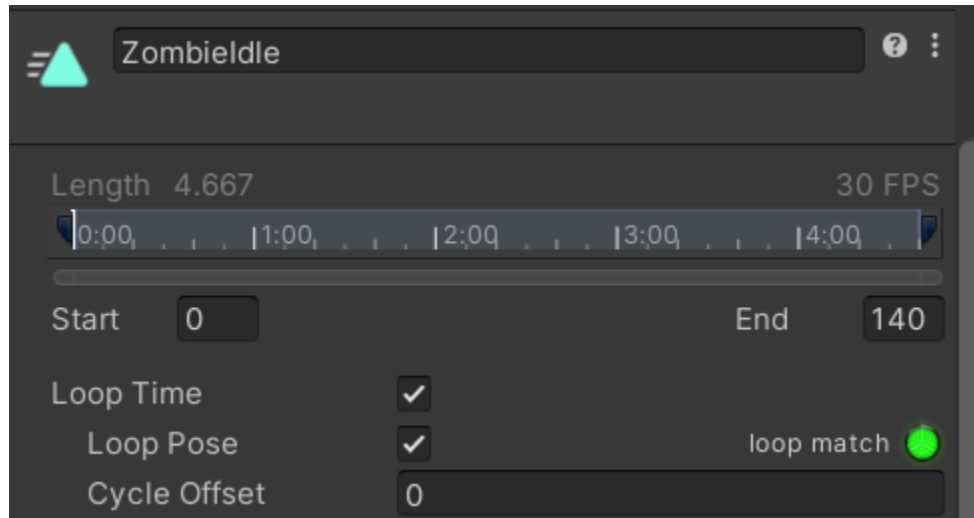
15.- En el caso del enemigo descargue también de la Asset Store el siguiente paquete: <https://assetstore.unity.com/packages/3d/characters/toony-tiny-people-demo-113188>. Lo importe también por medio del Package Manager. Y únicamente use el modelo del zombi junto con las animaciones que incluye el paquete.



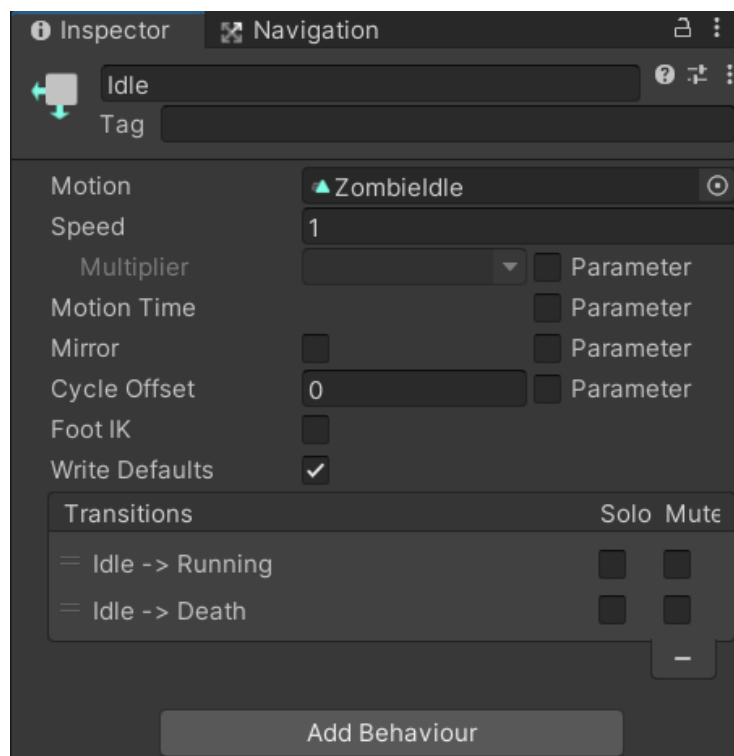
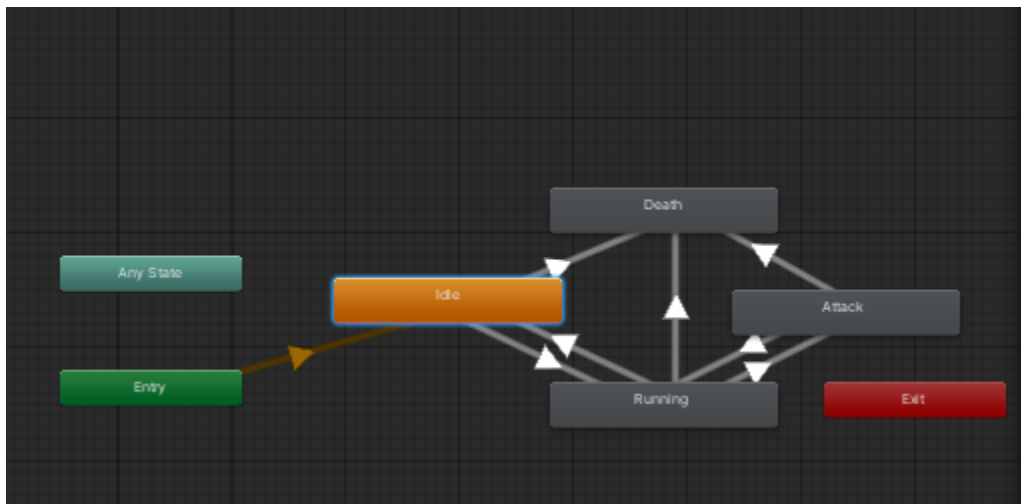
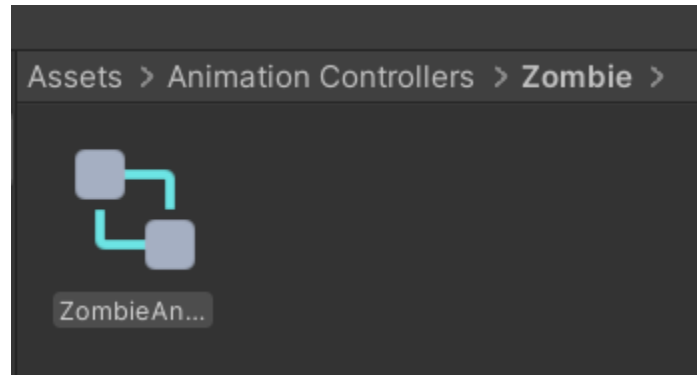
16.- A todas las animaciones con excepción de la de muerte les agregué la opción de Loop activado.

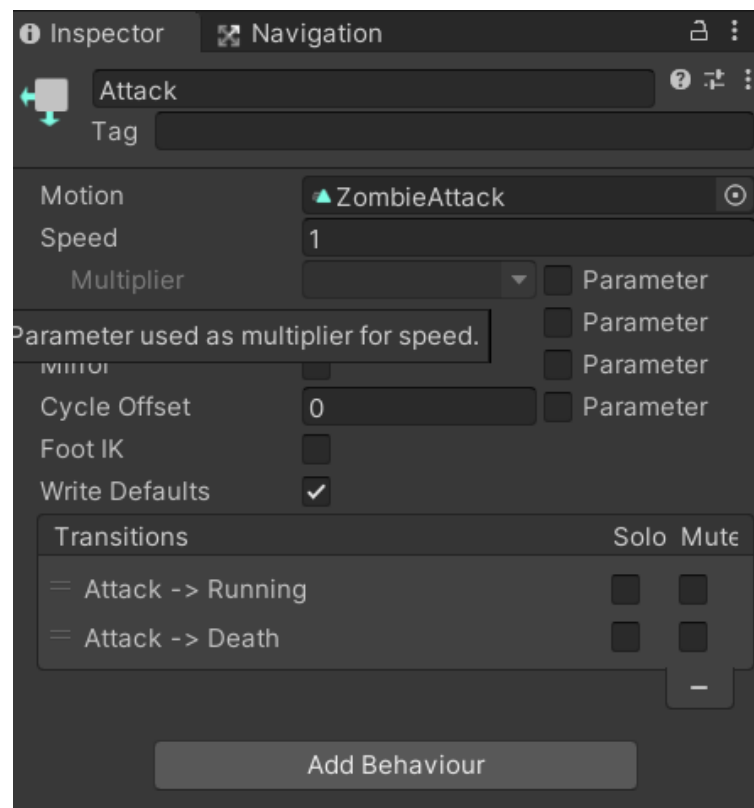


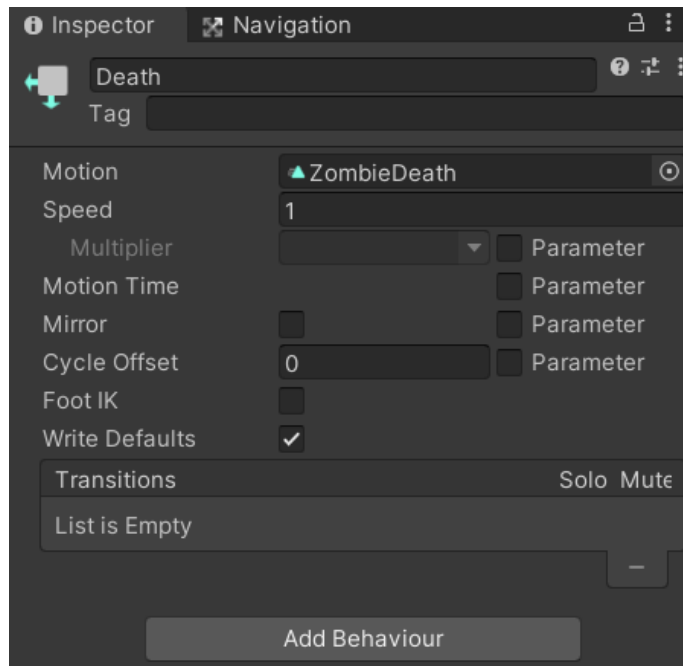




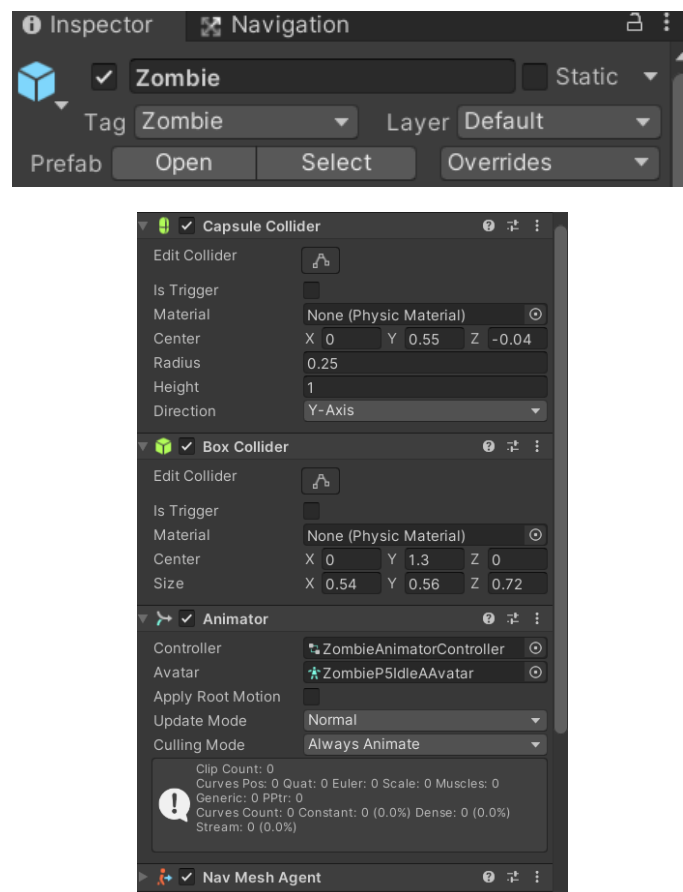
17.- Generé un Zombie Animator Controller al cual le agregue estados de espera, de ataque, de correr y de muerte. Basados en el valor de una variable entera.

