Introducción a la programación de videojuegos

Sebastián Díaz Arancibia

Universidad Técnica Federico Santa María - 2023

Tabla de contenido

01. Manejo de audio

02. Manejo de cámara

01.

Manejo de audio

El audio es uno de los elementos inmersivos más importantes de los videojuegos. Sin una buena banda sonora o efectos de sonido, un jugador no recibe un buen feedback de sus acciones y podría terminar con una mala experiencia de juego.

BRAND NAME

Manejo de audio

Al diseñar un videojuego se debe tener muy en cuenta que este debe ir acompañado de los **sonidos y música adecuados** a lo que se quiere transmitir, con el objetivo de **aumentar la inmersión** del jugador en la experiencia.

Es esencial que un **Game Designer** sepa elegir los sonidos y la música adecuada para cada momento del juego y saber trasladarlo al compositor. Esto, aunque parezca fácil, es una de las tareas más complicadas, ya que **se debe explicar con palabras** la música que se tiene en mente.

Asimismo, la música y los sonidos son **fundamentales para comprender la narrativa**: identificar los momentos de miedo, de tristeza, de alegría, si un personaje es bueno o malo, etc.

Manejo de audio

Títulos como **Rayman Legends** utilizan la música y los efectos de sonido para entregar un **feedback de acciones** del personaje que llevan la experiencia del jugador a otro nivel.

Uno de los aspectos más sobresalientes de las últimas entregas de Rayman (Origins y Legends) es su música, que combinada con los coloridos gráficos del UbiArt Framework resultan en juegos de plataformas únicos.

Es tan importante la música para Rayman Legends que **la música está planeada** incluso para ir en **perfecta coordinación** con el escenario.

Manejo de audio

Algunos videojuegos de ritmo se destacaron por el uso de controles alternativos y específicos. Por ejemplo, la emblemática guitarra de **Guitar Hero**, la batería de Rock Band o los bongos de Donkey Konga.

En el caso de Guitar Hero, la franquicia comenzó con Guitar Hero para Playstation 2.

Al día de hoy, la serie ha vendido alrededor de 15 millones de juegos y ganando aproximadamente US$1000 millones de dólares.

Manejo de audio

El audio es uno de los elementos más importantes para conseguir una correcta ambientación y experiencia cuando jugamos a un juego. El uso de audio incluye: música ambiental, efectos de sonidos (explosiones, disparos, aceleraciones, etc.)

Para manejar los audios del videojuego, Unity maneja tres conceptos:

- **Audio Clip**
- **Audio Source**
- **Audio Listener**

Puede descargar audios de los siguientes sitios:

https://www.zapsplat.com/

https://freesound.org/

http://soundbible.com/

** Siempre revise las licencias de uso de los Assets descargadas desde Internet para evitar problemas legales.

Manejo de audio

AudioClip

Un AudioClip en Unity es una clase que representa un clip de audio o sonido que se puede reproducir en el motor de Unity.

Proporciona una forma de **almacenar y acceder a datos de audio**, como música, efectos de sonido o voces.

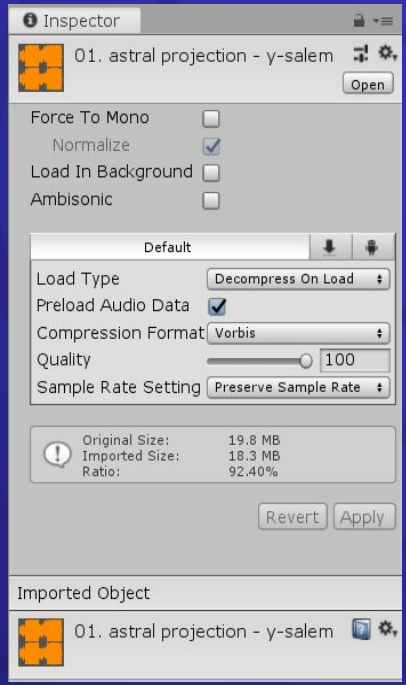
Son los archivos de audio que guardan el sonido que queremos reproducir. Son los sonidos que queremos reproducir.

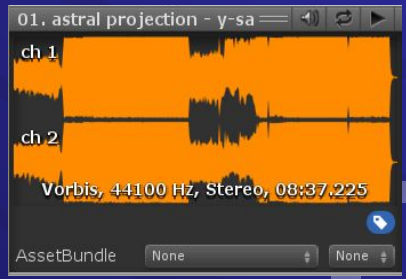
Para importar un audio clip, basta con **arrastrar los archivos de audio en la ventana de proyectos** del editor de Unity.

