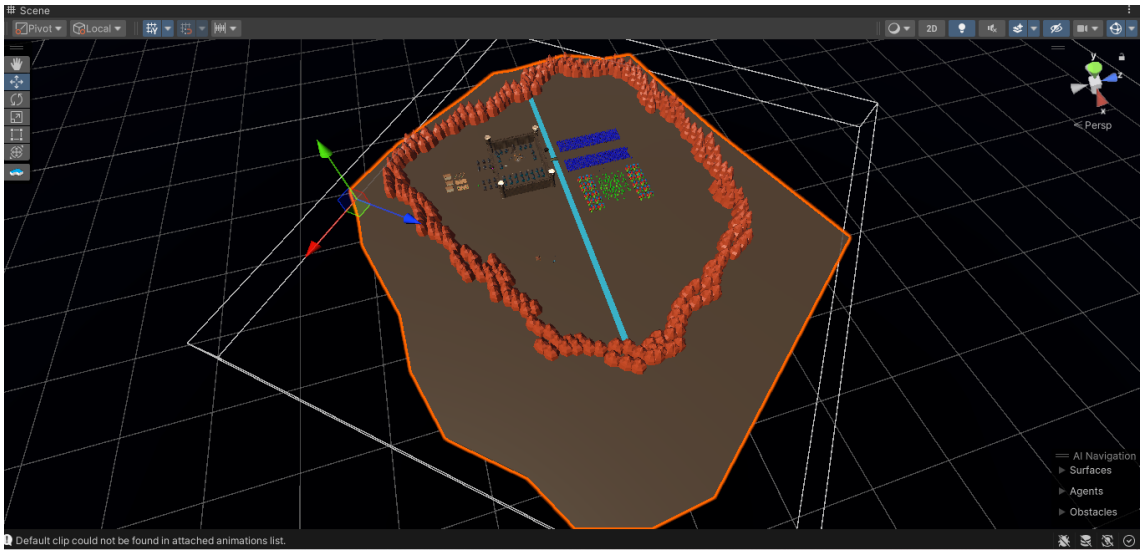
Memoria 3D proyecto

Jorge Wang Wang

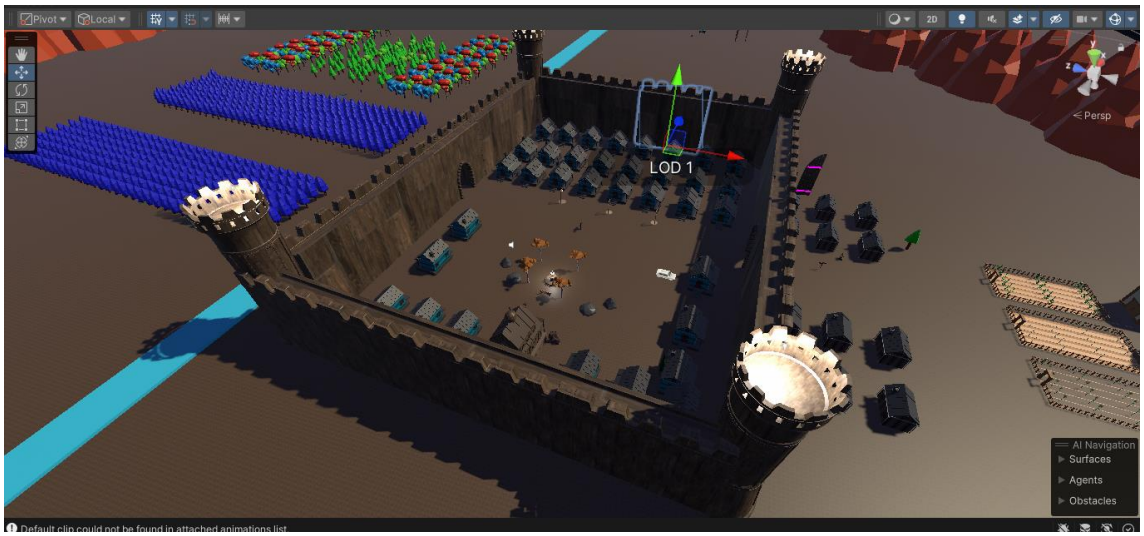
En este trabajo necesitábamos crear un nuevo escenario en donde se incluiría varios elementos de sonido e iluminación. Por lo que cree una nueva escena llamada escenario.