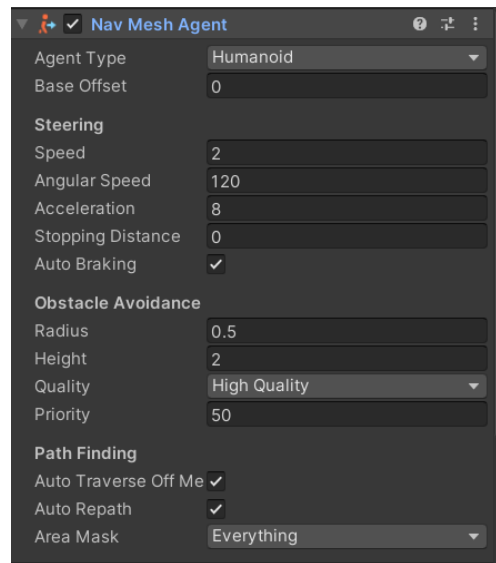




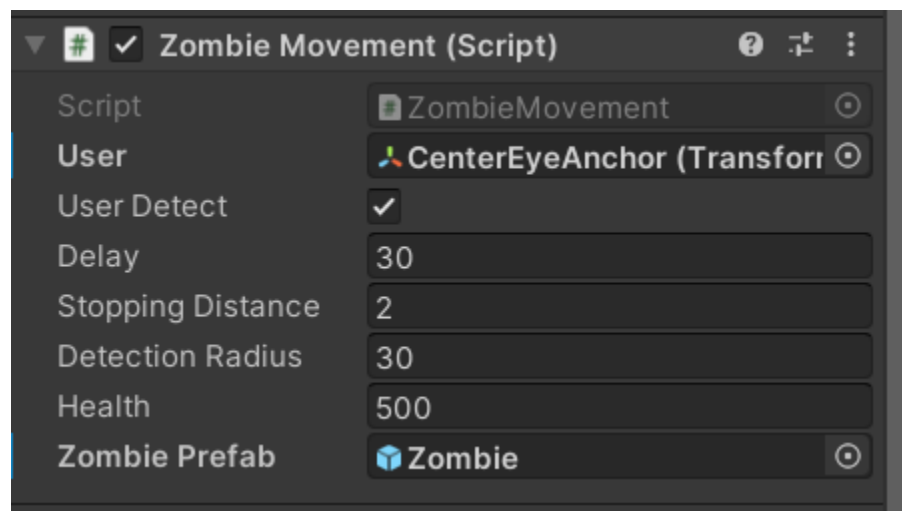
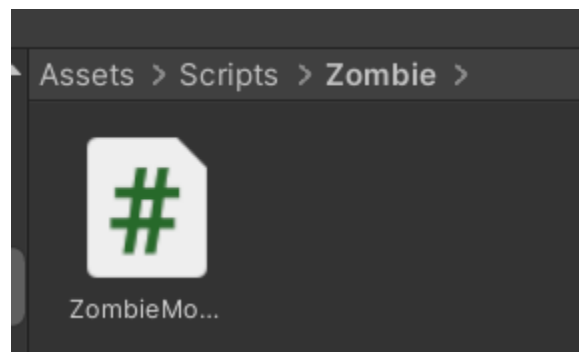


18.- Le agrege un tag de Zombie al propio Zombie, le agrege un Capsule Collider para el torso y un Box Collider para la cabeza además de un Animator y un Nav Mesh Agent.





19.- Generé un script llamado ZombieMovement, lo agregué al elemento de Zombie existente y le mandé los elementos necesarios. Posteriormente genere un prefab llamado Zombie también.



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

```
using UnityEngine;

public class ZombieMovement : MonoBehaviour
{
    public Transform user;
    private NavMeshAgent enemyAgent;
    public bool userDetect;
    public float delay; //Segundos a esperar para destruir tras animación
    de muerte.
    public float stoppingDistance = 1.5f; // Distancia de tolerancia para
    detenerse frente al jugador.
    public float detectionRadius = 10f; // Radio de detección
    private Animator enemyAnimator;
    private bool disparado = false;
    private Vector3 originalPosition; // Posición original del enemigo
    public int health = 500;
    public GameObject zombiePrefab; // Prefab del zombie para respawning
    private HealthBar playerHealthBar; // Referencia a la barra de salud
    del jugador
    private int lastAttackTime = -1; // Rastrea la última vez que se
    aplicó el daño
    private Collider[] colliders; // Arreglo para todos los colliders
    asociados

    public void OnTriggerEnter(Collider other)
    {
        userDetect = true;
    }

    // Start is called before the first frame update
    void Start()
    {
        // Obtiene los colliders
        colliders = colliders = GetComponents<Collider>();

        // Activa los colliders
        foreach (Collider collider in colliders)
        {
            collider.enabled = true;
        }

        // Busca el transform de la cámara principal usando el tag
        'MainCamera'
```

```
        if (user == null)
        {
            GameObject cameraGameObject =
GameObject.FindGameObjectWithTag("MainCamera");
            if (cameraGameObject != null)
            {
                user = cameraGameObject.transform;
            }
            else
            {
                Debug.LogError("No se encontró ningún objeto con el tag
'MainCamera'");
            }
        }

        // Encuentra la HealthBar por nombre
        GameObject healthBarGameObject = GameObject.Find("Health Bar");
        if (healthBarGameObject != null)
        {
            playerHealthBar =
healthBarGameObject.GetComponent<HealthBar>();
            if (playerHealthBar == null)
                Debug.LogError("El componente HealthBar no se encontró en
el objeto 'Health Bar'");
        }
        else
        {
            Debug.LogError("No se encontró ningún objeto llamado 'Health
Bar'");
        }

        health = 500;
        disparado = false;
        enemyAgent = GetComponent<UnityEngine.AI.NavMeshAgent>();
        enemyAnimator = GetComponent<Animator>();
        originalPosition = transform.position; // Almacena la posición
original al inicio
    }

    void Update()
    {
        // Calcula la distancia entre el enemigo y el jugador
        float distanceToUser = Vector3.Distance(transform.position,
user.position);
    }
}
```

```
// Calcula la distancia entre la posición actual y la posición original
float distanceToOriginal = Vector3.Distance(transform.position, originalPosition);

if(disparado)
{
    enemyAnimator.SetInteger("action",3);
    return;
}

// Verifica si el jugador está dentro del radio de detección
if(distanceToUser <= detectionRadius)
{
    userDetect = true;
}
else
{
    userDetect = false;
}

// Si el jugador ha sido detectado, persigue al jugador
if(userDetect)
{
    // Si el enemigo está lo suficientemente cerca del jugador,
    detenerse y reproducir la animación de ataque
    if(distanceToUser <= stoppingDistance)
    {
        enemyAnimator.SetInteger("action", 2);
        AnimatorStateInfo stateInfo =
        enemyAnimator.GetCurrentAnimatorStateInfo(0);
        if(stateInfo.IsName("Attack"))
        {
            int currentAttackTime = (int)(stateInfo.normalizedTime
            * 10); // Multiplique por 10 para obtener un índice más amplio
            if (currentAttackTime != lastAttackTime &&
            stateInfo.normalizedTime >= 0.5f)
            {
                playerHealthBar.TakeDamage(1);
                lastAttackTime = currentAttackTime; // Actualiza
                el ciclo de ataque
            }
        }
    }
    else
```



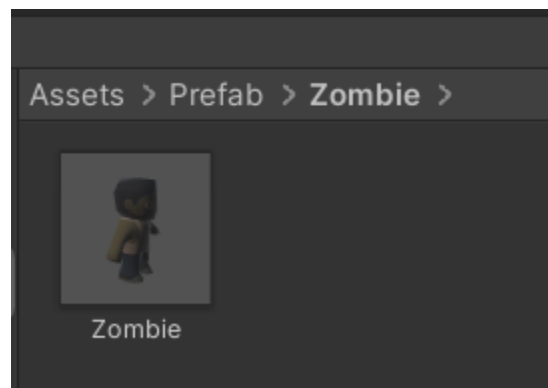
```
        {
            enemyAnimator.SetInteger("action", 1);
            enemyAgent.destination = user.position;
        }

    }
    else
    {
        // Si el jugador no ha sido detectado, regresa a su posición
original
        if (distanceToOriginal <= 0.5f) // Ajusta la tolerancia según
sea necesario
        {
            // Si el enemigo está en su posición original, cambia a la
animación Idle
            enemyAnimator.SetInteger("action",0);
            //Debug.Log("Cambio a IDLE");
        }
        else
        {
            enemyAgent.destination = originalPosition;
        }
    }
}

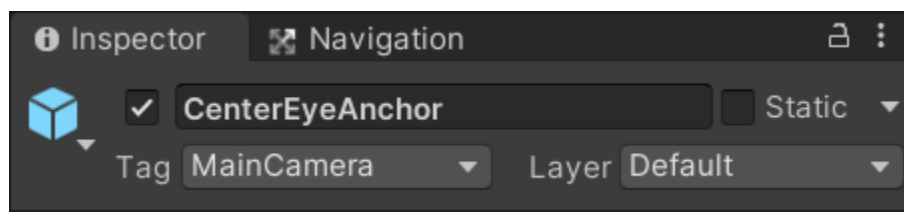
public void HitByRaycast()
{
    if (health > 0) // Solo reacciona si aún tiene vida
    {
        health -= 1; // Supongamos que cada golpe quita 50 de vida
        if (health <= 0)
        {
            if(!disparado)
                TriggerDeath();
        }
    }
}

private void TriggerDeath()
{
    Debug.Log("Zombie muere.");
    enemyAnimator.SetInteger("action", 3); // Asume que 3 es la
animación de muerte
    float animTime =
enemyAnimator.GetCurrentAnimatorStateInfo(0).length;
```

```
// Desactiva los colliders.  
foreach (Collider collider in colliders)  
{  
    collider.enabled = false;  
}  
  
Invoke(nameof(Respawn), animTime + delay);  
Destroy(gameObject, animTime + delay); // Asegura destruir después  
de la animación  
disparado = true;  
}  
  
private void Respawn()  
{  
    Instantiate(zombiePrefab, originalPosition, Quaternion.identity);  
}  
}
```



20.- Añadí el tag MainCamera al elemento CenterEyeAnchor que es la cámara dentro del OVRCameraRigInteraction. Este elemento es el que perseguirán los Zombies.

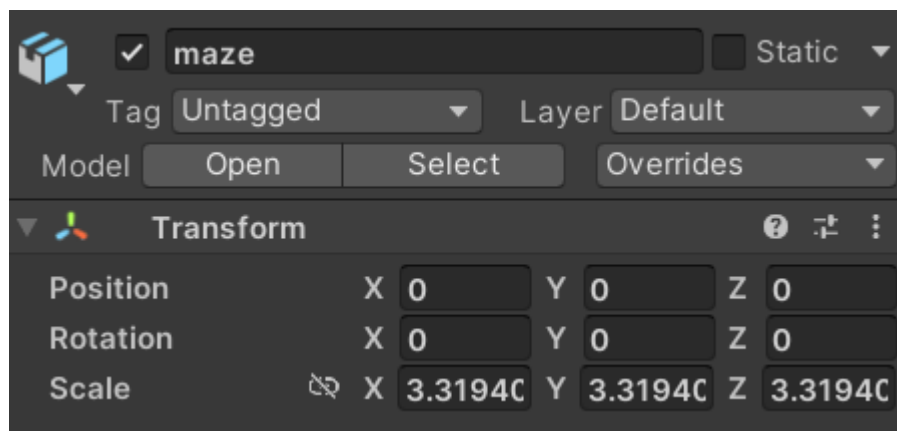
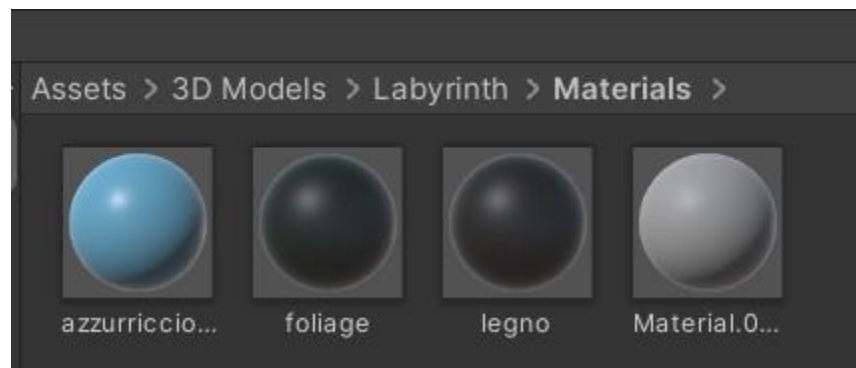
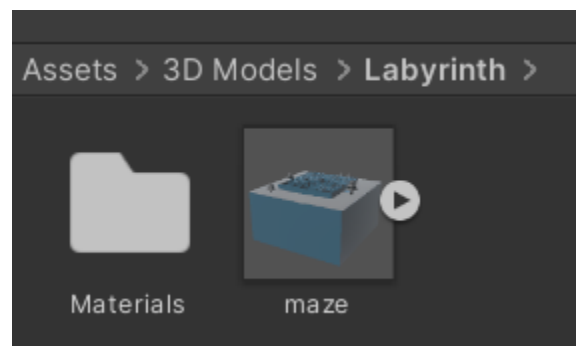


21.- Al final solo acomodé a los zombies en ciertas posiciones claves respecto a mi escenario. En total agregué cinco zombies.

