



IES AUGUSTO GONZALES DE LINARES
CFGS DAM2º

INVESTIGACIÓN SOBRE MOTORES DE VIDEOJUEGOS

PROGRAMACIÓN MULTIMEDIA Y
DISPOSITIVOS MÓVILES

Gutiérrez Valverde, Ramiro

Espinosa García, Daniel

Díez de Paulino, Albano

Barrios Fernández, Carmen



2023/2024

Índice

RA-4 Motores de Videojuegos.....	2
CE-b) Se han analizado los componentes de un motor de juegos.	2
CE-c) Se han analizado entornos de desarrollo de juegos.	3
CE-d) Se han analizado diferentes motores de juegos, sus características y funcionalidades.	4
CE-e) Se han identificado los bloques funcionales de un juego existente.	6
CE-f) Se han definido y ejecutado procesos de render.	8
CE-g) Se ha reconocido la representación lógica y espacial de una escena gráfica sobre un juego existente.	10
RA-5 Desarrollo de Videojuegos.....	11
CE-b) Se han creado objetos y definido los fondos.....	11
CE-c) Se han instalado y utilizado extensiones para el manejo de escenas.....	12
CE-e) Se ha incorporado sonido a los diferentes eventos del juego.....	13
Tabla de Ilustraciones.....	14

RA-4 Motores de Videojuegos

CE-b) Se han analizado los componentes de un motor de juegos.

Actualmente un motor de videojuegos esta compuesto de cinco grandes áreas para implementar gran variedad de funciones.

- **Bloque Principal:**

Es el componente básico de un motor de videojuegos, se encarga de gestionar todas las clases/objetos que tendrá el programador para implementar la lógica del videojuego(GDK), además permite al programador compilar y ejecutar sin salir del entorno.

- **Renderización:**

Junto al bloque principal, el motor de renderizado es fundamental para un motor de videojuegos, es el encargado de generar gráficos 2D/3D con el fin de mostrarlos en pantalla, para ello utilizan distintas APIs de comunicación con el hardware gráfico, como por ejemplo DirectX, OpenGL o Vulkan.

Actualmente implementan mejoras graficas con IA como por ejemplo el trazado de rayados(Ray Tracing), el reescalado dinámico de resolución o generación de fotogramas(DLSS), y reducción de picos en bordes (DLAA).

- **Audio:**

Es la parte encargada de la lectura y reproducción de música y efectos de sonido en el juego, los formatos más populares de audio soportado son WAV y OGG, aunque son soportados casi todos.

Permite a los desarrolladores manipular las diferentes magnitudes del audio como, por ejemplo, la amplitud o la frecuencia.

Para ello usa APIs al igual el renderizado, las mas conocidas son, OpenAL y DirectSound.

- **Inteligencia Artificial:**

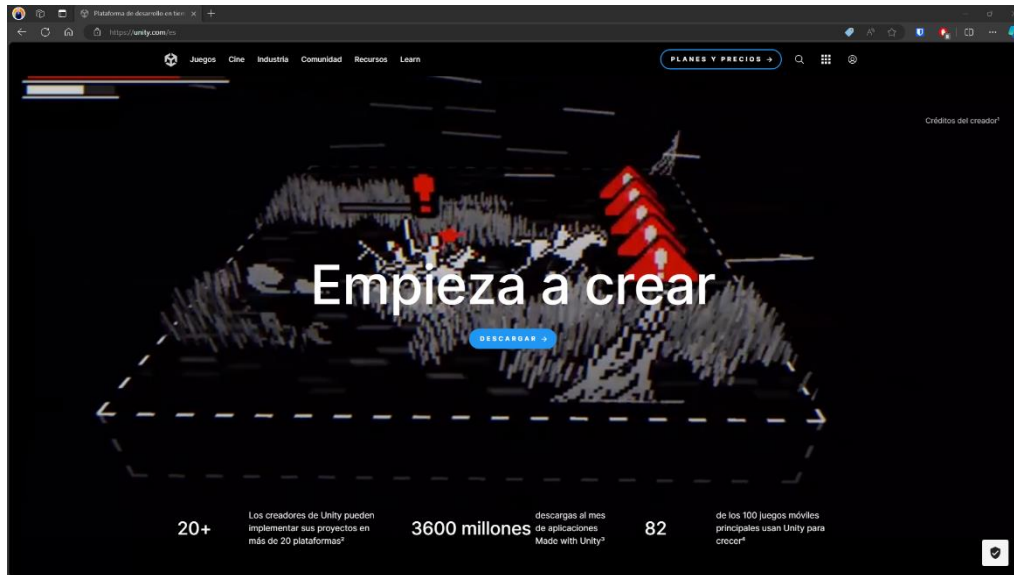
Es el modulo encargado de gestionar todos los movimientos de los personajes controlados por el ordenador, el progreso de la historia, las posibles interacciones del usuario con el entorno, etc.

