Instituto Tecnológico de Costa Rica

Carrera de Ingeniería en Computación

Campus Tecnológico Local San Carlos

Incremento a “Let Me Go”: Enemigos y combate rítmico

Práctica Profesional para optar por el título de Ingeniero en Computación con el grado académico de Bachiller Universitario

Leonardo Lizano Nájera

Costa Rica, 2021

Resumen Ejecutivo

El presente proyecto corresponde al desarrollo de una plataforma de software para la empresa Headless Chicken Games S.A, correspondiendo a un incremento de las funcionalidades del videojuego “Let Me Go”, el cual está diseñado para realidad virtual y está hecho en el motor gráfico Unity3D, agregando todo un módulo de combate con sus respectivas mecánicas y funcionalidades dentro de las que se puede mencionar el crear enemigos, un modo de combate, estadisticas de vida del jugador o los enemigos y otras funciones. La idea principal consiste en diseñar y crear un modo de combate que se ajuste a las estéticas rítmicas y musicales del juego. Para las pruebas de las mecánicas y el sistema en general se utilizan los visores de realidad virtual Oculus Quest y Oculus Quest 2.

Para realizar este proyecto se cuenta con un lapso de 17 semanas dentro de las cuales está la capacitación, toma de requerimientos, diseño, programación, pruebas y documentación de la plataforma de software y sus diferentes funcionalidades, quedando este documento y la presentación final como evidencia de la capacidad del estudiante de desempeñar profesionalmente el rol de programador y bajo la tutela y dirección de la empresa, al lado de los avances y evaluaciones presentadas al Instituto Tecnológico de Costa Rica como parte del curso de práctica profesional.

Se obtiene un sistema de software fácilmente integrable dentro del entorno de desarrollo de la plataforma Unity3D que contiene toda la lógica, funcionalidades y mecánicas para evitar o enfrentar enemigos dentro del entorno 3D del videojuego de realidad virtual, que se utiliza para rellenar “espacios vacíos” del escenario que requieren ser explorados más de una vez, manteniendo al jugador alerta y agregando valor al producto.

Palabras clave

Videojuego, realidad virtual, Unity3D, combate, enemigos, estéticas rítmicas y musicales, Oculus Quest.

Executive summary

The following project corresponds to the development of a software platform for Headless Chicken Games S.A, the corresponds to a functionalities increase for the videogame called “Let Me Go”, which is designed for virtual reality and currently developed the graphics motor Unity3D, adding a whole new combat module with it’s own mechanics and functionalities, to mention some examples, adding enemies, a full combat system, stats

about player or enemies health and other functions. The main idea is to design and create a combat module that fits the game's rhythmic and musical esthetics. To develop and test these mechanics and general system Oculus Quest and Quest 2 virtual reality headsets will be used.

The project deadline is 17 weeks covering capacitation, requirements writing, design, programming, testing and documentation of the software platform and all its functionalities, this document and the final presentation works as evidence of the student capacity to professionally develop a development role under the company guidance and supervision, on the side of the advances and evaluations shown to Instituto Tecnológico de Costa Rica as part of the professional practice course.

The final product is an easy to integrate combat system inside the development environment of Unity3D that contains all the combat logic, functionalities and mechanics to avoid or fight enemies inside the 3D environment of the virtual reality game, used to fulfill “empty gameplay spaces” of the scenario where the player has to explore more than once, keeping the player entertained and alert, adding a new value to the product

Keywords

Videogame, virtual reality, combat, enemies, rhythmic and musical esthetics, Oculus Quest.

Tabla de contenidos

Índice de figuras

Índice de tablas

Capítulo I

1 Introducción

El presente proyecto está enfocado en el área de desarrollo de videojuegos, es un producto de propiedad intelectual de la empresa Headless Chicken Games S.A, el cual es un incremento a un videojuego en desarrollo que ya existe.

El juego “Let Me Go” es un producto de propiedad intelectual interna, por lo que el desarrollo no es producto del contrato con algún cliente de la compañía. El juego está enfocado en realidad virtual, se ejecuta en los dispositivos de Facebook Oculus Quest y se está desarrollando en Unity3D, en la versión 2021.1.12f1 del motor de juegos. Unity3D ofrece su propio IDE, y la lógica de programación se realiza en Visual Studio Community, en el lenguaje C#.

El juego se encuentra en una etapa donde existen varias de las funcionalidades principales como lo es la inmersión en un entorno 3D y algunas mecánicas que aportan a la experiencia de exploración, las funcionalidades que ya posee el producto se detallan en el apartado de contexto del proyecto, aún no hay *releases* o acceso previo programado para el producto, y se espera presentarlo a posibles *publishers* (terceros encargados de invertir en el mercadeo y distribución del juego.). Durante el desarrollo de la práctica el estudiante estará encargado del desarrollo de un conjunto de mecánicas de suma importancia para el videojuego diseñado y desarrollado por la empresa.

1.1 Descripción de empresa

Headless Chicken Games S.A es una empresa de diseño y desarrollo de videojuegos, a nivel interno y también externo, producción de *assets* 3D/2D, modelado y riggeado de personajes para videojuegos. Fundada en 2013 por Jose Pablo Monge Chacón y ubicada en San José, San Pedro de Montes de Oca, la empresa cuenta con un equipo de catorce personas, distribuido en gestión de proyectos, desarrolladores y artistas. Uno de los valores más importantes que busca la empresa es la pasión y el talento, donde se evalúa mucho la forma de ser de las personas y no solo sus capacidades técnicas.

Como se mencionó antes, la empresa ofrece el desarrollo de videojuegos como un servicio de *outsourcing*, al igual que cuentan con una variedad de proyectos internos.

Dentro de los videojuegos desarrollados por la empresa cabe mencionar.

- Night Raid, es un videojuego en desarrollo multijugador cooperativo desarrollado en el motor Unreal Engine 4.
- Let Me Go, es un videojuego en desarrollo de realidad virtual, de tipo Aventura, el cual se ejecuta en los dispositivos de Facebook, Oculus Quest.
- Ciudad Kolbi, es un videojuego para dispositivos móviles con variedad de minijuegos desarrollado para Kolbi, ICE.

1.2 Contexto del proyecto

Este producto se ha desarrollado por períodos intermitentes de trabajo, y ha estado a cargo de distintos equipos de desarrollo. Esto se debe a que el tipo de producto depende de tendencias particulares de la industria, y que al ser un proyecto interno no siempre se cuenta con los recursos o el personal para desarrollarlo. Se escogió este momento del año y la oportunidad del proyecto de práctica para darle continuidad con la intención de que el juego tenga un *publisher* (entidad/socio comercial encargado de la distribución del videojuego) para fin de año.

“Let Me Go” es un videojuego de realidad virtual ambientado en la perspectiva y la mente de una persona que ha perdido sus recuerdos, es un juego surrealista, el videojuego está planteado en primera persona para la respectiva inmersión de la realidad virtual. El mismo cuenta con una gran variedad de mecánicas(son los componentes del juego o esquema lúdico, es decir, las piezas y sus conexiones que permiten el funcionamientos de este) asociadas a un reproductor de cassettes que porta el jugador, el cuál según la música que reproduzca puede generar diferentes reacciones en el entorno que le rodea, dentro de lo que cabe mencionar, la música juega un rol muy importante dentro del juego por lo que podría clasificarse incluso como un juego rítmico(el juego posee una estética muy musical, las estéticas de un juego tiene que ver con la emoción que se pretende generar al jugador) y se espera que varias de las mecánicas interactúen en conjunto con la música. El juego se clasifica como de aventura, exploración, rítmico y role playing game, el cual cuenta con escenarios tanto para interiores como exteriores, resolución de conflictos en el mundo abierto y resolución de rompecabezas lógicos dentro ciertas zonas del juego como pueden ser las mazmorras.

Dentro de las mecánicas principales del juego que ya existen se encuentran:

- Un sistema de inventario que está en proceso de desarrollo.
- Un guardado temporal del proceso de los niveles.
- Varias mecánicas asociadas a los cassettes.
    - Obtener y reproducir las canciones de los cassettes en el reproductor.
    - Retroceder en el tiempo un espacio.
    - Guardar objetos del entorno en un inventario.
- Diferentes entornos 3D para que el jugador pueda explorar.
- Una mecánica de teletransportación para permitir el movimiento del usuario en el entorno 3D.
- Sistemas de diálogos.
- Sistemas de subtítulos.
- Control de flujo del videojuego.
- Interacción con objetos de múltiples partes.
- Menús e interfaces.
- Una introducción de varios minutos al entorno y las principales mecánicas del juego.
- Las mecánicas de movimiento de la cámara del jugador, el movimiento de las manos en realidad virtual, la lógica para interactuar con una variedad de objetos.

Toda la lógica del juego se está construyendo en Unity3D, utilizando las herramientas de desarrollo del SDK de Oculus, de Facebook, para enfocar el juego hacia un público meta de usuarios de este tipo de visores de realidad virtual (Oculus Quest y Oculus Quest 2). Además de que la optimización del juego está pensada para la arquitectura android y capacidad de estos dispositivos.

1.3 Problema

Dentro del diseño del producto se encuentra la interacción con enemigos en el entorno, estas funcionalidades no existen. Todo el diseño del flujo de combate consiste en la interacción con enemigos en el entorno principal, combates individuales con estos siguiendo las estéticas rítmicas del juego. Todo esto como adición principal del proyecto para generar un producto mínimo viable para luego presentar el juego a posibles socios comerciales. (Nuevo)

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar un sistema de software en Unity3D que contenga la lógica de las mecánicas de combate rítmico para el videojuego de realidad virtual “Let Me Go”.

1.4.2 Objetivos específicos

1 Definir los requerimientos funcionales para un sistema de software que contenga la lógica de las mecánicas de combate rítmico para el videojuego de realidad virtual “Let Me Go”.

2 Diseñar un sistema de software que contenga la lógica de las mecánicas de combate rítmico para el videojuego de realidad virtual “Let Me Go”.

3 Desarrollar el sistema de software que contenga la lógica de las mecánicas de combate rítmico para el videojuego de realidad virtual “Let Me Go”.

4 Evaluar la funcionalidad y usabilidad del sistema de software que contenga la lógica de las mecánicas de combate rítmico para el videojuego de realidad virtual “Let Me Go”.

1.5 Justificación

El mercado de videojuegos en Costa Rica es un área que ha tenido pocos impulsores y muy marcados, siendo Headless Chicken Games una de las empresas de desarrollo de videojuegos con una trayectoria destacada. La empresa cuenta con variedad de proyectos de outsourcing, como también videojuegos de propiedad intelectual, lo cual es evidencia de un arduo esfuerzo y gran experiencia tanto en gestión del negocio como en desarrollo de software en sí.

“Let Me Go” es uno de los juegos de propiedad intelectual de la empresa, por lo que su planificación, producción y distribución está enteramente en manos de la compañía. Al aportar un incremento a este producto, su valor no solo aumenta, sino que se espera ofrezca realmente un atractivo a los posibles jugadores e inversionistas. Aprovechando el auge reciente de los videojuegos en realidad virtual, esta no es una oportunidad desatendida por la empresa, que espera lanzar su producto con temática de aventura para este mercado, en el país hay estudios y compañías que también han lanzado propuestas de negocios o videojuegos en realidad virtual, siendo que en el mismo país existen varios grupos y medios de distribución y/o promoción para contenido de videojuegos, como también existe un grupo considerable de posibles usuarios, aficionados de la realidad virtual.

Las propuestas “inmersivas” tanto para negocios, ocio o entretenimiento, están en su apogeo debido a la reciente puesta en el mercado de distintos dispositivos y experiencias de realidad virtual en general, por lo que, que un estudio costarricense manufacture este tipo de productos o experiencias, hace que sean proyectos altamente viables.

1.6 Cronograma de trabajo

La práctica se llevará a cabo a partir del 19 de julio al 12 de noviembre de 2021. Trabajo a tiempo completo de lunes a viernes, 40 horas por semana, con condición de horario flexible, hora de entrada entre 8 am y 12 md, 1 hora de almuerzo al día. Se pretende que el trabajo esté distribuido entre semana 0 y semana 17, un desglose más detallado de las tareas y el plan original se encuentra en el Anexo 6.

Resumen de distribución del cronograma:

- Semana 0 y Semana 1: Capacitación incluyendo introducción al proyecto y comprensión del código.
- Semana 2 y Semana 4: Diseño y prototipo de mecánicas de ataque y mecánicas de evasión, huida y defensa.
- Semanas 4 - 11: Diseño y prototipado de todo el combate con un enemigo (Lobos).
- Semana 12 y Semana 13: Integrar el combate sin cortes.
- Semana 14 - 17: Diseño y prototipado de enemigos de rango.

Capítulo II

1 Revisión de literatura

2.1 Marco teórico

Dentro del marco contextual del proyecto, Let Me Go posee su propio *Game Design Document* (GDD), lo cual es una práctica muy común en las empresas de desarrollo de videojuegos para llevar la trazabilidad de un proyecto y anotar en este documento toda la información fundamental del proyecto desde su etapa de diseño y conceptualización, como también elementos de sus etapas finales como la distribución del mismo.

El juego toma lugar en la mente de un anciano, que trata de entender qué es lo que le ocurre. Abel está teniendo un derrame, y en su mente aún se castiga por la muerte de su nieto (Monge, 2018). Siendo que bajo el contexto brindado se deduce que el ambiente del juego se desarrolla dentro de la mente de Abel, y narra la historia de un conflicto interno donde el personaje debe aprender a perdonarse en los últimos momentos de su vida.

2.1.1 Unity 3D

Entiéndase Unity3D como el motor gráfico para el desarrollo de videojuegos y aplicaciones interactivas enfocados en el manejo de objetos digitales tridimensionales, o que también permite realizar aplicaciones y videojuegos en 2D y lanzar estos en múltiples plataformas. Al manejarse un juego de realidad virtual que utiliza enteramente objetos y escenarios 3D.

Unity permite el desarrollo de escenarios y niveles mediante *scenes* (escenas), la cual contiene un apartado llamado *hierarchy* (jerarquía), el cual permite organizar y acomodar los diferentes objetos dentro de una escena.

Cada objeto dentro de la jerarquía posee un componente principal llamado *transform* el cual contiene la información esencial de cada objeto como su nombre, posición, rotación y escala, cada *gameObject* (objeto del juego) posee este componente.

A los diferentes objetos dentro del juego se les puede agregar otros componentes como puede ser el *mesh renderer* el cual es el que se encarga de representar el modelo 3D de los objetos. Elementos como el *Rigidbody* se agregar para agregar masa, gravedad y otras físicas. Finalmente a los objetos en escena se les añaden elementos de tipo *script* que son clases de C# que se extienden de una clase llamada *MonoBehavior* que es nativa del

motor y permite pegar estos *scripts* a los objetos dentro de la escena para añadir las diferentes lógicas de programación y algoritmos.

A través de los *scripts* se puede acceder a nivel de código a valores de los otros componentes de un *gameObject* y realizar modificaciones en estos, como cambiar la posición de un *transform,* activar o desactivar la gravedad de un objeto, también los scripts poseen sus propios métodos para detectar eventos en el espacio 3D como lo son los eventos *trigger* o los eventos *collision,* a los cuales también se les puede integrar parte de la funcionalidad y lógica de muchas mecánicas.

2.2 Trabajos relacionados

El término *diegetic music* como “categoría de música de videojuegos”, está directamente ligado con la capacidad de los juegos para contar historias. Es una tendencia humana discernible el querer añadir un valor narrativo a cualquier fenómeno abstracto inerte (Sexton, 2007). Dicho esto, la música da un significado a estas historias, ya sea confirmando el mensaje visual, o resolviendo ambigüedades en el mensaje principal. El uso de la música correcta en los momentos de angustia o de epifanía del jugador, permite alcanzar una experiencia humana más profunda y una narrativa mejor hilada. Este fundamento se puede aplicar a Let Me Go ya que en cierto punto del GDD se menciona que las canciones mágicas que permiten interactuar con el entorno del juego, al sonar juntas se desea transmitir un deseo de esperanza para el jugador.

Capítulo III

3 Solución planteada

3.1 Propuesta

La propuesta del proyecto está compuesto en varias etapas, dentro de las cuales se mencionan y detallan.

- Instalación del proyecto, configuraciones básicas y esenciales del proyecto de Unity3D, en la versión 2021.1.12f1 del motor de juegos, dentro de esta etapa se incluye la instalación de la versión más reciente del motor de juegos y las primera pruebas del sistema.
    - Se incluye un proceso de guiado de introducción y comprensión del código que ya existe, como también de prueba de las mecánicas con las que ya cuenta el juego.
- Diseño, prototipado y desarrollo de la mecánica principal del módulo de combate.
    - Diseño en conjunto con el director y desarrolladores de la mecánica que se pretende agregar, al no ser algo tangible y escrito en piedra este “módulo de combate” va a ir teniendo incrementos, pruebas, observaciones y cambios durante el desarrollo de todo el proyecto.
    - Se diseñan y se prototipa varios conceptos de los diferentes tipos de combate y de ataques según la estética rítmica y según el tipo de enemigo.
- Diseño y prototipado de enemigos para combate cuerpo a cuerpo.
    - Se plantea el desarrollo y diseño de dos a tres tipos de enemigos cuyo rango de efecto sea a corto alcance o cuerpo a cuerpo (un ejemplo de esto podría ser encontrar lobos en un bosque).
    - Se debe integrar este conjunto de enemigos con el sistema de combate base y con el resto del juego.