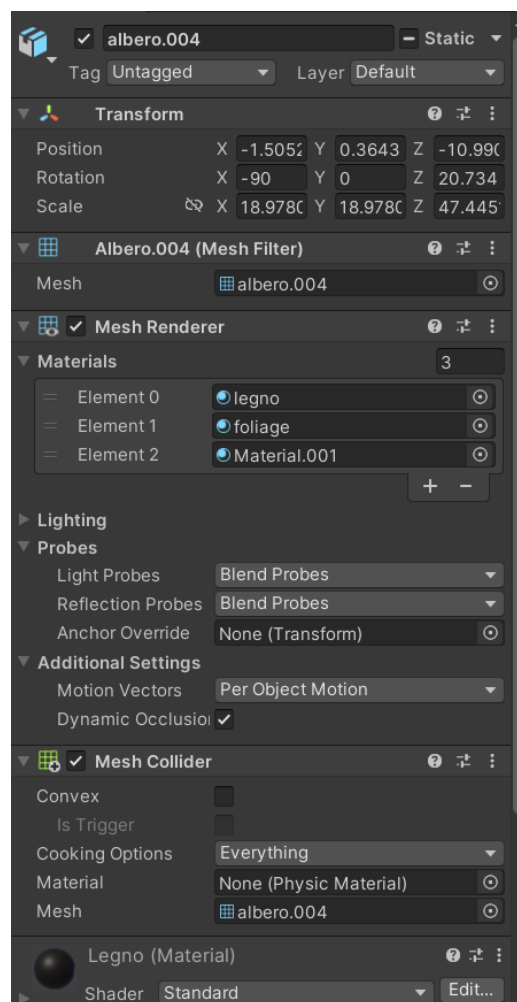
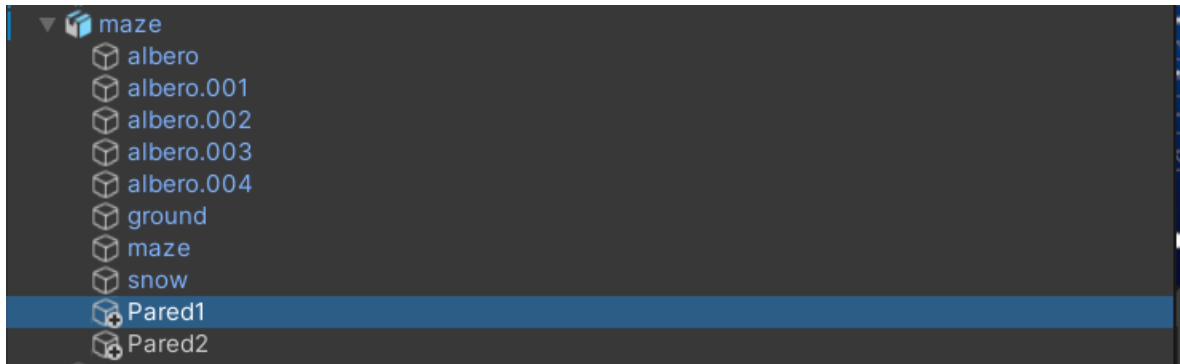


### PARA EL ESCENARIO

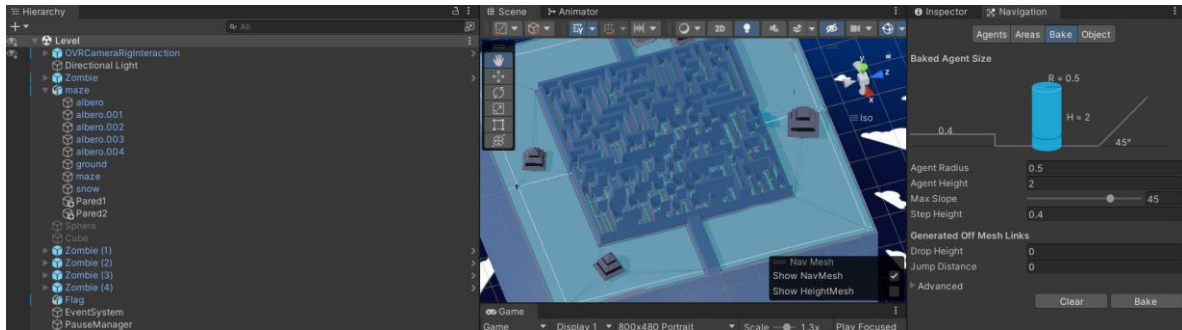
22.- Para el caso del escenario, este lo saque de aquí: <https://sketchfab.com/3d-models/winter-maze-3december2020-13b31219a53c4fe4afc2e72898ce2d83>. Lo descargue y lo importe. Le agregué sus respectivos materiales y lo escalé para acomodarlo en la escena.



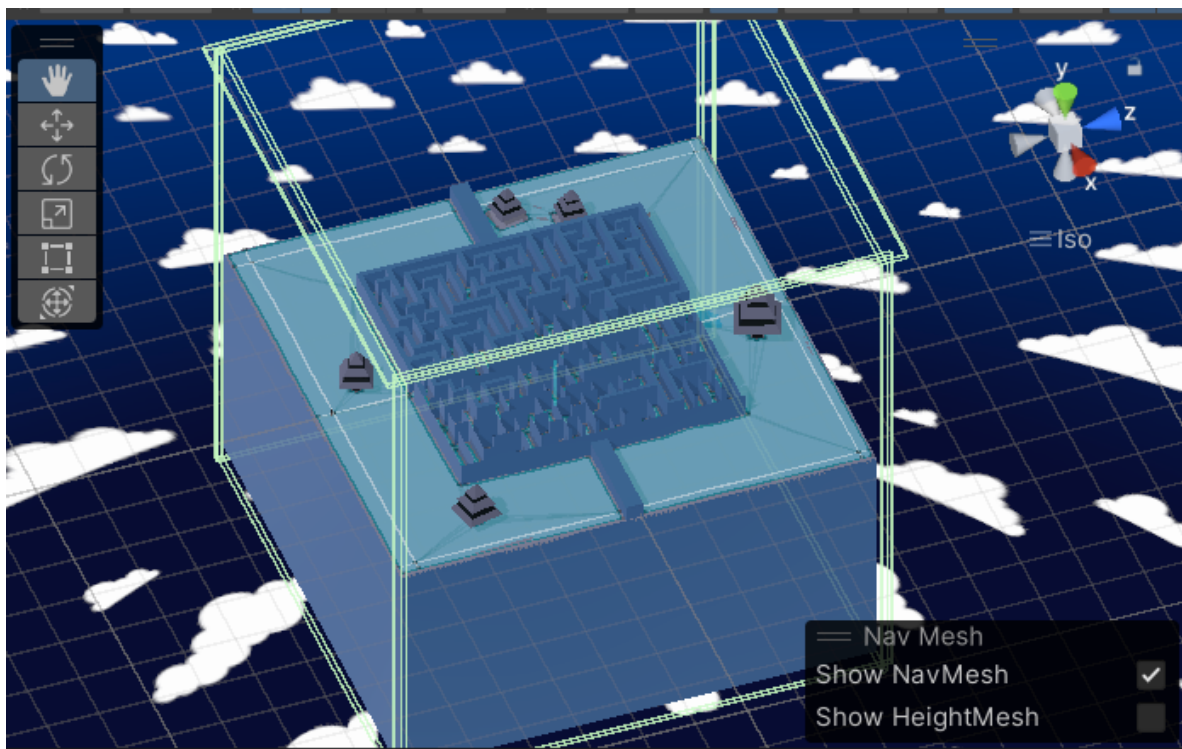
23.- Además tuve que añadir mesh colliders para cada elemento hijo del laberinto. Y añadir dos paredes para evitar que pudieras llegar al otro lado.



24.- Una vez acomodado el escenario simplemente fue cuestión de ir a Navigation y hacer un Bake del NavMesh. Seleccionando la escena completa, es importante destacarlo.

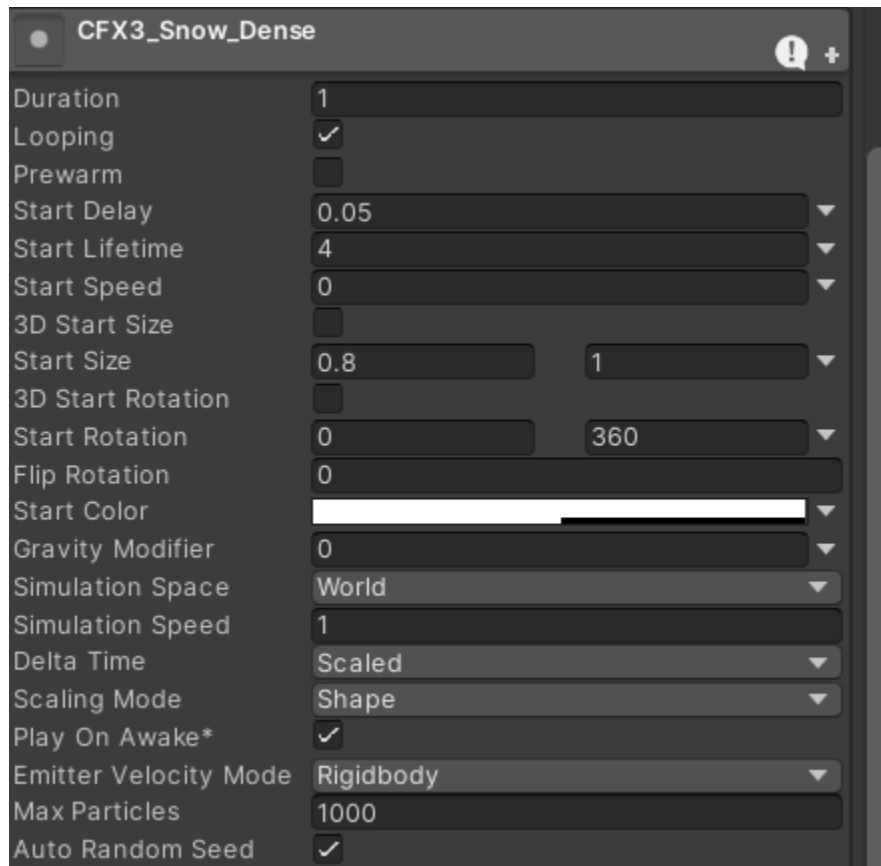


25.- Adicionalmente añadí colliders para los limites del escenario e incluso para un 'techo' falso.

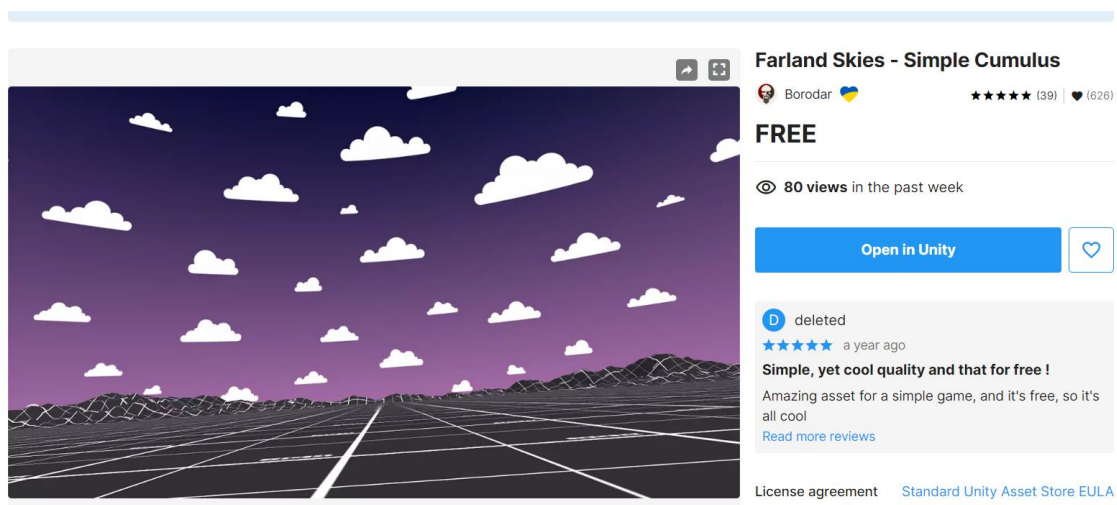


26.- Finalmente añadí partículas instanciadas directamente del mismo paquete que importe, en este caso de nieve. A estas tuve que editarles ciertos parámetros. Como el 3D size y el Max Particles.