Creación del escenario

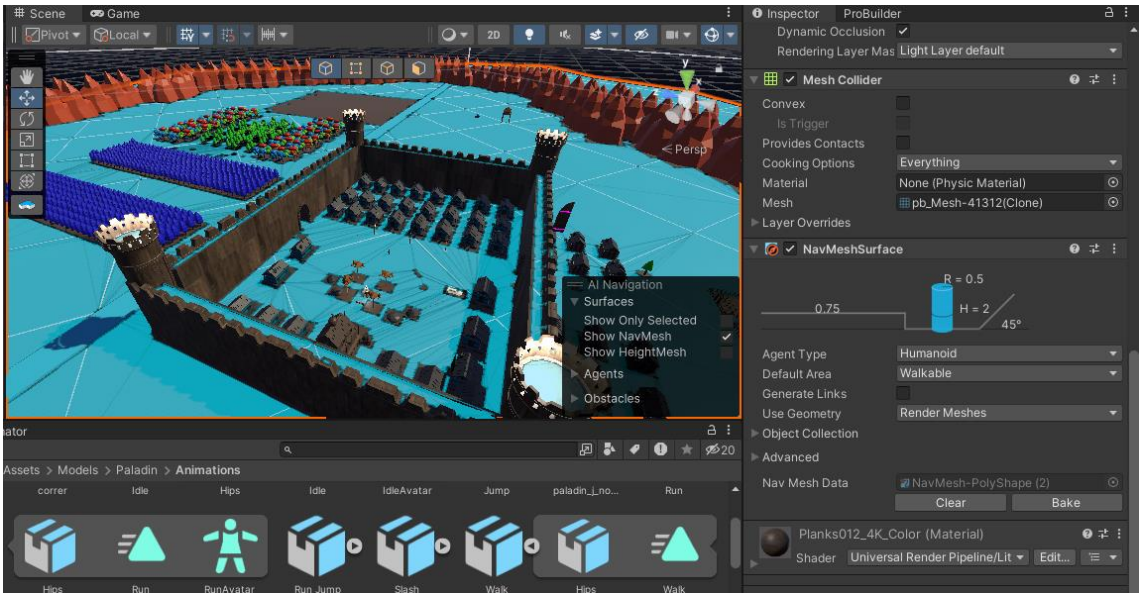
El escenario primero decidí crear usando ProBuilder un suelo para que su forma no sea cuadrada. Por lo que use su poli shape para crear un suelo poligonal con varios lados. Además, añadí montañas en un circulo para delimitar la zona además de poner un collider para que no pasen.



Además, ya viendo la foto se puede ver varias zonas, la primera es un castillo la cual contendrá la mayoría audios.

