UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE DISEÑO MULTIMEDIA

DISEÑO Y DESARROLLO DE UN VIDEOJUEGO EN REALIDAD VIRTUAL: PROGRAMACIÓN E IMPLEMENTACIÓN DE MECÁNICAS

Trabajo de titulación previo a la obtención del título de Licenciado en Diseño Multimedia

AUTOR: GEOVANNY NICOLÁS ORELLANA JARAMILLO

TUTOR: RAFAEL AUGUSTO CAMPOVERDE DURÁN, PHD.

Cuenca - Ecuador

2025

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Geovanny Nicolás Orellana Jaramillo con documento de identificación N° 0104978143 manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 17 de enero del 2025

Atentamente,

Geovanny Nicolás Orellana Jaramillo

0104978143

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Geovanny Nicolás Orellana Jaramillo con documento de identificación N° 0104978143, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Proyecto integrador: "Diseño y desarrollo de un videojuego en realidad virtual: programación e implementación de mecánicas", el cual ha sido desarrollado para optar por el título de: Licenciado en Diseño Multimedia, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 17 de enero del 2025

Atentamente,

Geovanny Nicolás Orellana Jaramillo

0104978143

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Rafael Augusto Campoverde Durán con documento de identificación N° 0102377520, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO Y DESARROLLO DE UN VIDEOJUEGO EN REALIDAD VIRTUAL: PROGRAMACIÓN E IMPLEMENTACIÓN DE MECÁNICAS, realizado por Geovanny Nicolás Orellana Jaramillo con documento de identificación N° 0104978143, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto integrador que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 17 de enero del 2025

Atentamente,

Rafael Augusto Campoverde Durán, PhD.

0102377520

1. DEDICATORIA

Dedico esta tesis a todas las personas que siempre han estado a mi lado, brindándome su apoyo, amor y motivación en cada paso de este largo camino. A mis abuelos, Tomás Cornelio Jaramillo Arizaga y Teresa de Jesús Sarmiento Seminario, por su sabiduría y por enseñarme a valorar los esfuerzos que nos permiten alcanzar nuestros más grandes sueños. A mis padres, Flavio Enrique Orellana Castro y Tania Esperanza Jaramillo Sarmiento, por su amor incondicional, su paciencia y por siempre confiar en mí, aun en los momentos de duda.

A mi hermano, Pedro José Orellana Jaramillo, por ser mi compañero incondicional, mi motivación y mi guía, por compartir tantos momentos de alegría y reflexión. A mis primos, por su apoyo constante y por hacerme sentir siempre rodeado de amor.

A toda mi familia, por su confianza, por creer en mi incluso cuando yo dudaba, y por ser mi fuerza en cada etapa de este proceso.

A mis amigos y compañeros más cercanos, quienes me han acompañado en esta travesía con risas, consejos y apoyo inquebrantable.

2. AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a todas las personas que, de alguna u otra manera, contribuyeron a la realización de esta tesis.

A mi director de tesis, Rafael Augusto Campoverde Durán, por su valiosa orientación, paciencia y apoyo académico durante todo este proceso. Sus consejos y agradecimiento han sido fundamentales para el desarrollo de este trabajo.

A mis amigos y compañeros, Jorge Andres Sisalima Toba, Cristián André Narváez Miranda y Robinson Samuel Calle Cabrera, por su apoyo incondicional, por compartir conmigo tantas horas de estudio y trabajo, y por estar siempre ahí cuando más los necesité.

Y, por supuesto, a mi familia, en especial a mi hermano, cuyo amor y respaldo constante han sido fundamentales para llegar hasta aquí.

3. CONTENIDO

4. RESUMEN

Este estudio abordó el desafío de la falta de conciencia sobre los residuos de aparatos eléctricos y electrónicos, utilizando un enfoque innovador a través del desarrollo de un videojuego en realidad virtual (VR). El objetivo principal fue educar a los usuarios sobre la importancia del reciclaje mediante mecánicas interactivas y una narrativa inmersiva. Utilizando la metodología de prototipado iterativo y el motor de desarrollo Unity, se creó un prototipo que fue evaluado a través de pruebas con usuarios. Los resultados indicaron que el videojuego mejoró significativamente la concienciación entre los jugadores. Las conclusiones sugieren que la VR es una herramienta efectiva para la educación ambiental, proporcionando una experiencia inmersiva y atractiva que potencia el aprendizaje. Este estudio contribuyó al campo del diseño multimedia y la educación ambiental, destacando la utilidad de la VR en la sensibilización sobre problemas ecológicos.

Palabras clave:

Realidad virtual

Residuos electrónicos

Educación ambiental

Videojuegos

Prototipado iterativo

Unity.

5. ABSTRACT

This study addressed the challenge of lack of awareness about Waste Electrical and Electronic Equipment, using an innovative approach through the development of a virtual reality (VR) video game. The main goal was to educate users on the importance of recycling using interactive mechanics and an immersive narrative. Employing the iterative prototyping methodology and the Unity development engine, a prototype was created and evaluated through user testing. The results indicated that the video game significantly improved awareness among players. The conclusions suggest that VR is an effective tool for environmental education, providing an immersive and engaging experience that enhances learning. This study contributed to the field of multimedia design and environmental education, highlighting the utility of VR in raising awareness about ecological issues.

Keywords:

Virtual reality

Electronic waste

Environmental education

Video games

Iterative prototyping

Unity.

Diseño y desarrollo de un videojuego en realidad virtual: Programación e implementación de mecánicas.

6. INTRODUCCIÓN

El crecimiento constante de los residuos provenientes de aparatos eléctricos y electrónicos representa un reto ambiental significativo. Este problema, causado en parte por el rápido avance tecnológico y la obsolescencia programada, genera impactos negativos tanto en el medio ambiente como en la salud humana. Según (Ministerio del Ambiente, Agua y Transición Ecológica, 2024), en Ecuador se generan más de 88 mil toneladas anuales de estos desechos, con un promedio de cinco kilogramos por persona.

Ante esta situación, las tecnologías emergentes, como la realidad virtual (VR) y los videojuegos, se presentan como alternativas innovadoras para promover la educación ambiental. Estas herramientas permiten involucrar a los usuarios de manera interactiva, sensibilizándolos sobre los efectos de la mala gestión de residuos electrónicos.

7. PROBLEMÁTICA

La problemática de los residuos de aparatos eléctricos y electrónicos en Ecuador representa un desafío ambiental y social de creciente magnitud. La acumulación inadecuada de estos residuos conlleva grandes riesgos para la salud

humana y el medio ambiente, contaminando suelos, agua y aire con sustancias tóxicas. Ante esta situación, es imperativo buscar soluciones innovadoras y efectivas que promuevan la gestión responsable de dichos residuos y fomenten la conciencia ambiental en la población.

En este contexto, el presente proyecto propone el desarrollo de una experiencia interactiva en VR como una herramienta innovadora para abordar la problemática. La elección de los videojuegos, y en particular la VR, como medio de comunicación se justifica por su capacidad para generar inmersión, interacción y motivación en los usuarios. La VR ofrece un entorno virtual inmersivo que permite simular situaciones y experiencias que serían difíciles o imposibles de replicar en el mundo real, permitiendo a los usuarios experimentar de forma directa las consecuencias de la mala gestión de los residuos electrónicos y eléctricos y explorar soluciones de manera interactiva y atractiva. Este enfoque lúdico y experiencial tiene el potencial de generar un mayor impacto en la concienciación y el aprendizaje en comparación con métodos tradicionales.

Para el desarrollo de esta experiencia en VR, se ha seleccionado el motor de videojuegos Unity y el lenguaje de programación C#. Unity es una plataforma líder en la industria del desarrollo de videojuegos, reconocida por su versatilidad, su potente conjunto de herramientas y su amplio soporte para el desarrollo multiplataforma, incluyendo VR. Su interfaz intuitiva y su editor visual facilitan la creación de prototipos rápidos y la iteración en el diseño. C#, por su parte, es un lenguaje de programación moderno, orientado a objetos y ampliamente utilizado en Unity, que ofrece un equilibrio entre facilidad de uso y rendimiento, permitiendo la implementación de lógicas de juego complejas y la optimización del rendimiento

en entornos de VR. La combinación de Unity y C# proporciona un entorno de desarrollo robusto y eficiente para la creación de experiencias inmersivas y de alta calidad.

Se espera que el desarrollo de este videojuego en realidad virtual utilizando Unity y C# tenga un impacto positivo en la concienciación sobre la problemática de los residuos de aparatos eléctricos y electrónicos en Ecuador. Al ofrecer una experiencia interactiva e inmersiva, este proyecto busca generar un cambio de actitud y comportamiento con relación a la gestión de los residuos electrónicos y eléctricos, contribuyendo a la construcción de un futuro más sostenible.

8. MARCO TEÓRICO

El desarrollo de videojuegos es un campo que combina creatividad, habilidades técnicas y capacidad para resolver problemas. Dentro de este proceso, la programación juega un papel clave, ya que es la base sobre la que se construyen las mecánicas del juego, la lógica del sistema y la interacción con el usuario. Los programadores de videojuegos funcionan como arquitectos de mundos virtuales, convirtiendo ideas en código que da vida a experiencias interactivas.