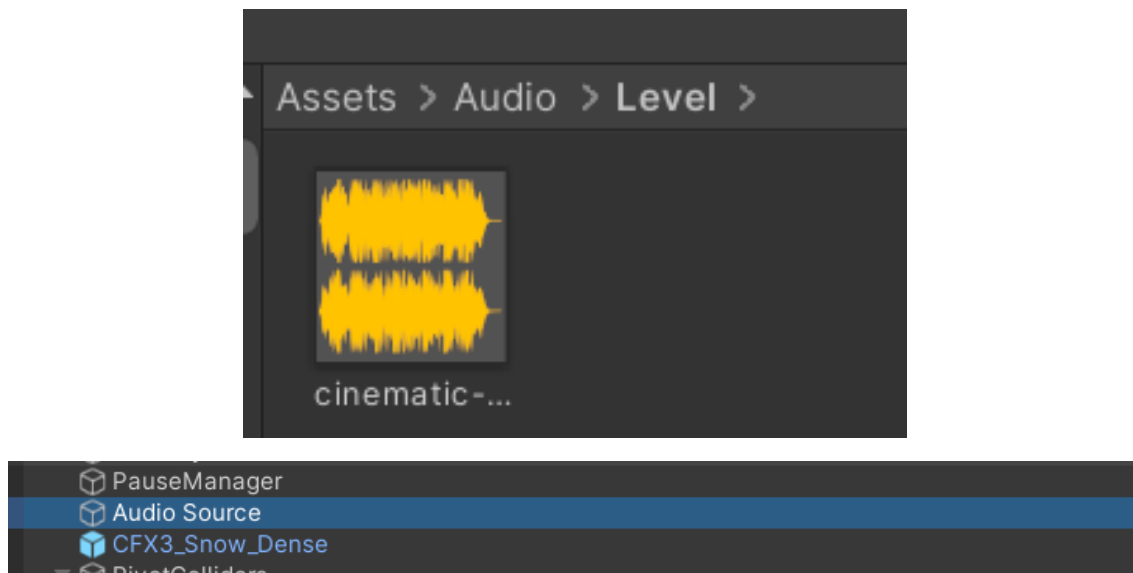


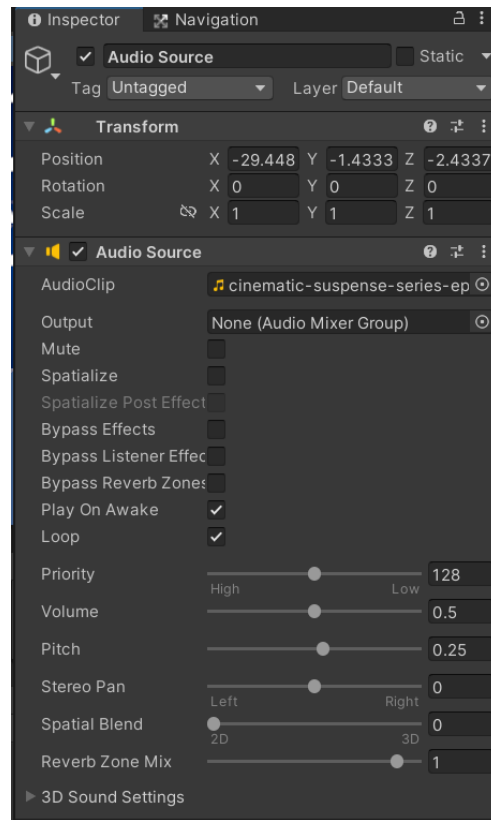
27.- Para el skybox utilicé el siguiente paquete de la Asset Store: <https://assetstore.unity.com/packages/2d/textures-materials/sky/farland-skies-simple-cumulus-62565>. Lo importe a mi proyecto con el Package Manager y simplemente lo arrastre a mi escena.



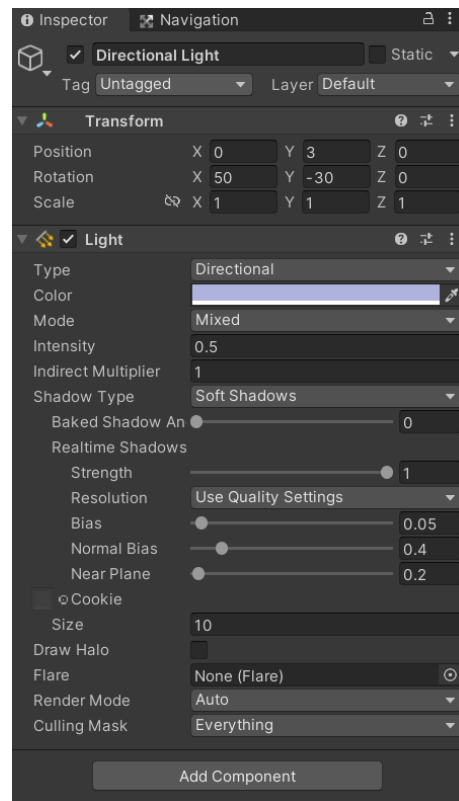


28.- Para la música del nivel la saque de: <https://filmmusic.io/song/5786-cinematic-suspense-series-episode-001>. Simplemente agregué un Audio Source a la escena que la reproduzca en Loop. Con ciertas modificaciones.





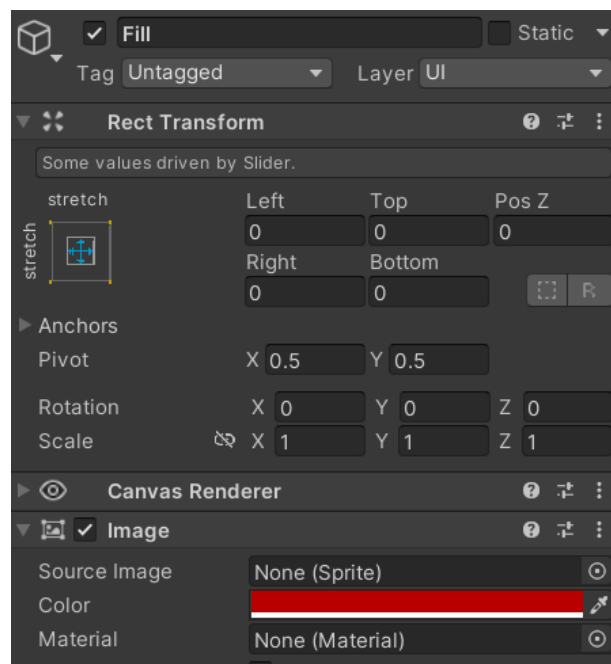
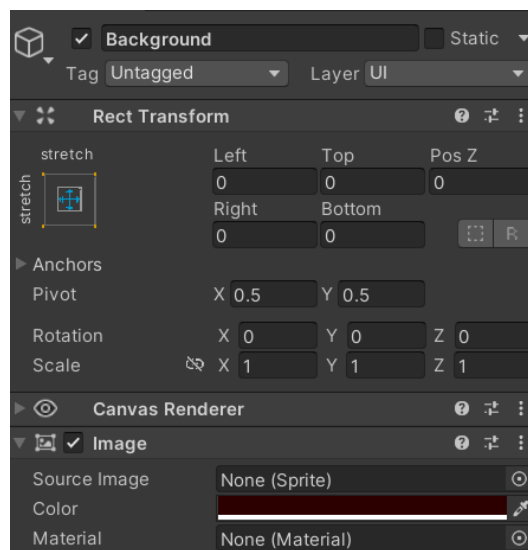
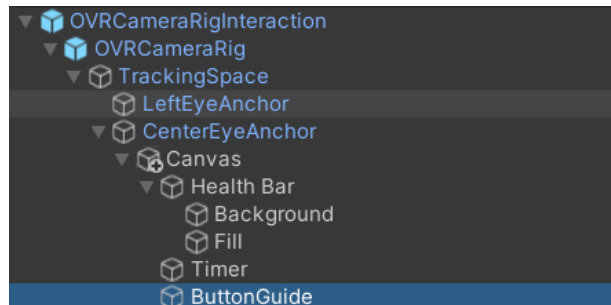
29.- Configuré el color y todo de la Directional Light.



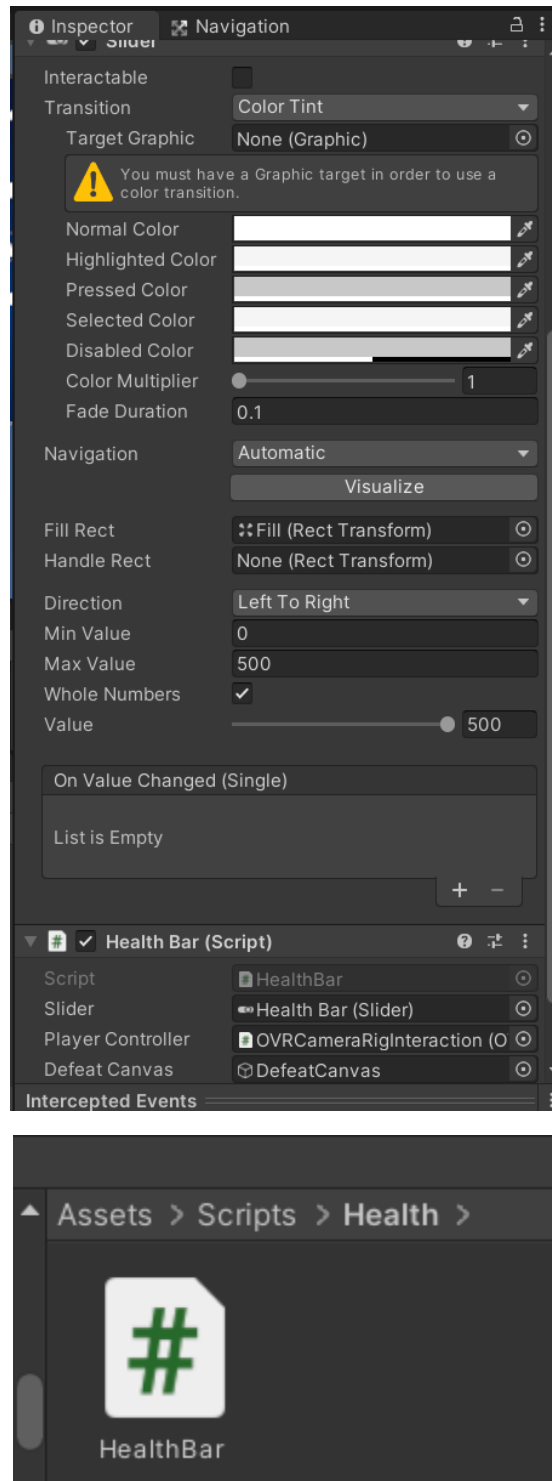


### PARA LA INTERFAZ DE VIDA Y TIEMPO

30.- En este caso agregue un canvas para la vida y el tiempo. La barra de vida tiene un background con color oscuro y un fill de color rojo.



31.- La barra de vida tambien tiene un slider y un script llamado HealthBar.