- **Motor de físicas:**

Es el modulo encargado de la emulación de las leyes físicas de forma realista, concretamente simula variables como la gravedad, masas, fricción, fuerza y colisiones.

CE-c) Se han analizado entornos de desarrollo de juegos.

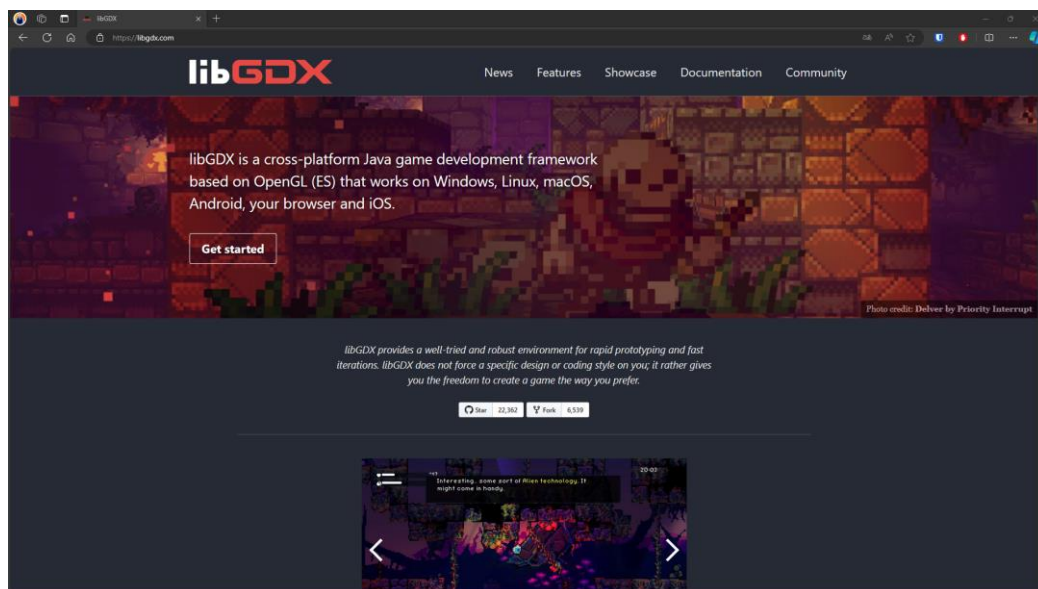
Desde la salida del motor de videojuegos Unity, el desarrollo de videojuegos ha crecido exponencialmente, debido a que Unity permite desarrollar un juego de forma “gratuita” y de forma sencilla.



Web 1- Pagina Web de Unity

Pero además se puede desarrollar en otros múltiples motores, tanto gratuitos como de pago, que analizamos en el apartado CE-D.

Además, existe la posibilidad de desarrollar con librerías nativas(GDK) de los diferentes lenguajes de programación, por ejemplo, LibGDX en java o pygame en Python.