Este proyecto se centra en aplicar principios de programación para el desarrollo de videojuegos, enfrentando tanto los retos técnicos como los creativos que surgen al diseñar experiencias innovadoras.

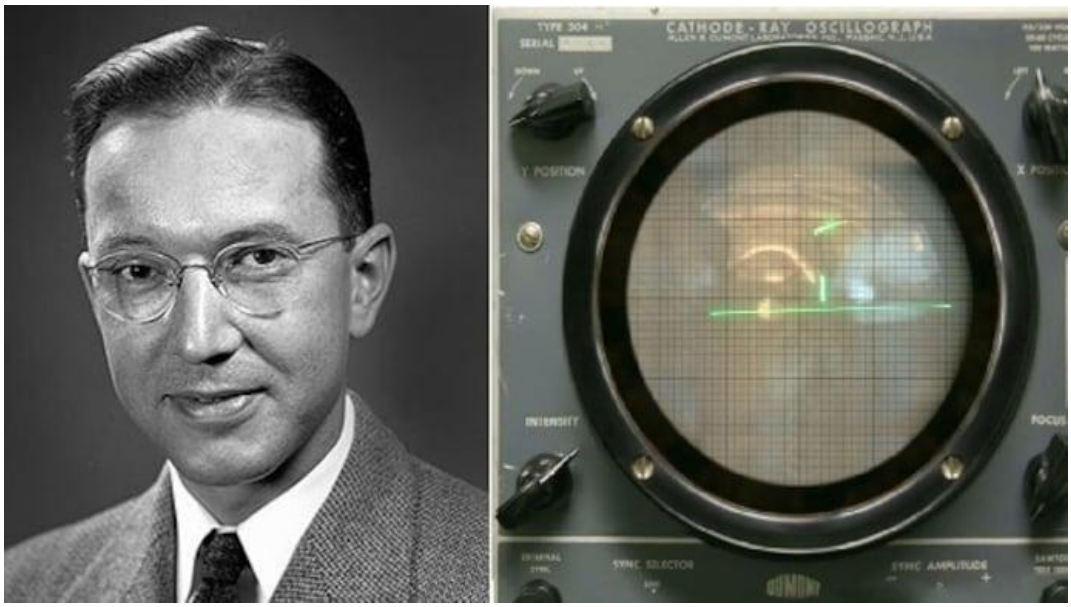
8.1 El inicio de los videojuegos:

Los videojuegos tuvieron sus inicios en la década de 1950 con desarrollos pioneros como “Tennis for Two”, un experimento realizado por William Higginbotham que

simulaba un partido de tenis. A pesar de su simplicidad, marcó el comienzo de un camino hacia experiencias cada vez más sofisticadas.

Figura 1

Videojuego Tennis for two y su creador William Higginbotham



Nota. El físico estadounidense William Higinbotham (izq.) desarrolló el primer videojuego del mundo, 'Tennis for Two' (der.), en 1958. Tomado de (India Today, 2024)

Cuatro años más tarde Steve Russell, alumno del Instituto de Tecnología de Massachussets, desarrollo el primer juego informático “Spacewar!”, sin embargo no alcanzo éxito significativo fuera del entorno universitario (Belli y López Raventós, 2008, pp. 5-7)

Los primeros videojuegos, con gráficos simples y mecánicas básicas, evolucionaron rápidamente hacia complejas narrativas interactivas y mundos virtuales cada vez más inmersivos. El surgimiento de consolas domésticas como

Atari y el fulgor subsiguiente de los videojuegos en la década de 1980 marcaron un punto de inflexión en la cultura popular, consolidando a los videojuegos como una forma de entretenimiento masivo. (Martínez Cantudo, pp. 12-16)

Figura 2

Así jugábamos: las primeras generaciones de consolas (1991-1993)

Nota. Primeras consolas de videojuegos de la historia. Tomado de (Herranz, 2021)

8.2 La realidad virtual y su convergencia con los videojuegos:

El concepto de realidad virtual, aunque con diferentes nombres, se remonta a la década de 1960 con el Sensorama de Morton Heilig, un dispositivo que intentó recrear una experiencia multisensorial. Posteriormente, el trabajo de Ivan Sutherland en las décadas de 1960 y 1970 (Machuca Contreras et al., 2024) sentó las bases para la VR moderna, desarrollando los primeros cascos de visualización. Sin embargo, las limitaciones tecnológicas de la época impidieron un desarrollo masivo de la VR durante varios años.

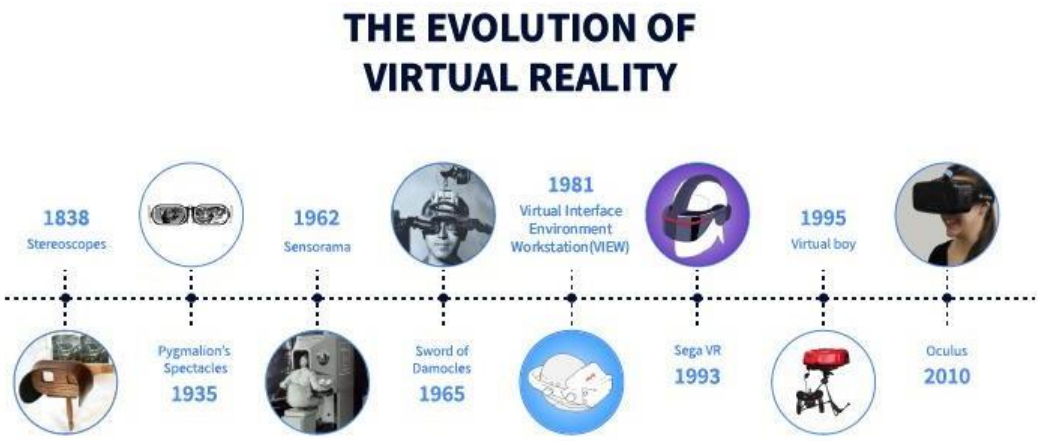
Entre los años 1977 y 1982, investigadores pertenecientes a la Universidad de Illinois, Daniel J. Sandin, Thomas A. DeFanti y Richard Sayre, desarrollaron los primeros guantes conectados a computadoras. Estos guantes utilizaban fotocélulas para detectar la cantidad de luz entrante y el movimiento de los dedos. En 1987, la empresa VPL Research Inc. comenzó a comercializar el producto como "Data Glove", y más tarde lanzó un casco de visualización llamado "Eye Phone" (Henderson et al., 2021, p. 17).

Desde sus inicios, la VR ha estado estrechamente ligada a los videojuegos. La realidad virtual busca sumergir al usuario en entornos simulados, brindándole la sensación de estar presente en un espacio digital. Los primeros intentos de integrar la VR en los videojuegos, como el Virtual Boy en Julio de 1995 (Zachara & Zagal, 2009, p. 101), aunque con limitaciones técnicas, demostraron el potencial de esta combinación.

El siglo XXI presenció un resurgimiento de la VR gracias a los avances en pantallas, sensores y potencia de procesamiento. La aparición de Oculus Rift y otros dispositivos de VR marcó un nuevo hito, revitalizando el interés en esta tecnología y abriendo un nuevo capítulo en la historia de los videojuegos. La VR ahora ofrece experiencias inmersivas de alta calidad, permitiendo la creación de juegos con mecánicas innovadoras y una mayor sensación de presencia.

Figura 3

The evolution of virtual reality



Nota. Evolución de la realidad virtual, desde su concepto, hasta su realización. Tomada de (the-story-of-virtual-reality, 2019)

8.3 Programación de videojuegos en Realidad Virtual:

Según (Delgado Yunquera, 2022), el desarrollo de un videojuego exitoso depende de la colaboración entre diversas disciplinas. Los programadores trabajan de la mano con artistas para diseñar los elementos visuales, con diseñadores de sonido para crear la atmósfera auditiva, con escritores para dar forma a la narrativa y con diseñadores de niveles para estructurar los escenarios del juego. Esta dinámica multidisciplinaria requiere una comunicación clara y una visión global del proceso de desarrollo.

En este contexto, la programación funciona como el vínculo entre la idea creativa y la experiencia interactiva. Un diseñador puede imaginar un sistema de combate complejo, pero es el programador quien convierte esa visión en código funcional, estableciendo las reglas de juego, la física de los proyectiles y la

inteligencia artificial de los enemigos. Por ello, la programación es una herramienta clave para transformar conceptos en experiencias jugables.

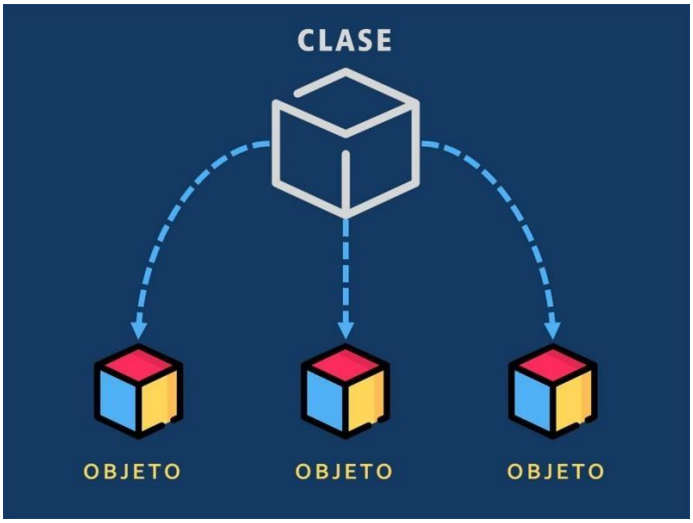
8.4 Fundamentos de la Programación orientada a objetos

La programación orientada a objetos (POO) se basa en conceptos como clases y objetos para estructurar sistemas complejos de manera más eficiente. Gracias a principios como la herencia, el encapsulamiento y el polimorfismo, es posible crear código más organizado, reutilizable y fácil de escalar (Equipo Geek - NTT DATA, 2019).