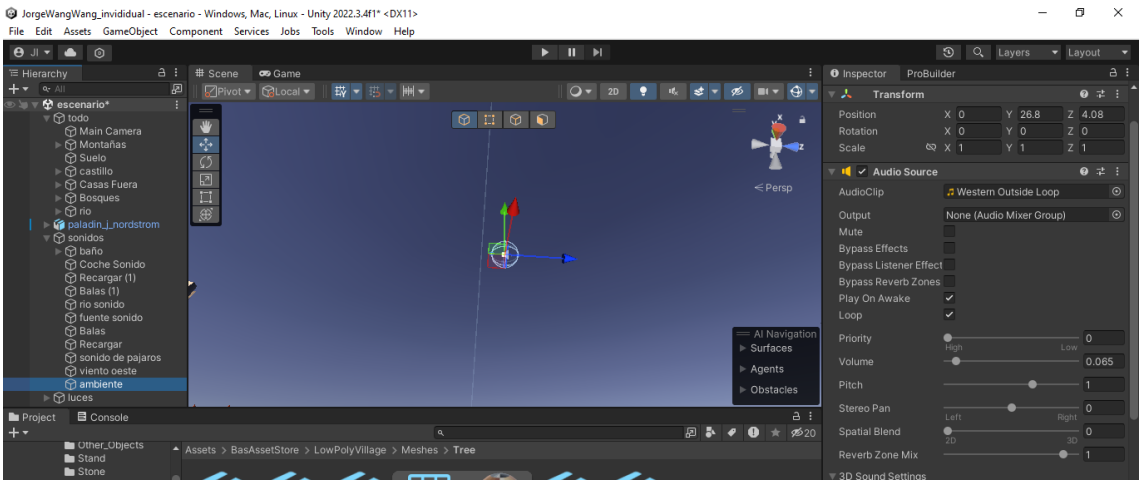


Y luego estaría el exterior del castillo en donde lo use como ambiente, y tambien añadí algunos sonidos que detallare más adelante. Tambien al suelo le añadí el componente de navMesh que sirve para marcar una zona en donde se podrían hacer patrullajes con otros personajes.