3.2 Involucrados

Headless Chicken Games es una empresa de desarrollo que cuenta con una variedad de proyectos, algunos de estos para clientes externos, por lo que el alcance de estos y la confidencialidad son un valor profesional importante, es común que la empresa maneje un acuerdo de no divulgación, asociado a que la persona puede saber cosas de otros proyectos que no deben mencionarse al público, por lo que igual es importante contar con este documento durante la práctica. Parte del proceso de diseño y toma de decisiones va de la mano del director del proyecto, en este caso Jose Pablo Monge, que también es productor, dueño y principal inversor del producto.

Dentro de los principales stakeholders del proyecto están:

- Jose Pablo Monge: Director del proyecto
- Jose Andrés Chinchilla: Desarrollador senior de videojuegos.
- Pedro Bastos: Productor suplente.
- Leonardo Lizano: Desarrollador de Unity3D y VR.
- Oscar Víquez: Profesor asesor encargado del seguimiento.

3.3 Procedimiento metodológico

El flujo de trabajo de la empresa en general se maneja con metodologías ágiles, se realiza el manejo de sprints tipo scrum, se realizan reuniones diarias, en los cuales se promueve mucho la comunicación, el consenso, el apoyo y el consejo. Parte del procedimiento de diseño y prototipado va acompañado de pruebas de usabilidad, como parte de la retroalimentación del desarrollo del producto, el criterio de evaluación de integración consiste en que todas las funcionalidades nuevas que se agreguen no afecten ninguna funcionalidad anterior del producto, cada desarrollador está encargado de probar y pulir las mecánicas que le estén asignadas antes de las revisiones de cada sprint, como también otros miembros de la empresa prueban el producto después de cada etapa de desarrollo.

Tabla 1: Matriz para el procedimiento metodológico

Objetivo Específico	Tarea	Meta	Indicador
Definir los requerimientos funcionales para un	A. Participar en las reuniones de diseño de las	A. Lista del conjunto de tareas con sus respectivas	A. Aprobación del director sobre la lista del conjunto de

sistema de software que contenga la lógica de las mecánicas de combate rítmico para el videojuego de realidad virtual “Let Me Go”.	funcionalidades del sistema. B. Analizar las necesidades del proyecto y descomponer la propuesta del módulo de combate en tareas. C. Definir un orden secuencial que respete la prioridad de los deberes e interdependencias. D. Redactar una lista de tareas o *backlog* como base para iniciar la etapa de diseño/desarrollo.	dependencias. B. Distribuir las tareas en los diferentes sprints según su complejidad y prioridades.	tareas con sus respectivas dependencias. B. Cronograma de la lista de actividades distribuida en los meses del período de práctica aprobado por el director.
Diseñar un sistema de software que contenga la lógica de las mecánicas de combate rítmico para el videojuego de realidad virtual “Let Me Go”.	A. Diseñar las mecánicas de combate rítmico base para el juego. B. Diseñar las mecánicas de ataque del jugador. C. Diseñar las mecánicas de evasión del jugador. D. Diseñar las mecánicas de ataque a distancia del jugador. E. Diseñar las mecánicas de ataque a distancia del jugador. F. Diseñar el concepto de dos a tres enemigos.	A. Generar un concepto base claro sobre la mecánica de combate base. B. Definir las formas de ataque a corta distancia del jugador. C. Definir las formas de defensa estimadas para el jugador. D. Definir las formas de evasión estimadas para el jugador. E. Definir las formas de atacar a distancia estimadas para el jugador. F. Definir el concepto de los tipos de enemigos que se van a integrar con el módulo de combate.	A. Documento de *gamedesign* del módulo de combate, que describa las mecánicas de ataque, defensa, evasión y ataques a distancia del jugador, aprobado por el director. B. Documento de *gamedesign* con la explicación de los conceptos, mecánicas y estéticas de los enemigos, aprobado por el director.
Desarrollar el sistema de software que contenga la lógica de las mecánicas de combate rítmico	A. Desarrollar la funcionalidad del módulo de combate base. B. Desarrollar la funcionalidad de la	A. Generar un sistema de combate rítmico mantenible y extensible. B. Generar mecánicas de	A. Código fuente y archivos del sistema de combate del juego, evaluado por el director. B. Código fuente y

para el videojuego de realidad virtual “Let Me Go”.	mecánica de ataque. C. Desarrollar la funcionalidad de la mecánica de defensa. D. Desarrollar la funcionalidad de la mecánica de evasión. E. Desarrollar la funcionalidad de la mecánica de ataque a distancia. F. Desarrollar las funcionalidades y mecánicas de los enemigos diseñados previamente.	ataque reutilizables en diferentes contextos. C. Generar mecánicas de defensa reutilizables en diferentes contextos. D. Generar mecánicas de evasión reutilizables en diferentes contextos. E. Generar mecánicas de ataques a distancia reutilizables en diferentes contextos. F. Generar un conjunto de *prefabs* (objetos idénticos reutilizables dentro del juego) de los enemigos.	archivos utilizados en las mecánicas de ataque, defensa, evasión del módulo de combate, evaluado por el director. C. Código fuente, archivos base y archivos *prefabs* de los enemigos para su uso constante dentro del juego, evaluado por el director.
Evaluar la funcionalidad y usabilidad del sistema de software que contenga la lógica de las mecánicas de combate rítmico para el videojuego de realidad virtual “Let Me Go”.	A. Evaluar las funcionalidades de combate cuerpo a cuerpo del módulo de combate. B. Evaluar las funcionalidades de combate a distancia del módulo de combate. C. Evaluar las funcionalidades de combate con los enemigos desarrollados.	A. Probar la usabilidad de las mecánicas de combate cuerpo a cuerpo. B. Probar la usabilidad de las mecánicas de combate a distancia. C. Probar la usabilidad durante la interacción con enemigos en combate.	A. Documentación y retroalimentación de las pruebas de combate cuerpo a cuerpo, aprobado por el director del proyecto. B. Documentación y retroalimentación de las pruebas de combate a distancia, aprobado por el director del proyecto. C. Documentación y retroalimentación de las mecánicas de combate de los enemigos, aprobado por el director del proyecto.

Fuente: Elaboración propia

3.5 Análisis de riesgos

El análisis de las diferentes mecánicas a desarrollar genera la siguiente lista de riesgos que pueden afectar la usabilidad o integridad del sistema.

Documentación de riesgos

Referencia	Objetivo específico 2
Código	R-001
Nombre	El jugador se aburra
Categoría	Error de diseño
Causa	Alguna de las mecánicas o funcionalidades implementadas aburre al jugador y este quiere dejar de jugar.
Impacto	El jugador abandona el juego por culpa de una de las mecánicas implementadas.
Estrategia de evasión	Revisiones periódicas por parte del desarrollador, diseñador y otros miembros del equipo para retroalimentación.
Estrategia de mitigación	Incluir enemigos y mecánicas de combate dinámicas que no hagan que el juego sea repetitivo o pesado.
Estrategia de contingencia	Si a las personas que prueban los prototipos definitivamente les disgusta una mecánica, el diseñador puede optar por eliminarla del alcance.

Referencia	Objetivo específico 4
Código	R-002
Nombre	Que el jugador se quede bloqueado
Categoría	Defectos
Causa	Alguna de las mecánicas está mal programada, de forma que hagan que el jugador se quede atascado en un lugar sin poder hacer nada.
Impacto	Se arruina la experiencia del juego y genera frustración.
Estrategia de evasión	Las revisiones periódicas y pruebas de usabilidad tienen como objetivo generar retroalimentación y detectar defectos que puedan hacer que el sistema esté comprometido.
Estrategia de mitigación	El jugador debe poseer los medios para poder abandonar o reiniciar el juego en cualquier momento.
Estrategia de contingencia	El desarrollador debe hacer pruebas intensivas al sistema con la intención de que no quede ningún error de este tipo en el producto

	final.

Referencia	Objetivo específico 2
Código	R-003
Nombre	El jugador se desorienta
Categoría	Error de diseño
Causa	Los objetivos, mecánicas o peligros dentro del juego son poco intuitivos y el jugador no sabe qué hacer.
Impacto	Arruina la experiencia de juego y genera frustración, además de desorientación, detiene el flujo del juego
Estrategia de evasión	Las pruebas de diseño y de usabilidad ayudan a generar retroalimentación sobre las mecánicas desarrolladas.
Estrategia de mitigación	El desarrollador puede agregar funcionalidades o mensajes de UI/UX para orientar al jugador, no todo debe ser indicado de forma escrita.
Estrategia de contingencia	El flujo completo de todas las mecánicas debe ser probado varias veces por varias personas para probar que el sistema sea utilizable y fácil de usar. El desarrollador debe asegurarse de que todas las mecánicas funcionan de forma correcta y sin comprometer la integridad del sistema.

Referencia	Objetivos específicos 3 y 4
Código	R-004
Nombre	El jugador rompe el juego
Categoría	Personas
Causa	El jugador usa comportamientos raros o “fuerza bruta” para romper la lógica del juego para obtener algún beneficio o comprometiendo el flujo del juego
Impacto	Arruina o compromete la experiencia del juego.
Estrategia de evasión	El desarrollador y personas que prueban el sistema pueden encontrar este tipo de errores de desarrollo fatales durante la etapa de desarrollo.
Estrategia de mitigación	En caso de aparecer un error de este tipo ya lanzado el producto, se debe agregar la información del error a la lista de *bugs* para arreglarlo y lanzar una versión actualizada del producto.

Estrategia de contingencia	El desarrollador debe asegurarse de que no existan aperturas para generar este tipo de errores que comprometan la funcionalidad del sistema.

Referencia	Objetivo específico 2
Código	R-005
Nombre	Prototipo no aprobado
Categoría	Error de diseño
Causa	Se desarrolla una funcionalidad en base a un requerimiento del diseñador y durante la revisión el diseñador no aprueba el prototipo.
Impacto	Atrasos en el desarrollo del producto, disgusto por parte del equipo de diseño o de desarrollo y re trabajo.
Estrategia de evasión	No asumir las tareas, durante los sprint planning debe haber comunicación clara para que lo que se desarrolle sea lo que el diseñador pide.
Estrategia de mitigación	El diseñador y los desarrolladores se reúnen frecuentemente o se envían avances en los canales de comunicación de la empresa para que se supervise el trabajo de los desarrolladores más de cerca.
Estrategia de contingencia	Si una mecánica se diseña, se desarrolla, se modifica e igualmente no se aprueba por el diseñador, este puede eliminarla del alcance del proyecto.

Referencia	Objetivo específico 1
Código	R-006
Nombre	Cambios drásticos en el alcance del proyecto
Categoría	Error de producción
Causa	Durante el desarrollo del sistema el cliente o productor decide cambiar los objetivos o mecánicas del proyecto.
Impacto	Atrasos con el proyecto Compromete la integridad del proyecto de práctica profesional
Estrategia de evasión	Se asume un compromiso por parte del estudiante y de la empresa en que las tareas que realiza el estudiante sean acorde a lo estipulado originalmente.
Estrategia de mitigación	Algunos requerimientos o mecánicas sí están sujetas a cambios repentinos durante el desarrollo siempre y cuando no se pretenda asignar un rol o conjunto de tareas totalmente diferente a lo acordado

	originalmente.
Estrategia de contingencia	En un caso muy extremo de que el cliente quiera cambiar totalmente el trabajo del estudiante, se puede dar una intervención por parte de la universidad.

Referencia	Objetivo específico 3
Código	R-007
Nombre	Falta de recursos para financiar el proyecto
Categoría	Económico
Causa	La empresa, el productor, el cliente o el principal inversor deciden detener el proyecto por falta de recursos económicos.
Impacto	Se congela el proyecto
Estrategia de evasión	Como el estudiante está atravesando un proceso de práctica profesional, no se considera como un empleado de la empresa ni tampoco es estrictamente necesario un pago de salario.
Estrategia de mitigación	Existe un compromiso por parte del estudiante y de la empresa sobre el proyecto que se estipuló originalmente que va a realizar el estudiante, la inversión económica se deja en manos de la empresa.
Estrategia de contingencia	En caso de que la empresa no permita al estudiante continuar con el desarrollo, puede realizarse una intervención por parte de la universidad.

Capítulo IV

4 Requerimientos y diseño

4.1 Definición de los requerimientos

El proceso de toma de la información en lo que respecta al proyecto en curso del videojuego “Let Me Go”, se basa en la comunicación directa y constante con el dueño, director y principal inversor del producto de software, Jose Pablo Monge, en conjunto con trabajos multidisciplinarios donde se ven involucradas áreas artísticas como diseño visual o musical, el equipo de desarrollo y el ya mencionado director del proyecto.

Dentro de las tareas del estudiante practicante se encuentra la asistencia continua a reuniones diarias para verificar el progreso del proyecto, como también las reuniones donde se analiza en retrospectiva lo realizado, como también el planeamiento de cada período de iteración y la asignación y descripción de las tareas, al utilizarse metodología ágil existe la posibilidad de que lo presentado en el prototipo final difiera levemente con la idea original descrita, como también se agreguen o se eliminen funcionalidades si no son aprobadas por el director del proyecto.

A continuación se detallan los requerimientos del sistema.

4.1.1 Diagrama de Casos de uso

Con el análisis de las peticiones del productor y director del proyecto, se extraen los siguientes casos de uso.

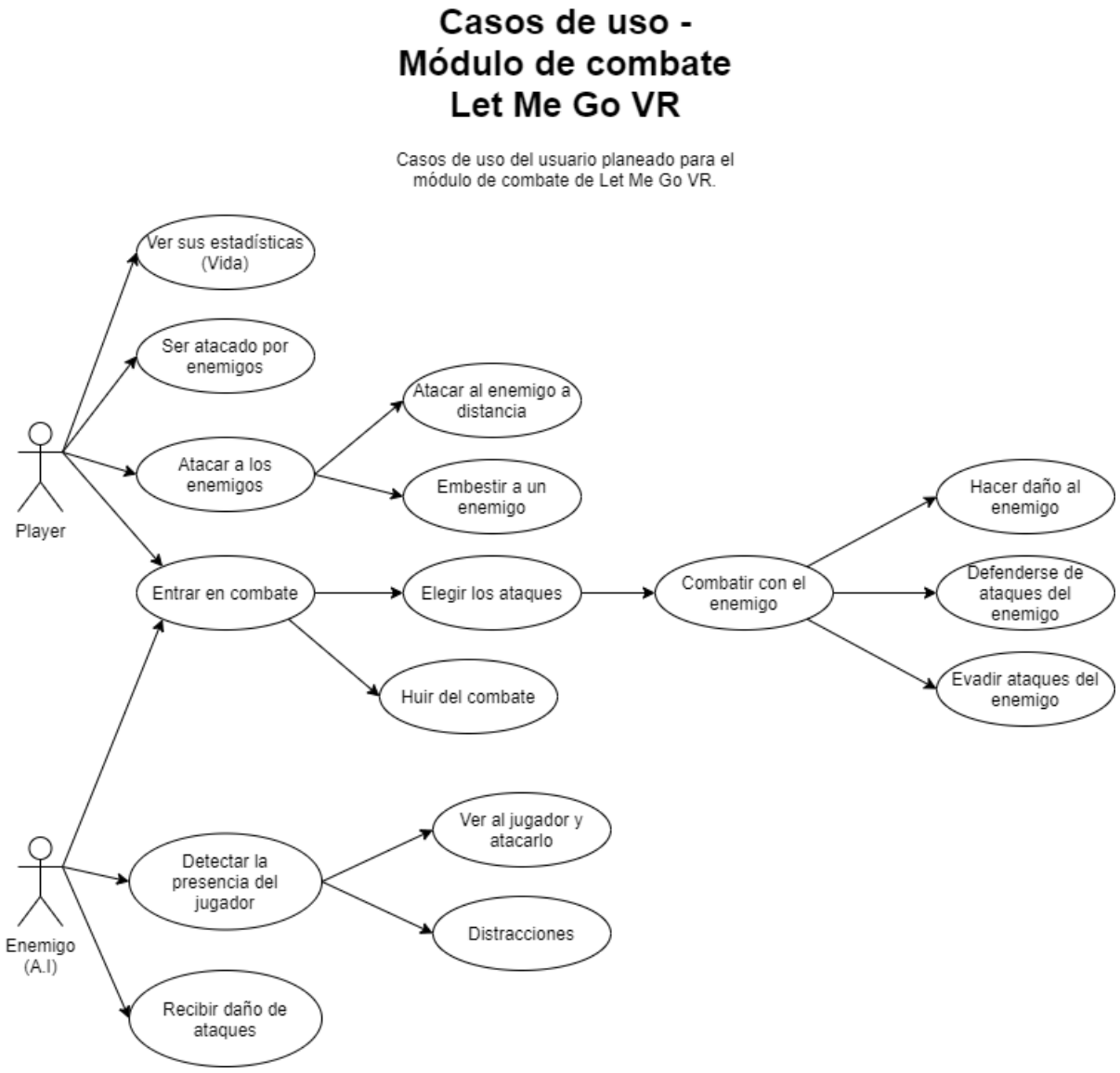

Figura 1: Diagramas de caso de uso del sistema del módulo de combate.