En el caso de Unity, por ejemplo, los **GameObjects** actúan como contenedores a los que se les pueden añadir diferentes componentes para definir su comportamiento, lo que facilita la modularidad y flexibilidad en el desarrollo de videojuegos.

Figura 4

Programación Orientada a Objetos



Nota. Ejemplo del funcionamiento de la POO. Tomado de (MGPanel, 2022)

8.4.1 Ventajas de la POO en el desarrollo de videojuegos:

La POO ofrece numerosas ventajas en el desarrollo de videojuegos. La modularidad permite dividir el código en componentes independientes, facilitando el trabajo en equipo y el mantenimiento del código. La reutilización de código a través de la herencia reduce la redundancia y acelera el desarrollo. La organización que proporciona la POO mejora la legibilidad y la escalabilidad del proyecto.

8.4.2 Aplicación de la POO en C# para Unity:

En Unity, los componentes que se agregan a los GameObjects están implementados como clases en C#. Estas clases determinan el comportamiento y las propiedades de los objetos dentro del juego. Por ejemplo, un script en C# puede controlar el movimiento de un personaje, gestionar la lógica de un disparo o definir cómo interactúa un objeto con su entorno.

La herencia juega un papel clave en este proceso, ya que permite crear componentes más específicos a partir de una base común, lo que facilita la reutilización de código y la implementación de sistemas más complejos sin necesidad de empezar desde cero.

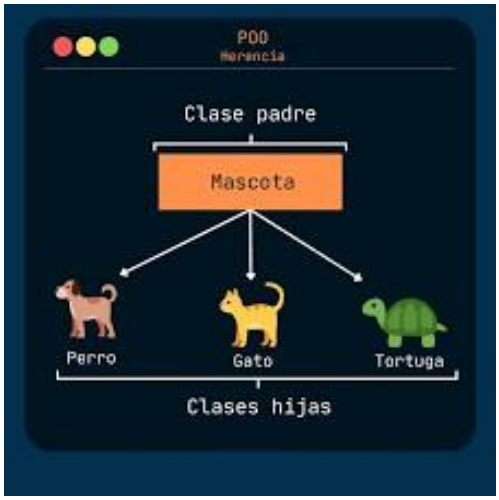
8.5 Herencia

La herencia es uno de los pilares fundamentales de la POO y tiene un rol crucial en el desarrollo de videojuegos. Gracias a este concepto, una clase "padre" puede compartir sus propiedades y métodos con clases "hijas", lo que permite

reutilizar código y mantener una estructura más organizada (Equipo Geek - NTT DATA, 2019).

Figura 5

Herencia POO



Nota. En este ejemplo se muestra un ejemplo de herencia en Java, con sus respectivas clases. Tomado de (Ortega, 2022)

8.5.1 Definición de herencia

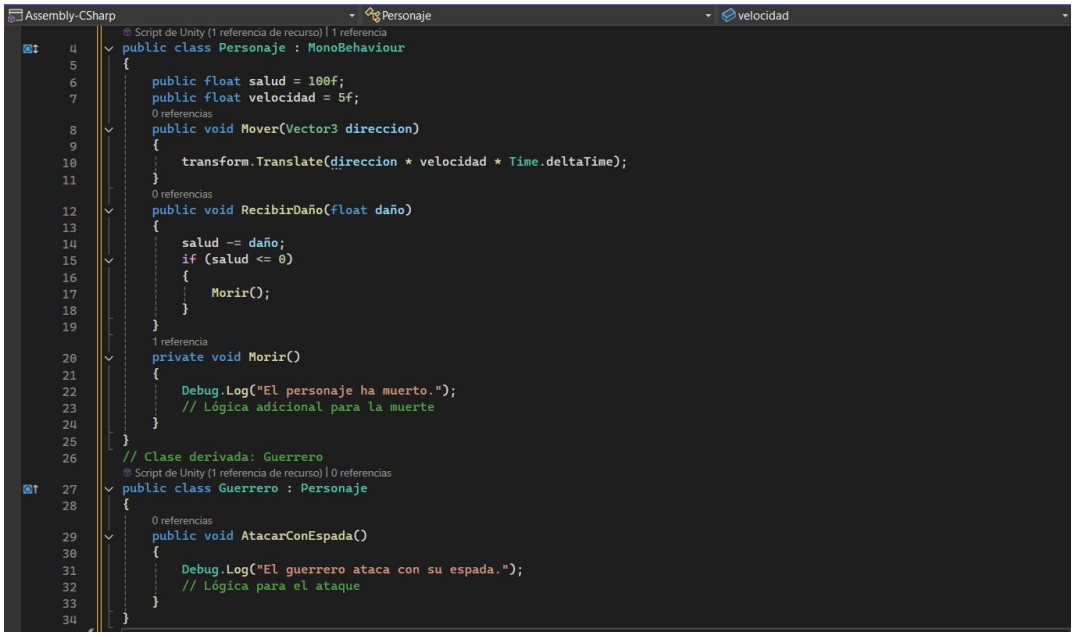
La herencia permite establecer relaciones jerárquicas entre objetos, donde una clase base define atributos y comportamientos generales, y las clases derivadas los amplían o personalizan según sea necesario.

Por ejemplo, en un videojuego, una clase base "Personaje" podría incluir atributos como salud y velocidad, junto con métodos como Mover() o RecibirDaño(). A partir de esta clase, se podrían crear subclases como "Jugador" o "Enemigo", que heredan estas características pero también agregan funcionalidades específicas, como habilidades únicas o patrones de comportamiento.

Este enfoque ayuda a mantener el código más limpio, organizado y fácil de mantener, ya que evita la duplicación de código y facilita la escalabilidad del proyecto.

Figura 6

Ejemplo de herencia



Nota. La clase “hijo” guerrero hereda propiedades de la clase “padre” personaje.

Las clases derivadas Guerrero y Mago pueden heredar estos atributos y métodos, añadiendo características específicas como AtacarConEspada().

8.5.2 Ventajas de la herencia en videojuegos

- **Reutilización de código:** Las propiedades y métodos comunes solo se tienen que definir una vez en la clase base.
- **Extensibilidad:** Las clases derivadas pueden agregar funcionalidades específicas sin modificar o cambiar la clase base.

- **Organización:** Facilita la estructuración del código en un orden lógico, mejorando la claridad y el mantenimiento.

8.6 Polimorfismo

El polimorfismo es otro principio clave de la POO y se refiere a la capacidad de una clase derivada para modificar o sobrescribir los métodos de una clase base, permitiendo que un mismo método funcione de distintas maneras según el contexto (López Blasco, 2023).

En el desarrollo de videojuegos, el polimorfismo resulta especialmente útil para crear sistemas más flexibles y escalables. Por ejemplo, si existe una clase base "Personaje" con un método Atacar(), distintas clases derivadas como "Guerrero" o "Mago" pueden sobrescribir este método para ejecutar ataques diferentes según el tipo de personaje.

8.6.1 Definición de polimorfismo

El polimorfismo es una herramienta que admite diferentes objetos, aunque sean de clases derivadas distintas, respondan de una manera propia a una misma llamada de método. Esto se logra a través de:

- **Sobreescritura de Métodos:** Una clase derivada redefine un método de la clase base para cambiar su comportamiento.
- **Sobrecarga de Métodos:** Un método tiene diferentes implementaciones según el número o tipo de parámetros.