Unity los almacena en forma de Assets dentro de la ventana de proyecto y soporta diversos formatos: **mp3**, **ogg**, **wav**, aiff, aif, mod, s3m, xm.

Manejo de audio

- **Audio data**: Almacena los datos de audio reales que componen el clip.
- **Duración**: Indica la duración del AudioClip en segundos. Es útil para sincronizar eventos y manipular el tiempo de reproducción.
- **Canales**: Especifica si el audio es mono o estéreo.
- **Frecuencia de muestreo**: Representa el número de muestras tomadas por segundo para representar el audio. Es medida en hertz (Hz) y afecta la calidad y el tamaño del archivo de audio.
- **Compresión**: Los AudioClips pueden ser comprimidos para reducir su tamaño en disco, es útil para ahorrar espacio de almacenamiento y mejorar el rendimiento.





Manejo de audio

AudioSource

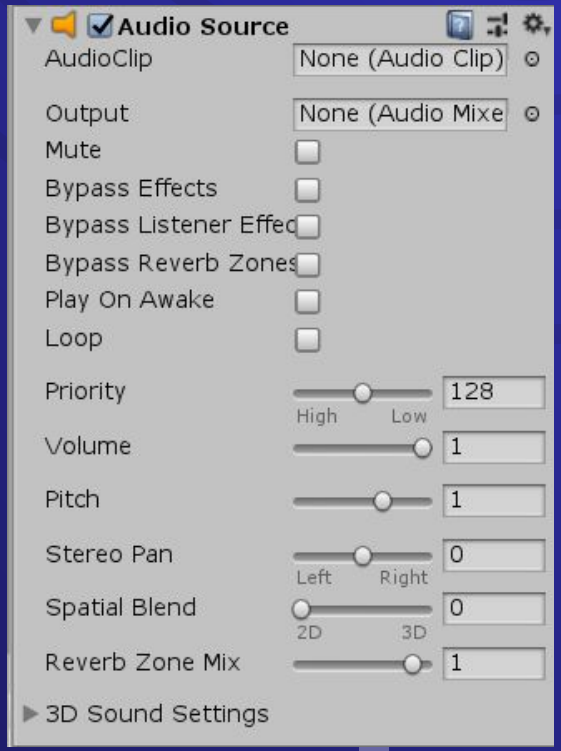
Un Audio Source es un **componente** que se asocia a un GameObject en la escena y permite reproducir un AudioClip.

- Representa una **fuente de audio que emite sonidos** desde un AudioClip hacia el entorno del juego.

El Audio Source está encargado de:

- Volumen
- Tono
- Reproducir el AudioClip al cargar el objeto en escena.
- Realizar un loop del Audio Clip
- Configuración 3D

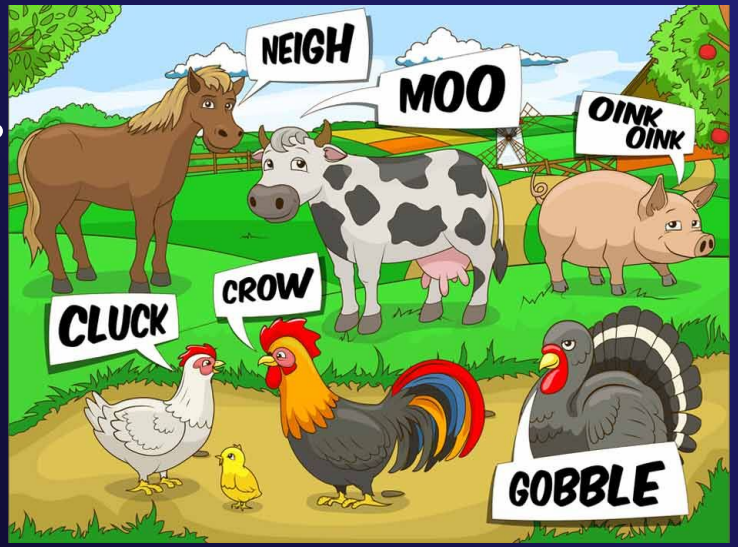
**Tener en cuenta que si la variable public AudioClip está vacía, deberá agregar el archivo de audio por código en un Script.