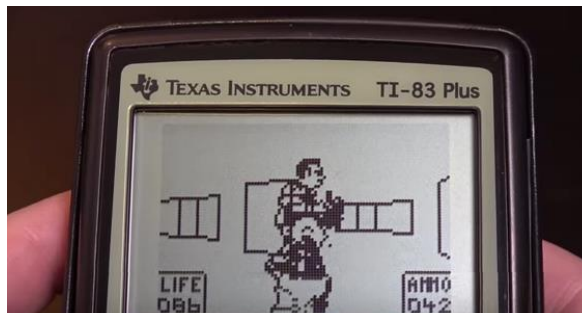


Web 2 – Pagina Web LibGDX

CE-d) Se han analizado diferentes motores de juegos, sus características y funcionalidades.

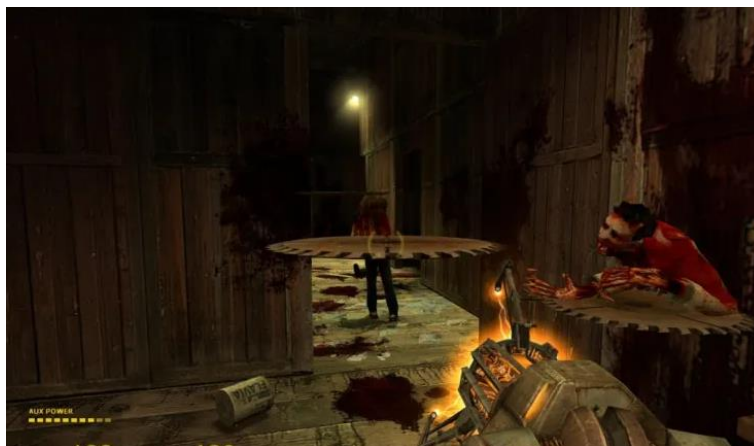
A lo largo de la historia de la informática han existido múltiples motores de videojuegos, pero algunos a destacado por encima del resto por diferentes motivos, a continuación, se muestra una lista cronológica de dichos motores, que aportaron al desarrollo de videojuegos y que juegos populares fueron desarrolladores con dicho motor.

- Id Tech 1
 - Primer motor de videojuegos que permita el desarrollo 3D, con el tiempo fue liberado, lo que permitió que los juegos desarrollados por este motor puedan ser ejecutados en una cantidad de dispositivos infinita gracias al soporte de la comunidad.
 - Juegos: Doom, Quake.



Fotografía 1 – Doom ejecutándose en una calculadora

- GoldSource:
 - Motor desarrollado por Valve a partir de la mejora del id tech 2, añadió múltiples funcionalidades, pero la principal fue la gestión de la historia y los NPCs
 - Juegos: Half Life 1, Counter Strike, Team Fortress
- Source
 - Evolución del motor GoldSource, aparte de mejoras técnicas y visuales, el principal añadido fue un motor de físicas realistas.
 - Juegos: Half Life 2, Portal 1 y 2, Counter Strike Global Offensive, Titan Fall, Apex Legends



Fotografía 2 – Half Life 2 | Arma controladora de gravedad

- RAGE
 - Motor desarrollado por Rockstar Games. Permite el desarrollo de grandes mundos abiertos, con una cantidad muy alta de NPCs en Hardware no muy potente, a lo largo de los años han ido implementado mejoras visuales.
 - Juegos: Saga GTA, Red Dead Redemption, Bully.



Fotografía 3 – Red Dead Redemption 2

- Unity
 - Motor de videojuegos que permitió a cualquier persona desarrollar un videojuego, lo que permitió un aumento del desarrollo indie.
 - Juegos: Cities Skylines, Pokemon Go, HearthStone, Crossy Road, Hollow Knight.
- Unreal Engine 3|5
 - Implemento mejoras visuales nunca vistas, llegando al punto del foto realismo, actualmente es usado a parte del desarrollo de videojuegos para producciones cinematográficas.
 - Juegos: Fortnite, Unreal Tournament, Tekken 7, Street Fighter 5



Fotografía 4 – Set de grabación serie The Mandalorian

CE-e) Se han identificado los bloques funcionales de un juego existente.