Dentro de las funcionalidades más explícitas que se solicitan están: Que los enemigos tengan su propia inteligencia artificial básica que les permita moverse y detectar a un jugador en el espacio, el enemigo debe poder distraerse con objetos arrojables, como también debe perseguir al jugador para atacar cuando lo detecta, cuando inicia un combate el enemigo puede recibir y lanzar ataques y lleva sus propias estadísticas aún por definir.

El jugador puede ver sus propias estadísticas, puede ser atacado por los enemigos o atacar a los enemigos, ya sea entrando en combate directo, lanzar objetos que hagan daño o embestir a un enemigo por sorpresa. En los combates se planea que el jugador pueda elegir diferentes secuencias de ataques, el jugador debe poder defenderse de algunos ataques o evadirlos, en cualquier momento se puede dar vuelta e intentar huir del combate.

Los ataques o respuestas que realiza el jugador debe reflejar un daño sobre el enemigo. Las mecánicas se detallan más en el diagrama de arquitectura.

4.1.2 Especificación de casos de uso

Siendo las prioridades: A = Alta, M = Media, B = Baja

Documentación de requerimientos

Código	Nombre	Descripción	Prioridad
Módulo de combate del videojuego Let Me Go VR			
RF-01	***Enemies Patrol***	Los enemigos pueden moverse en un área definida por un rango de movimiento.	**A**
RF-02	***Enemies field of view***	Los enemigos poseen un rango de visión en el cual pueden detectar al jugador. Ver anexo 1.	**A**
RF-03	***Enemies distraction***	El jugador puede arrojar objetos que permitan distraer a los enemigos de su rumbo de patrullaje.	**A**
RF-04	***Enemy ambush***	Cuando un enemigo detecta al jugador, debe "correr" hacia este para detonar el evento de combate.	**A**
RF-05	***Enemy start combat***	Cuando un enemigo que va hacia el jugador está a cierta distancia de este, inicia el evento de combate. ● El jugador y el enemigo se llevan a una zona aparte donde se desarrolla el combate	**A**
RF-06	***Enemy attacking notification***	Al jugador se le debe avisar cuando un enemigo está corriendo hacia este.	**A**
RF-07	***Player run away from combat***	El jugador puede dar la espalda al enemigo y mediante un movimiento de manos simulando correr, se abandona el combate.	**A**
RF-08	***Immunity frames***	Cuando el jugador derrota a un enemigo o abandona un combate, vuelve a la zona principal, los demás enemigos en la zona principal ignoran la presencia del jugador durante unos segundos.	**A**
RF-09	***Combat boxes***	El jugador puede golpear cajas, que van hacia este sobre dos filas y en secuencias para realizar daño al enemigo con el que está combatiendo. Ver anexo 2.	**A**
RF-10	***Combat sequences***	El jugador puede elegir distintas secuencias de ataques de mayor o menor dificultad para aplicar	**A**

		distintos daños a los enemigos en combate.	
RF-11	***Enemy health***	El enemigo posee una estadística de vida que se va reduciendo según el desempeño del combate.	**A**
RF-12	***Player damage***	Según el tipo de combate que se implemente el jugador puede recibir daño de ataques en secuencia de los enemigos.	**A**
RF-13	***Player evasion***	El jugador puede evadir algunos ataques de los enemigos para que no se le refleje daño a su salud.	**A**
RF-14	***Player defense***	El jugador puede “sostener” algunas secuencias de ataques de los enemigos para defenderse de estos ataques.	**A**
RF-15	***Player Health***	El jugador puede ver su nivel de vida o salud, en un reloj en su mano.	**A**
RF-16	***Turn based combat***	Prototipar utilizando los *Combat boxes* un sistema de combate por turnos, donde el jugador y el enemigo toman turnos para atacar.	**A**
RF-17	***Active Time Battle Combat***	Prototipar utilizando los *Combat boxes* un sistema de combate que permita hacer ataques ligeros o pesados basado en tiempos de carga. Ver anexo 3.	**A**
RF-18	***Real time combat***	Prototipar utilizando los *Combat boxes* un sistema de combate que se asemeje a un combate en la vida real sin turnos, esperas, ni inmunidades, donde el jugador tenga que aprender los patrones entre ataque, evasión y defensa.	**A**
RF-19	***Player ambush***	El jugador puede emboscar por sorpresa a un enemigo que no lo esté mirando en ese momento y derrotarlo sin entrar en combate.	**A**
RF-20	***Ranged bow***	El jugador puede usar un arco para realizar ataques a distancia a los enemigos en la zona principal sin entrar en combate.	**A**
RF-21	***Melee enemies***	Prototipar enemigos con ataques de corto alcance para simular el combate cuerpo a cuerpo.	**A**
RF-22	***Range enemies***	Prototipar enemigos que puedan atacar al jugador desde una distancia considerable o que vuelan.	**A**

Código	Nombre	Descripción	Prioridad
Requerimientos no funcionales			

RNF-01	***Combat transition***	Existe un espacio o pausa de transición entre la zona principal e iniciar el combate con un enemigo.	**M**
RNF-02	***Combat area***	El combate en sí se desarrolla en un espacio aparte a la zona principal del juego, donde solo interactúan el jugador y el enemigo.	**A**
RNF-03	***Combat experience***	El jugador al desarrollar varios combates con el mismo tipo de enemigo puede desarrollar "experiencia" que le permita evitar u omitir un combate. Esto no es una estadística que el jugador pueda ver en todo momento.	**M**
RNF-04	***Integration***	Los cambios desarrollados o funcionalidades nuevas deben integrarse con la escena principal del juego sin afectar el funcionamiento de esta.	**A**

4.2 Diseño de la plataforma de software

Al utilizar metodologías ágiles existe una base de tareas de la cual partir para desarrollar las diferentes funcionalidades, al utilizar metodologías como *scrum* se genera un *backlog* el cual va a contener la variedad de tareas y durante los *sprint planning* se detallan más a fondo estas, por la naturaleza del desarrollo de videojuegos en Unity, para desarrollar e integrar las diferentes funcionalidades se debe desarrollar una cantidad incalculable de *scripts* (clases de C# que se extienden de la clase para objetos en el juego *Monobehaviour* de Unity 3D), los cuales se integran con los objetos en el entorno 3D para crear las diferentes funcionalidades, se detallan los diagramas más esenciales a continuación.

4.2.1 Arquitectura conceptual de la solución

Tras la redacción del alcance del proyecto se redacta el siguiente modelo de arquitectura conceptual de la solución. El mismo se encuentra dividido en partes y el diagrama completo se puede ver en el Anexo 5.

Figura 1: Inicio arquitectura conceptual.

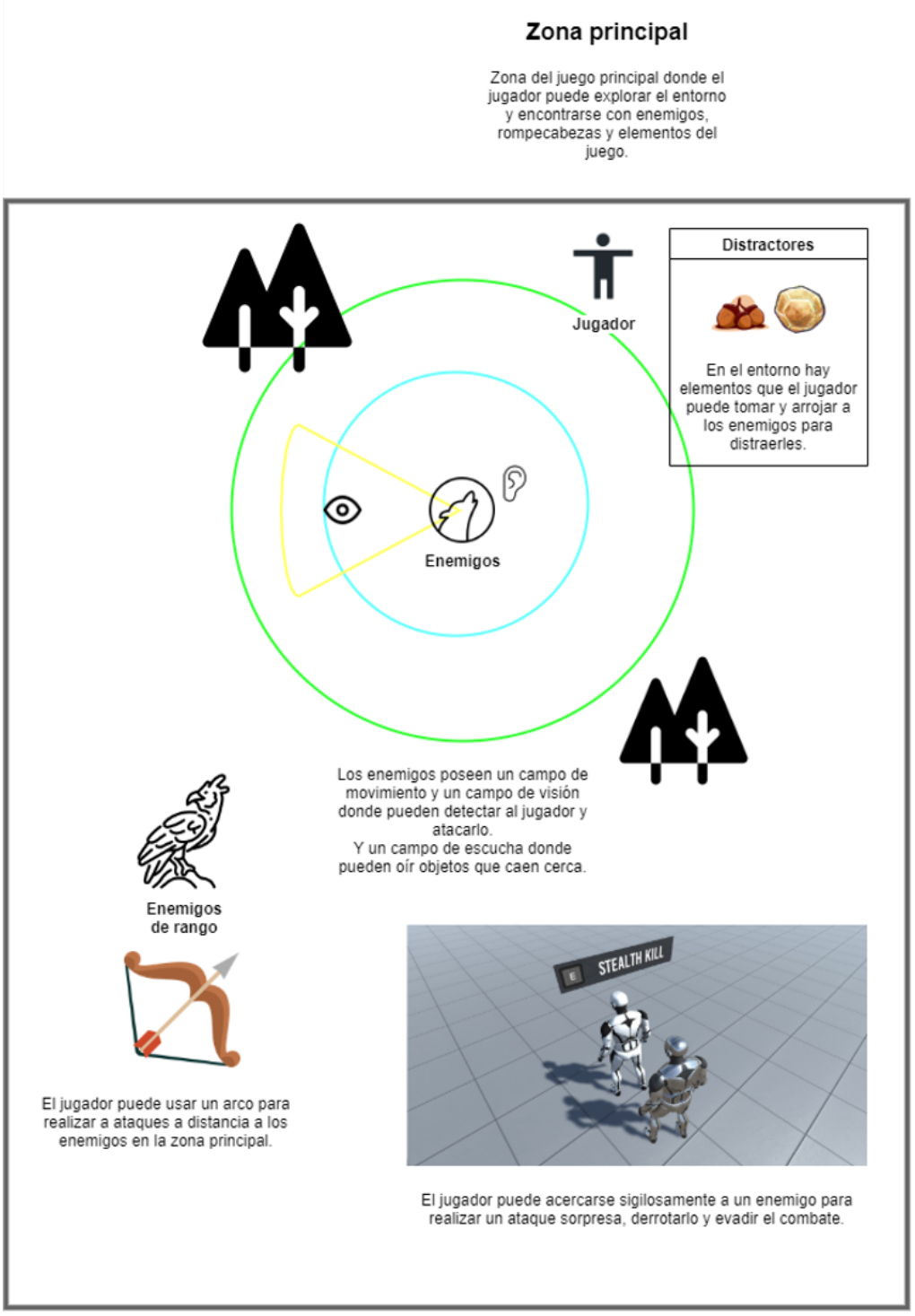

Figura 2: Apartado de zona principal, diagrama de arquitectura conceptual.

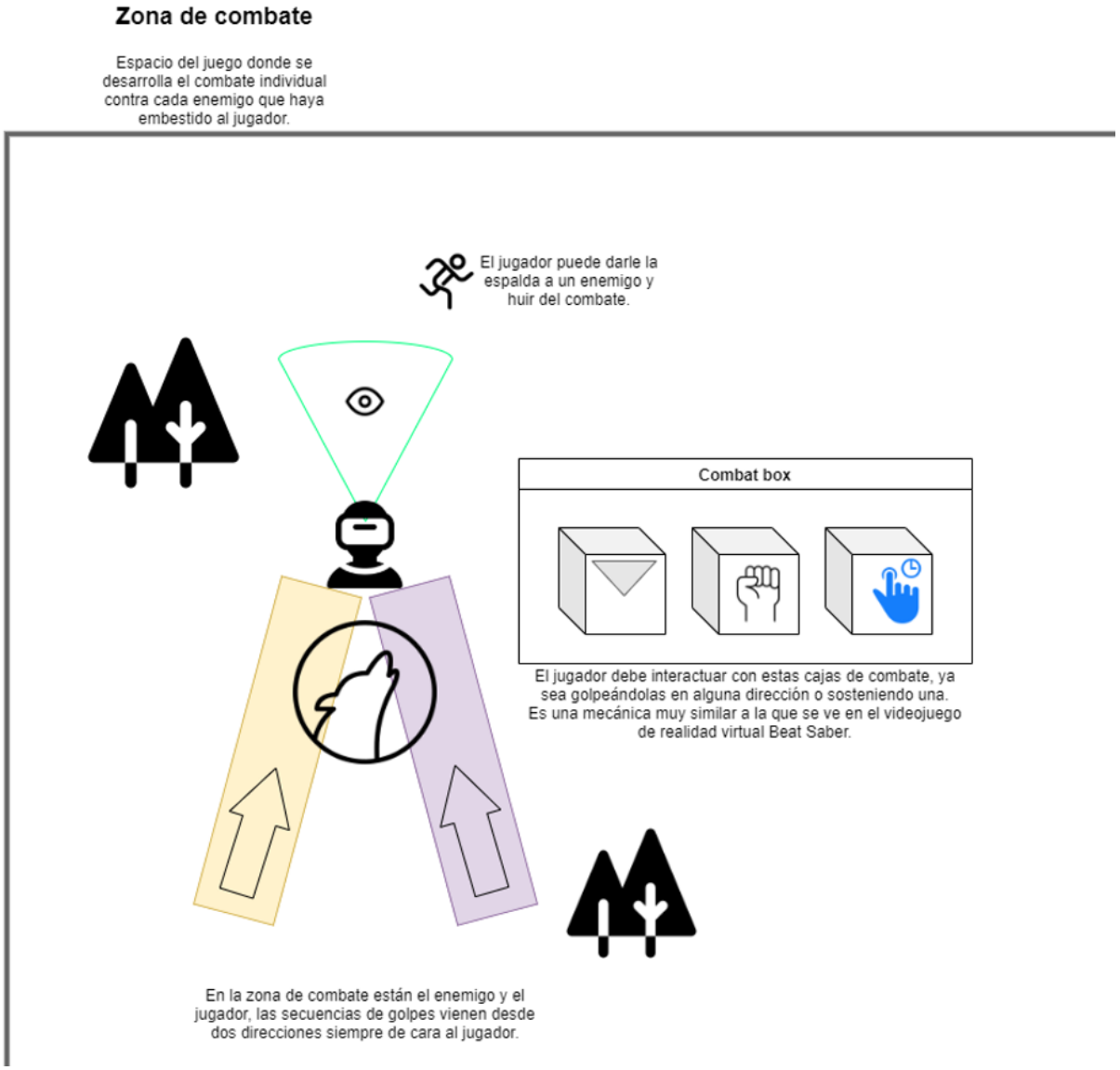

Figura 3: Descripción del combate, diagrama de arquitectura conceptual.

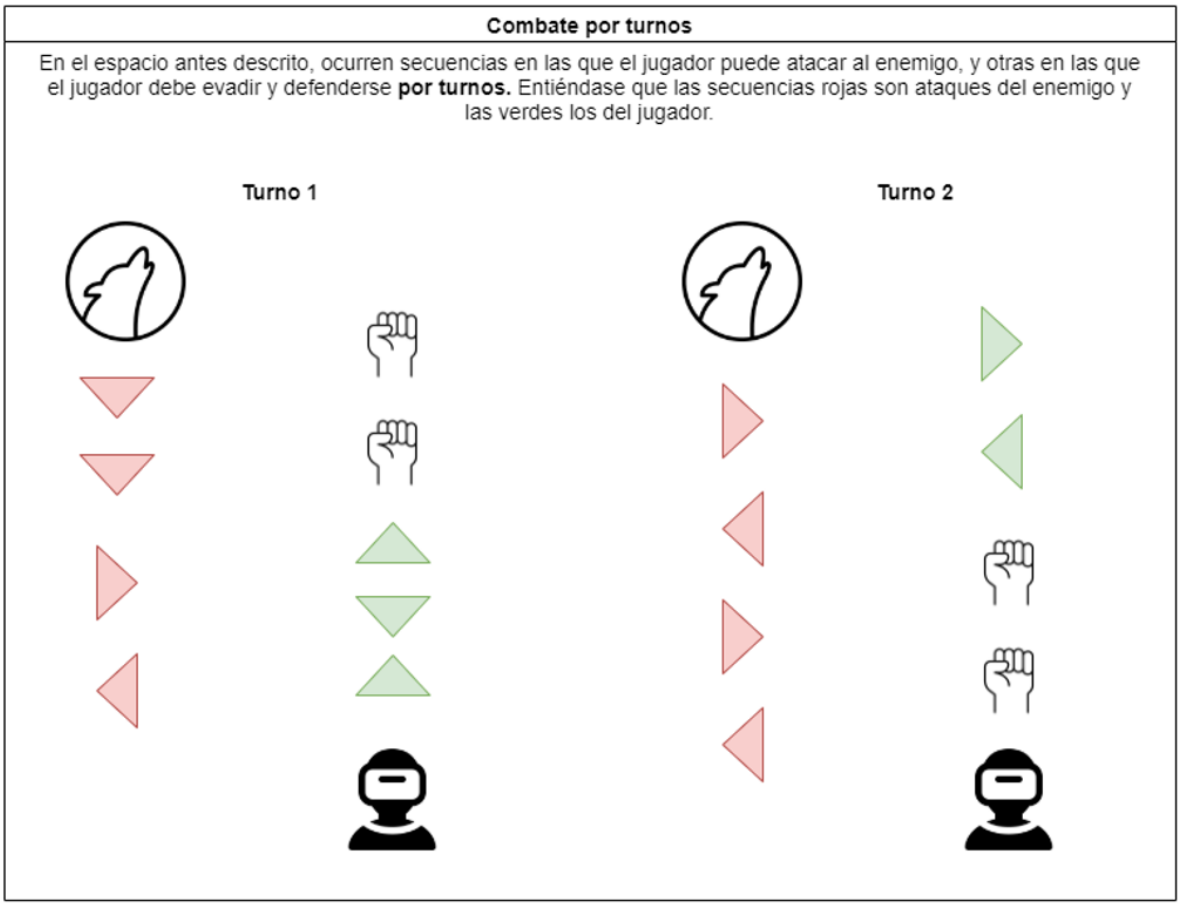

Figura 4: Descripción del combate por turnos, diagrama de arquitectura conceptual.