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

```
using UnityEngine.SceneManagement;

public class HealthBar : MonoBehaviour
{
    public Slider slider; // Referencia al Slider UI
    public OVRPlayerController playerController; // Referencia al
    OVRPlayerController que se desactivará al finalizar el tiempo
    public GameObject defeatCanvas;

    void Start()
    {
        if (slider == null)
        {
            slider = GetComponent<Slider>();
        }
    }

    // Método público para establecer la salud
    public void SetHealth(int health)
    {
        slider.value = health;
    }

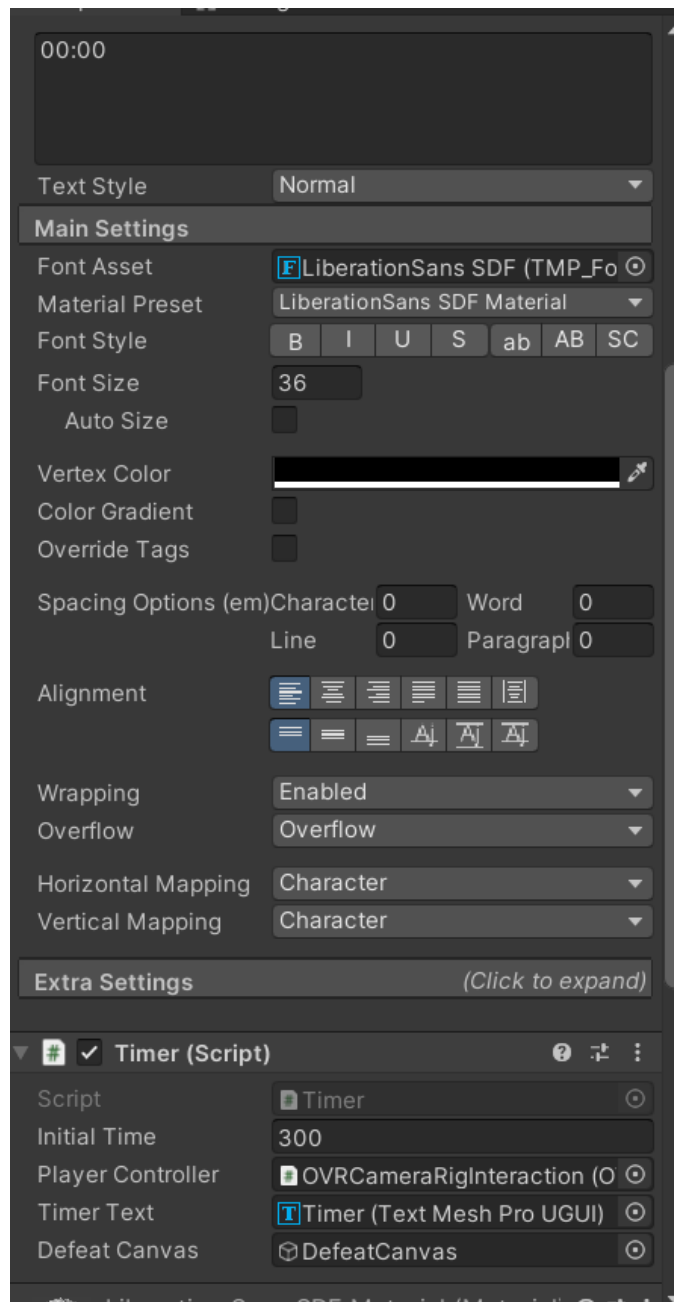
    // Método para decrementar la salud
    public void TakeDamage(int damage)
    {
        slider.value -= damage;
        if (slider.value < 0)
            slider.value = 0;
    }

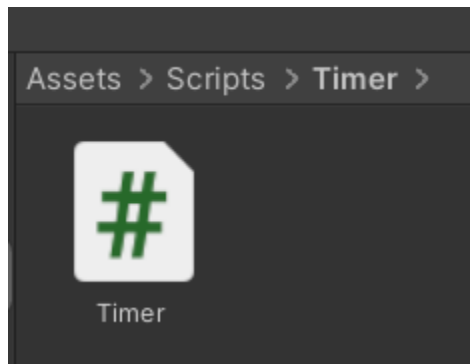
    // Verificar si la salud es 0 y actuar en consecuencia
    void Update()
    {
        if (slider.value <= 0)
        {
            Die();
        }
    }

    // Manejar la "muerte" del jugador
    private void Die()
    {
        // Logica de muerte
        // Pausar el juego
    }
}
```

```
    defeatCanvas.SetActive(true);  
    Time.timeScale = 0f;  
    playerController.enabled = false;  
}  
  
}
```

32.- El timer tiene un TextMeshPro y un script llamado Timer.





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

public class Timer : MonoBehaviour
{
    public float initialTime = 300f; // 5 minutos en segundos
    private float currentTime;
    public OVRPlayerController playerController; // Referencia al
    OVRPlayerController que se desactivará al finalizar el tiempo
    public TMPro.TextMeshProUGUI timerText;
    public GameObject defeatCanvas;

    void Start()
    {
        currentTime = initialTime;
    }

    void Update()
    {
        // Reducir el tiempo restante
        currentTime -= Time.deltaTime;

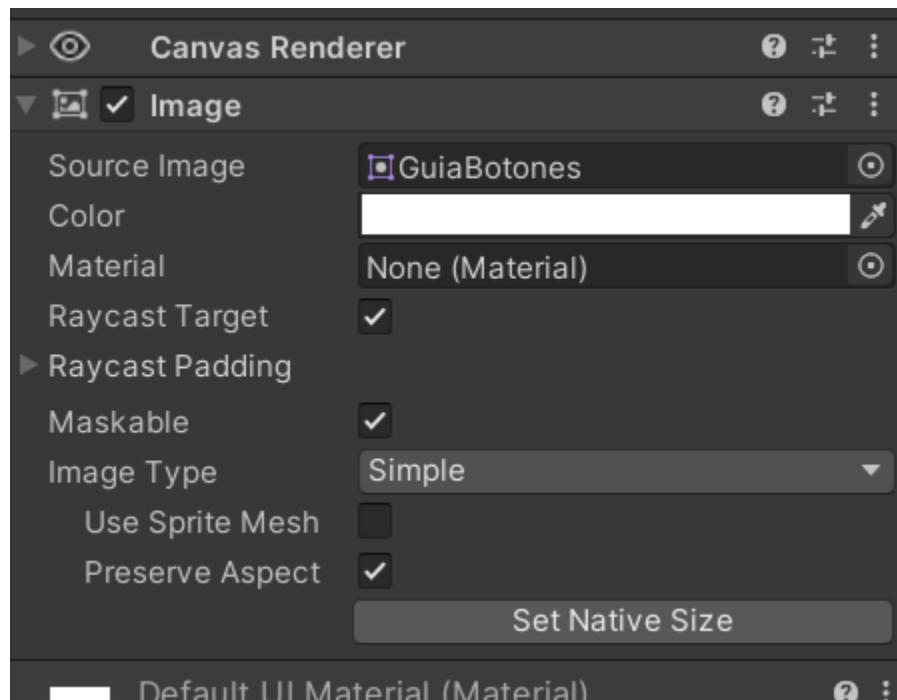
        // Actualizar el texto del temporizador en la interfaz de usuario
        DisplayTime(currentTime);

        // Verificar si el tiempo restante ha llegado a cero
        if (currentTime <= 0)
        {
            EndGame(); // Aquí deberías implementar tu lógica de derrota
        }
    }

    void DisplayTime(float timeToDisplay)
```

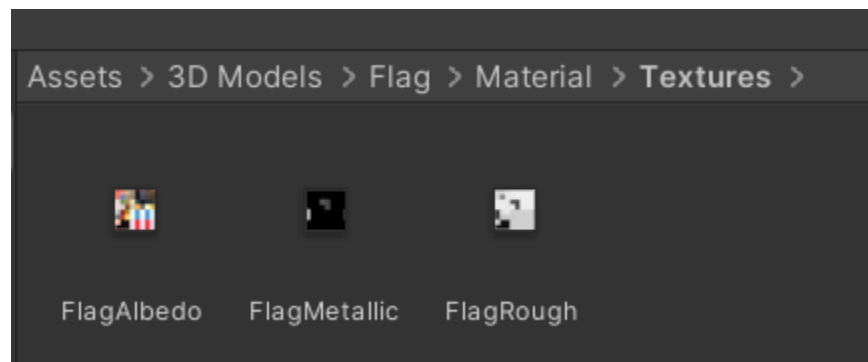
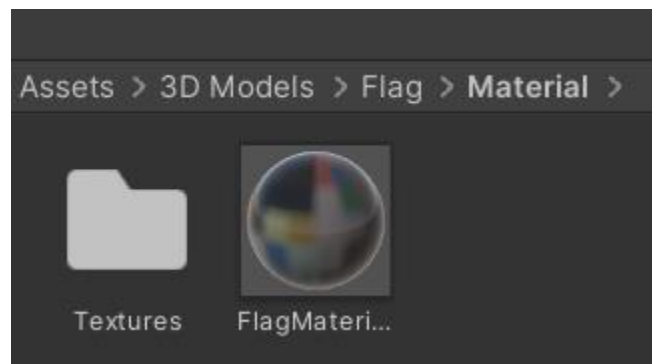
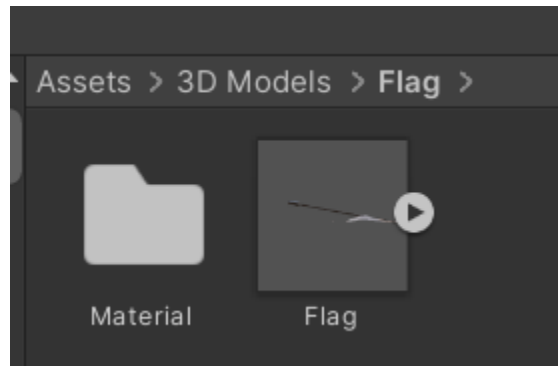
```
{  
    // Convertir el tiempo a minutos y segundos  
    float minutes = Mathf.FloorToInt(timeToDisplay / 60);  
    float seconds = Mathf.FloorToInt(timeToDisplay % 60);  
  
    // Actualizar el texto en el formato MM:SS  
    timerText.text = string.Format("{0:00}:{1:00}", minutes, seconds);  
}  
  
void EndGame()  
{  
    // Pausar el juego  
    defeatCanvas.SetActive(true);  
    Time.timeScale = 0f;  
    playerController.enabled = false;  
}  
}
```

33.- Finalmente el ButtonGuide solo es una imagen de guía de los controles.



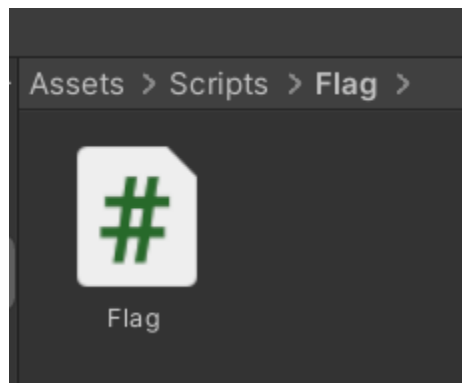
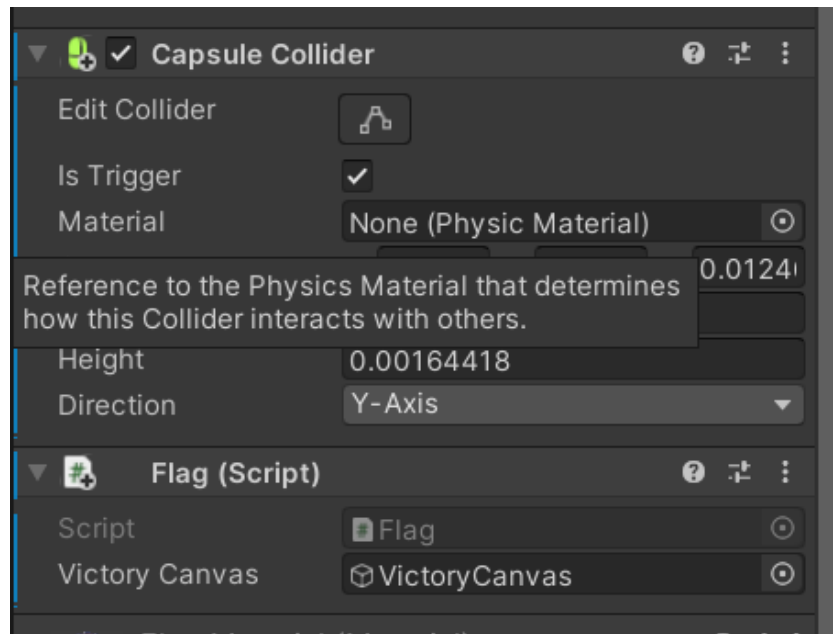
**PARA LA VICTORIA, DERROTA Y LA PAUSA**

34.- Para la victoria agregue un modelo de bandera sacado de: <https://sketchfab.com/3d-models/flag-e947a018194745598df88444bf43de11>. Lo importe en mi proyecto y extraje sus materiales correspondientes con sus texturas.



35.- La agregué a la escena, le puse un collider y le generé un script llamado Flag.





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

public class Flag : MonoBehaviour
{
    public GameObject victoryCanvas;

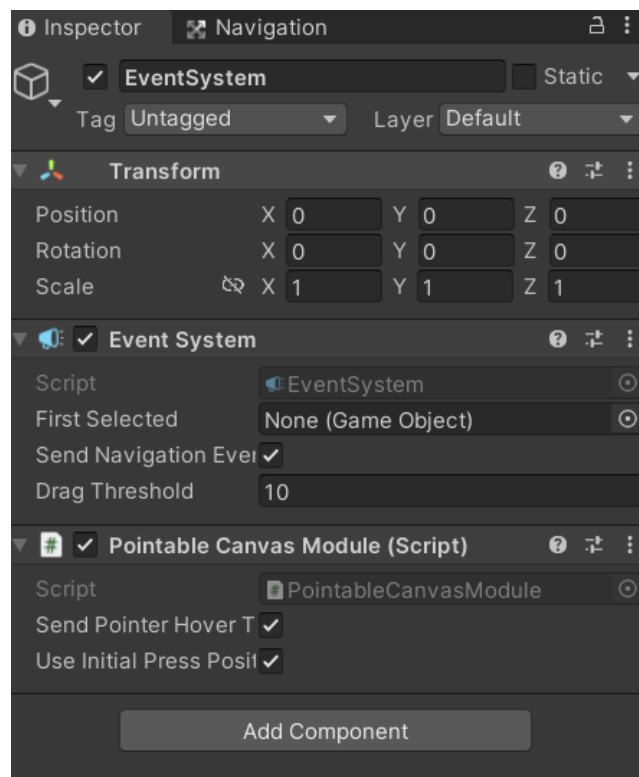
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            // Activar el Canvas de victoria cuando el jugador colisiona
            con la bandera
            if (victoryCanvas != null)
            {
                victoryCanvas.SetActive(true);
            }
        }
    }
}
```