Manejo de audio

- **AudioClip**: Especifica el AudioClip que se reproducirá desde el AudioSource.
- **Volumen**: Controla el nivel de volumen del sonido reproducido.
- **Bucle**: Permite configurar si el sonido se repetirá en bucle o si se reproducirá una sola vez.
- **Efectos de sonido**: Aplicación de efectos de sonido, como reverberación, atenuación y filtros, para modificar la calidad y la percepción del sonido reproducido.
- **Pitch**: Controla la frecuencia de reproducción del sonido, lo que afecta su tono, creando efectos de sonido únicos o simular variaciones en la velocidad.
- **Espacialización**: Permite simular la posición y dirección del sonido en el espacio 3D para que el sonido se ajuste según la ubicación del AudioSource en relación con el oyente del juego.

Manejo de audio



Un **AudioSource** permite hacer lo siguiente:

- **Play**: Reproducir el AudioClip que tiene asignado.
- **Pause**: Pausar el AudioClip que tiene asignado.
- **Stop**: Detener el AudioClip que tiene asignado.
- **Clip**: Modificar el AudioClip asignado.
- **Volume**: Cambiar el volumen de la música o sonido entre 0.0 y 1.0.
- Etc.

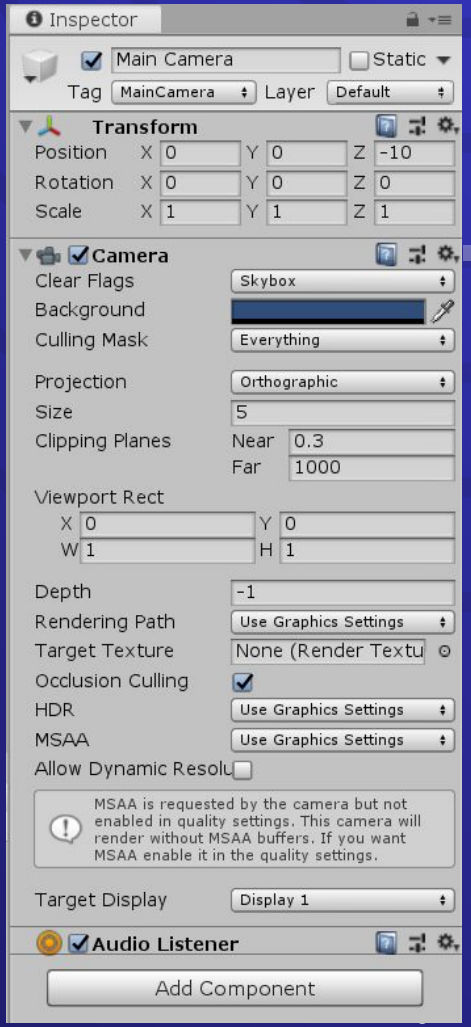
Manejo de audio

AudioListener

Componente necesario para que **se pueda escucha**r cualquier AudioSource. Un Audio Listener representa el punto desde el cual el jugador va a escuchar el sonido.

Este componente está añadido por defecto al GameObject Main Camera y sólo puede haber un Audio Listener activo en la escena.

La **posición** del Audio Listener va a influir en el sonido que escucha el jugador debido a la **distancia entre el jugador y el Audio Source** que emite el sonido.



Manejo de audio

Implementación de audio en un videojuego

En primer lugar, se debe asegurar que cuente con los 3 conceptos claves de Audio en Unity:

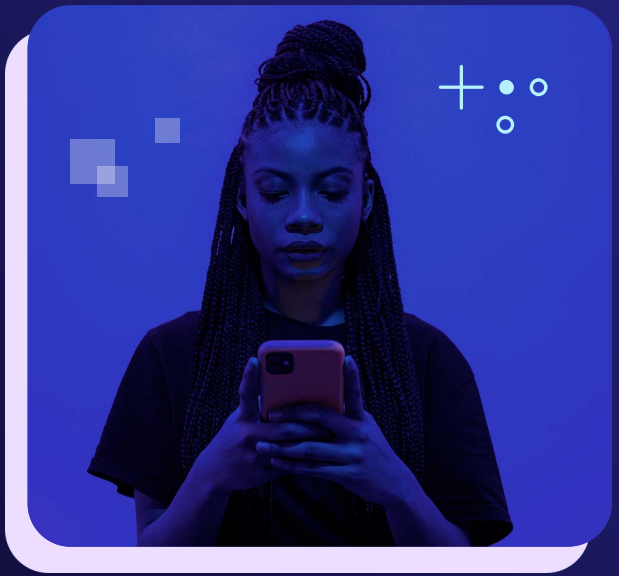
- Debe **cargar** en su proyecto todos aquellos **archivos de audio** que desee utilizar para convertirlos en AudioClip.
- **Crear** el objeto que será la **fuente emisora** o AudioSource del AudioClip deseado. No olvide arrastrar el AudioClip en el componente AudioSource.
- La cámara principal de su proyecto debe tener añadido el **componente AudioListener** para escuchar los sonidos del juego.