4.2.2 Interfaces de usuario

Dentro del ámbito de la realidad virtual, al ser completamente un entorno alterno diferente a estar sentado frente a una pantalla, existen diferentes formas de representar una interfaz de usuario dentro del entorno 3D. Debido a que pegar elementos visuales a la cámara en el espacio tridimensional que representa la vista del jugador, puede generar *motion sickness* (o cinetosis es una sensación de mareo, angustia y náuseas que se manifiesta en una persona después de un rato usando dispositivos de realidad virtual, como las gafas VR) lo cual es un problema muy común en las aplicaciones de VR, se opta por otras alternativas como poner los menús, opciones, estadísticas y elementos en el espacio global, u otra práctica funcional muy usada es pegar estos objetos pero no a la cara del usuario, sino en las manos de realidad virtual.

Figura 5: Interfaz de usuario con elementos de texto pegado a la mano de realidad virtual.

En la figura anterior, se muestra como se le representa al jugador la estadística de su vida con elementos de texto, de forma que no esté pegado a la cara y no genere ninguna molestia visual.

Figura 6: Menú principal del juego, en el espacio global.

En la figura del menú, se muestra cómo se interactúa con elementos de interfaz de usuario, en el espacio global, para interactuar con el menú se mueve la perilla del televisor. Esta interfaz el usuario la puede ver pero no está pegado a su cabeza, sino que se encuentra en el entorno 3D.

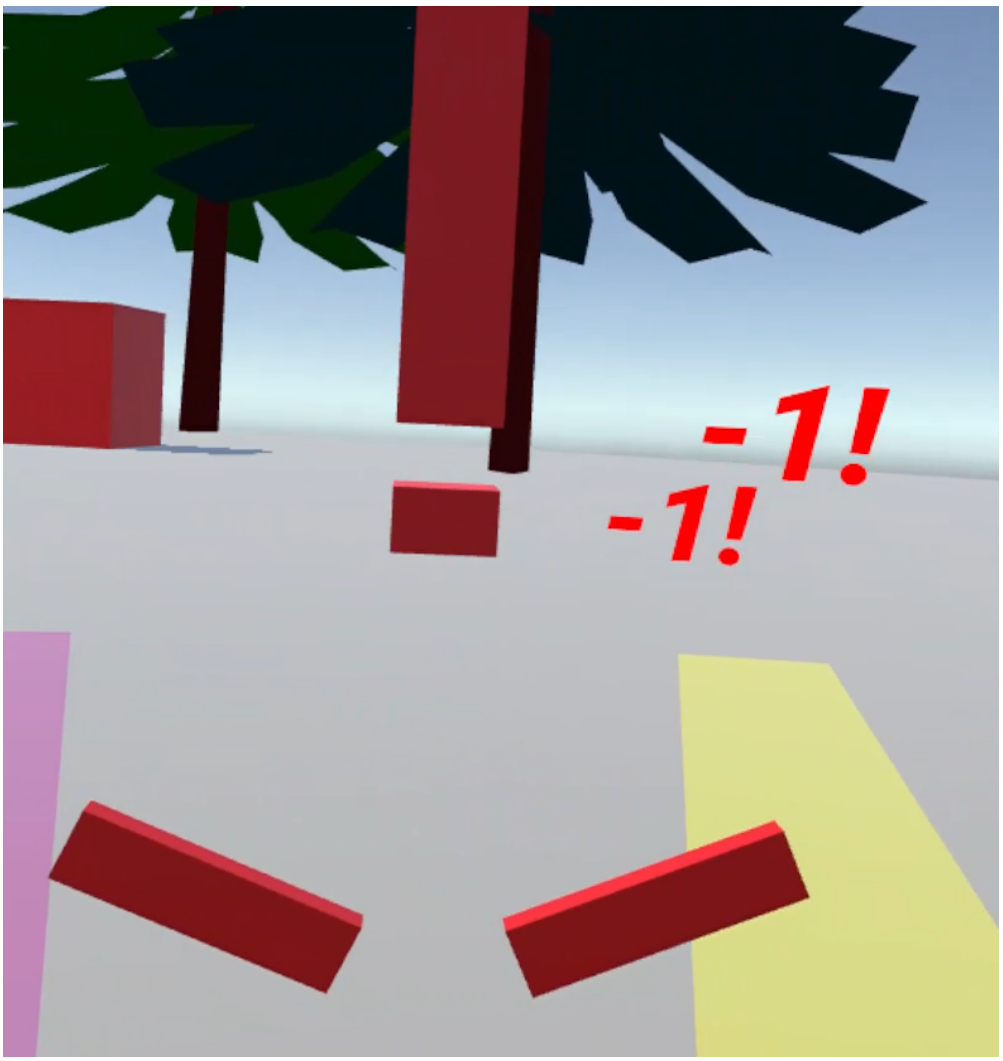

Figura 7: Representación del daño de los enemigos.

En esta figura se aprecia cómo se integran elementos de interfaz de usuario en el entorno 3D para indicarle al jugador que está aplicando daño al enemigo, estos mensajes de daño están unidos al enemigo en el espacio 3D, pero no al jugador.

Capítulo V

5 Plataforma de software

5.1 Aplicación

5.1.1 Introducción al desarrollo de la aplicación

5.1.1.1 Plataformas de desarrollo

El desarrollo de la aplicación se realiza con una computadora con las capacidades para ejecutar aplicaciones de realidad virtual pese a su alto coste computacional, y se planea lanzar para los dispositivos de Facebook de realidad virtual Oculus. El equipo también debe ser capaz de correr el motor de juegos Unity3D.

5.1.1.2 Oculus Rift S, Oculus Quest y Oculus Quest 2

La empresa cuenta con dispositivos Oculus Quest y Quest 2, con los cuales el encargado del proyecto y otro desarrollador pueden tanto programar, como ejecutar y probar las funcionalidades añadidas diseñadas y creadas por el estudiante, en este caso particular y en vista de lanzar el juego en plataformas de realidad virtual en PC, el estudiante cuenta con un visor Oculus Rift S, que funciona conectado a la PC que se encarga del procesado, mientras que los dispositivos Quest funcionan como *standalone* sin necesidad de estar conectados al computador.

5.1.1.3 Motor gráfico Unity3D

Al ser la aplicación un incremento, existe una base de código que el estudiante debe descargar y comprender para poder desarrollar sobre eso. El proyecto ya se estaba desarrollando en el motor gráfico y para videojuegos Unity3D, en la versión 2021.1.9f1, sobre la cual ya se había estado desarrollando este año antes de que el estudiante ingresara al proyecto.

5.1.1.4 NewtonVR Package

La integración de los módulos de realidad virtual básicos como son la cámara dentro del entorno 3D que se mueve con la orientación del visor, las manos que se manejan con los controles del visor de realidad virtual, y sistema básico para las físicas de realidad virtual (tomar, sujetar, empujar o golpear objetos 3D) que se implementa en el proyecto, es el de NewtonVR, cuya última versión de desarrollo es algo antigua y la actual se encuentra modificada por el equipo de la empresa. Este paquete permite un control de físicas dentro del entorno 3D más realistas a la hora de interactuar con los objetos.

El objeto NVRPlayer es la base para las aplicaciones de realidad virtual desarrolladas con este paquete ya que este es el que posee la cabeza y las manos del jugador dentro del entorno 3D y que se controlan con el visor de realidad virtual y mandos de este.

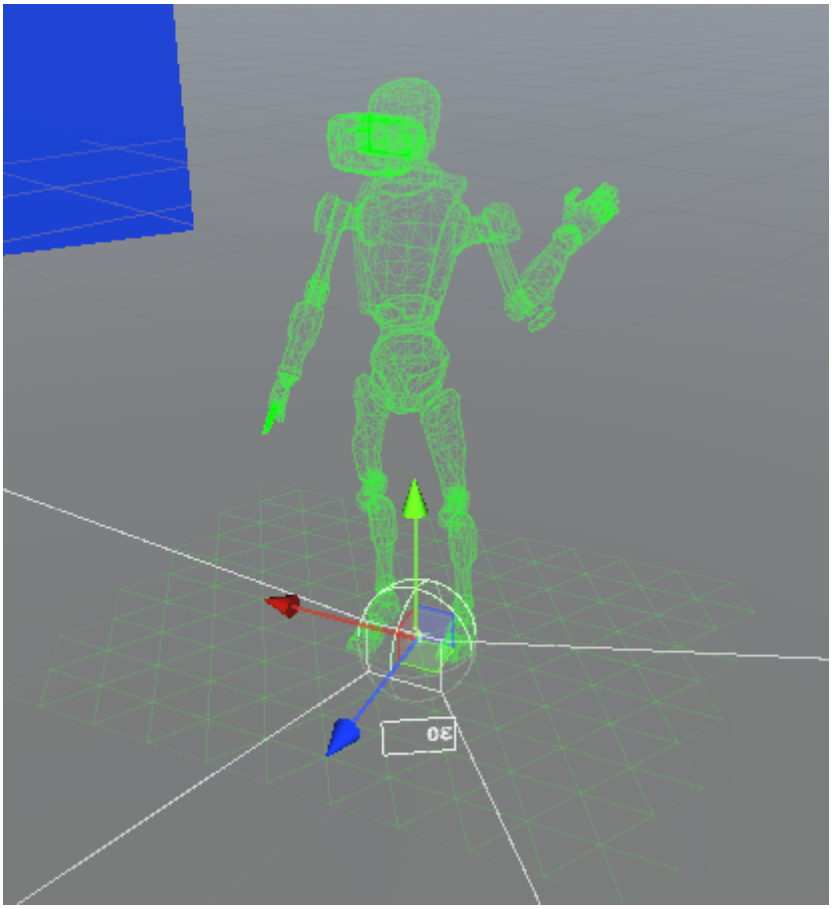

Figura 8: Objeto NVRPlayer en la escena de prototipado.

5.1.1.5 Scripts de C#

El motor Unity3D permite al desarrollador crear un conjunto de *scripts* de lenguaje C#, que son clases que se extienden de una clase base(*monobehavior*) que ya trae el motor y permite al usuario pegar estos *scripts* a los objetos en el entorno 3D para generar los diferentes comportamientos que se desea programar. Todas las funcionalidades se crean en base a estos *scripts* y otros componentes del motor como las físicas, colisiones, entre otros. Estas clases generadas por el desarrollador permiten la integración de programación orientada a objetos normal de C# y la integración de patrones de diseño.

5.1.2 Tareas realizadas para desarrollar la plataforma de software

5.1.2.1 Capacitación

Durante las primeras dos semanas se envía al practicante a capacitación de los motores gráficos, tanto Unity3D como de Unreal Engine, para adquirir conocimiento general

sobre las herramientas, siguiendo las guías oficiales para principiantes que existen en los portales web oficiales de estas mismas.

5.1.2.2 Definición de los requerimientos

Desde el período de entrevistas, reuniones y desarrollo del anteproyecto ya se han ido definiendo las diferentes funcionalidades que se desea agregar al videojuego. Al manejarse metodología ágil la empresa y el estudiante en conjunto llenan un backlog con los requerimientos principales o globales y en estos requerimientos se van llenando las condiciones de satisfacción conforme el estudiante va desarrollando estas funcionalidades.

5.1.2.3 Espacio de prototipado

La empresa brinda al estudiante libertad para integrar una variedad de *assets* para el juego según las necesidades de los requerimientos. Para no comprometer la integridad de la aplicación que ya existe el estudiante debe tomar lo esencial para el funcionamiento de la aplicación en realidad virtual y llevarlo a una escena propia para el debido prototipado de las diferentes funcionalidades.

La escena DevLeoPrototyping cuenta con los elementos esenciales para desarrollar las funcionalidades según lo que se ha estipulado. Dentro de sus componentes principales existe el objeto jugador NVRPlayer, que es la unidad básica para aplicaciones de realidad virtual que incluye el paquete NewtonVR, un contenedor con los enemigos en “la escena principal” y un contenedor con un espacio específicamente dedicado a los enemigos de combate. Cada enemigo en movimiento tiene ligado un enemigo en combate, para luchar el jugador es transportado a otra zona únicamente dedicada al combate.

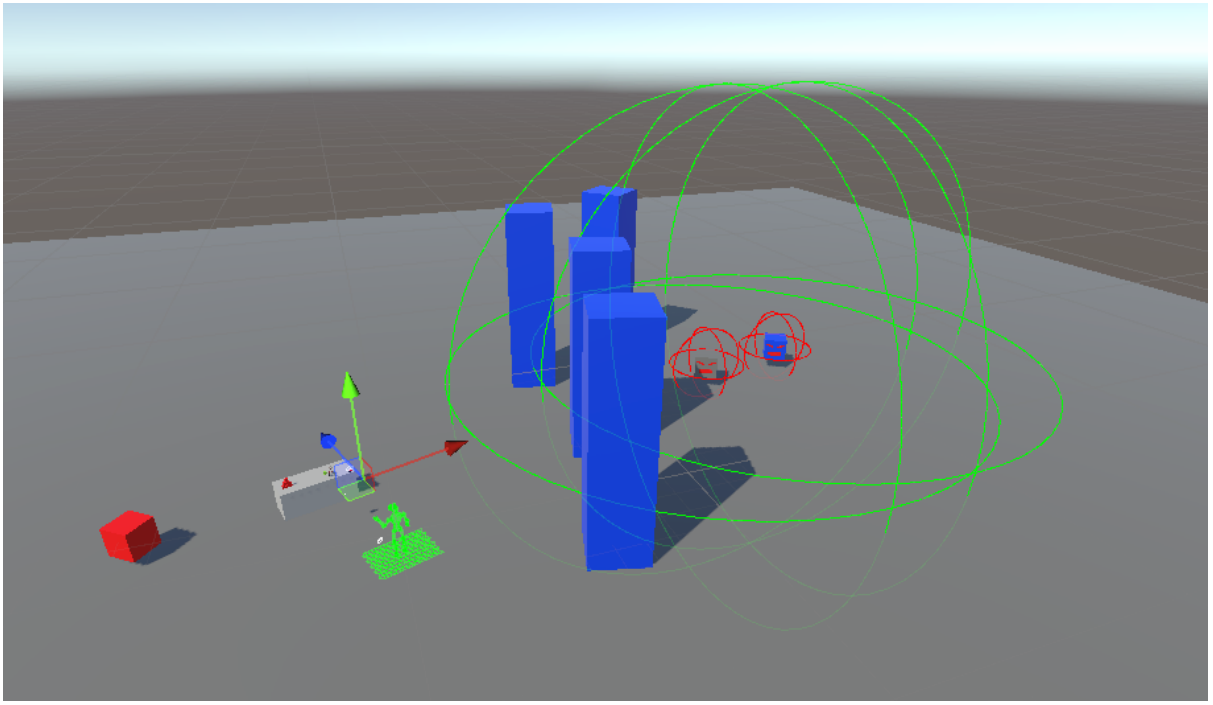

Figura 9: Escena de prototipado del estudiante.

Como se puede apreciar en la figura anterior, la escena de trabajo del estudiante no cuenta con el arte de la escena principal, ya que el único propósito de la misma es crear y probar las funcionalidades desarrolladas antes de integrar estas con la escena principal.

5.1.2.4 Enemigos en movimiento

Los enemigos en la zona principal se desarrollan ignorando las cuestiones de arte y concentrándose en las funcionalidades principales. Estos enemigos poseen movimiento sobre el suelo tipo patrulla, este puede ser lineal o generado aleatoriamente según los parámetros que se les brinden. Los enemigos poseen un rango de movimiento, un rango de ataque, un campo de visión, un campo de escucha, velocidad de movimiento y un punto débil en la espalda. Para el movimiento automático de los enemigos se usa una herramienta que ya incluye el motor de Unity3D para inteligencia artificial básica llamada *navmesh*.