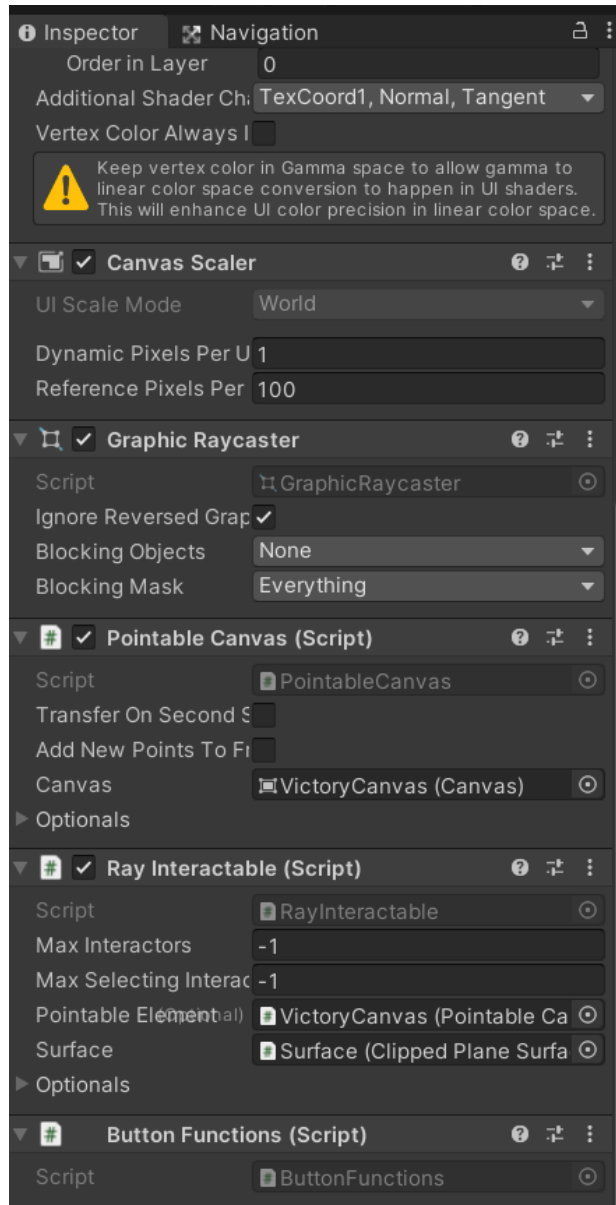
```
        Time.timeScale = 0f; // Pausar el juego al activar el
        Canvas de victoria
        // Desactivar el script OVRPlayerController
        DesactivarScript(other.GetComponent<OVRPlayerController>())
    );
    }
}

// Método para desactivar un script en un componente MonoBehaviour
private void DesactivarScript(MonoBehaviour script)
{
    if (script != null)
    {
        script.enabled = false;
    }
}
}
```

36.- Generé un elemento de Event System y le cambie su input module por el Pointable Canvas Module.



37.- Generé un VictoryCanvas con los scripts Pointable Canvas, Ray Interactable y Buttons Functions. El ultimo lo generé yo.



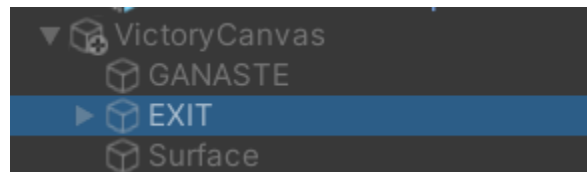
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

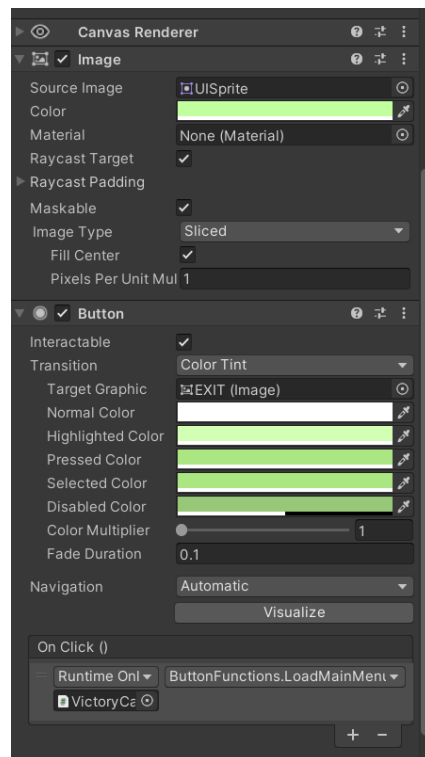
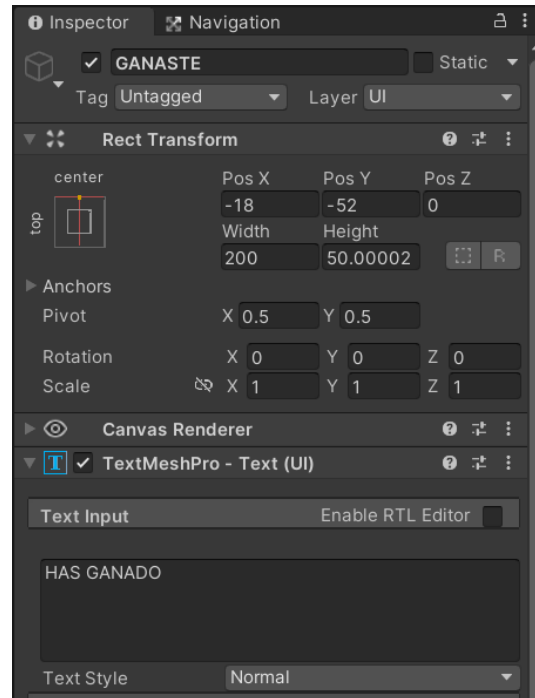
public class ButtonFunctions : MonoBehaviour
{
    // Función para cargar el menu
    public void LoadMainMenu()
    {

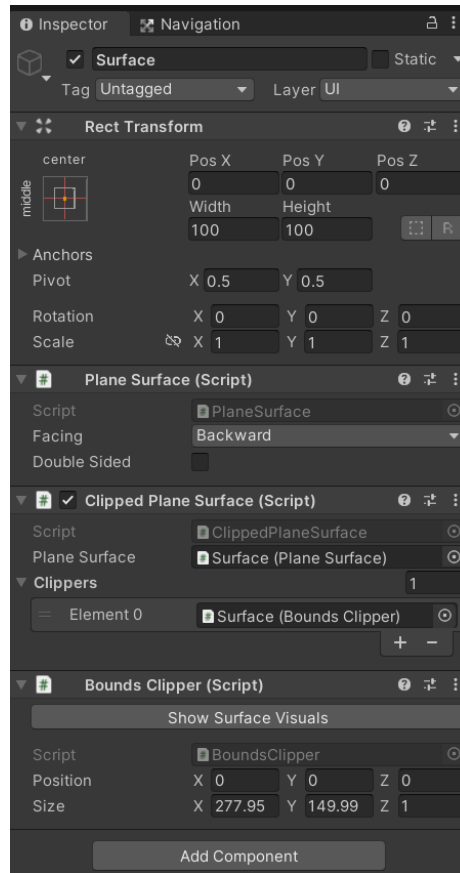
```

```
        Time.timeScale = 1f;  
        SceneManager.LoadScene(0);  
    }  
  
    // Función para cargar el nivel  
    public void LoadLevel()  
    {  
        Time.timeScale = 1f;  
        SceneManager.LoadScene(1);  
    }  
  