Siguiendo los puntos del apartado CE-B y el videojuego Half-Life 2, vamos a identificar los diferentes bloques funcionales.

- **Interfaz de Usuario (UI):**

Es el componente que da información importante al usuario para la jugabilidad. En Half Life 2 disponemos de punto de mira para ver con facilidad donde se está apuntando, y múltiples marcadores para mostrar vida y munición.



Fotografía 5 – Half Life 2 UI

- **Cámara:**

En un videojuego es muy importante determinar la posición de la cámara, existe la cámara en primera persona que nos permite una mayor inmersión y poder empatizar mejor con el personaje que controlamos, o la cámara de tercera persona que nos permite ver mejor todos los movimientos de nuestro personaje, por ejemplo, como se agacha o salta. En el caso de Half Life 2 es una cámara de primera persona en todo momento.

- **Audio:**

Al audio bien implementado en un videojuego permite la inmersión completa en el mundo que quiere contar, en el caso de Half Life 2 hay sonidos puntuales de todas las armas y música ambiental en zonas importantes o cinemáticas.

- **Cinemáticas:**

Secciones del videojuego que progresan en la historia en las cuales el usuario pierde todo el control de su personaje, aunque algunos juegos en estas partes permiten cierto control del personaje.

Half Life 2 dispone de cinemáticas, tanto sin controles como con controles. Por norma todas permiten el movimiento menos las más importantes.



Fotografía 6 – Cinemática Half Life 2

CE-f) Se han definido y ejecutado procesos de render.

El término "render" proviene del campo de la gráfica por computadora y se aplica a la creación de imágenes a partir de datos tridimensionales.

En un videojuego, el proceso de render involucra varios aspectos:

- **Escena 3D:**

La representación del entorno del juego se modela en un espacio tridimensional. Esto incluye personajes, objetos, terrenos y otros elementos del juego.

- **Cámara:**

Se define la posición y rotación de la cámara que simula la perspectiva del jugador en el mundo del juego.

- **Iluminación:**

Se aplican fuentes de luz al entorno para simular sombras, reflejos y otros efectos visuales realistas. La iluminación puede ser estática o dinámica, dependiendo de cómo cambien las condiciones de luz en el juego. Por ejemplo, la rotación solar sobre el suelo.

- **Texturización:**

Las superficies de los objetos se aplican con texturas para darles detalles visuales. Las texturas pueden incluir colores, patrones y mapas de relieve para simular la apariencia de materiales reales.

- **Renderización en Tiempo Real:**

Utilizando el motor gráfico del juego, el hardware de la computadora o la consola realiza cálculos para convertir la escena 3D en una imagen 2D que se mostrará en la pantalla del jugador.