8.6.2 Ventajas del polimorfismo en videojuegos

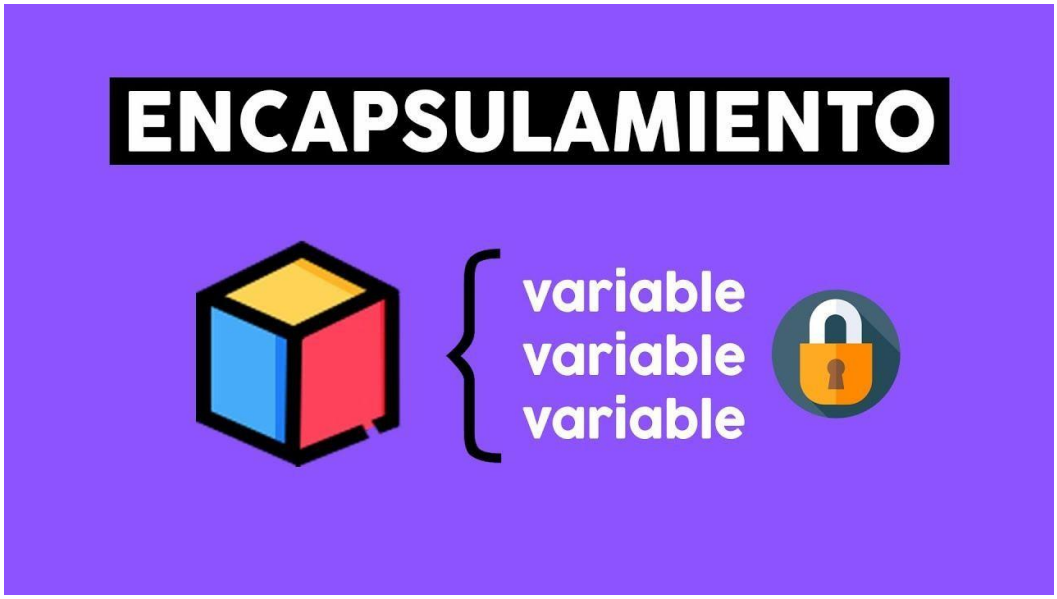
- **Flexibilidad:** Permite manejar clases derivadas mediante referencias de la clase base, simplificando la lógica del código.
- **Reutilización:** Evita duplicar código al definir comportamientos genéricos en la clase base que las clases derivadas pueden especializar.
- **Escalabilidad:** Facilita la adición de nuevas clases derivadas sin modificar el código existente.

8.7 Encapsulamiento

El encapsulamiento es otro de los principios fundamentales de la POO y se refiere a la capacidad de una clase para proteger sus datos y exponer únicamente los métodos y atributos necesarios para interactuar con ellos. Este enfoque promueve la modularidad, la seguridad y la facilidad de mantenimiento en el desarrollo de videojuegos, asegurando que los componentes del juego interactúen de manera controlada y segura (Quirino Rodríguez et al., 2024).

Figura 7

Encapsulamiento



Nota. Ejemplo de encapsulamiento en POO. Tomado de (Nico Programando, 2021).

8.7.1 Definición de encapsulamiento

El encapsulamiento implica ocultar la implementación interna de una clase y proporcionar acceso controlado a través de métodos públicos (getters y setters). Esto permite:

- **Controlar el Acceso:** Restringir el acceso directo a los atributos de una clase, definiéndolos como private o protected.
- **Proteger la Integridad:** Asegurar que los datos internos sean válidos al imponer restricciones mediante métodos.

Figura 8

Ejemplo de encapsulamiento

```
void Start()
{
    Personaje guerrero = new Personaje(100f);

    Debug.Log("Salud inicial: " + guerrero.ObtenerSalud());
    guerrero.AjustarSalud(-50f); // Resta salud
    Debug.Log("Salud después del daño: " + guerrero.ObtenerSalud());
}
```

Nota. Uso del encapsulamiento en videojuegos

8.8 Arquitectura y Patrones de Diseño en el Desarrollo de Videojuegos:

8.8.1 Arquitectura de un videojuego:

Un videojuego se compone de varios sistemas interconectados. El motor de juego (Unity, Unreal Engine, etc.) proporciona la infraestructura básica, incluyendo el sistema de renderizado (que se encarga de dibujar los gráficos en pantalla), el sistema de físicas (que simula el comportamiento de los objetos en el mundo virtual), el sistema de audio (que gestiona los efectos de sonido y la música) y el sistema de entrada (que procesa las acciones del usuario). Estos sistemas trabajan en conjunto para crear la experiencia de juego.

La organización del código en videojuegos es fundamental para garantizar su mantenimiento y escalabilidad. Patrones como MVC (Modelo-Vista-Controlador) y Singleton facilitan la gestión de datos y la interacción entre diferentes elementos del juego. Unity permite implementar estos patrones mediante herramientas integradas y componentes personalizables.

8.8.2 Patrón MVC (Modelo-Vista-Controlador):

El patrón de diseño MVC tiene el fin de reducir el esfuerzo de programación, útil en la implementación de sistemas múltiples y sincronizados de los mismos datos, a partir de normalizar el diseño de las aplicaciones (Fernández Romero & Díaz González, 2012).

Figura 9

Patrón de diseño MVC

Nota. Funcionamiento del patrón de diseño MVC. Tomado de (Easy App Code, 2020)

En un juego de disparos, el *Modelo* podría ser la clase 'Arma' que contiene información sobre el tipo de munición, la cadencia de disparo y el daño. La *Vista* sería la representación visual del arma en la pantalla del jugador. El *Controlador* se encargaría de recibir las entradas del jugador (presionar el botón de disparo) y

ejecutar la lógica correspondiente en el Modelo (restar munición, calcular el daño, etc.), actualizando luego la vista para reflejar los cambios.

8.8.3 Patrón singleton

El patrón Singleton es un patrón de diseño creacional que garantiza que una clase tenga una única instancia en todo el sistema, proporcionando un punto de acceso global a ella. Es útil en situaciones donde se necesita una instancia única para gestionar funciones esenciales, como un controlador de audio, un gestor de escenas o configuraciones globales (Blanco, pp. 23-26). Este patrón se implementa haciendo el constructor de la clase privado, creando una instancia estática y proporcionando un método público para acceder a esa instancia.

En Unity, el Singleton es comúnmente utilizado en sistemas globales como el *Game Manager*, que controla el flujo del juego, o un *Audio Manager* para manejar la música y los efectos sonoros. La implementación asegura que no se creen instancias adicionales accidentalmente, utilizando verificaciones en el método *Awake()* para destruir duplicados y mantener una única instancia a lo largo de la ejecución. Además, el método *DontDestroyOnLoad()* se emplea para preservar el objeto al cambiar de escena.

8.8.4 Beneficios de utilizar patrones de diseño:

El uso de patrones de diseño en el desarrollo de videojuegos mejora la organización del código, facilita el mantenimiento y la escalabilidad del proyecto, y promueve la reutilización de código. Al seguir patrones establecidos, los

desarrolladores pueden crear un código más robusto, legible y fácil de entender para otros miembros del equipo.

8.9 Características y capacidades de Unity para el desarrollo de RV:

Unity se destaca por ser una plataforma versátil que admite el desarrollo para múltiples dispositivos, incluyendo sistemas de realidad virtual. Su integración con herramientas como XR Interaction Toolkit permite implementar mecánicas inmersivas, mientras que su Asset Store ofrece recursos adicionales que aceleran el proceso creativo.

- **Soporte para dispositivos de RV:** Unity se integra con una amplia gama de dispositivos de VR, como Oculus, HTC Vive y otros, facilitando la configuración y el desarrollo para estas plataformas.
- **Herramientas para la interacción en VR:** Unity proporciona herramientas para la implementación de interacciones naturales en VR, como el seguimiento de manos (hand tracking), la teletransportación y la manipulación de objetos virtuales.
- **Optimización para VR:** El motor ofrece herramientas para optimizar el rendimiento de las aplicaciones de RV, lo cual es fundamental para mantener una alta tasa de fotogramas y evitar el mareo por movimiento (motion sickness)."
- **XR Interaction Toolkit:** Unity igualmente ofrece un paquete de interacción de alto nivel que se basa en componentes para crear experiencias de realidad virtual y realidad aumentada (AR) (Unity, 2024).

8.10 Desarrollo con Unity y C#:

8.10.1 Unity como entorno de desarrollo integral:

Unity ofrece un editor visual intuitivo que permite a los desarrolladores crear escenas, añadir objetos, configurar componentes y programar el comportamiento del juego sin necesidad de escribir código desde cero. Además, Unity proporciona un potente sistema de físicas, herramientas de animación, soporte para diferentes plataformas y una gran cantidad de recursos disponibles en la Asset Store.

8.10.2 C# como lenguaje de programación para Unity:

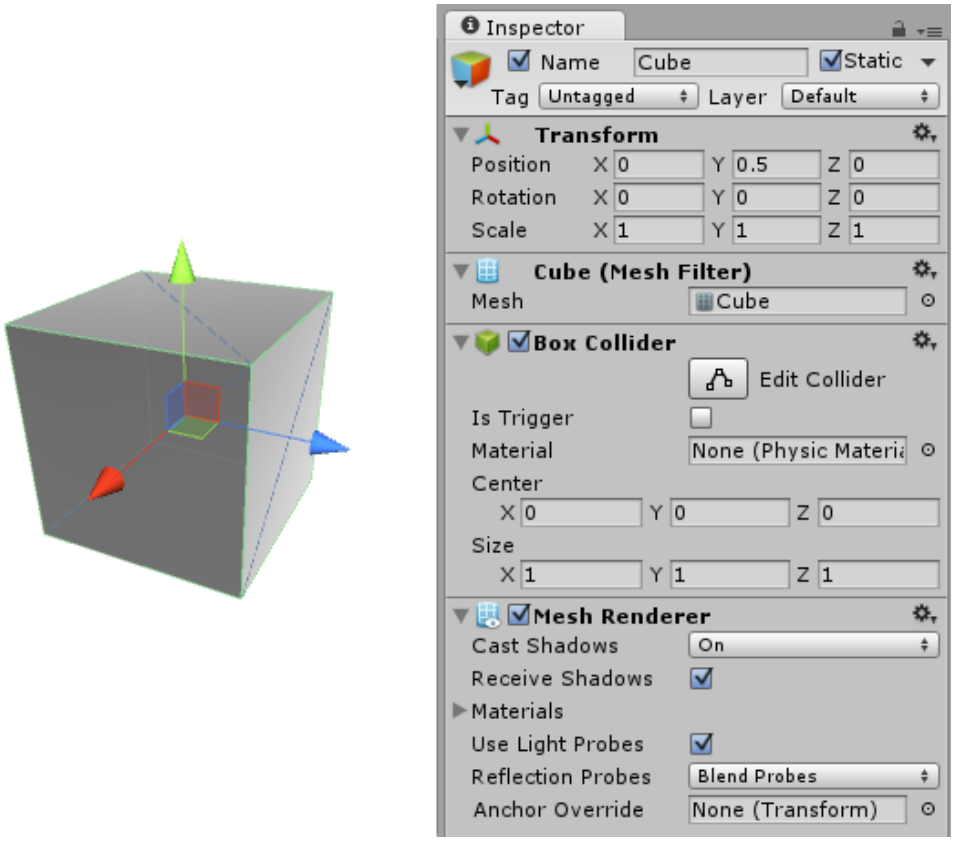
C# es un lenguaje de programación moderno y orientado a objetos que se integra perfectamente con Unity. Su sintaxis clara y concisa facilita la escritura y el mantenimiento del código. Además, C# ofrece características avanzadas como el manejo de eventos y la programación asíncrona, que son útiles para el desarrollo de videojuegos complejos (Microsoft Corporation, 2025).