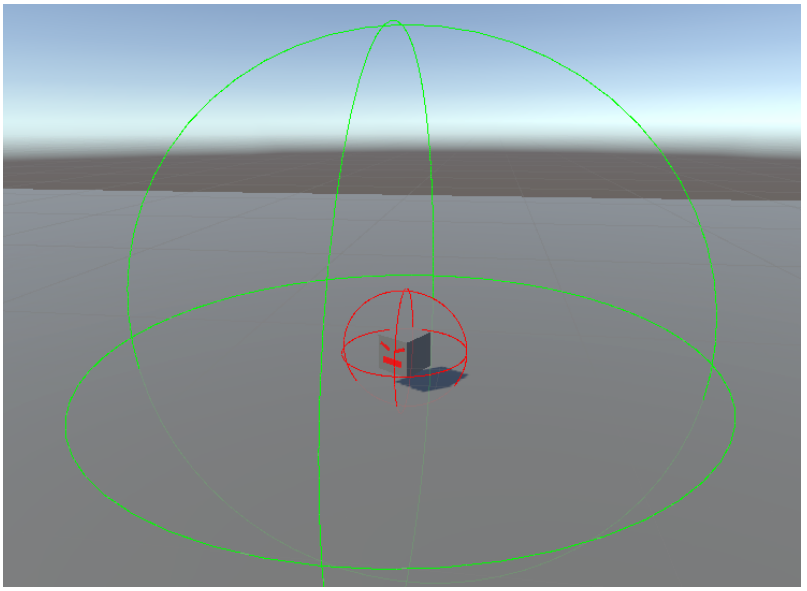

Figura 10: Radio de movimiento del enemigo en movimiento.

Estos parámetros como el rango de movimiento, la velocidad de movimiento, el identificador, si patrulla o se mueve de forma aleatoria, se controlan mediante el *script* de EnemyController.cs.

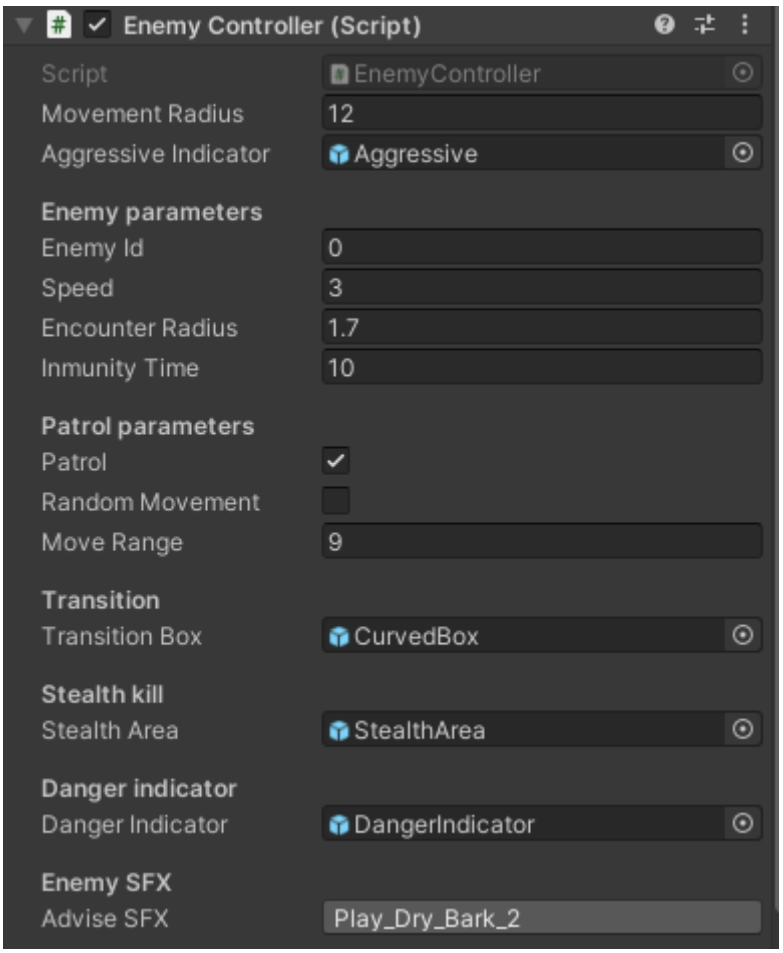

Figura 11: Script de EnemyController.cs desde el editor.

Para la detección de enemigos dentro de un campo de visión **ver anexo 1**, y el rango de escucha se utiliza una herramienta de un paquete llamado *Sensor Toolkit*, el cual consiste en un conjunto de sensores y espacios de colisión asociados a ciertos *scripts* que permiten la detección de objetos dentro de espacios 3D. Permiten definir objetos a detectar o a excluir de la detección, permiten no detectar cuando hay obstáculos, permite generar un campo de visión o un campo de escucha, entre otras funciones. Se implementa un Range sensor de este paquete para la escucha del enemigo y la detección de distractores, y se implementa un Trigger sensor con un *field of view collider*, que es la figura en forma de cono frente al enemigo, el rango, forma y tamaño de estos sensores se puede editar a preferencia del desarrollador en el IDE de Unity.

Para la distracción de los enemigos se utiliza el *range sensor* del paquete antes mencionado, y se desarrolla el *script* Distractor.cs, el cual cuenta con un parámetro de tiempo de distracción y otro para saber si el objeto se está sujetando o no, en caso de haber sido lanzado, este objeto al colisionar con el suelo cerca de un enemigo “generará ruido” haciendo que el enemigo se distraiga y vaya hacia este. Se utilizan los eventos de NewtonVR para saber cuando se toma y cuando se suelta este objeto distractor.

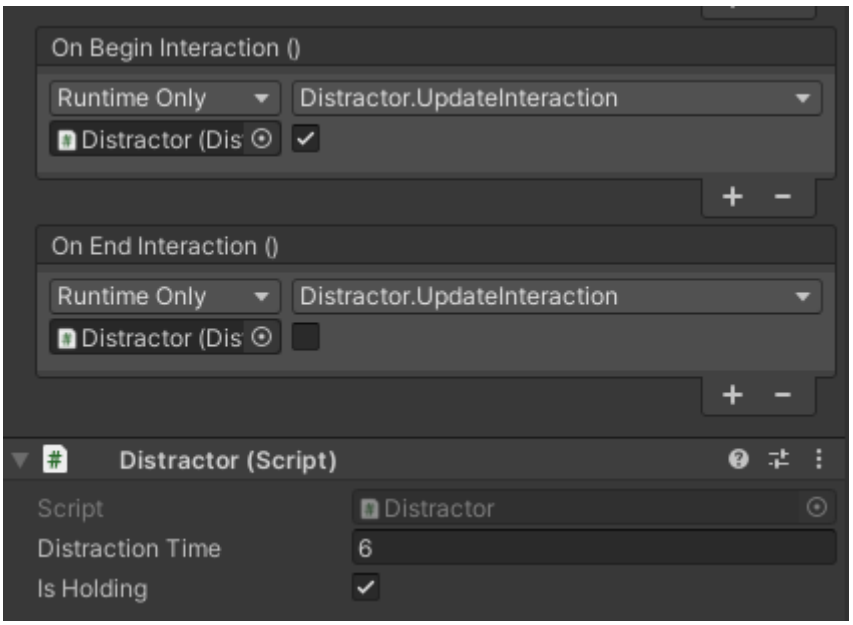

Figura 12: Lógica de un objeto distractor desde el editor.

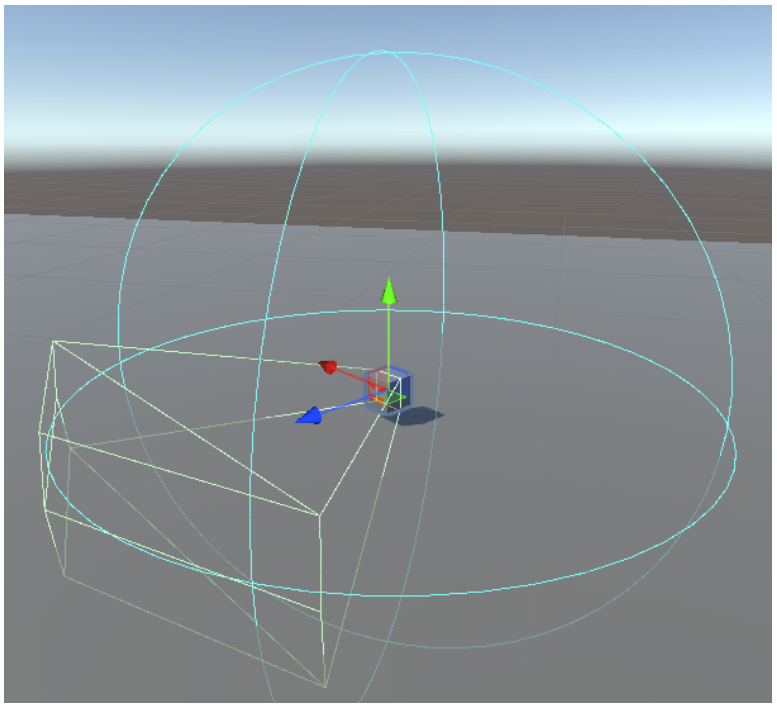

Figura 13: Sensor de visión(verde) y de escucha(celeste) del enemigo.

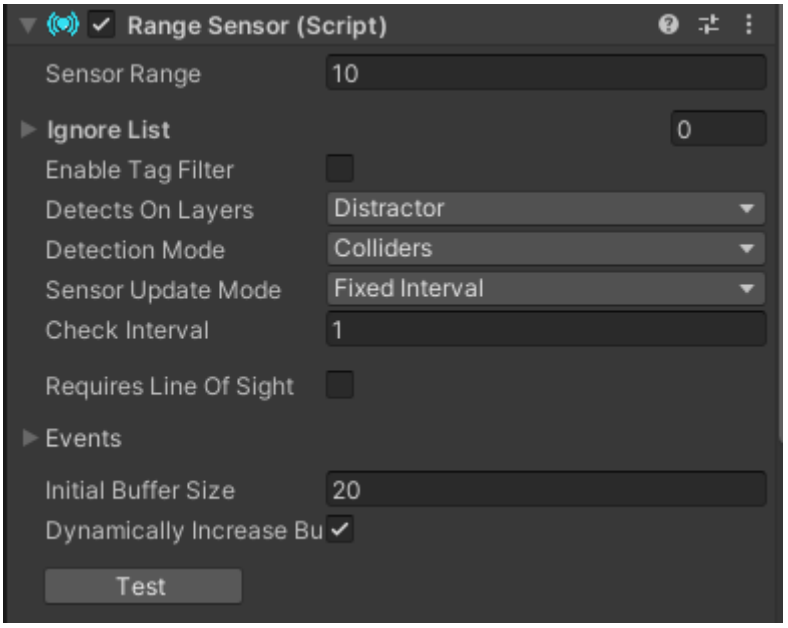

Figura 14: Sensor de escucha en el editor.

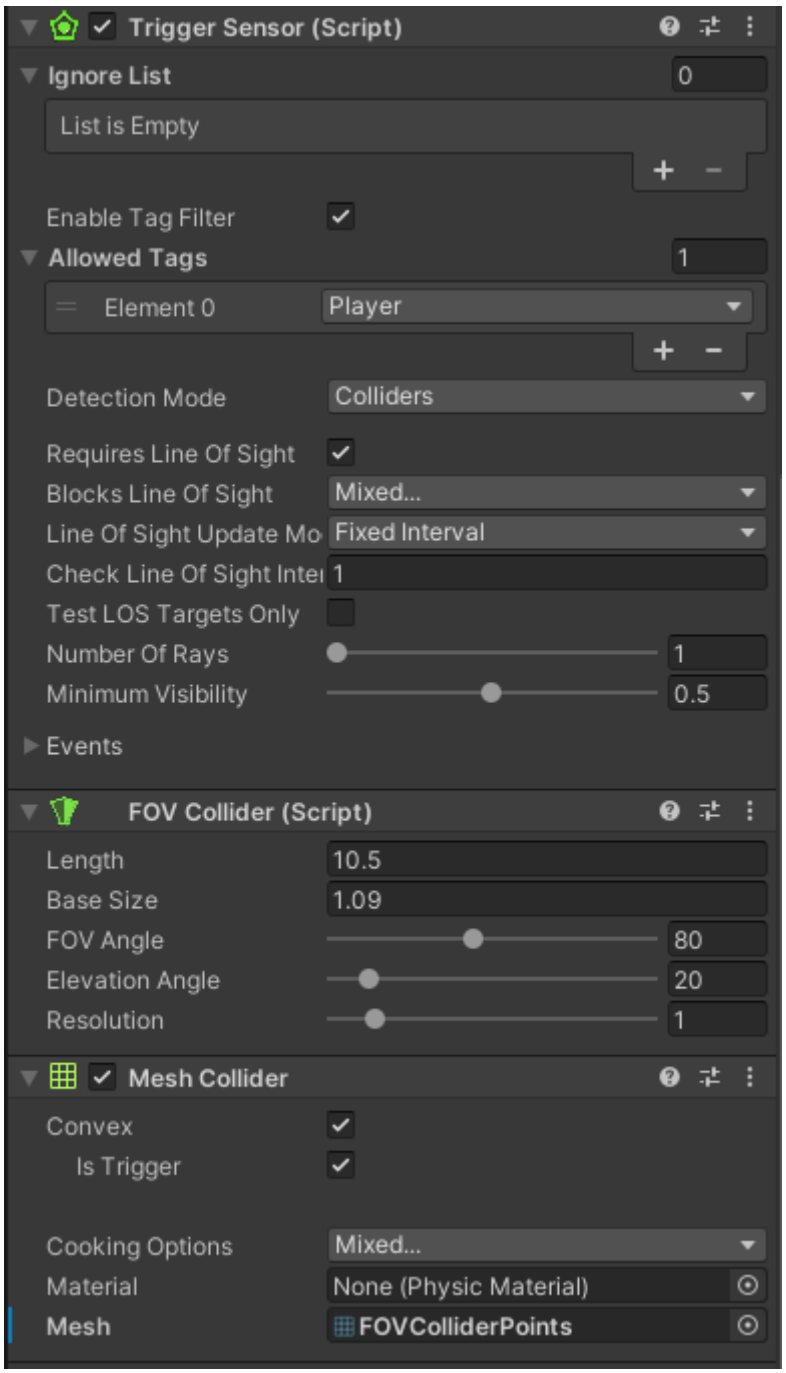

Figura 15: Sensor de visión en el editor.

5.1.2.5 Enemigos en combate

El enemigo en combate solo se activa según la orden de su homónimo en la zona principal, el área de combate es un espacio limitado y el jugador no debe poder moverse libremente por esta zona sino que debe enfrentar al enemigo que tiene enfrente. Si bien existe la opción de dar la vuelta y huir, tanto el jugador como el enemigo poseen estadísticas de vida y se pueden intercambiar golpes mutuamente siguiendo la mecánica rítmica de golpear las secuencias de cajas.

5.1.2.6 Vida del jugador

El jugador cuenta con una interfaz de usuario unida a un modelo 3D de un reloj, que está sujeto a la mano de realidad virtual del jugador, esta interfaz muestra un sencillo texto con la información sobre la salud del jugador, por ejemplo “50/100” indicaría que el jugador posee 50 puntos de vida de 100 que poseía originalmente, el reloj añadido a la mano cuenta con un “líquido interactivo” que se va vaciando conforme baja la vida del jugador. Este líquido se gestiona mediante un *shader* gráfico, y el llenado se controla mediante un script que modifica el valor del *shader*. Toda la lógica del reloj se maneja en dos *scripts*, uno que lleva la salud del jugador, y otro que hace que el reloj se pegue a la mano del jugador al entrar al mundo.

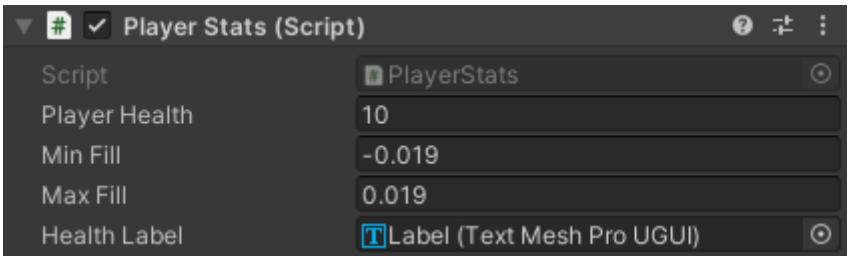

Figura 16: Script que controla la estadística de vida del jugador y el llenado del líquido.