Un vez se tengan los **3 conceptos de Audio** en Unity aplicados en el proyecto, se puede manejar por Script de la siguiente forma:

```
AudioSource auds;
auds = GetComponent<AudioSource>();
auds.Play();
```

Puedes reemplazar la línea del GetComponent si es que declaras la variable auds como pública y le pasas a través del inspector el **AudioSource** correspondiente.

```
public AudioSource auds;
```



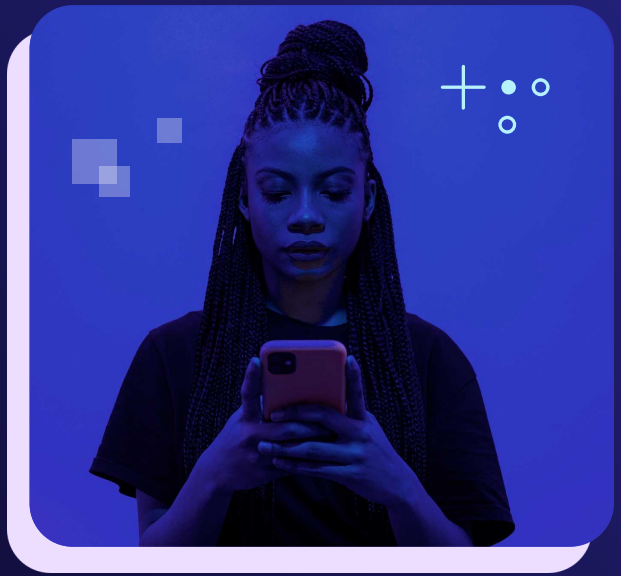
Ejercicio 1

Cree un Script que permita que al presionar las tecla 1, reproduzca un sonido. Repita el proceso para las teclas 2 y 3 con diferentes sonidos. Dar play y presionar las teclas para oír el resultado.

Ejercicio 1

```
public class Sounds : MonoBehaviour
{
    public AudioSource auds;
    public AudioClip clip1;
    public AudioClip clip2;
    public AudioClip clip3;

    Mensaje de Unity | 0 referencias
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Alpha1))
        {
            auds.Stop();
            auds.clip = clip1;
            auds.Play();
        }
        if (Input.GetKeyDown(KeyCode.Alpha2))
        {
            auds.Stop();
            auds.clip = clip2;
            auds.Play();
        }
        if (Input.GetKeyDown(KeyCode.Alpha3))
        {
            auds.Stop();
            auds.clip = clip3;
            auds.Play();
        }
    }
}
```



Ejercicio 2

Crear un objeto que al colisionar con otro que tenga el tag “Enemy”, emitirá un sonido. Este sonido debe ser seleccionado de forma aleatoria de un arreglo de 3 sonidos. Para resolver este ejercicio debe crear la clase Player y la clase Enemy.

Ejercicio 2

```
public class Player : MonoBehaviour
{
    public float speed = 5f;

    Mensaje de Unity | 0 referencias
    void Update()
    {
        float x = Input.GetAxisRaw("Horizontal");
        float y = Input.GetAxisRaw("Vertical");

        transform.position += new Vector3(x, y) * speed * Time.deltaTime;
    }

    Mensaje de Unity | 0 referencias
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.CompareTag("Enemy"))
        {
            collision.gameObject.GetComponent<Enemy>().PlayClip();
        }
    }
}
```

Ejercicio 2

```
public class Enemy : MonoBehaviour
{
    public AudioClip[] clips;
    public int clipIndex;
    public AudioSource audSource;

    Mensaje de Unity | 0 referencias
    void Start()
    {
        audSource = GetComponent<AudioSource>();
        clipIndex = Random.Range(0, clips.Length);
    }

    1 referencia
    public void PlayClip()
    {
        audSource.Stop();
        clipIndex = Random.Range(0, clips.Length);
        audSource.clip = clips[clipIndex];
        audSource.Play();
    }
}
```

02.