8.10.3 Componentes y scripts en Unity:

Un GameObject en Unity representa un objeto en la escena del juego. Los componentes se adjuntan a los GameObjects para añadirles funcionalidad. Un script en C# es un componente que contiene el código que define el comportamiento del GameObject. Por ejemplo, un script podría controlar el movimiento de un personaje, la animación de un objeto o la respuesta a las entradas del usuario.

Figura 10

Gameobject y componentes Unity



Nota. Un GameObject de cubo simple con varios componentes. Tomado de (Unity, 2024).

8.11 Flujo de trabajo de desarrollo en Unity con C#:

El flujo de trabajo típico en Unity con C# implica la creación de escenas, la importación de assets (modelos 3D, texturas, sonidos, etc.), la creación de scripts en C# para controlar el comportamiento de los objetos y la configuración de las interacciones del usuario. El editor de Unity permite la previsualización en tiempo real del juego, lo que facilita la iteración y la depuración del código. La combinación de la interfaz visual de Unity con la potencia de C# ofrece un entorno de desarrollo flexible y eficiente.

8.12 Mecánicas de videojuegos:

8.12.1 ¿Qué son las Mecánicas de Videojuegos?

Las mecánicas de videojuegos (Rodríguez Mira, 2020) son el conjunto de reglas, sistemas y procesos que definen la interacción del jugador con el mundo del juego. Son los elementos que determinan cómo se juega, qué acciones puede realizar el jugador y cuáles son las consecuencias de esas acciones. En esencia, son el "cómo" del juego, en contraposición al "qué" (la narrativa, la estética, etc.).

Las mecánicas son la base de la jugabilidad. Definen:

- **Objetivos:** ¿Qué debe lograr el jugador? (Ejemplo: llegar al final del nivel, derrotar a un jefe, recolectar objetos).
- **Acciones:** ¿Qué puede hacer el jugador? (Ejemplo: saltar, disparar, interactuar con objetos, resolver puzles).
- **Reglas:** ¿Cómo funcionan las acciones y qué consecuencias tienen? (Ejemplo: daño por impacto, sistema de puntos, físicas de los objetos).
- **Restricciones:** ¿Qué no puede hacer el jugador? (Ejemplo: no puede volar sin un objeto específico, tiene un límite de inventario).

8.12.2 Tipos de Mecánicas en Videojuegos:

- **Movimiento:** Definen cómo se desplaza el jugador en el mundo del juego. (Ejemplos: caminar, correr, saltar, nadar, conducir, volar.)
- **Combate:** Rigen las interacciones de combate entre personajes o entidades. (Ejemplos: ataque cuerpo a cuerpo, disparos a distancia, uso de habilidades especiales, sistemas de cobertura.)

- **Interacción con el Entorno:** Permiten al jugador interactuar con objetos y elementos del mundo del juego. (Ejemplos: recolectar objetos, abrir puertas, resolver puzles, manipular interruptores.)
- **Gestión de Recursos:** Implican la administración de recursos por parte del jugador. (Ejemplos: gestión de inventario, economía, construcción de bases, gestión de unidades.)
- **Progresión:** Definen cómo avanza el jugador en el juego. (Ejemplos: sistemas de niveles, árboles de habilidades, desbloqueo de contenido.)
- **Sociales:** Facilitan la interacción entre jugadores. (Ejemplos: chat, comercio, clanes, cooperativo, competitivo.)

Un videojuego usualmente combina múltiples tipos de mecánicas para crear una experiencia completa.

8.12.3 Implementación de las Mecánicas de Videojuegos:

La implementación de mecánicas depende en gran medida del motor de juego y el lenguaje de programación que se utilice. A grandes rasgos, se realiza mediante:

- **Código (Scripts):** Se utilizan lenguajes de programación como C# (Unity), C++ (Unreal Engine) o Lua (en algunos motores) para definir la lógica de las mecánicas. Esto incluye la detección de entradas del jugador, la actualización del estado del juego y la respuesta a las acciones del jugador.
- **Componentes (en motores como Unity o Unreal Engine):** Los componentes son módulos prefabricados que ofrecen funcionalidades específicas, como físicas, audio o animaciones. Se pueden combinar y

configurar componentes para crear mecánicas complejas sin necesidad de escribir todo el código desde cero.

- **Diseño de niveles:** El diseño de niveles influye directamente en la forma en que se experimentan las mecánicas. La disposición de los elementos del entorno, los obstáculos y los desafíos pueden potenciar o limitar el uso de ciertas mecánicas.

9. OBJETIVOS:

9.1 Objetivo general

Desarrollar e implementar las mecánicas de juego e interfaz de usuario, mediante el motor grafico Unity, lenguaje de programación C# y el IDE Visual Studio, para el desarrollo del proyecto.

9.2 Objetivos específicos

- Implementar mecánicas interactivas que optimicen la experiencia del usuario mediante el uso del motor gráfico Unity y programación en C#.
- Configurar y optimizar los elementos visuales y funcionales de la UI para garantizar compatibilidad y facilidad de uso en dispositivos VR.
- Asegurar la integración eficiente de modelos 3D, animaciones y sonidos en las mecánicas del videojuego.

10. METODOLOGIA

10.1 Prototipado Iterativo

El desarrollo de este proyecto se llevará a cabo utilizando la metodología de Prototipado Iterativo, una estrategia ampliamente empleada en la creación de videojuegos y entornos interactivos. Esta metodología se centra en la construcción de prototipos funcionales que se evalúan y perfeccionan a través de ciclos continuos de pruebas y retroalimentación.

El prototipado iterativo resulta ideal para el diseño y desarrollo de videojuegos en realidad virtual, ya que permite detectar problemas técnicos y de diseño desde etapas tempranas, asegurando que las mecánicas y la experiencia del usuario sean óptimas antes de finalizar el producto.

10.2 Fases del Proceso

10.2.1 Planeación y Definición de Objetivos:

En esta fase inicial, se identifican los requisitos específicos del proyecto y se definen las metas de cada iteración. Se crea un plan de desarrollo que prioriza las funcionalidades principales del videojuego, tales como mecánicas de movimiento, interacción con objetos y navegación por la interfaz de usuario.

10.2.2 Diseño Inicial:

Aquí se realiza un diseño preliminar que incluye las mecánicas básicas del videojuego y los elementos del entorno virtual. Este diseño sirve como base para el desarrollo del primer prototipo.

Actividades:

- Creación de entornos iniciales en Unity.
- Configuración básica de la interfaz gráfica.

- Resultado: Prototipo básico del entorno virtual.

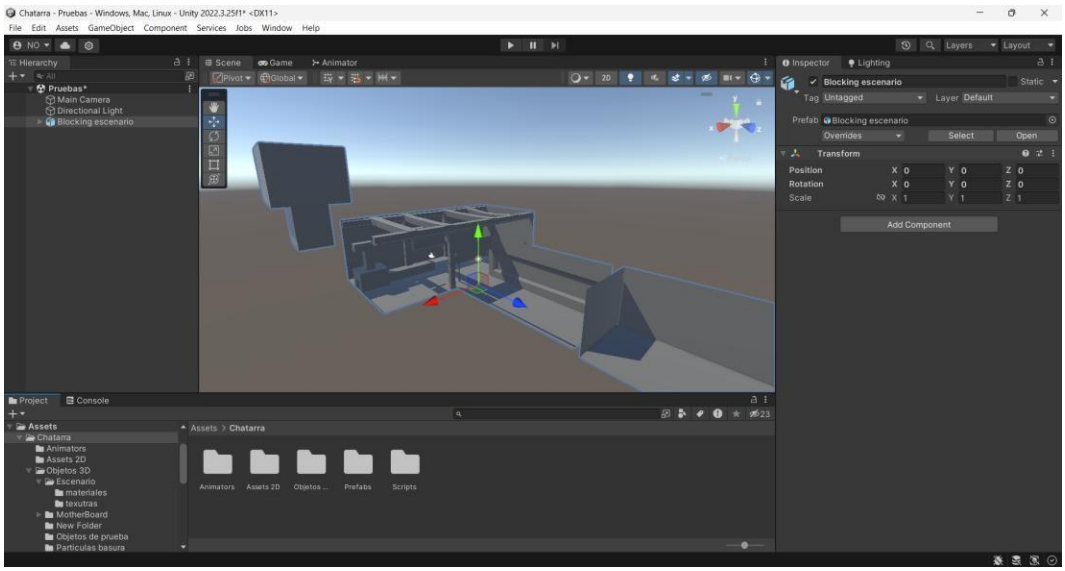
10.2.3 Desarrollo del Prototipo:

Se implementan las funcionalidades definidas en la etapa de planeación, utilizando Unity y el lenguaje de programación C#. Este prototipo incluye elementos básicos que se evaluarán en pruebas iniciales.