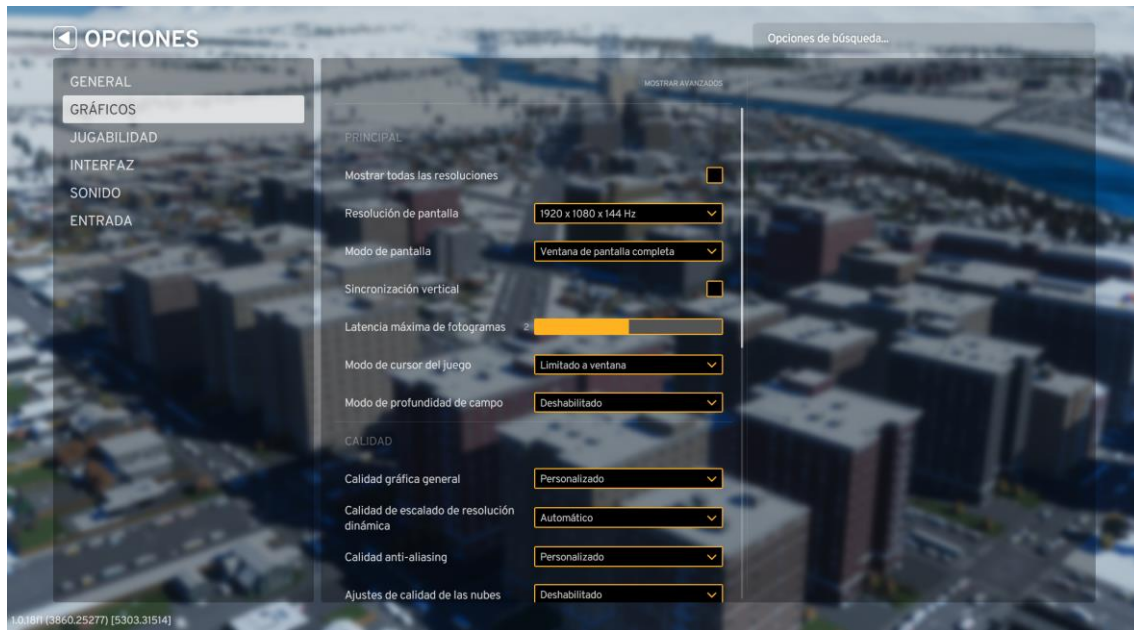
- **Efectos Visuales:**

Se aplican efectos visuales como partículas, post-procesamiento y filtros para mejorar la calidad y la apariencia estética de la escena.

▪ **Optimización:**

Se implementan técnicas para optimizar el rendimiento, como el uso eficiente de recursos y la adaptación dinámica de la calidad gráfica según la capacidad del hardware.

Actualmente los videojuegos permiten modificar múltiples parámetros de la renderización en el apartado de ajustes.



Fotografía 7 – Ajustes Gráficos Cities Skylines 2

CE-g) Se ha reconocido la representación lógica y espacial de una escena gráfica sobre un juego existente.

Los videojuegos pretenden contar una historia, para ello hay que desarrollar un escenario, este puede tener dos dimensiones (2D), o tres dimensiones (3D). Además, hay muchos juegos que mezclan los dos técnicas



Fotografía 8 – Undertale (2D)



Fotografía 10- League of Legends (2,5D)



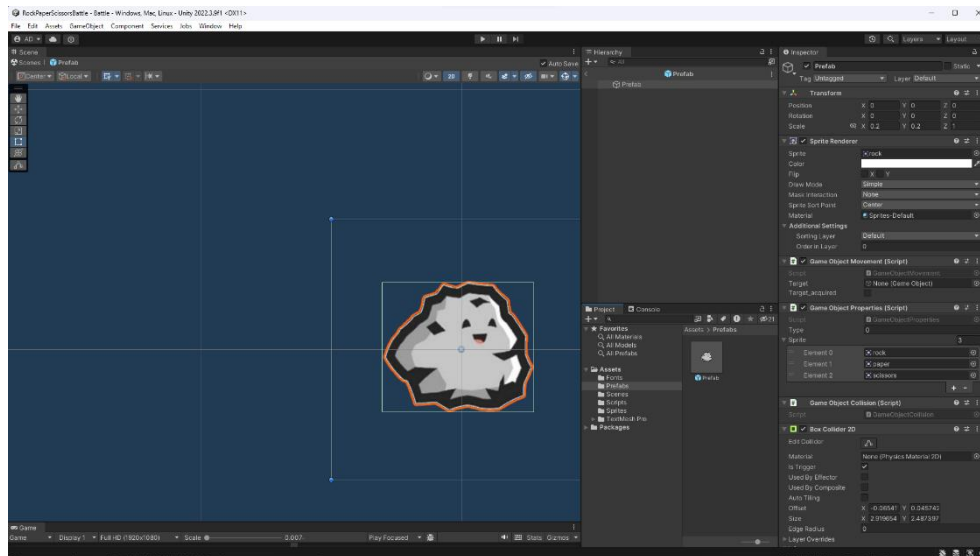
Fotografía 9 – Minecraft (3D)

Estos escenarios son coherentes con la historia que quieren contar, por ejemplo, si se pretende contar una historia ambientada en un futuro distópico, el escenario no va a ser un castillo medieval o un foro romano.