Manejo de cámara

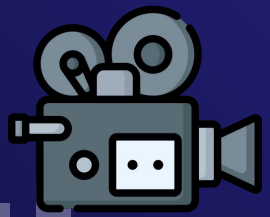
La cámara es comparable a los ojos del jugador, ya que muestra los objetos que tiene a su alrededor pero en un ambiente virtual.

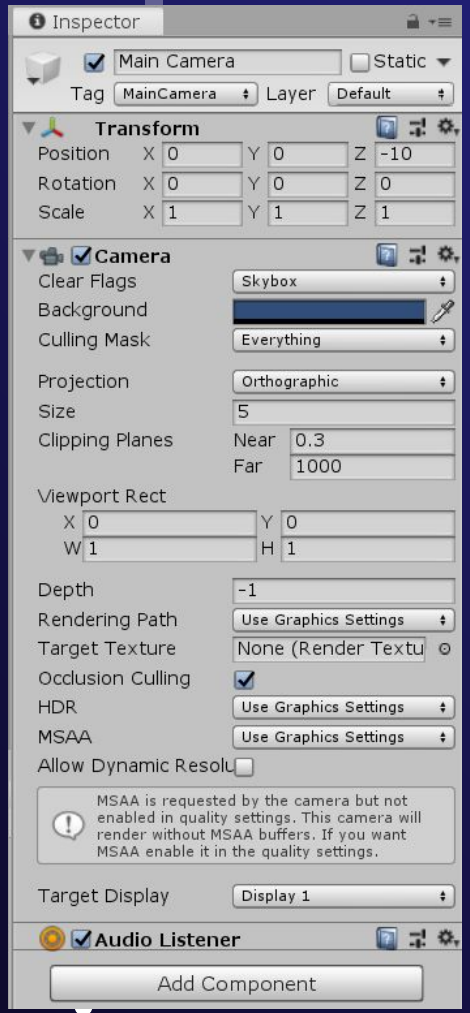
Manejo de cámara

Camera es un componente fundamental utilizado para representar una cámara en el mundo del juego.

- Es responsable de capturar y **mostrar la vista desde una perspectiva específica** en el escenario.

Puede representar una cámara en primera persona, en tercera persona o cualquier otra perspectiva deseada. La cámara proyecta una imagen del mundo virtual en el área de visualización.





Manejo de cámara

La **posición** de la Camera **define el punto de vista**, mientras que sus **ejes Z e Y definen la dirección de la vista** y la parte superior de la pantalla respectivamente.

El componente Camera también define el tamaño y la forma de la región que recae dentro de la vista.

Con estos parámetros establecidos, **la cámara puede mostrar lo que "ve"** actualmente en la pantalla. Si el objeto de cámara se mueve y rota, la vista mostrada también se moverá y rotará en consecuencia.

Manejo de cámara

Al crear una nueva escena en Unity, esta contiene una **cámara por defecto llamada Main Camera**, sin embargo, puede tener más de una cámara en escena y sus puntos de vista pueden combinarse de diferentes formas.

Por defecto, una cámara renderiza su punto de vista para cubrir toda la pantalla y por lo tanto solo puede verse el punto de vista de una cámara a la vez (la cámara visible es la que tiene el mayor valor de propiedad depth).

Al deshabilitar una cámara y habilitar otra desde un Script, se puede cambiar de un punto de vista de una escena a otro completamente distinto. Esto puede ser útil para cambiar entre una vista del mapa desde arriba y una vista en primera persona.

```
public class CambioCamara : MonoBehaviour
{
    public Camera cam1;

    // Start is called before the first frame update
    void Start()
    {
        cam1.enabled = true;
    }
}
```

Manejo de cámara

Para activar o desactivar una cámara, tenemos dos formas de hacerlo:

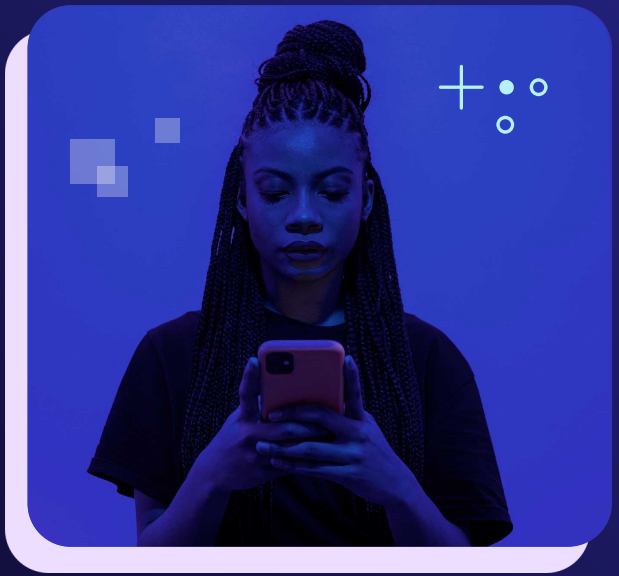
- Desactivando el gameobject
- Desactivando el componente