Herramientas: Unity, Visual Studio.

Figura 11

Blocking escenario



Nota. Prototipo del escenario principal del videojuego

10.2.4 Debugging:

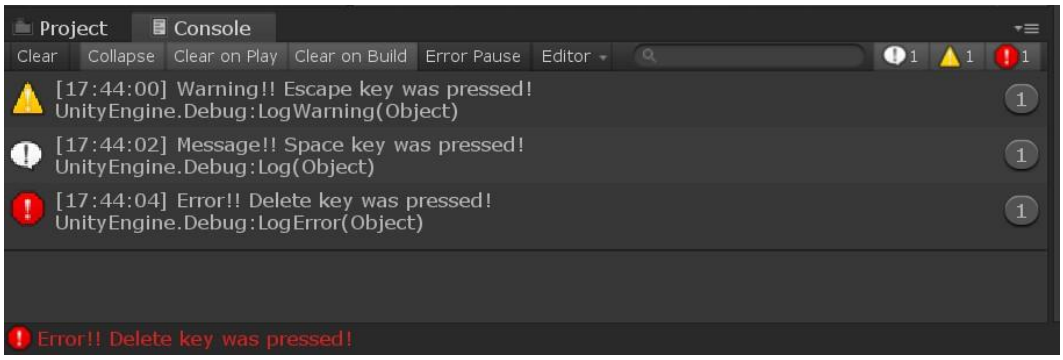
Este término se refiere a la identificación, análisis y corrección de errores o fallos en el código que puedan afectar el rendimiento, la funcionalidad o la experiencia del usuario.

Herramientas:

- **Consola de Unity:** Muestra mensajes de error, advertencias y logs personalizados que ayudan a rastrear problemas específicos en el código.
- **Debug.Log():** Una función en C# que permite registrar mensajes en la consola para rastrear el comportamiento del programa durante la ejecución.

Figura 12

Consola de Unity



Nota. Consola de unity con sus tipos de mensajes: Advertencias, Mensajes y Errores. Tomado de (javatpoint, s.f.)

10.2.5 Proceso de Debugging

- **Identificación de Errores:**

Durante las pruebas de usuario y la ejecución del prototipo, se recopilan datos sobre problemas técnicos o de rendimiento.

Los errores más comunes incluyen fallos en las mecánicas de interacción, tiempos de carga prolongados o inconsistencias en la UI.

- **Análisis:**

Se determina la causa raíz del error utilizando las herramientas anteriormente mencionadas como la consola de depuración de Unity.

- **Corrección:**

Se modifican los scripts o configuraciones correspondientes para resolver el problema.

- **Verificación:**

Tras implementar los cambios, se realizan pruebas para confirmar que el error ha sido solucionado sin generar nuevos problemas.

10.2.6 Pruebas de Usuario:

El prototipo es evaluado por usuarios internos o externos para identificar problemas de usabilidad, rendimiento o diseño. Estas pruebas ayudan a validar las mecánicas implementadas y garantizan una experiencia de usuario inmersiva y fluida.

Figura 13

Pruebas de usuario

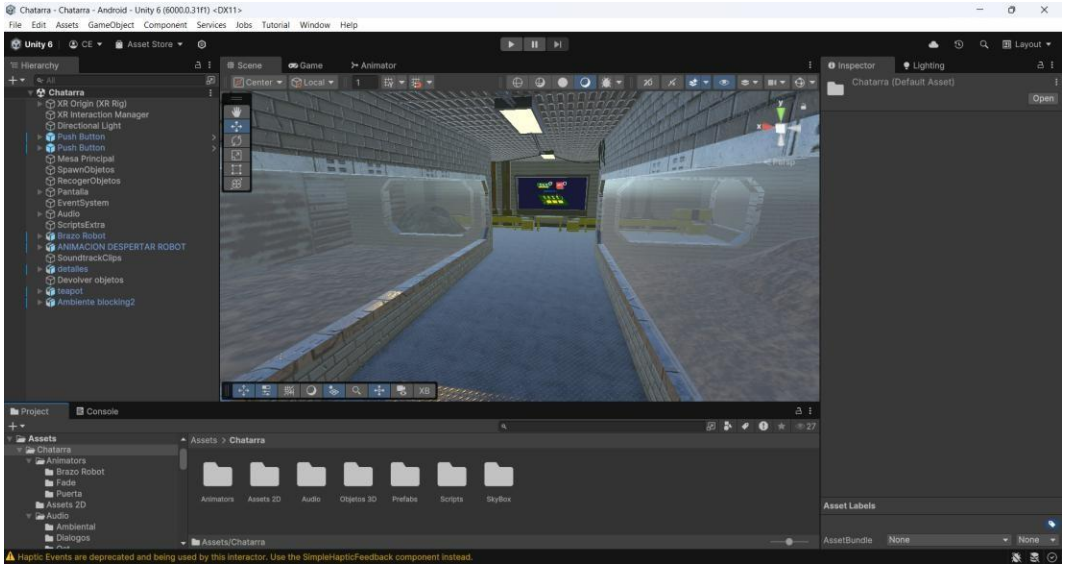
Nota. Pruebas realizadas a los estudiantes.

10.2.7 Producto Final:

Tras completar múltiples iteraciones, se obtiene un videojuego funcional y optimizado. Este producto final refleja el resultado de las pruebas, retroalimentación y mejoras realizadas durante todo el proceso.

Figura 14

Chatarra



Nota. Viewport del proyecto Chatarra finalizado.

11. TRABAJO PRÁCTICO

11.1 Planeación inicial

La planeación inicial del proyecto se centró en establecer una base sólida que permitiera desarrollar un videojuego inmersivo en realidad virtual, alineado con los objetivos de educar sobre el impacto de los residuos eléctricos y electrónicos mediante mecánicas de reciclaje en un entorno interactivo.

11.1.1 Conceptualización del proyecto

Narrativa y escenario

- La idea de un robot protagonista en un mundo postapocalíptico surge como una forma de conectar emocionalmente al jugador con el problema creciente de los residuos electrónicos y eléctricos.

- La narrativa fue diseñada en conjunto de todo el equipo de trabajo, sobre todo la dirección, buscando representar las consecuencias de una mala gestión de estos residuos y la esperanza que trae consigo el reciclaje como solución
- La elipsis al final del juego se planteó desde el inicio como un recurso narrativo para destacar el impacto positivo de las acciones del jugador.

Mecánicas iniciales

- **Movimiento y rotación en VR:** Garantizar que fuera intuitivo, y compatible con dispositivos como Oculus Quest 2 y Oculus Quest 3, utilizando los inputs de los controles, en este caso el joystick izquierdo para el movimiento, y el joystick derecho para la rotación
- **Interacción con objetos:** Centrada en recoger y armar dispositivos como: motherboard, batería y pistola, rearmándolos, usando componentes reciclados.
- Uso de cintas transportadoras y una mesa de trabajo para estructurar el flujo de reciclaje dentro del juego.

11.1.2 Selección de herramientas y recursos

Software

- **Unity:** Elegido como el motor de desarrollo principal debido a su robustez y capacidades avanzadas para la creación de experiencias en VR.
- **C#:** El lenguaje de programación utilizado para implementar todas las mecánicas y funcionalidades del videojuego.

- **XR Interaction Toolkit:** Integrado para permitir interacciones intuitivas en entornos de realidad virtual, como la manipulación de objetos y la navegación en el entorno.
- **Blender:** Herramienta seleccionada para modelado y animación 3D, con ajustes específicos para garantizar la compatibilidad con Unity.

Hardware

- **Dispositivos VR:** Oculus Quest 2 fue elegido como el dispositivo principal para pruebas y desarrollo debido a su accesibilidad y compatibilidad con Unity.

11.1.3 Organización y distribución de roles

- **Dirección y sonido:** Jorge Andres Sisalima Tobar
- **Desarrollo y programación:** Geovanny Nicolás Orellana Jaramillo
- **Modelado 3D y texturización:** Cristian André Narváez Miranda
- **Riggin y animación:** Robinson Samuel Calle Cabrera

12. RESULTADOS

12.1 Contexto practico

El videojuego se diseñó para abordar la problemática de los residuos eléctricos y electrónicos (RAEE) mediante una experiencia inmersiva en realidad virtual. En un mundo postapocalíptico, los jugadores toman el rol de un robot cuyo objetivo es reciclar componentes electrónicos, destacando la importancia del reciclaje como solución a los problemas ambientales.