RA-5 Desarrollo de Videojuegos

CE-b) Se han creado objetos y definido los fondos.

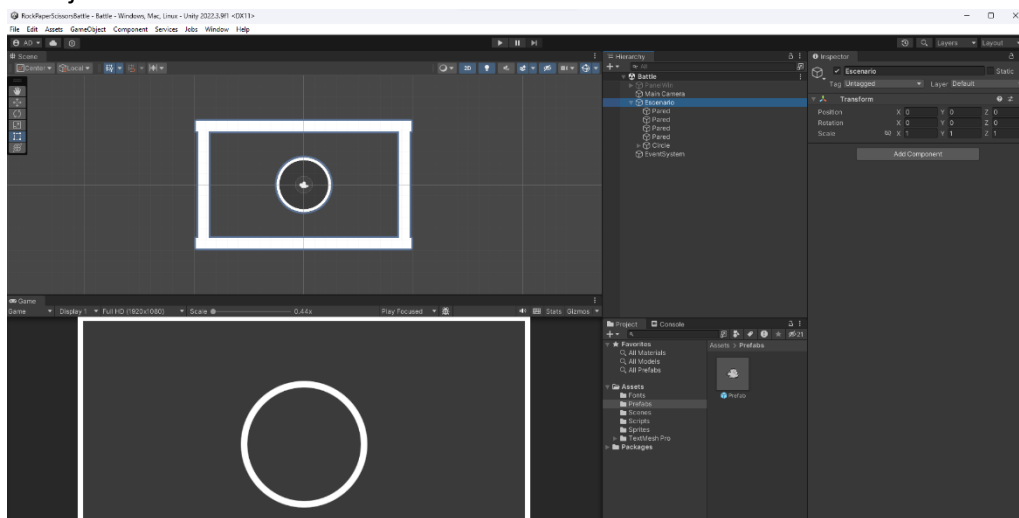
Para nuestro proyecto de uso de inteligencia artificial hemos usado un prefab (Pre-ajuste de un gameObject), el cual será introducido en la escena por un algoritmo.



Fotografía 11 – Prefab Unity

Un gameObject en Unity es un conjunto de funciones para definir un objeto en el escenario, el componente básico es el Sprite Render para que la tarjeta gráfica renderice el objeto, además se pueden introducir una gran variedad de componentes, por ejemplo, scripts, colliders, rigidbody, etc.

Además, hemos definido un fondo para representar un campo de batalla, con varios gameObjects.



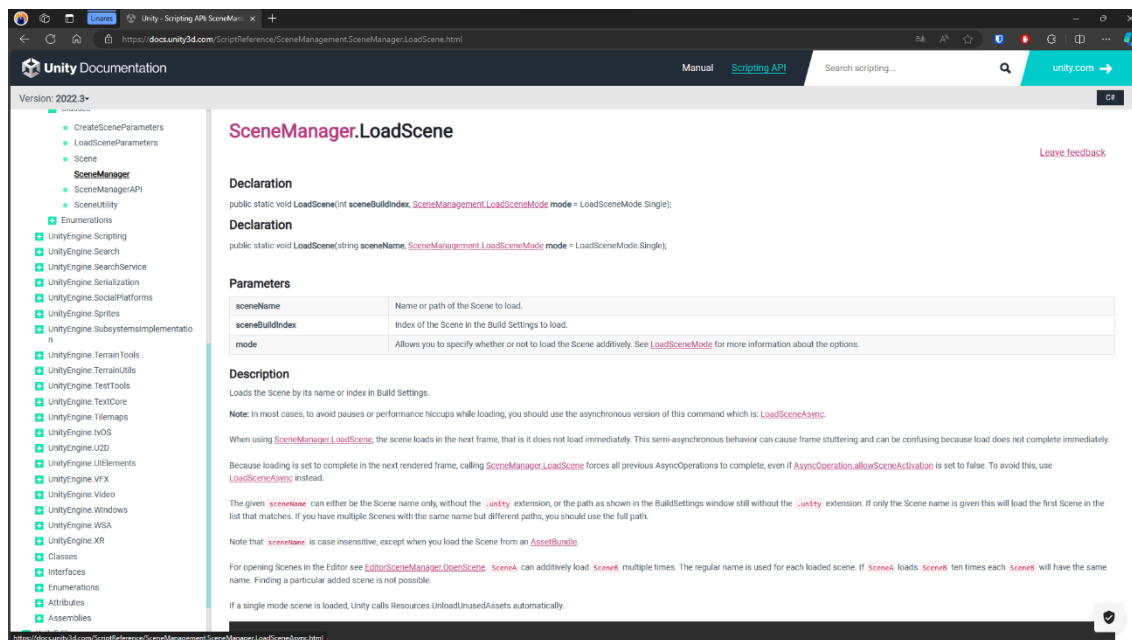
Fotografía 12 – Fondo en una escena en Unity