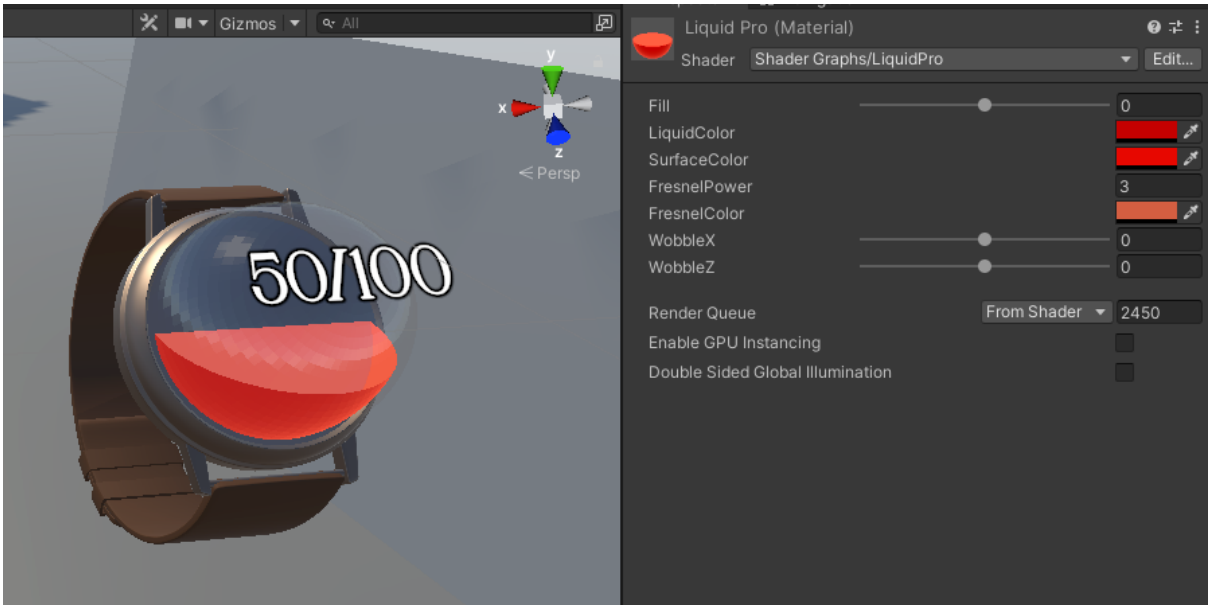

Figura 17: Modelo 3D del reloj, con el material del líquido interactivo.

5.1.2.7 Combate rítmico

Las cajas de combate poseen un funcionamiento singular, estas aparecen bajo el suelo, se elevan y van hacia el jugador, según la secuencia y los golpes que atine el jugador

a estas cajas puede aplicar diferentes cantidades de daño al enemigo, en caso de ser el enemigo el que está atacando, si el jugador no golpea las cajas recibe daño directo a su estadística de vida/salud. Cuando un enemigo pierde toda la vida el jugador vuelve a la zona principal y el enemigo ya no está ahí, si el jugador pierde se retorna al mismo a un *checkpoint* antes del combate en que murió.

5.1.2.8 Tipos de combate

Para decidir qué tipo de combate se va a implementar finalmente en el producto se propone prototipar tres tipos de combate, probar y generar retroalimentación sobre estos antes de integrar finalmente uno al juego.

5.1.2.8.1 Combate turnado

El combate por turnos es como su nombre lo indica mediante turnos, el jugador puede elegir sus ataques hacia el enemigo, después de ejecutar estos el enemigo lanza sus ataques y así sucesivamente hasta que alguno de los dos caiga o el jugador abandone el combate.

5.1.2.8.2 Combate basado en tiempos de ataque

El combate basado en tiempos de ataque se detalla más cláramente en el **anexo 3**, se basa en que el jugador tiene varias opciones de ataques pero estas no siempre van a estar disponibles, sino que según la eficacia de la secuencia de un ataque estos tienen un tiempo de carga, hasta que un ataque esté cargado el jugador no lo podrá usar.

5.1.2.8.3 Combate en tiempo real

El combate en tiempo real se desencadena en que el intercambio de golpes no posee ninguna pausa y el jugador está en todo momento enfocado en las cajas y patrones de ataques de los enemigos, sea tanto para atacar como para defenderse.

5.1.2.9 Integración del módulo de combate

Para la integración del módulo de combate y bajo la supervisión del otro desarrollador en el proyecto, Jose Andrés Chinchilla, se elabora un *prefab* de Unity, que es un objeto de juego que va a contener los diferentes componentes principales del combate para poder migrar este desde la escena de prototipado hacia la escena principal, este objeto posee como objetos “hijos” el contenedor con los enemigos en movimiento, el área de combate con su escenario y enemigos de combate previamente organizados (cada enemigo en movimiento posee un identificador numérico para su duplicado de combate), y finalmente el reloj que se “pega” a la mano del jugador para poder tener al alcance sus estadísticas de vida. Este objeto que contiene toda la lógica de combate se prueba en escenas nuevas y se corrigen las fallas que se presentan, antes de agregarlo a la escena principal.

5.1.3 Resultados obtenidos en el desarrollo de la plataforma de software

5.1.3.1 Enemigos en movimiento

Los enemigos en la zona principal se mueven en su espacio de movimiento designado, como también son capaces de detectar al jugador y embestir contra este para iniciar un combate. El script de EnemyController.cs funciona con normalidad y gestiona correctamente todos las funcionalidades y mecánicas de movimiento y detección de los enemigos en la zona principal. El sensor de escucha funciona correctamente y los enemigos persiguen objetos distractores, el sensor de visión funciona correctamente y el enemigo persigue al jugador. La velocidad de movimiento del enemigo es lenta mientras se mueve libre y más rápida cuando detecta al jugador y lo persigue, el jugador puede huir y esconderse de los enemigos. También al abandonar o finalizar un combate los enemigos de zona donde está el jugador lo ignoran durante unos segundos.

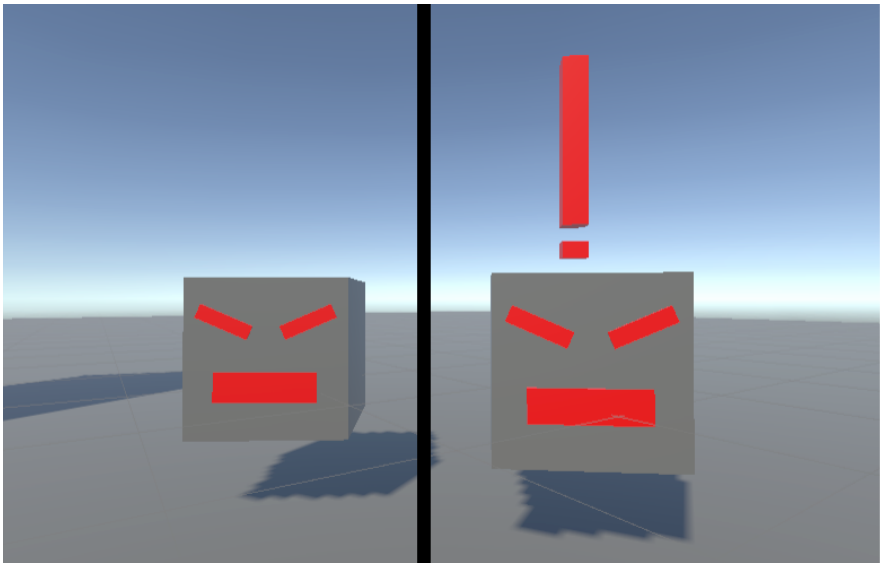

Figura 18: Ilustración de un enemigo cuando está en movimiento libre y cuando ha detectado al jugador.

5.1.3.2 Enemigos en combate

Los enemigos en combate se integran correctamente con el tipo de combate escogido y con la mecánica de las cajas de combate, atacan al jugador y también pueden recibir daño de los ataques del jugador. Cuando se inicia un combate, el identificador numérico del enemigo en movimiento enciende el duplicado de este mismo para combate en el área de combate.

5.1.3.3 Combate rítmico

Las cajas de combate funcionan según lo estipulado originalmente y similarmente a lo mostrado en el **anexo 2**, las cajas de combate se usan tanto para los ataques del jugador como para los del enemigo y se integran como una mecánica rítmica del videojuego.

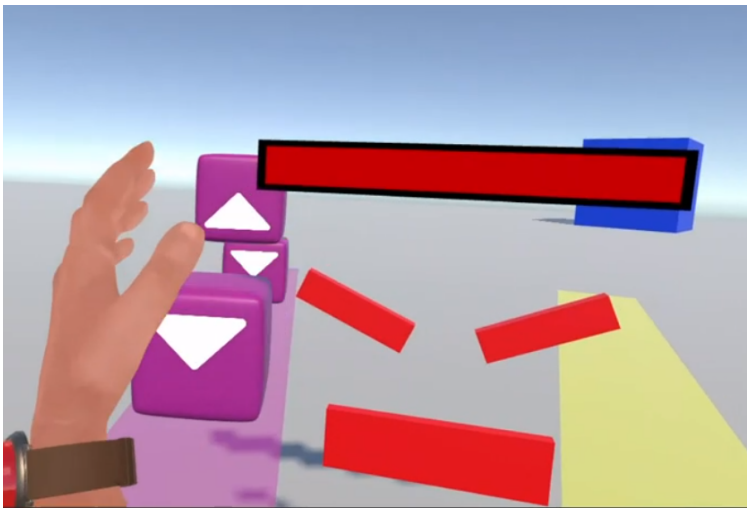

Figura 19: Combo básico de combate.

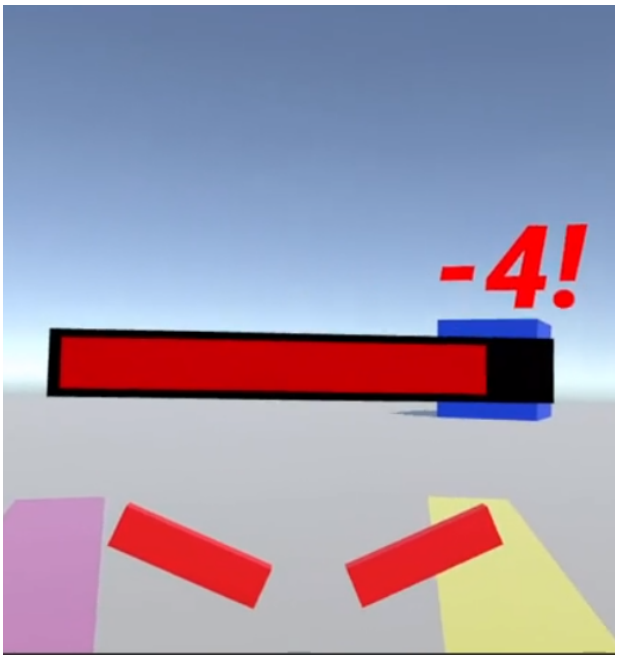

Figura 20: Daño aplicado al enemigo al golpear correctamente un combo de ataques.

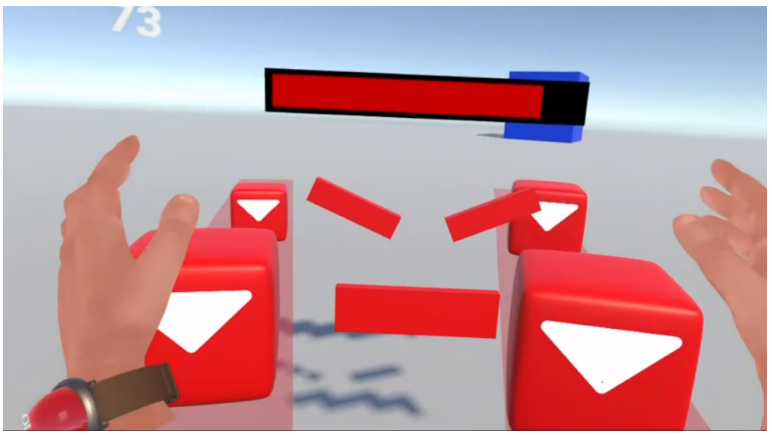

Figura 21: Secuencia de ataques del enemigo.

Figura 22: Salud del jugador afectada por no defenderse correctamente de los ataques del enemigo.

5.2 Evaluación

5.2.1 Introducción a la evaluación de la plataforma de software

Para la evaluación del correcto funcionamiento de la aplicación el desarrollador debe estar encargado de detectar y corregir las fallas de funcionamiento antes de pasar cada "actualización" del código a los otros involucrados del proyecto para su respectiva prueba y retroalimentación. La retroalimentación en las etapas de desarrollo se analiza y se integra según sea oportuno.

Se define un conjunto de casos de uso para pruebas manuales y de usabilidad según los requerimientos establecidos en el *backlog* de la aplicación.

Al ser una aplicación de realidad virtual, los métodos de entrada como *bots* o pruebas automatizadas no se pueden realizar.

5.2.2 Tareas realizadas para la evaluación de la plataforma de software

5.2.2.1 Definición de los casos de uso

El jugador dentro del entorno 3D puede desencadenar una serie de eventos según parámetros definidos como su ubicación, interacción con los objetos en el entorno, interacción con los enemigos o interfaces, entre otros.

Se define la siguiente lista de casos de prueba manuales que pretende no dejar sin evaluar ninguna de las funcionalidades principales.

Documentación de casos de prueba

Caso de prueba	
ID: CP-001	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Movimiento de los enemigos con *patrol*.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Verificar que el enemigo se mueva correctamente en un área sin obstáculos.	**Condiciones previas:** - El movimiento de los enemigos debe estar completamente

	desarrollado y listo para probarse. **Dependencias:** - No debe haber árboles, bloques o paredes en el rango de movimiento del enemigo. - El parámetro de *RandomMovement* del componente EnemyController debe estar desmarcado.
Pasos: 1. Colocar al enemigo en un espacio de movimiento limpio. 2. Colocar los parámetros de movimiento: a. En el editor sobre el componente de EnemyController se marca la casilla de *patrol* como verdadero.	
Resultado esperado: El enemigo con el parámetro de *patrol* activo se mueve libremente dentro de su rango de movimiento.	**Resultado Obtenido:** El enemigo se mueve entre dos puntos, sin embargo se mueve en dos puntos respecto a las coordenadas globales (“norte y sur”) y no a las coordenadas locales (al frente y atrás **de sí mismo**).
Estado (Pasa/Falla): Pasar	**Notas:** Se corrigió este error a nivel de programación.

Caso de prueba	
ID: CP-002	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Movimiento con obstáculos de los enemigos con *patrol*.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Verificar que el enemigo se mueva correctamente en un área con obstáculos, el enemigo ve un obstáculo y debe rodear este.	**Condiciones previas:** - El movimiento de los enemigos debe estar completamente desarrollado y listo para probarse. **Dependencias:** - Debe haber bloques con el componente *navmesh obstacle* dentro del área de movimiento del enemigo. - El parámetro de *RandomMovement* del componente EnemyController debe estar desmarcado.

Pasos: 1. Colocar al enemigo en un espacio de movimiento con obstáculos (*navmesh obstacle*). 2. Colocar los parámetros de movimiento: b. En el editor sobre el componente de EnemyController se marca la casilla de *patrol* como verdadero.	
Resultado esperado: El enemigo con el parámetro de *patrol* activo se mueve libremente dentro de su rango de movimiento, si encuentra un obstáculo lo rodea y continúa con su camino.	**Resultado Obtenido:** El enemigo encuentra obstáculos en su camino y los rodea.
Estado (Pasa/Falla): Pasar	**Notas:** Se añadió una referencia visual en el editor para mostrar la trayectoria del enemigo.

Caso de prueba	
ID: CP-003	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Movimiento de los enemigos con *randomMovement*.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Verificar que el enemigo se mueva correctamente en un área sin obstáculos.	**Condiciones previas:** - El movimiento de los enemigos debe estar completamente desarrollado y listo para probarse. **Dependencias:** - No debe haber árboles, bloques o paredes en el rango de movimiento del enemigo. - El parámetro de *Patrol* del componente EnemyController debe estar desmarcado.
Pasos: 1. Colocar al enemigo en un espacio de movimiento limpio. 2. Colocar los parámetros de movimiento: c. En el editor sobre el componente de EnemyController se marca la casilla de *randomMovement* como verdadero.	
Resultado esperado: El enemigo con el parámetro de *randomMovement* activo se mueve libremente dentro de su rango de movimiento.	**Resultado Obtenido:** El enemigo se mueve libremente en el rango de los tres puntos generados aleatoriamente.

Estado (Pasa/Falla): Pasar	**Notas:** Se añadió una referencia visual en el editor para mostrar la trayectoria del enemigo.

Caso de prueba	
ID: CP-004	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Movimiento con obstáculos de los enemigos con *randomMovement*.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Verificar que el enemigo se mueva correctamente en un área con obstáculos, el enemigo ve un obstáculo y debe rodear este.	**Condiciones previas:** - El movimiento de los enemigos debe estar completamente desarrollado y listo para probarse. **Dependencias:** - Debe haber bloques con el componente *navmesh obstacle* dentro del área de movimiento del enemigo. - El parámetro de *Patrol* del componente EnemyController debe estar desmarcado.
Pasos: 1. Colocar al enemigo en un espacio de movimiento con obstáculos (*navmesh obstacle*). 2. Colocar los parámetros de movimiento: d. En el editor sobre el componente de EnemyController se marca la casilla de *randomMovement* como verdadero.	
Resultado esperado: El enemigo con el parámetro de *randomMovement* activo se mueve libremente dentro de su rango de movimiento, si encuentra un obstáculo lo rodea y continúa con su camino.	**Resultado Obtenido:** El enemigo de vez en cuando genera un punto dentro de un obstáculo y se queda mirándolo fijamente bloqueado. Se debe corregir esto a nivel de código.
Estado (Pasa/Falla): Pasar	**Notas:** Se añadió una referencia visual en el editor para mostrar la trayectoria del enemigo.