Este enfoque combina educación y entretenimiento, utilizando la realidad virtual para sensibilizar al jugador sobre las consecuencias de una gestión inadecuada de residuos electrónicos, y mostrando cómo pequeñas acciones individuales pueden contribuir a un cambio significativo.

12.2 Briefing del proyecto

12.2.1 Nombre del proyecto

Chatarra

12.2.2 Objetivo principal

Crear un videojuego en realidad virtual que eduque a los jugadores sobre los residuos eléctricos y electrónicos mediante mecánicas de reciclaje y una narrativa inmersiva.

12.2.3 Elementos clave

- **Narrativa:** Un robot protagonista guiado por una IA en un mundo devastado por residuos electrónicos.
- **Mecánicas:** Reciclaje de componentes electrónicos mediante cintas transportadoras y mesas de trabajo.
- **Impacto:** Mostrar al jugador cómo sus acciones contribuyen a salvar el planeta, culminando con un mensaje esperanzador tras una elipsis narrativa.

12.2.4 Público objetivo

- Estudiantes y jóvenes interesados en aprender sobre sostenibilidad y reciclaje a través de medios interactivos.

- Usuarios de dispositivos VR interesados en experiencias educativas.

12.2.5 Proceso creativo

- **Conceptualización:** Diseño de una narrativa educativa y mecánicas de reciclaje, integrando una representación visual de un mundo afectado por residuos eléctricos.
- **Diseño y desarrollo:** Creación de modelos 3D y animaciones en blender. Música y sonidos ambiente. Implementación de mecánicas interactivas mediante XR Interaction Toolkit.
- **Pruebas de usuario:** Pruebas iniciales con estudiantes de diseño multimedia, quienes interactuaron con el prototipo y proporcionaron retroalimentación mediante encuestas y observaciones directas.

12.2.6 Producto final

Mecánicas implementadas

- Movimiento fluido e intuitivo en VR.
- Interacción con objetos en cintas transportadoras y mesas de trabajo.
- Reciclaje de componentes como elemento central del juego.

Narrativa

Una narrativa inmersiva que guía al jugador desde un escenario introductorio hasta un desenlace esperanzador tras completar su misión.

Optimización

Ajustes técnicos para garantizar una tasa de fotogramas constante y evitar errores en las colisiones de los objetos interactivos.

12.2.7 Evaluación

Las pruebas realizadas con estudiantes nos permitieron identificar y solucionar errores clave, como:

- **Objetos que desaparecían:** Solucionado al hacer que reaparecieran en la cinta transportadora.
- **Colisiones:** Limitación de la colisión entre el usuario y objetos interactivos para evitar que salieran del escenario.
- **Espacios sin límites:** Implementación de colliders para evitar que los jugadores cayeran del entorno.

13. CONLUSIONES

El desarrollo del videojuego en realidad virtual destinado a educar sobre los residuos de aparatos eléctricos y electrónicos ha demostrado ser una herramienta eficaz para concienciar sobre la importancia del reciclaje y la gestión ambiental. A lo largo del proyecto, se han enfrentado diversos desafíos técnicos y creativos, pero también se han obtenido aprendizajes valiosos que han enriquecido la experiencia del equipo de desarrollo y, se espera, también enriquecerán a los usuarios finales del juego.

13.1 Impacto Educativo y de Concienciación

El videojuego ha cumplido su objetivo de sensibilizar al público sobre los problemas generados por los RAEE, mostrando de manera práctica cómo acciones individuales pueden contribuir significativamente a la solución de problemas ambientales. La narrativa del juego y sus mecánicas interactivas facilitan un aprendizaje inmersivo que es tanto educativo como entretenido.

13.2 Desafíos Técnicos y Soluciones Implementadas:

A lo largo del proyecto, se enfrentaron varios desafíos, como la integración de modelos 3D, la precisión de las colisiones y la gestión de objetos dentro del entorno virtual. La colaboración efectiva entre programadores y diseñadores permitió superar estos retos mediante ajustes técnicos y creativos, tales como la optimización de colisiones y la implementación de sistemas de reaparición de objetos para asegurar una experiencia de usuario fluida y libre de frustraciones.

13.3 Lecciones Aprendidas:

El proceso de desarrollo reafirmó la importancia crítica de la programación en la creación de videojuegos. Las habilidades en programación no solo se aplicaron para resolver problemas técnicos sino también para implementar y refinar las mecánicas de juego que son esenciales para la educación y el entretenimiento.

Además, las pruebas de usuario fueron fundamentales para identificar problemas no previstos y confirmar que el diseño del juego era intuitivo y accesible para el público objetivo.

13.4 Recomendaciones para Futuros Proyectos:

Se recomienda continuar explorando y expandiendo el uso de realidad virtual en educación ambiental, aprovechando las capacidades inmersivas de esta tecnología para crear experiencias educativas aún más profundas y efectivas.

Para futuros desarrollos, se sugiere una planificación aún más detallada en las fases iniciales del diseño de juego, especialmente en la prueba y ajuste de mecánicas de interacción en VR para garantizar que los elementos del juego funcionen armoniosamente en un amplio rango de dispositivos de realidad virtual.

13.5 Visión a Futuro:

A partir de los resultados obtenidos y la experiencia adquirida, hay un claro potencial para ampliar el proyecto con nuevas mecánicas, niveles y narrativas que continúen promoviendo la conciencia ambiental a través de la gamificación y la educación interactiva.

14. BIBLIOGRAFÍA

Belli, S., y López Raventós, C. (2008). *Athenea Digital.* Revista de pensamiento: https://www.redalyc.org/pdf/537/53701409.pdf

Blanco, C. (s.f.). Patrones de Diseño. *Patrones de Diseño.* Universidad de Cantabria, Cantabria.

Delgado Yunquera, T. (31 de 03 de 2022). *Como se crea un videojuego*. Bit: https://bit.coit.es/como-se-crea-un-videojuego/

Easy App Code. (02 de Septiembre de 2020). *Patrón de diseño MVC. ¿Qué es y cómo puedo utilizarlo?* Easy App Code:

https://www.easyappcode.com/patron-de-diseno-mvc-que-es-y-como-puedo-utilizarlo

Equipo Geek - NTT DATA. (5 de Junio de 2019). *¿Qué es la Herencia en programación orientada a objetos?* if geek then: https://ifgeekthen.nttdata.com/s/post/herencia-en-programacion-orientada-objetos-MCPV3PCZDNBFHSROCCU3JMI7UIJQ?language=es

Fernández Romero, Y., y Díaz González, Y. (2012). Patrón modelo-vista-controlador. *Telem@tica*, 1.

Henderson, J., Condell, J., Connolly, J., Kelly, D., y Curran, K. (2021). *Review of Wearable Sensor-Based Health Monitoring Glove Devices for Rheumatoid Arthritis.*

Herranz, S. (16 de enero de 2021). *jugabamos-primeras-generaciones-1991-1993-784823*. HOBBYCONSOLAS: https://www.hobbyconsolas.com/reportajes/jugabamos-primeras-generaciones-1991-1993-784823

India Today. (28 de Octubre de 2024). *indiatoday*. A physicist created the world's first-ever video game: https://www.indiatoday.in/education-today/gk-current-affairs/story/william-higinbotham-physicist-invented-worklds-first-video-game-tennis-for-two-2622810-2024-10-25

javatpoint. (s.f.). *Unity Console*. https://www.javatpoint.com/: https://www.javatpoint.com/unity-console

López Blasco, J. (2023 de Noviembre de 2023). *Open Webinars*. introduccion-a-poo-en-java-herencia-y-polimorfismo: https://openwebinars.net/blog/introduccion-a-poo-en-java-herencia-y-polimorfismo/

Machuca Contreras, F., Lepez, C. O., y Canova Barrios, C. (2024). *Influence of virtual reality and augmented reality on mental health.* Santiago de Chile.

Martínez Cantudo, R. (2015). *LA GENERACIÓN QUE CAMBIÓ LA HISTORIA DEL VIDEOJUEGO.* SÍNTESIS, S. A.

MGPanel. (Agosto de 2022). *¿Qué es la Programación Orientada a Objetos?* MGPanel: https://www.mgpanel.org/post/-que-es-la-programacion-orientada-a-objetos-

Microsoft Corporation. (2025). *csharp*. microsoft.com: https://dotnet.microsoft.com/es-es/languages/csharp

Ministerio del Ambiente, Agua y Transición Ecológica. (30 de 10 de 2024). *La gestión responsable de los residuos de aparatos eléctricos y electrónicos se presentó a representantes de los medios de comunicación*. www.ambiente.gob.ec: https://www.ambiente.gob.ec/la-gestion-responsable-de-los-residuos-de-aparatos-electricos-y-electronicos-se-presento-a-representantes-de-los-medios-de-comunicacion/#:~:text=En%20Ecuador%2C%20solo%20en%202020,de%20cinco%20kilogramos%20por%20a%C3%B1o.