Gutiérrez Valverde, Ramiro
Espinosa García, Daniel
Diez de Paulino, Albano
Barrios Fernández, Carmen

CE-c) Se han instalado y utilizado extensiones para el manejo de escenas.

Actualmente Unity disponible de una clase para el manejo de las escenas, esta se llama SceneManager.

Dentro de esta clase hay muchos métodos para múltiples funciones, pero el más importante es LoadScene("NombreEscenaACargar") que nos permite descargar la escena actual y cargar la que nos interese, para ello primero debes de tener un archivo de tipo escena guardado en el proyecto y cargarlo en el build settings.



The screenshot shows the Unity Documentation page for the `SceneManager.LoadScene` method. The page includes a navigation sidebar on the left with categories like 'CreateSceneParameters', 'LoadSceneParameters', 'Scene', 'SceneManager', 'SceneManagerAPI', and 'SceneUtility'. The main content area is titled 'SceneManager.LoadScene' and contains the following sections:

- Declaration:**

```
public static void LoadScene(int sceneBuildIndex, SceneManager.LoadSceneMode mode = LoadSceneMode.Single);
```
- Declaration:**

```
public static void LoadScene(string sceneName, SceneManager.LoadSceneMode mode = LoadSceneMode.Single);
```
- Parameters:**

Parameter	Description
<code>sceneName</code>	Name or path of the Scene to load.
<code>sceneBuildIndex</code>	Index of the Scene in the Build Settings to load.
<code>mode</code>	Allows you to specify whether or not to load the Scene additively. See LoadSceneMode for more information about the options.
- Description:**

Loads the Scene by its name or index in Build Settings.

Note: In most cases, to avoid pauses or performance hiccups while loading, you should use the asynchronous version of this command which is [LoadSceneAsync](#).

When using [SceneManager.LoadScene](#), the scene loads in the next frame, that is it does not load immediately. This semi-asynchronous behavior can cause frame stuttering and can be confusing because load does not complete immediately. Because loading is set to complete in the next rendered frame, calling [SceneManager.LoadScene](#) forces all previous AsyncOperations to complete, even if [AsyncOperation.allowSceneActivation](#) is set to false. To avoid this, use [LoadSceneAsync](#) instead.

The given `sceneName` can either be the Scene name only, without the `.unity` extension, or the path as shown in the BuildSettings window still without the `.unity` extension. If only the Scene name is given this will load the first Scene in the list that matches. If you have multiple Scenes with the same name but different paths, you should use the full path.

Note that `sceneName` is case insensitive, except when you load the Scene from an [AssetBundle](#).

For opening Scenes in the Editor see [EditorSceneManager.OpenScene](#). `SceneName` can additively load `SceneName` multiple times. The regular name is used for each loaded scene. If `SceneName` loads `SceneName` ten times each `SceneName` will have the same name. Finding a particular added scene is not possible.

If a single mode scene is loaded, Unity calls [Resources.UnloadUnusedAssets](#) automatically.

Web 3 – Unity Documentation – SceneManager.LoadScene

CE-e) Se ha incorporado sonido a los diferentes eventos del juego.