    // Función para salir del juego  
    public void ExitGame()  
    {  
        Application.Quit();  
    }  
}
```

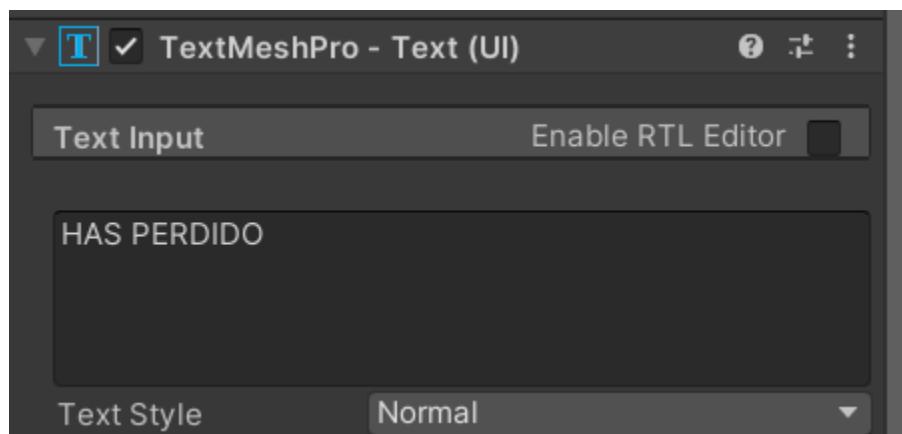
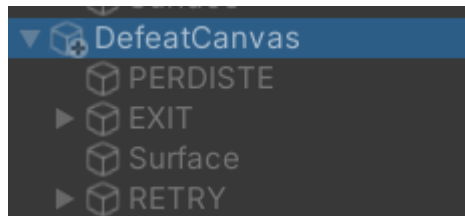
38.- Dentro del canvas oculto de victoria hay un Text Mesh Pro. Llamado ganaste. Un botón con un Sprite. Y un elemento en blanco Surface que tiene los scripts Plane Surface, Clipped Plane Surface y Bounds Clipper.

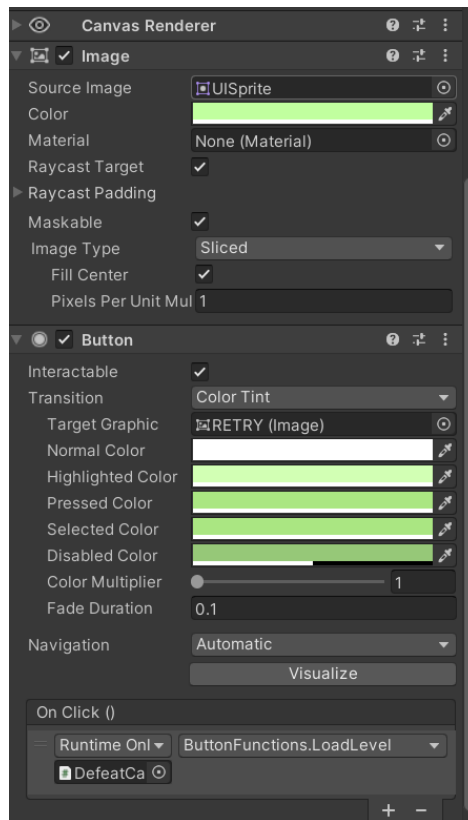




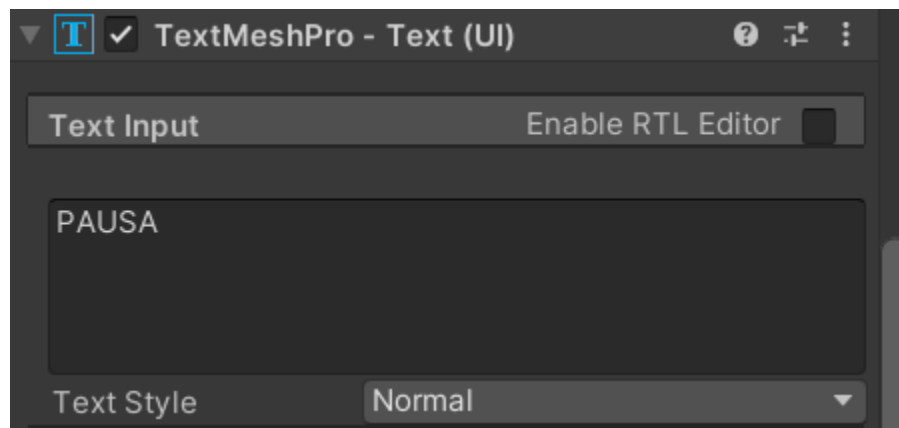
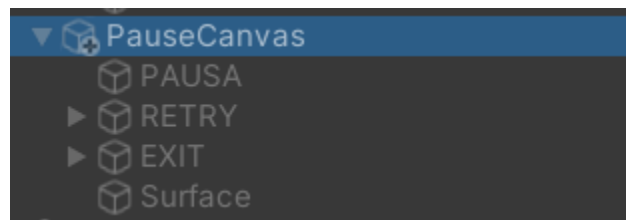


39.- Para el canvas de derrota se tienen exactamente los mismos elementos, solo cambia el texto. Y se agrega un botón adicional de reintentar.

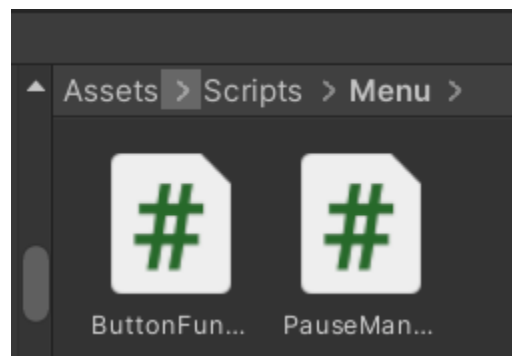
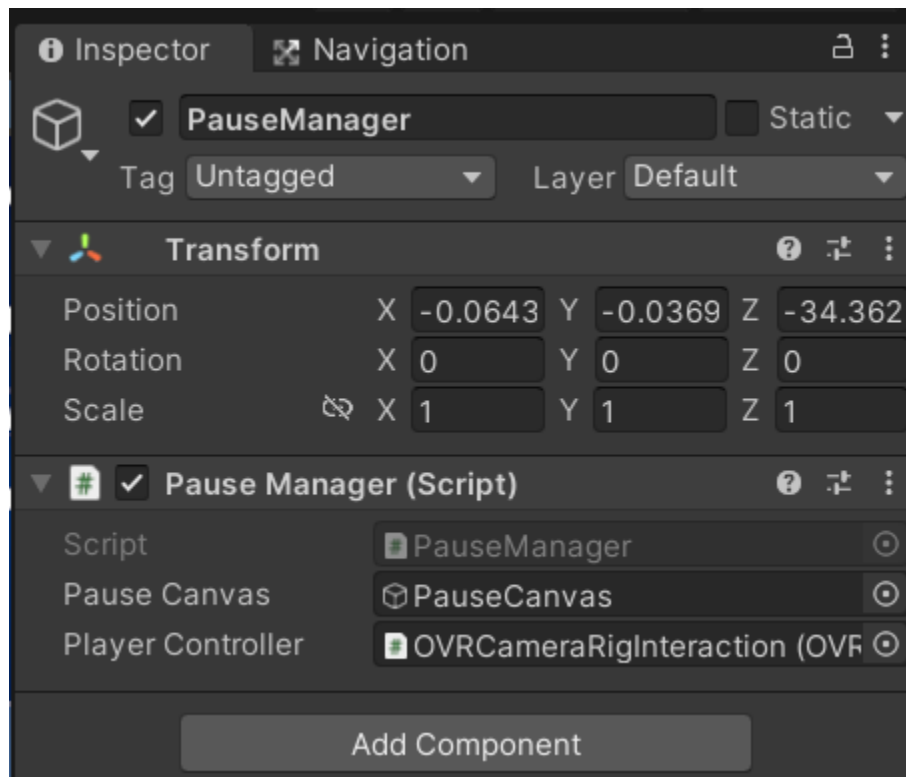




40.- Para el canvas de Pausa se tienen exactamente los mismos elementos que el de derrota. Únicamente cambia el texto.



41.- Adicionalmente para agregar la funcionalidad de pausa se agrego a la escena un componente vacío llamado PauseManager, el cual solo incluye un script llamado PauseManager.



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

public class PauseManager : MonoBehaviour
{
    public GameObject pauseCanvas; // Referencia al canvas que se mostrará
    al pausar la escena
    public OVRPlayerController playerController; // Referencia al
    OVRPlayerController que se desactivará al finalizar el tiempo
}
```

```
private bool _isPaused = false;

void Start()
{
    // Establecer el timeScale en 1 al inicio
    Time.timeScale = 1f;
    playerController.enabled = true;

    // Asegurarse de que el canvas de pausa esté desactivado al inicio
    if (pauseCanvas != null)
    {
        pauseCanvas.SetActive(false);
    }
}

void Update()
{
    // Verificar la entrada para pausar y despausar la escena
    if (OVRInput.GetDown(OVRInput.Button.Start,
OVRInput.Controller.LTouch))
    {
        if (_isPaused)
        {
            ResumeGame();
        }
        else
        {
            if (!IsActive())
                PauseGame();
        }
    }
}

// Método para pausar la escena
void PauseGame()
{
    _isPaused = true;
    Time.timeScale = 0f; // Congelar el tiempo
    playerController.enabled = false;

    // Mostrar el canvas de pausa si está asignado
    if (pauseCanvas != null)
    {
        pauseCanvas.SetActive(true);
    }
}
```



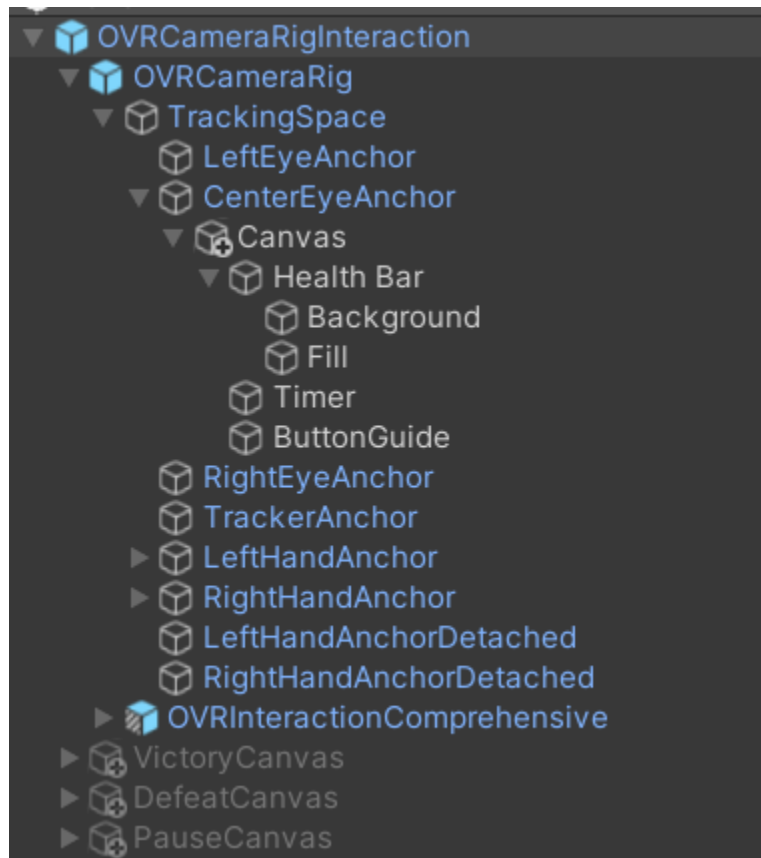
```
}

// Método para despausar la escena
void ResumeGame()
{
    _isPaused = false;
    Time.timeScale = 1f; // Reanudar el tiempo
    playerController.enabled = true;

    // Ocultar el canvas de pausa si está asignado
    if (pauseCanvas != null)
    {
        pauseCanvas.SetActive(false);
    }
}

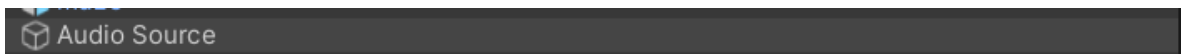
// Método para verificar si hay algún elemento con el tag "UI" activo
en la escena
bool IsUIActive()
{
    GameObject[] uiElements = GameObject.FindGameObjectsWithTag("UI");
    foreach (GameObject uiElement in uiElements)
    {
        if (uiElement.activeSelf)
        {
            return true;
        }
    }
    return false;
}
}
```

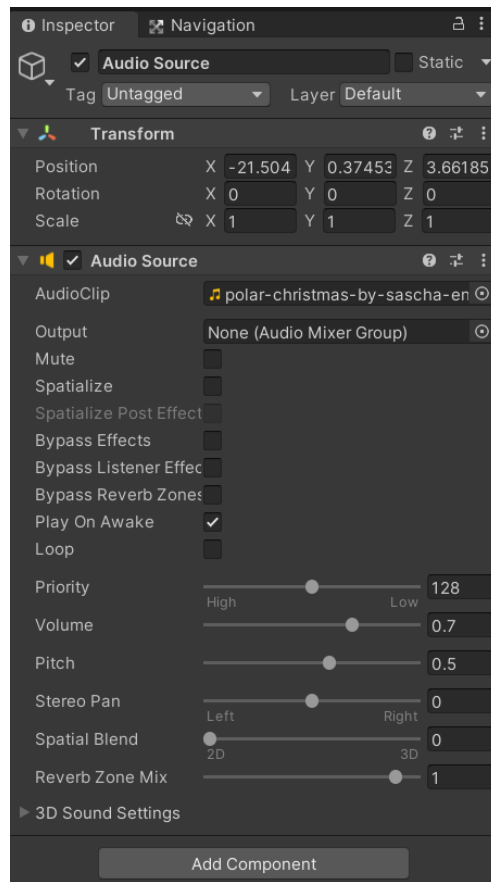
42.- Es importante recalcar que TODOS los canvas están como hijos del OVRCameraRigInteraction. Y están acomodados específicamente para que se vean desde el punto de vista del jugador.



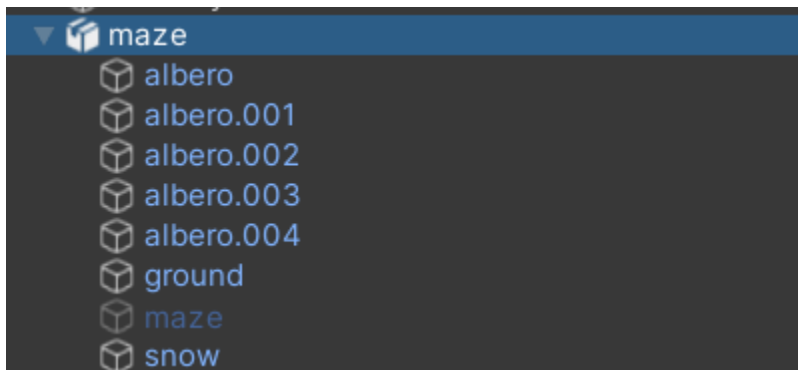
#### PARA EL MENÚ PRINCIPAL

43.- Para el menú principal saque una canción de: <https://filmmusic.io/song/5497-polar-christmas>. La coloque en un Audio Source en la escena y le modifique algunos parámetros. La configure para loop y que empiece sola.