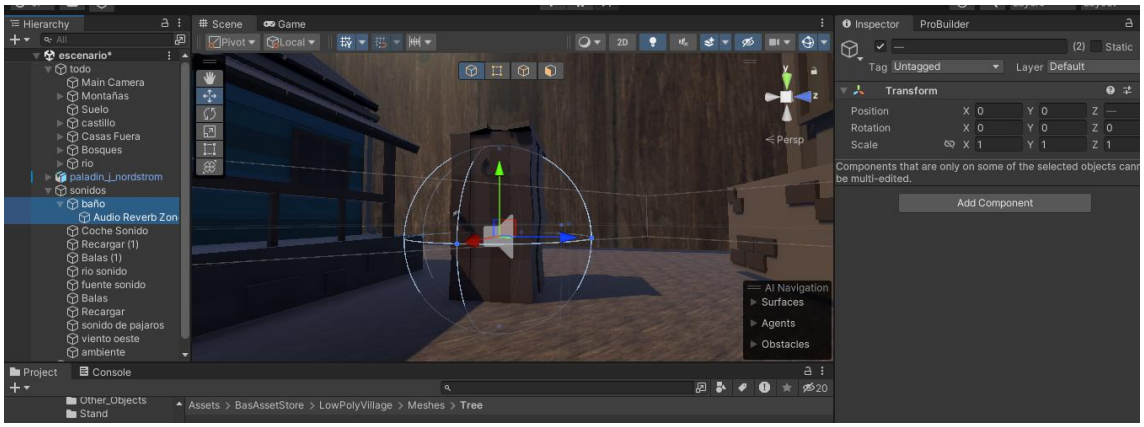


Audios y Luces

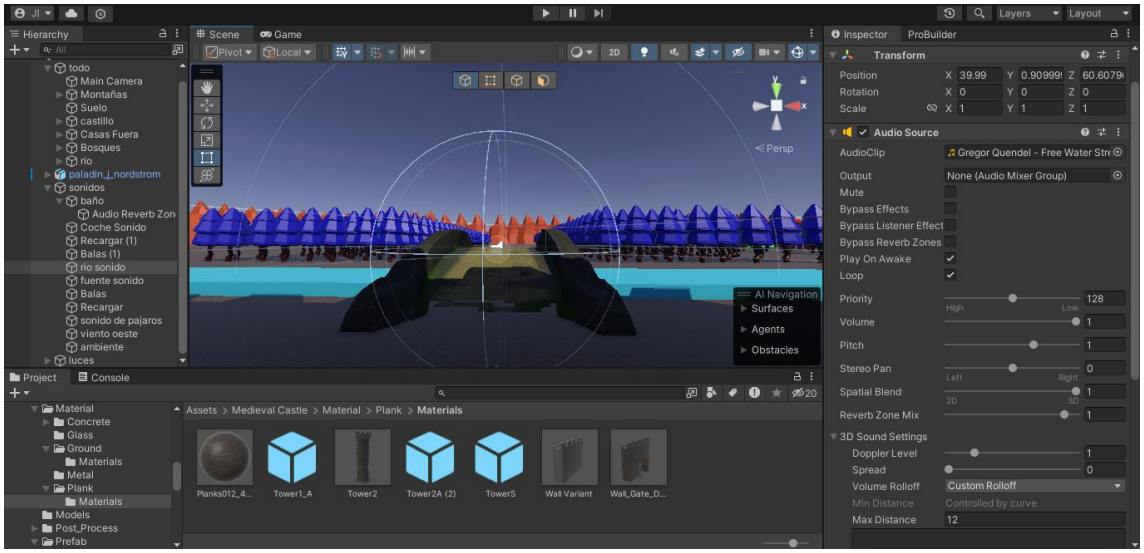
En el tema de audios añadí 8 sonidos la cuales hay uno ambiental en donde va sonando un sonido del oeste todo el rato. Además, le puse que tenga prioridad baja para que no moleste a otros audios. Y para que se escuche en toda la escena, aplique que su spatial blend en 2D,



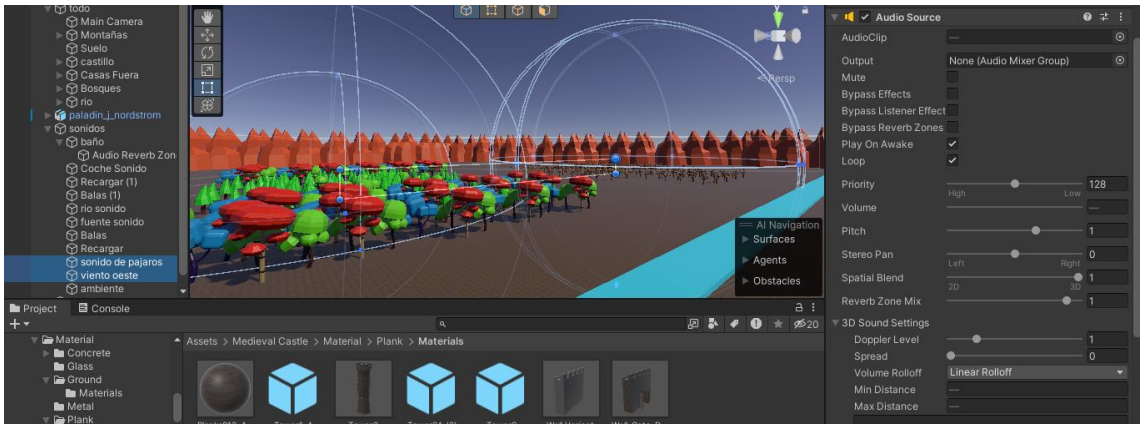
Un sonido con propiedad Logarithimic roloff significa que el sonido mientras más lejos estes más bajo se escucha. en el baño para que se escuche de mas o menos dependiendo lo cerca que estes, igual que al sonido de los disparos ya que suelen ser altos que se escuchan lejos. Tambien otro sonido con esta propiedad que añadí un sonido de baño que se escucha al estar cerca del gameObject que representé al lavabo. Y en este último audio le añadí tambien un audio reverb zone que sirve como para aplicar filtros, en este caso tenían uno de baño.



En cuanto los sonidos con propiedad de custom rolouff la cual te permite modificar a tu gusto la curva de atenuación de sonido. Se lo añadí a un coche y al puente de la zona del rio.



Y los ultimo con propiedad de linear rolouff que significa que al entrar en la zona de audio se escucha el sonido y al salir desaparece al instante. Se lo puse a las armas de recarga y las zonas de bosque para darles ambiente.