Para desactivar un **GameObject**, se utiliza la **función SetActive**, la cual recibe como parámetro un booleano que indica si hay que activar o desactivar el objeto.

```
GameObject go;
go.SetActive(true);
go.SetActive(false);
```

Para desactivar un **Component**, se utiliza la **propiedad enabled** y se le asigna un valor true para activarlo o false para desactivarlo.

```
Collider2D col;
col.enabled = true;
col.enabled = false;
```



Ejercicio 3

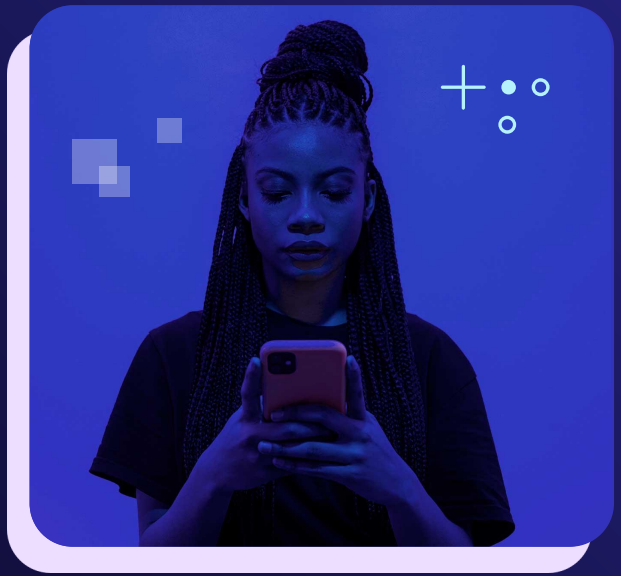
Agregue una nueva cámara a escena y cambie su posición en el eje Z para que se vea desde más arriba. Cree un nuevo Script llamado CambioCamara que controle que al presionar la tecla 1, active la cámara 1 y desactive la cámara 2. Mismo principio para la cámara 2. Arrastre el Script a un GameObject vacío. Dar play y observe el resultado.

Ejercicio 3

```
public class ChangeCamera : MonoBehaviour
{
    public Camera cam1;
    public Camera cam2;

    Unity Message | 0 references
    private void Start()
    {
        cam1.gameObject.SetActive(true);
        cam2.gameObject.SetActive(false);
    }

    Unity Message | 0 references
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Alpha1))
        {
            cam1.gameObject.SetActive(true);
            cam2.gameObject.SetActive(false);
        }
        else if (Input.GetKeyDown(KeyCode.Alpha2))
        {
            cam1.gameObject.SetActive(false);
            cam2.gameObject.SetActive(true);
        }
    }
}
```



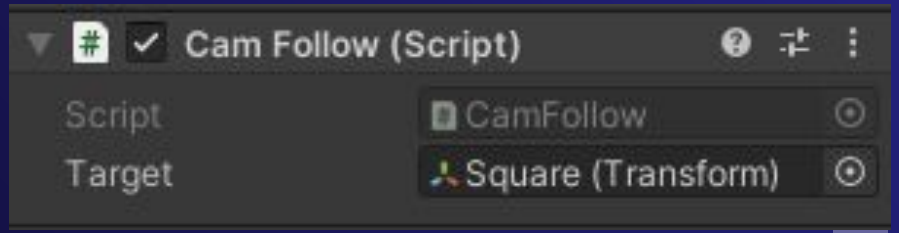
Ejercicio 4

Crear un Script para que una cámara siga a un personaje en la escena.

Ejercicio 4

```
public class CamFollow : MonoBehaviour
{
    public Transform target;

    Mensaje de Unity | 0 referencias
    private void LateUpdate()
    {
        Vector3 desiredPosition = new Vector3(target.position.x, target.position.y, transform.position.z);
        transform.position = desiredPosition;
    }
}
```



Introducción a la programación de videojuegos

Sebastián Díaz Arancibia

Universidad Técnica Federico Santa María - 2023