Nico Programando. (1 de Febrero de 2021). *ENCAPSULAMIENTO | Programacion Orientada a Objetos en Java*. YouTube: https://www.youtube.com/watch?v=MHQZyCniPkk

Ortega, G. (28 de Junio de 2022). *sobrecodigo*. Instagram: https://www.instagram.com/sobrecodigo/p/CfXj8HasCfP/?img_index=1

Que-es-la-programacion-orientada-a-objetos. (24 de 07 de 2022). Universidad Europea: https://universidadeuropea.com/blog/programacion-orientada-objetos/#que-es-la-programacion-orientada-a-objetos

Quirino Rodríguez, L. G., Huerta Mora, E. A., Valenzuela Carrasco, A. C., y López López, H. L. (2024). PARADIGMA DE LA PROGRAMACION ORIENTADA A OBJETOS (POO). *Revista Digital de Tecnologías Informáticas y Sistemas*, *8*(1), 147-150. https://doi.org/10.61530

Rodríguez Mira, A. (21 de Enero de 2020). *www.tokioschool.com/noticias/mecanicas-de-juego-habituales-en-videojuegos/*. www.tokioschool.com: https://www.tokioschool.com/noticias/mecanicas-de-juego-habituales-en-videojuegos/

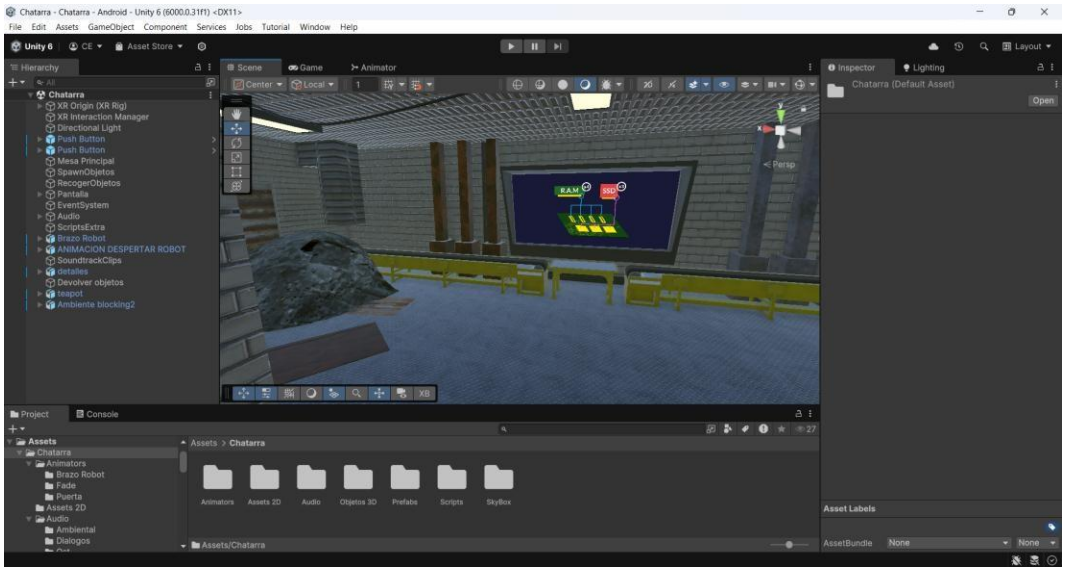
the-story-of-virtual-reality. (04 de Diciembre de 2019). ib.cricket: https://ib.cricket/the-story-of-virtual-reality/

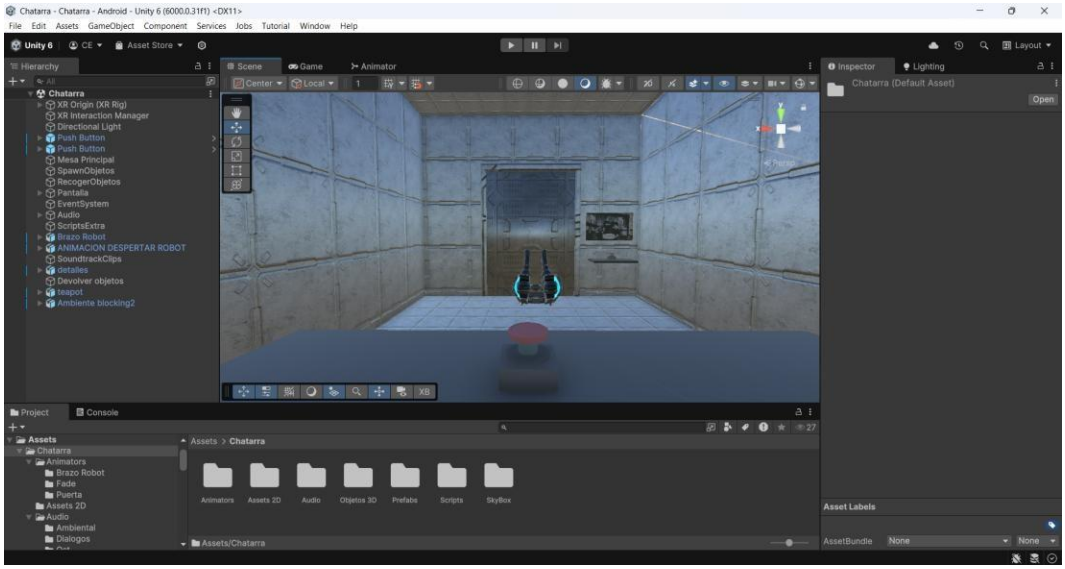
Unity. (2024). *Unity*. XR Interaction Toolkit: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@latest/

Unity Technologies. (2023). *Unity Documentation*. Unity: https://docs.unity3d.com/2023.2/Documentation/Manual/AssetStore.html

Zachara, M., y Zagal, J. P. (2009). Challenges for success in stereo gaming: a Virtual Boy case study. *Challenges for success in stereo gaming: a Virtual Boy case study*. College of Computing and Digital Media DePaul University, Chicago.

15. ANEXOS



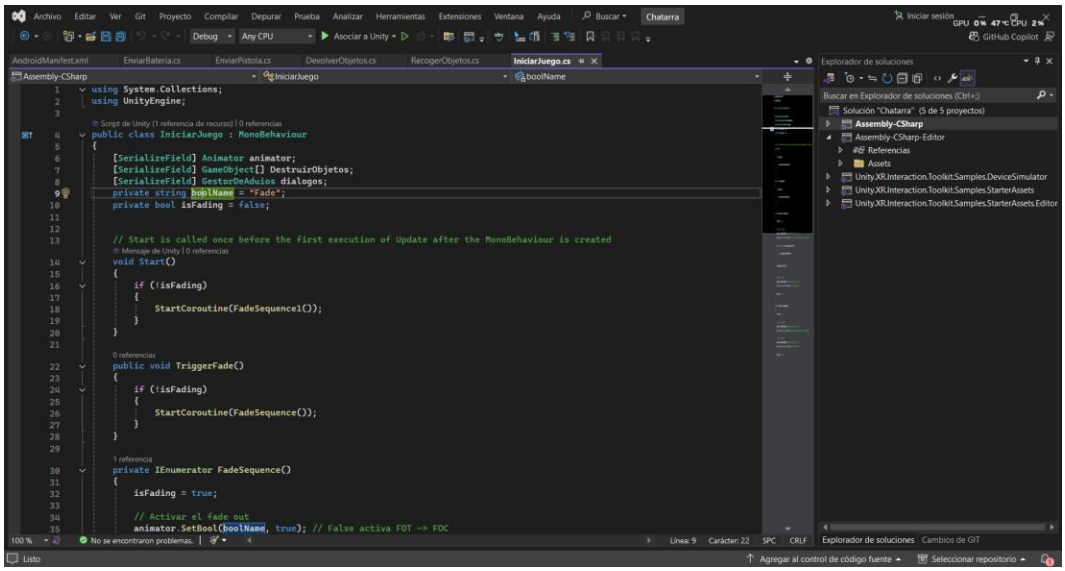



Assets > Chatarra > Prefabs > Bateria
caja b...
caja az...
caja az...
tapa c...


Assets > Chatarra > Prefabs > MotherBoard
mother...
mother...
mother...


Assets > Chatarra > Prefabs > Pistola de Agua
cuerpo...
mango...
mangu...
tanque...


Assets > Chatarra > Scripts
AbrirPuerta
ChangeL...
CintaTran...
Comprob...
Devolver...
EnviarBat...
EnviarPist...
EnviarPlaca
GestorDe...
IniciarEsc...
IniciarJue...
MusicaFinal
RecogerO...
SocketsC...
Soundtra...


using System.Collections;
using UnityEngine;
public class IniciarJuego : MonoBehaviour
{
[SerializeField] Animator animator;
[SerializeField] GameObject[] DestruirObjetos;
[SerializeField] GestorDeAudios dialogos;
private string boolName = "Fade";
private bool isFading = false;
// Start is called once before the first execution of Update after the MonoBehaviour is created
void Start()
{
if (!isFading)
{
StartCoroutine(FadeSequence1());
}
}
public void TriggerFade()
{
if (!isFading)
{
StartCoroutine(FadeSequence());
}
}
private IEnumerator FadeSequence()
{
isFading = true;
// Activar el fade out
animator.SetBool(boolName, true); // False activa FDT -> FDC