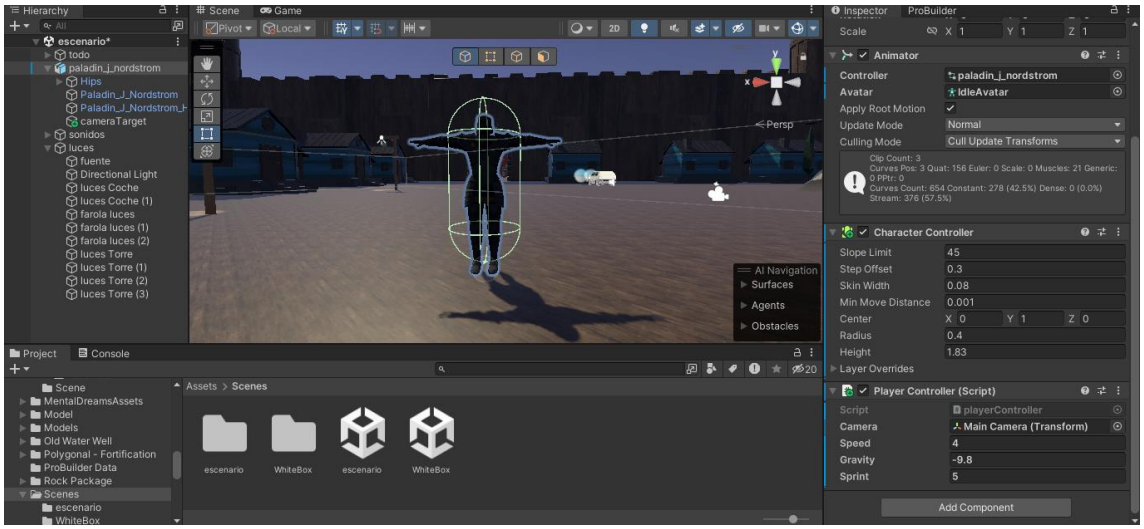


En cuanto a luces use spotlight que es como un cañón en donde dispara la luz a la dirección que el usuario pida, en el proyecto los incluido en las farolas para que siempre tiren luz apuntado hacia abajo, y en el coche para que simule como las luces delanteras de un coche.

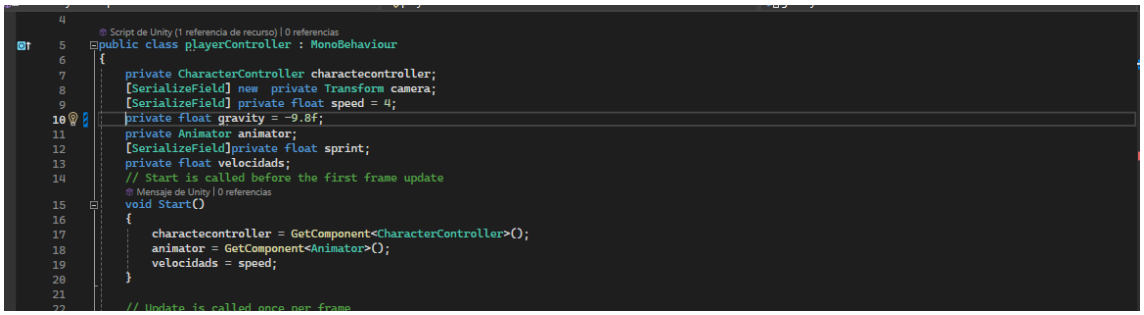
Otra componente de luz que use fue point light que es como un área de luz que ilumina un área en concreto, en el proyecto lo puse en la fuente y los cuatros torres, la razón es porque no se me ocurría donde ponerlo y en la torre le da como un toque más épico.

Control Personaje

En el tema del movimiento del personaje, usé un asset de un video de YouTube, en la cual seguí tambien los pasos y le añadí algunos extras. La cual te permite mover el personaje en tercera persona. Para ello use dos scripts distintos, uno del movimiento del monigote y otro para el posicionamiento de la cámara.