Unity implementa un sistema de Cliente-Servidor para el sistema de audio, es decir a una o múltiples fuentes de audio que emiten el sonido, y un receptor que recibe ese audio, este se suele implementar en el personaje para tener un sonido más inmersivo.

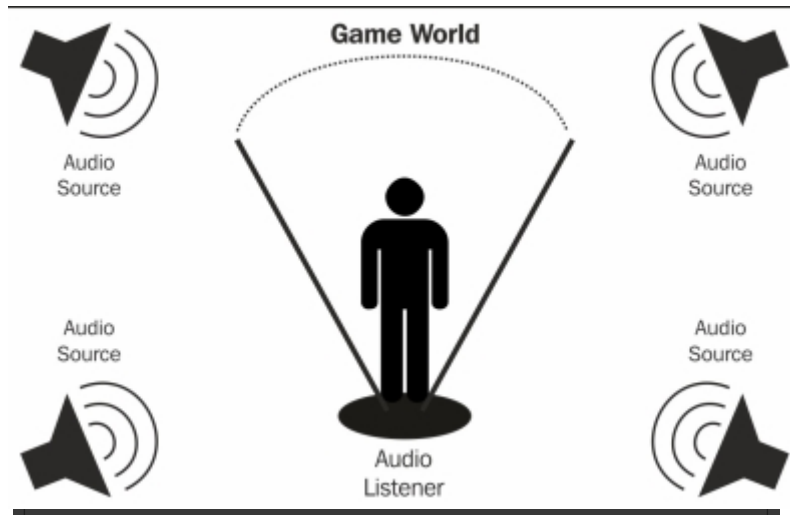
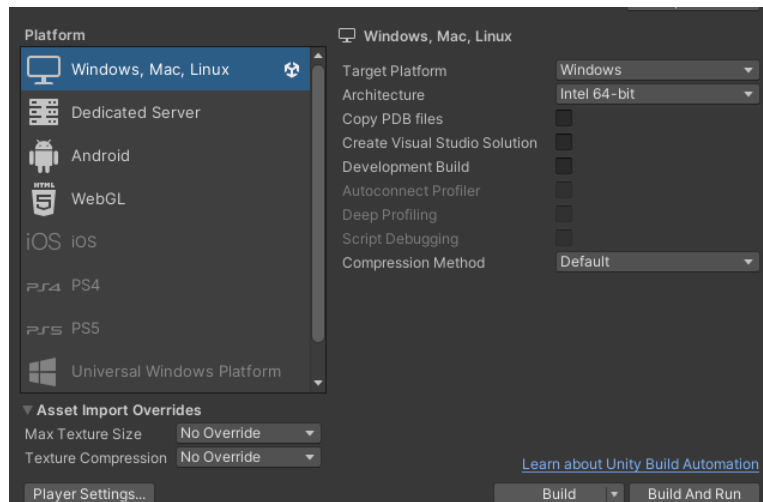


Diagrama 1 – Sistema Audio Unity



Fotografía 13 – Build Settings Unity

Tabla de Ilustraciones

Diagrama 1 – Sistema Audio Unity.....	13
Web 1- Pagina Web de Unity	3
Web 2 – Pagina Web LibGDX	3
Web 3 – Unity Documentation – SceneManager.LoadScene	13
Fotografía 1 – Doom ejecutándose en una calculadora	4
Fotografía 2 – Half Life 2 Arma controladora de gravedad	4
Fotografía 3 – Red Dead Redemption 2	5
Fotografía 4 – Set de grabación serie The Mandalorian	5
Fotografía 5 – Half Life 2 UI.....	6
Fotografía 6 – Cinemática Half Life 2	7
Fotografía 7 – Ajustes Gráficos Cities Skylines 2	9
Fotografía 8 – Undertale (2D)	10
Fotografía 9 – Minecraft (3D).....	10
Fotografía 10- League of Legends (2,5D)	10
Fotografía 11 – Prefab Unity	11
Fotografía 12 – Fondo en una escena en Unity	11
Fotografía 13 – Build Settings Unity.....	13