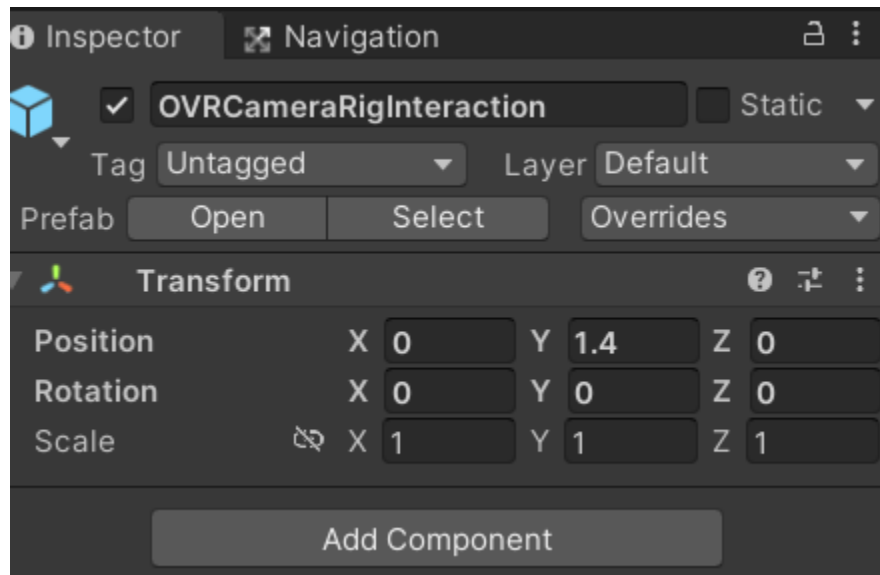


44.- Añadí mi escenario, pero le quite las paredes del laberinto.

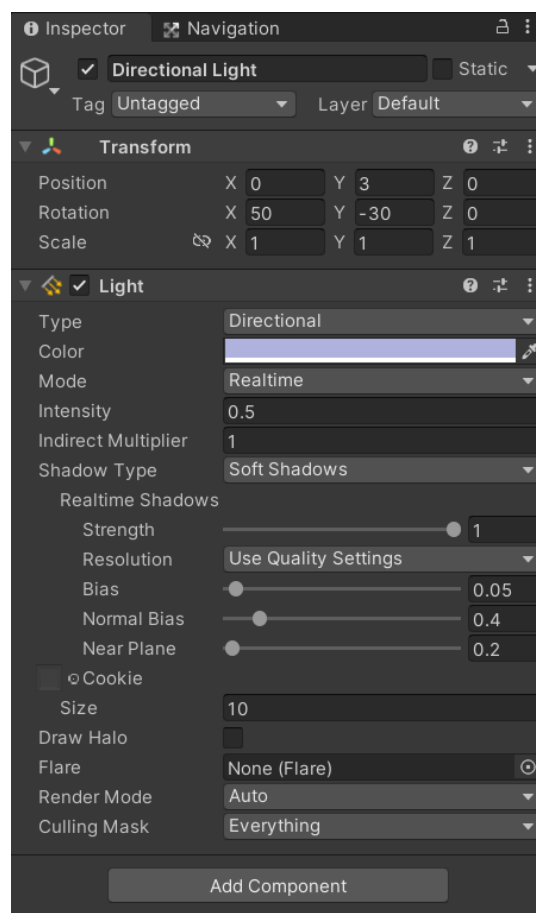


45.- También agregué las mismas partículas de nieve. Pero adentro de mi cámara. Además esta vez estoy usando el OVRCameraRigInteraction puro, directo como viene el prefab.

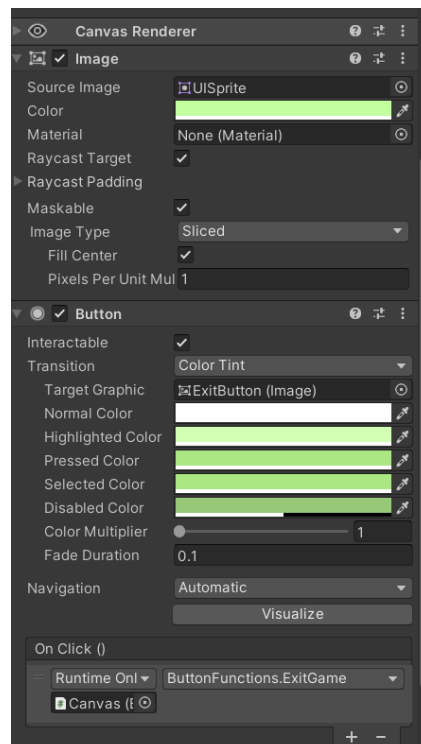
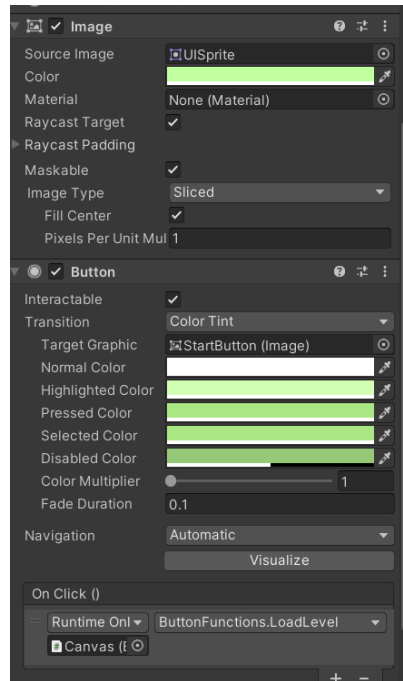
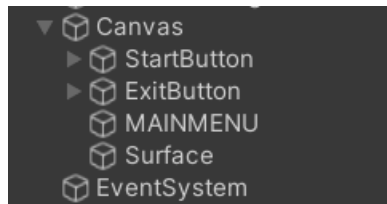


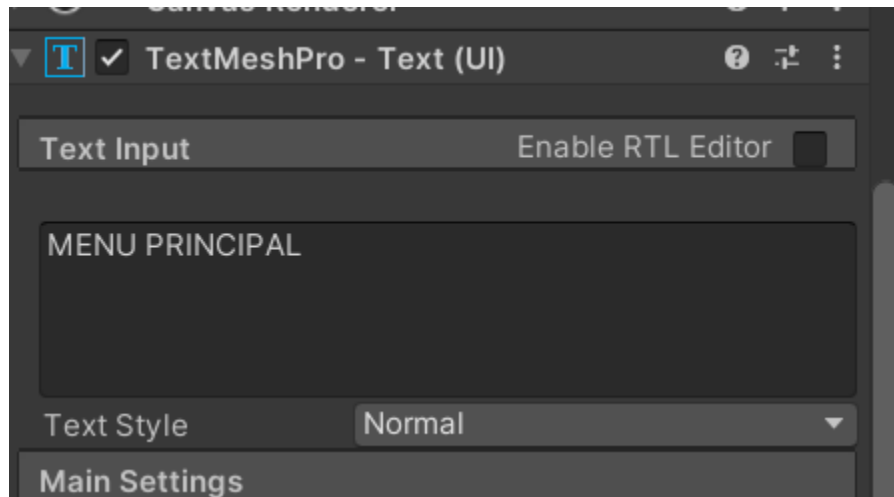


46.- Configure el Directional Light igual.



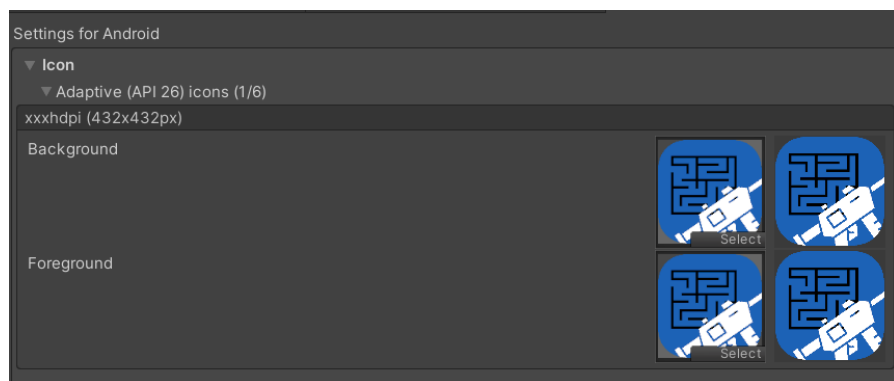
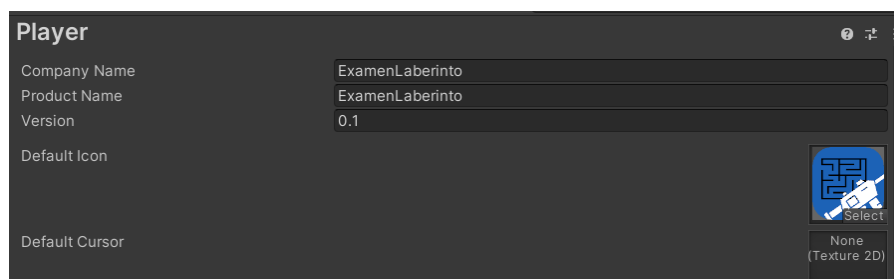
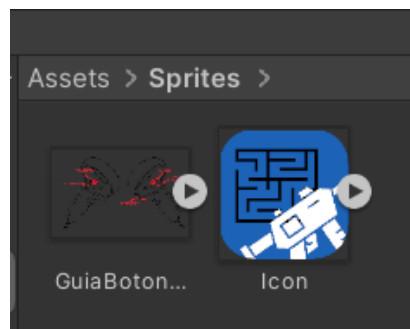
47.- Finalmente agregué un canvas con las mismas configuraciones y un Event System con las mismas configuraciones en scripts. Solo que ahora el Canvas tiene un texto diferente y dos botones que dicen Comenzar y Salir, cada uno llamando a diferentes funciones del script ButtonFunctions.





### PARA EL ICONO

48.- El icono lo realice usando GIMP y simplemente lo importe como Sprite al proyecto para usarlo en la parte de Player Settings.



## Conclusiones

En la realización de este proyecto considero que el aprendizaje mas importante es el trabajar adecuadamente con el SDK de Meta para Unity, ya que, si bien ya tenia experiencia trabajando con el mismo debido a que realice el proyecto final para la misma plataforma y lo concluí antes de este examen, si considero que me sirvió para reforzar mi conocimiento sobre el uso del mismo e incluso entendí como hacer nuevas modificaciones necesarias que en mi proyecto no lo fueron, como el Player Controller para detectar el movimiento del jugador con el stick izquierdo como cualquier control, cosa para la que tuve que comparar dos prefabs que ofrecía el SDK y agregar lo que le faltaba al prefab que estaba usando yo personalmente basándome en el prefab que tiene movimiento. También destaco la lectura de los botones de los controles para mis propias validaciones, ya que en el proyecto no tuve la necesidad de hacerlo pero aquí si para desarrollar la mecánica del disparo. Además, considero que en la realización de este examen se consolidaron conceptos de practicas anteriores como el NavMesh que entendí mejor como realizarlo, siendo necesario seleccionar toda la escena para que tome en cuenta todos los elementos a la hora de hacer el Bake o seleccionando solo los que quiero que tome en cuenta; o el raycast que habíamos revisado en las primeras prácticas y que en su momento no entendí tan bien como ahorita.

También fue nuevo para mi el uso de partículas implementadas en el editor de partículas de Unity, que si bien fueron partículas descargadas si tuve que entender un poco los parámetros para ajustarlas a mis necesidades.

## Repositorio

[https://github.com/Yoaddan/OculusExam\\_VR](https://github.com/Yoaddan/OculusExam_VR)

## Referencias:

- <https://www.youtube.com/watch?v=dXgb6J46yjk>
- [https://www.youtube.com/watch?v=q2auix\\_Vqd8](https://www.youtube.com/watch?v=q2auix_Vqd8)
- <https://www.youtube.com/watch?v=3JjBJfoWDCM>
- <https://www.youtube.com/watch?v=xk-LLY7scvM&list=PL3jOzOoZU5TjiVT5IFRSyH2FZUnkXT5UA&index=12>
- <https://www.youtube.com/watch?v=xk-LLY7scvM&t=2197s>