Como se puede ver en la imagen, además de los componentes que le dan el diseño al personaje, tiene un objeto vació que es el cameraTarget que esta localizado en la cabeza, su utilidad es para que en el script de la cámara orbite alrededor de ella. En la parte derecha se pueden ver los componentes, la cual están formado por un character controller que servirá para aplicar los movimientos, un animator para que dependiendo de la acción haga una animación u otra. Y su script correspondiente.



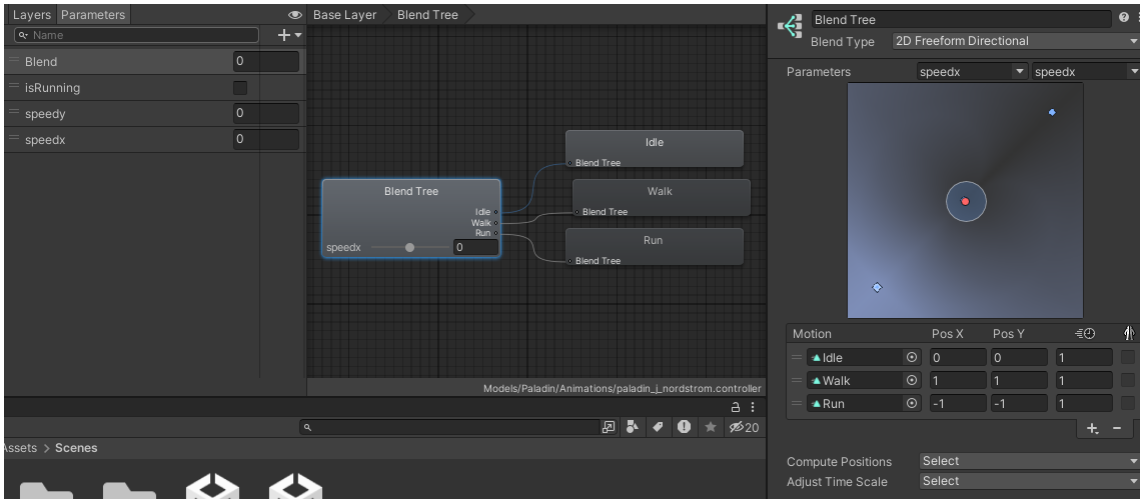
```
Script de Unity (1 referencia de recurso) | 0 referencias
public class playerController : MonoBehaviour
{
    private CharacterController charactecontroller;
    [SerializeField] new  private Transform camera;
    [SerializeField] private float speed = 4;
    private float gravity = -9.8f;
    private Animator animator;
    [SerializeField]private float sprint;
    private float velocidads;
    // Start is called before the first frame update
    Mensaje de Unity | 0 referencias
    void Start()
    {
        charactecontroller = GetComponent<CharacterController>();
        animator = GetComponent<Animator>();
        velocidads = speed;
    }

    // Update is called once per frame
```

Como se puede ver en el script primero creamos los objetos, en donde cogeremos la cámara, declararemos las velocidades, la gravedad y su sprint, además de coger los componentes para poder modificarlos luego.

```
void Update()
{
    float hor = Input.GetAxis("Horizontal");
    float ver = Input.GetAxis("Vertical");
    float movementspeed = 0f;
    float sprintd = 1;
    Vector3 movement = Vector3.zero;
    speed = velocidads;
    if (Input.GetKey(KeyCode.LeftShift) || Input.GetKey(KeyCode.RightShift))
    {
        speed += sprint;
        sprintd = -1;
    }
    if (hor != 0 || ver != 0)
    {
        Vector3 forward = camera.forward;
        forward.y = 0;
        forward.Normalize();

        Vector3 right = camera.right;
        right.y = 0;
        right.Normalize();

        Vector3 direction = forward * ver + right * hor;
        movementspeed = Mathf.Clamp01(direction.magnitude) * sprintd;
        direction.Normalize();
        movement = direction * speed * Time.deltaTime;
        transform.rotation = Quaternion.Slerp(transform.rotation,Quaternion.LookRotation(direction),1);

    }

    animator.SetFloat("speedx", movementspeed);
    movement.y += gravity*Time.deltaTime;
    charactecontroller.Move(movement);
}
```

En cuanto la parte del update, cada vez que pulsas un teclado de movimiento se guarda su valor y se harán los cálculos siguientes, primero con la camera te recoge la dirección que están recogiendo en dos objetos distintos, la cual se normalizan para que evite que los movimientos diagonales vayan más rápidos. Y tras ello se hacen sus cálculos para crear otro Vector3 que se usara para aplicar el movimiento.

Además, como se puede ver, al pulsar shift aumenta la velocidad.



Por otro lado en el script se puede ver que tambien aplico las animaciones a través de animator, usando un blend tree guardo las animaciones y con un solo valor en este caso el movementspeed modifico el valor speedx del animator para que vaya a una posición o otra, este se calcula con la magnitud de la dirección, con el Mathf.Clamp10 te limita a resultado entre 1 o 0, el sprintd sirve para saber si esprinto poniendo negativo el resultado del calculo para que en el animator se posicione en esa animación.

En cuanto la cámara,

```
private Vector2 angle= new Vector2(90 * Mathf.Deg2Rad,0);
[SerializeField]private Transform follow;
[SerializeField]private float distancia;
[SerializeField] private Vector2 sensibility;
Mensaje de Unity | 0 referencias
void Start()
{
    Cursor.lockState = CursorLockMode.Locked;
}
```

Estos son los objetos iniciales declarados, la cuales son: angle para colocar la cámara en la posición inicial, en este caso detrás del objeto follow, y la distancia es el radio que orbitara la cámara, y la sensibilada la velocidad en la que se mueva, al ser Vector2 puedes modificar la velocidad en el movimiento Y e X. En el void start, la línea de código que contiene el método sirve para esconder el ratón y hacer que se quede bloqueado hasta que pulses escape, sirve para que al mover la cámara no moleste.

```
Mensaje de Unity | 0 referencias
void Update()
{
    float hor = Input.GetAxis("Mouse X");
       if(hor != 0)
    {
        angle.x += hor * Mathf.Deg2Rad * sensibility.x;
    }
    float ver = Input.GetAxis("Mouse Y");
    if (ver != 0)
    {
        angle.y += ver * Mathf.Deg2Rad * sensibility.y;
        angle.y = Mathf.Clamp(angle.y,-80 * Mathf.Deg2Rad,80 * Mathf.Deg2Rad);
    }

}
Mensaje de Unity | 0 referencias
private void LateUpdate()
{
    Vector3 orbit = new Vector3(
        Mathf.Cos(angle.x) * Mathf.Cos(angle.y),
        -Mathf.Sin(angle.y),
        -Mathf.Sin(angle.x) * Mathf.Cos(angle.y)
        );
    transform.position = follow.position + orbit * distancia;
    transform.rotation = Quaternion.LookRotation(follow.position - transform.position);
}
```

Luego para aplicar la lógica, use los métodos Update y lateUpdate, para que primero haga los cálculos y luego aplique el movimiento. En el método update, se calcula el ángulo x y ángulo y la cual se guarda en dos float cada vez que realizas un movimiento de ratón. Se calcula tambien en radianes debido a que la mayoría de los métodos usan esas unidades. El Mathf.Clamp sirve para delimitar las zonas en donde el ratón tenga libertad para mover su eje Y, debido a que es fácil marearse si al estar todo el rato moviendo a través del eje y la cámara de vueltas sin parar, por lo que la cámara cuando este en la altura mas alta o mas baja se parara y no podrá moverse mas hacia delante por lo que tendrá que volver hacia atrás.

Luego en el método LateUpdate, se calcula la posición de la orbita del Vector3 orbit con cálculos matemáticos para que gire alrededor de un círculo, y luego lo aplica a la posición de la cámara y su rotación.

Videos de referencia que saque estos scripts.

Cámara.

Movimiento