Caso de prueba	
ID: CP-005	**Diseñado por:** Leonardo Lizano N.

Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Detección de los enemigos en movimiento.	**Fecha de ejecución de la prueba:** 06/11/2021

Descripción: Los enemigos que se mueven por la zona principal deben poder detectar al jugador cuando este se encuentra frente a ellos.	**Condiciones previas:**
	- El movimiento de los enemigos debe estar completamente desarrollado y listo para probarse.
	Dependencias:
	- El enemigo debe poseer un componente de Trigger Sensor con los siguientes parámetros.
	  - Enable Tag Filter: true
	  - Allowed Tags: Player
	  - Detection Mode: Colliders
	  - Requires Line of sight: true
	  - Blocks Line of sight: Obstruction, ground
	- El enemigo debe poseer un componente FOV Collider con los siguientes parámetros (Este componente automáticamente agrega un Mesh Collider que debe ser Trigger).
	  - Length: 10.5f
	  - Base Size: 1.09
	  - FOV Angle: 80
	  - Elevation Angle: 20
	  - Resolution: 1

Pasos: 1. Usar la mecánica de teletransporte del jugador para colocarse frente a un enemigo.	
Resultado esperado: El enemigo detecta al jugador y se mueve hacia este.	**Resultado Obtenido:** El enemigo detecta al jugador y se mueve hacia este.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-006	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Fuera del rango de visión de los enemigos en movimiento.	**Fecha de ejecución de la prueba:** 06/11/2021

Descripción: Si el jugador se coloca detrás o a un lado de un enemigo, este no debería detectarlo al estar fuera del rango de visión.	**Condiciones previas:**
	- El movimiento de los enemigos debe estar completamente desarrollado y listo para probarse.
	Dependencias:
	- El enemigo debe poseer un componente de Trigger Sensor con los siguientes parámetros.
	  - Enable Tag Filter: true
	  - Allowed Tags: Player
	  - Detection Mode: Colliders
	  - Requires Line of sight: true
	  - Blocks Line of sight: Obstruction, ground
	- El enemigo debe poseer un componente FOV Collider con los siguientes parámetros (Este componente automáticamente agrega un Mesh Collider que debe ser Trigger).
	  - Length: 10.5f
	  - Base Size: 1.09
	  - FOV Angle: 80
	  - Elevation Angle: 20
	  - Resolution: 1

Pasos: 1. Usar la mecánica de teletransporte del jugador para colocarse detrás de un enemigo.	
Resultado esperado: El enemigo no detecta al jugador y sigue con su rumbo.	**Resultado Obtenido:** El enemigo no detecta al jugador y sigue con su rumbo.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-007	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Esconderse de los enemigos en movimiento.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Si hay un objeto marcado en la capa física de *Obstruction* y se encuentra entre el jugador y el enemigo, dentro del rango de visión del enemigo, este no debería detectar al jugador.	**Condiciones previas:** - El movimiento de los enemigos debe estar completamente desarrollado y listo para probarse. - El objeto que debe estar en medio debe estar en el Layer de *Obstruction*.

	Dependencias:
	- El enemigo debe poseer un componente de Trigger Sensor con los siguientes parámetros.
	  - Enable Tag Filter: true
	  - Allowed Tags: Player
	  - Detection Mode: Colliders
	  - Requires Line of sight: true
	  - Blocks Line of sight: Obstruction, ground
	- El enemigo debe poseer un componente FOV Collider con los siguientes parámetros (Este componente automáticamente agrega un Mesh Collider que debe ser Trigger).
	  - Length: 10.5f
	  - Base Size: 1.09
	  - FOV Angle: 80
	  - Elevation Angle: 20
	  - Resolution: 1

Pasos: 1. Usar la mecánica de teletransporte del jugador para colocarse detrás de un bloque con el layer *Obstruction* y dentro del rango de visión del enemigo.	
Resultado esperado: El enemigo no detecta al jugador y sigue con su rumbo.	**Resultado Obtenido:** El enemigo no detecta al jugador y sigue con su rumbo.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-008	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Inicio del combate	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Cuando un enemigo se acerca lo suficiente al jugador, este debe ser teletransportado a la zona de combate para iniciar el combate	**Condiciones previas:** - El movimiento de los enemigos debe estar completamente desarrollado y listo para probarse. **Dependencias:** - El enemigo debe poseer el Trigger Sensor de tipo Field of View correctamente configurado y probado.

Pasos: 1. Usar la mecánica de teletransporte del jugador para acercarse a un enemigo y colocarse frente a este. 2. Esperar a que el enemigo se acerque lo suficiente como para transicionar al combate.	
Resultado esperado: El enemigo detecta al jugador y al estar a menos de 1m de distancia de este, se inicia la transición a combate y el jugador aparece en la zona de pelea.	**Resultado Obtenido:** El enemigo detecta al jugador y al estar a menos de 1m de distancia de este, se inicia la transición a combate y el jugador aparece en la zona de pelea.
Estado (Pasa/Falla): Pasar	**Notas:** La distancia de un metro es relativa, este es un parámetro float llamado encounterRadius en EnemyController.

Caso de prueba	
ID: CP-009	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Huida del combate	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Cuando el jugador entra en un combate, tiene la posibilidad de dar la espalda al enemigo, y simular un movimiento de correr con los brazos para “huir corriendo” del combate y retornar a la zona principal.	**Condiciones previas:** - El inicio de combate debe estar correctamente desarrollado. - La mecánica de huir moviendo los brazos debe estar correctamente desarrollada. **Dependencias:**
Pasos: 1. Usar la mecánica de teletransporte del jugador para acercarse a un enemigo y colocarse frente a este. 2. Esperar a que el enemigo se acerque lo suficiente como para transicionar al combate. 3. Dar la espalda al enemigo girando el cuerpo real del jugador para girar en el entorno 3D. 4. Mover ambos brazos arriba y abajo repetidamente para simular “correr”. 5. Esperar hasta que se teletransporte de nuevo a la zona principal.	
Resultado esperado: El jugador da la espalda al enemigo, comienza a sacudir los brazos hasta que retorna a la principal y los enemigos lo ignoran.	**Resultado Obtenido:** El jugador da la espalda al enemigo, comienza a sacudir los brazos hasta que retorna a la principal y los enemigos lo ignoran.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-010	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Huida incorrecta brazo izquierdo.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Cuando el jugador entra en un combate, tiene la posibilidad de dar la espalda al enemigo, y simular un movimiento de correr con los brazos para “huir corriendo” del combate y retornar a la zona principal. Se debe mover ambos brazos para que esta mecánica se active.	**Condiciones previas:** - El inicio de combate debe estar correctamente desarrollado. - La mecánica de huir moviendo los brazos debe estar correctamente desarrollada. **Dependencias:**
Pasos: 1. Usar la mecánica de teletransporte del jugador para acercarse a un enemigo y colocarse frente a este. 2. Esperar a que el enemigo se acerque lo suficiente como para transicionar al combate. 3. Dar la espalda al enemigo girando el cuerpo real del jugador para girar en el entorno 3D. 4. **Solo el brazo izquierdo** arriba y abajo repetidamente para simular “correr”. 5. Esperar hasta que se teletransporte de nuevo a la zona principal.	
Resultado esperado: La mecánica de huida no se activa porque el movimiento de huida debe ser de ambos brazos.	**Resultado Obtenido:** La mecánica de huida no se activa porque el movimiento de huida debe ser de ambos brazos.
Estado (Pasa/Falla): Fallar.	**Notas:**

Caso de prueba	
ID: CP-012	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Huida incorrecta brazo derecho.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Cuando el jugador entra en un combate, tiene la posibilidad de dar la espalda al enemigo, y simular un movimiento de correr con los brazos para “huir corriendo” del combate y retornar a la zona principal. Se debe mover ambos	**Condiciones previas:** - El inicio de combate debe estar correctamente desarrollado. - La mecánica de huir moviendo los brazos debe estar correctamente desarrollada.

brazos para que esta mecánica se active.	**Dependencias:**
Pasos: 1. Usar la mecánica de teletransporte del jugador para acercarse a un enemigo y colocarse frente a este. 2. Esperar a que el enemigo se acerque lo suficiente como para transicionar al combate. 3. Dar la espalda al enemigo girando el cuerpo real del jugador para girar en el entorno 3D. 4. **Solo el brazo derecho** arriba y abajo repetidamente para simular “correr”. 5. Esperar hasta que se teletransporte de nuevo a la zona principal.	
Resultado esperado: La mecánica de huida no se activa porque el movimiento de huida debe ser de ambos brazos.	**Resultado Obtenido:** La mecánica de huida no se activa porque el movimiento de huida debe ser de ambos brazos.
Estado (Pasa/Falla): Fallar.	**Notas:**

Caso de prueba	
ID: CP-013	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Cajas de combate básicas.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Al golpear las cajas de colores que vienen hacia el jugador de esta forma se le aplica daño al enemigo, las cajas básicas aplican 1 punto de daño si son bien golpeadas.	**Condiciones previas:** - El funcionamiento de las cajas de combate debe estar correctamente desarrollado. - Se debe haber iniciado un combate. **Dependencias:** - Al iniciar el combate se debe lanzar un ataque que no sea en combo.
Pasos: 1. En la zona principal caminar hacia un enemigo y permitir que este corra hacia el jugador hasta iniciar un combate. 2. Golpear las cajas de combate correctamente. 3. Verificar la salud del enemigo.	
Resultado esperado: El enemigo pierde un punto de vida por cada caja que es golpeada correctamente.	**Resultado Obtenido:** El enemigo pierde un punto de vida por cada caja que es golpeada correctamente.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba

ID: CP-014	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Cajas de combate básicas - dejar pasar.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Al dejar pasar sin golpear las cajas de combate entonces el enemigo no recibe ningún daño.	**Condiciones previas:** - El funcionamiento de las cajas de combate debe estar correctamente desarrollado. - Se debe haber iniciado un combate. **Dependencias:** - Al iniciar el combate se debe lanzar un ataque que no sea en combo.
Pasos: 1. En la zona principal caminar hacia un enemigo y permitir que este corra hacia el jugador hasta iniciar un combate. 2. No golpear las cajas de combate. 3. Verificar la salud del enemigo.	
Resultado esperado: El enemigo no pierde puntos de vida si las cajas dejan pasar.	**Resultado Obtenido:** Al dejar pasar sin golpear las cajas de combate entonces el enemigo no recibe ningún daño.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-015	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Cajas de combate básicas mal golpeadas.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Al golpear las cajas de colores que vienen hacia el jugador de forma errónea a la dirección que estas indican, no surte ningún efecto sobre el enemigo.	**Condiciones previas:** - El funcionamiento de las cajas de combate debe estar correctamente desarrollado. - Se debe haber iniciado un combate. **Dependencias:** - Al iniciar el combate se debe lanzar un ataque que no sea en combo.
Pasos:	

1. En la zona principal caminar hacia un enemigo y permitir que este corra hacia el jugador hasta iniciar un combate. 2. Golpear las cajas de combate en una dirección diferente a la indicada. 3. Verificar la salud del enemigo.	
Resultado esperado: El enemigo no pierde puntos de vida si las cajas se golpean erróneamente.	**Resultado Obtenido:** El enemigo no pierde puntos de vida si las cajas se golpean erróneamente.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-015	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Cajas de combate combos.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Al golpear ciertas secuencias de cajas en combo, se aplica un daño mayor al enemigo, para que esto funcione se deben golpear correctamente todas las cajas de un combo.	**Condiciones previas:** - El funcionamiento de las cajas de combate debe estar correctamente desarrollado. - Se debe haber iniciado un combate. **Dependencias:** - Al iniciar el combate se debe lanzar un combo de ataques.
Pasos: 1. En la zona principal caminar hacia un enemigo y permitir que este corra hacia el jugador hasta iniciar un combate. 2. Golpear las cajas de combate correctamente. 3. Verificar la salud del enemigo.	
Resultado esperado: El enemigo pierde varios puntos de vida de golpe, no uno por cada caja.	**Resultado Obtenido:** El enemigo pierde varios puntos de vida de golpe, no uno por cada caja.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-016	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.

Título de la prueba: Cajas de combate con combos incorrectos.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Al golpear incorrectamente una de las cajas de combate de un combo, este pierde su efecto totalmente y no se aplica ningún daño al enemigo al terminar el combo.	**Condiciones previas:** - El funcionamiento de las cajas de combate debe estar correctamente desarrollado. - Se debe haber iniciado un combate. **Dependencias:** - Al iniciar el combate se debe lanzar un combo de ataques.
Pasos: 1. En la zona principal caminar hacia un enemigo y permitir que este corra hacia el jugador hasta iniciar un combate. 2. Golpear las cajas de combate incorrectamente. 3. Verificar la salud del enemigo.	
Resultado esperado: El enemigo no pierde ningún punto de vida por efectuar un combo de ataque incorrecto.	**Resultado Obtenido:** El enemigo no pierde ningún punto de vida por efectuar un combo de ataque incorrecto.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-017	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Ataques del enemigo	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Para continuar con la mecánica de cajas de combate en secuencias, el enemigo puede invocar secuencias de cajas de color rojo que funcionan igual a las cajas de combate, si el jugador las golpea no recibe daño.	**Condiciones previas:** - El funcionamiento de las cajas de combate debe estar correctamente desarrollado. - Se debe haber iniciado un combate. **Dependencias:** - Esperar al turno del enemigo.
Pasos: 1. En la zona principal caminar hacia un enemigo y permitir que este corra hacia el jugador hasta iniciar un combate. 2. Golpear las cajas de combate correctamente. 3. Verificar la salud del jugador en el reloj.	
Resultado esperado: El jugador no pierde ningún punto de vida por golpear correctamente los ataques del enemigo.	**Resultado Obtenido:** El jugador no pierde ningún punto de vida por golpear correctamente los ataques del enemigo.

Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-018	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Ataques del enemigo no golpeados	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Si el jugador deja pasar una de las cajas de combate del enemigo, entonces pierde puntos de salud	**Condiciones previas:** - El funcionamiento de las cajas de combate debe estar correctamente desarrollado. - Se debe haber iniciado un combate. **Dependencias:** - Esperar al turno del enemigo.
Pasos: 1. En la zona principal caminar hacia un enemigo y permitir que este corra hacia el jugador hasta iniciar un combate. 2. Dejar las cajas de combate del enemigo sin golpear. 3. Verificar la salud del jugador en el reloj.	
Resultado esperado: El jugador pierde dos puntos de vida por cada caja del enemigo que deje pasar y no golpee.	**Resultado Obtenido:** El jugador pierde dos puntos de vida por cada caja del enemigo que deje pasar y no golpee.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-019	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Ataques del enemigo mal golpeados	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Si el jugador golpea una de las cajas de combate del enemigo en una dirección errónea entonces este recibe los dos puntos de daño.	**Condiciones previas:** - El funcionamiento de las cajas de combate debe estar correctamente desarrollado. - Se debe haber iniciado un combate. **Dependencias:**

	- Esperar al turno del enemigo.
Pasos: 1. En la zona principal caminar hacia un enemigo y permitir que este corra hacia el jugador hasta iniciar un combate. 2. Golpear las cajas de combate incorrectamente. 3. Verificar la salud del jugador en el reloj.	
Resultado esperado: El jugador pierde dos puntos de vida por cada caja del enemigo que deje pasar y no golpee.	**Resultado Obtenido:** El jugador pierde dos puntos de vida por cada caja del enemigo que deje pasar y no golpee.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-020	**Diseñado por:** Leonardo Lizano N.
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Distracción de los enemigos en movimiento.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Si el jugador arroja un objeto distractor cerca de los enemigos estos lo escuchan y se mueven hacia este quedándose quietos durante unos segundos.	**Condiciones previas:** - La lógica de distracción de los enemigos debe estar correctamente desarrollada. **Dependencias:** - Debe haber al menos un objeto distractor en la escena.
Pasos: 1. En la zona principal tomar un objeto distractor con las manos. 2. Arrojar el objeto distractor cerca de un enemigo en movimiento. 3. Verificar el comportamiento del enemigo. 4. Esperar a que el tiempo de distracción termine. 5. Verificar que los enemigos retornen a su ruta original.	
Resultado esperado: El enemigo escucha un objeto distractor caer y se mueve hacia este, se queda quieto, espera y finalmente retorna a su rumbo original.	**Resultado Obtenido:** El enemigo escucha un objeto distractor caer y se mueve hacia este, se queda quieto, espera y finalmente retorna a su rumbo original.
Estado (Pasa/Falla): Pasar	**Notas:**

Caso de prueba	
ID: CP-021	**Diseñado por:** Leonardo Lizano N.

Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:** 26/09/2021
Nombre del módulo: Módulo de combate	**Prueba ejecutada por:** Leonardo Lizano N.
Título de la prueba: Ataque sorpresa del jugador.	**Fecha de ejecución de la prueba:** 06/11/2021
Descripción: Si el jugador se acerca por la espalda a un enemigo distraído puede golpear a este por sorpresa y derrotarlo sin entrar en combate.	**Condiciones previas:** - La lógica de distracción de los enemigos debe estar correctamente desarrollada. **Dependencias:** - Debe haber al menos un objeto distractor en la escena.
Pasos: 1. En la zona principal tomar un objeto distractor con las manos. 2. Arrojar el objeto distractor cerca de un enemigo en movimiento. 3. “Caminar” hasta estar justo detrás del enemigo. 4. Golpear el punto débil del enemigo.	
Resultado esperado: El enemigo desaparece tras haber sido derrotado por sorpresa.	**Resultado Obtenido:** El enemigo desaparece tras haber sido derrotado por sorpresa.
Estado (Pasa/Falla): Pasar	**Notas:**

5.2.3 Resultados obtenidos en la evaluación de la plataforma de software

Una vez finalizadas las pruebas manuales de las mecánicas principales de la plataforma de software del incremento de funcionalidades al videojuego “Let Me Go” se puede destacar el siguiente conjunto de resultados de las pruebas realizadas.

5.2.3.1 Evaluación de funcionalidades del jugador

Tanto la habilidad del jugador para golpear las cajas en combate como representar sus propios puntos de salud pegado al reloj en la muñeca funcionan correctamente.

5.2.3.2 Evaluación de enemigos en movimiento

Los enemigos en movimiento en la zona principal pasaron la mayoría de pruebas sin problemas.

Dentro de las pruebas que fallaron se encuentran CP-001 y CP-004, el caso de prueba primero corresponde al movimiento de patrulla del enemigo, que debe ser el

enemigo caminando hacia al frente y hacia atrás, este caso de prueba falló porque se calculaban mal las coordenadas de los puntos de patrullaje, como contramedida se refactoriza el código para que los puntos de patrullaje se calculen usando el verdadero frente y espalda del enemigo.

Para el caso de prueba segundo (CP-004) se tuvo que diseñar un contador de tiempo interno del enemigo para verificar si el mismo se encuentra detenido colisionando contra un objeto porque su punto de patrullaje está dentro del obstáculo, esta es una situación “difícil” de controlar porque los puntos se generan aleatoriamente, si bien se hizo que los puntos tengan un mínimo de distancia entre sí, también este contador de tiempo interno se encarga de decirle al enemigo que se mueva al siguiente punto si este se queda bloqueado.

5.2.3.3 Evaluación del combate

Los distintos elementos de UX/UX, al igual que la mecánica de combate en general se integran correctamente.

Las cajas de combate del enemigo fue la mecánica que más refactorización necesitó antes de realizar las pruebas, actualmente funcionan correctamente.

Capítulo VI

6 Conclusiones

Dentro de los objetivos de desarrollo de la plataforma de software de incremento al videojuego “Let Me Go” se determinó lo siguiente.

Se definió correctamente el conjunto de mecánicas y funcionalidades a añadir a la plataforma de software.

Mediante los procesos de *dailys*, revisión de sprints y desarrollo scrum se diseñó el conjunto de mecánicas a desarrollar, durante el proceso de desarrollo se fueron agregando o quitando pequeñas o grandes funcionalidades y finalmente el sistema de software cumple con integridad todas las funciones que se determinó.

Se desarrolló correctamente y mediante un proceso interactivo supervisado todo el conjunto de mecánicas solicitadas, la etapa de integración se encuentra en pausa por motivos de que simultáneamente al proceso de práctica se está realizando un remodelado completo del mapa del juego.

Se evaluó y corrigió las diferentes mecánicas y funcionalidades diseñadas y desarrolladas para el videojuego, las pruebas manuales fueron provechosas tomando en cuenta que se detectó y corrigió pequeños fallos dentro de las mecánicas más importantes que podrían desencadenar en un producto deficiente o carente de sentido.

Uno de los objetivos no explícitos dentro del documento pero sí dentro del proceso de desarrollo es que este módulo de combate y enemigos agreguen un valor extra en el sentido de entretenimiento, ya que el plan se sostiene en hacer aparecer enemigos cerca del jugador cuando este tiene que explorar áreas en las que ya ha estado, de forma que volver a recorrer estos lugares no se vuelva tedioso y se sienta como un mundo vivo y lleno. La presencia de los enemigos moviéndose alrededor realmente genera una sensación de tensión y alerta en el jugador.

Desarrollar videojuegos es en todos los sentidos posibles un trabajo multidisciplinario, por lo que exponer al estudiante a cuestiones de diseño, prototipado, diseño de arte, diseño de sonido, entre otras tareas, desencadena un proceso de crecimiento muy acelerado y sumamente enriquecedor, como también el poder trabajar junto a colaboradores expertos de las diferentes áreas como diseño, programación y arte digital.

6.1 Recomendaciones

En este apartado se abordan tanto recomendaciones generales sobre cuestiones del proyecto, perspectiva del estudiante y aprendizajes adquiridos durante el desarrollo. Se describen las recomendaciones.

Cabe mencionar que durante el proceso de desarrollo hubo “un reemplazo” del productor, siendo asignado al seguimiento del proyecto Pedro Bastos, de la compañía productora Seat 7 Entertainment, lo cual permitió continuar dando seguimiento y rastreo al trabajo del estudiante, pero, con ciertas limitaciones lo cual lleva a la siguiente recomendación.

Dentro de los involucrados del proyecto se encuentra Jose Andres Chinchilla el cual cuenta con un dispositivo de Oculus Quest proporcionado por la empresa por lo que él en ciertas ocasiones fue capaz de probar las mecánicas, al igual que Jose Pablo (director y productor) que cuenta con un Oculus Quest 2 para probar el juego, sin embargo, al integrarse Veronica Morera como artista del proyecto y Pedro Bastos como productor ninguno de los dos fue capaz de probar o medir el espacio 3D, generando algunas inconsistencias.

La integración de los diferentes componentes de arte para la parte de combate fue un éxito gracias al trabajo y comunicación rápida con la artista digital, sin embargo lo antes mencionado acabó en que el estudiante adquiera entre sus responsabilidades pequeñas cuestiones sobre el arte.

En cierto momento el colaborador Jose Andres Chinchilla recomendó comenzar a trabajar en un *prefab* de todo el sistema de combate para poder empezar a integrarlo con el resto del mapa, lo cual fue un paso sumamente importante para la etapa de integración, pero esta última etapa se vio interrumpida por el proceso continuo de remodelar toda la escena del bosque, que además se vio retrasado por el hecho de que Veronica Morera no cuenta con un dispositivo de realidad virtual para poder visualizar la escala o posición de los objetos en realidad virtual, la recomendación sería procurar que todos los involucrados con tareas importantes cuenten con los recursos y medios necesarios para poder probar lo que se está desarrollando.

6.1 Experiencia de la práctica

Como estudiante y profesional en formación, no cabe más que agradecer a Jose Pablo Monge por abrirme las puertas a la oportunidad de desempeñar la práctica profesional como desarrollador en Headless Chicken Games, mucho de lo que conocía superficialmente sobre el desarrollo de videojuegos ha cambiado, mejorando y ampliando muchísimo mi perspectiva profesional sobre este ámbito y la industria en general.

Durante un tiempo a inicios del proyecto todo el seguimiento se realizaba únicamente entre el productor Jose Pablo y el estudiante, con el paso del tiempo se asignaron otros recursos al equipo como son Veronica Morera como artista digital y Jose Andres Chinchilla como desarrollador, ambos profesionales sumamente talentosos y comprometidos con el proyecto, las conversaciones en dailys con Jose Andres desencadenó en varios procesos de mejora del trabajo en general y de las mecánicas, siendo este un apoyo incluso durante el desarrollo de algunas mecánicas o solucionando errores que aparecían durante el proceso.

Durante los últimos 60 días particularmente la artista digital Veronica Morera se esmeró mucho en sugerir, crear, modificar o cambiar muchos de los elementos referentes a todo el módulo de combate, lo cual finalizó en un producto sumamente impresionante y llamativo a la vista, tanto los elementos en el entorno 3D como los enemigos, como también los elementos de interfaces de usuario.

Extiendo un enorme agradecimiento a todos los involucrados por estar siempre atentos a dudas, por los consejos y por el esfuerzo que han puesto en el proyecto en general o en aportar para el crecimiento profesional del estudiante, desde mi perspectiva personal, el juego ha tenido un crecimiento increíble gracias a todos los involucrados en esta etapa del proyecto, y es de admirar mucho del trabajo que ya había antes de que el estudiante se integrara al proceso de desarrollo.

Al ser un proyecto multidisciplinario siempre se toma en cuenta la opinión de todos y esto genera una variedad de aportes que poco a poco van creando en el producto un inmenso valor como proyecto de software y como videojuego.

Bibliografía

Bibliografía

Monge Chacon J.P, (2018). Let Me Go Game Design Document, *Headless Chicken Games*.

Sexton J., (2007). Music, Sound and Multimedia: From the Live to the Virtual. *Edinburgh University Press*.

Anexos

Anexos

Dentro del material de referencia importante para la comprensión en profundidad del proyecto se adjunta el acceso a los siguientes elementos informativos.

Anexo 1: Representación del *Field of View* o campo de visión es un espectro de como funciona la visión en la vida real. Siendo que para inteligencia artificial esto representaría lo que el modelo u objeto puede ver/detectar dentro de un espacio y ángulo designado.

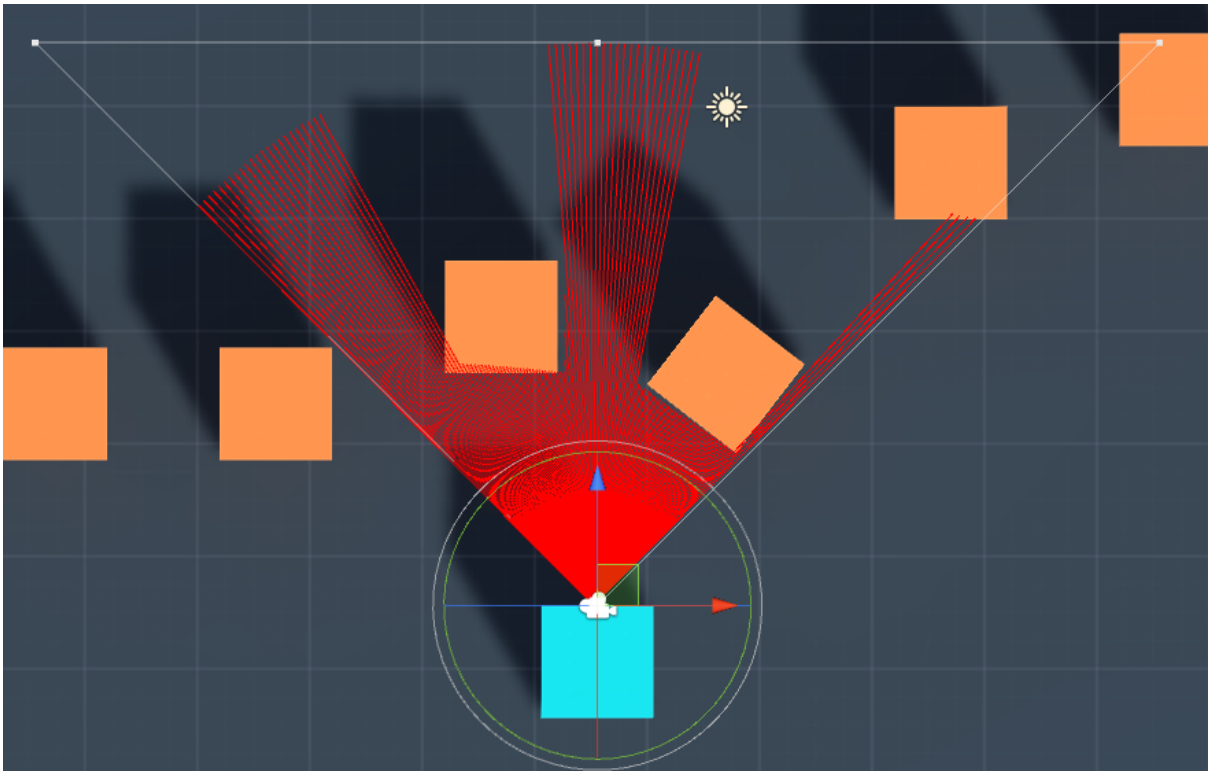

Figura 23: Representación gráfica de cómo funciona la detección de elementos en el entorno 3D con un campo de visión.

Anexo 2: Representación de las cajas de combate u objetivos a golpear en el juego, tomando como referencia la mecánica principal del videojuego para realidad virtual Beat Saber. Video de referencia: Dance Monkey (Tones and I) - Beat Saber - Normal Difficulty obtenido de YouTube, del usuario Techie49.

Anexo 3: *Active Time Battle* (ATB) es un sistema de combate para videojuegos el cual funciona como alternativa al combate de turnos de uno a uno, en esta mecánica de combate

introducida en la franquicia de videojuegos de Final Fantasy, en el cual tanto jugador, personajes como enemigos pueden lanzar sus ataques según el tiempo de carga de cada ataque y podrán lanzarlo en cualquier momento después de que éste esté cargado, ver más información en: Active Time Battle (Concept).

Anexo 4:

Formato de la tabla para el diseño de los casos de prueba.

Elaboración propia.

Caso de prueba	
ID: CP-00N	**Diseñado por:** Nombre de la persona
Prioridad de la prueba (Alto - Medio - Bajo): Alto	**Fecha de prueba de diseño:**
Nombre del módulo: Nombre	**Prueba ejecutada por:** Nombre de la persona
Título de la prueba: Nombre de la prueba	**Fecha de ejecución de la prueba:**
Descripción:	**Condiciones previas:** **Dependencias:**
Pasos: 1. Hacer X 2. Golpear X	
Resultado esperado:	**Resultado Obtenido:**
Estado (Pasa/Falla): Pasar	**Notas:**

Anexo 5: Diagrama completo de arquitectura conceptual de la solución: ModeloConceptual.png

Anexo 6: Cronograma de distribución de las tareas.

Inducción	10días?	07/19/2021	07/30/2021
Introducción al proyecto	5días?	07/19/2021	07/23/2021
Comprensión del código	5días?	07/26/2021	07/30/2021
Sprint 1	20días?	08/02/2021	08/27/2021
Diseño y prototipo de mecánicas de ataque	10días?	08/02/2021	08/13/2021
Diseño y prototipo de mecánicas de defensa	8días?	08/16/2021	08/25/2021
Revisión del sprint	2días?	08/26/2021	08/27/2021
Sprint 2	20días?	08/30/2021	09/24/2021
Diseño y prototipo de mecánicas de evasión	8días?	08/30/2021	09/08/2021
Diseño y prototipo de ataques de rango	10días?	09/09/2021	09/22/2021
Revisón del sprint	2días?	09/23/2021	09/24/2021
Sprint 3	20días?	09/27/2021	10/22/2021
Integración del combate sin cortes	10días?	09/27/2021	10/08/2021
Diseño y prototipado del enemigo 1	9días?	10/11/2021	10/21/2021
Revisión del sprint	1día?	10/22/2021	10/22/2021
Sprint 4	15días?	10/25/2021	11/12/2021
Diseño y prototipado del enemigo 2	13días?	10/25/2021	11/10/2021
Revisión del sprint	2días?	11/11/2021	11/12/2021

Figura 24: Cronograma general estimado del proyecto

Fuente: Elaboración propia

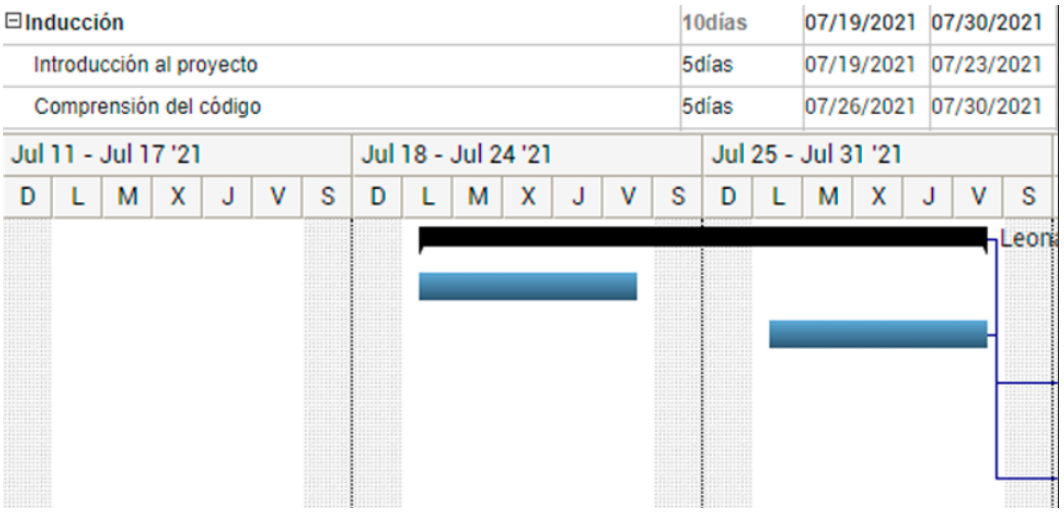

Figura 25: Cronograma de la etapa de inducción.

Fuente: Elaboración propia.

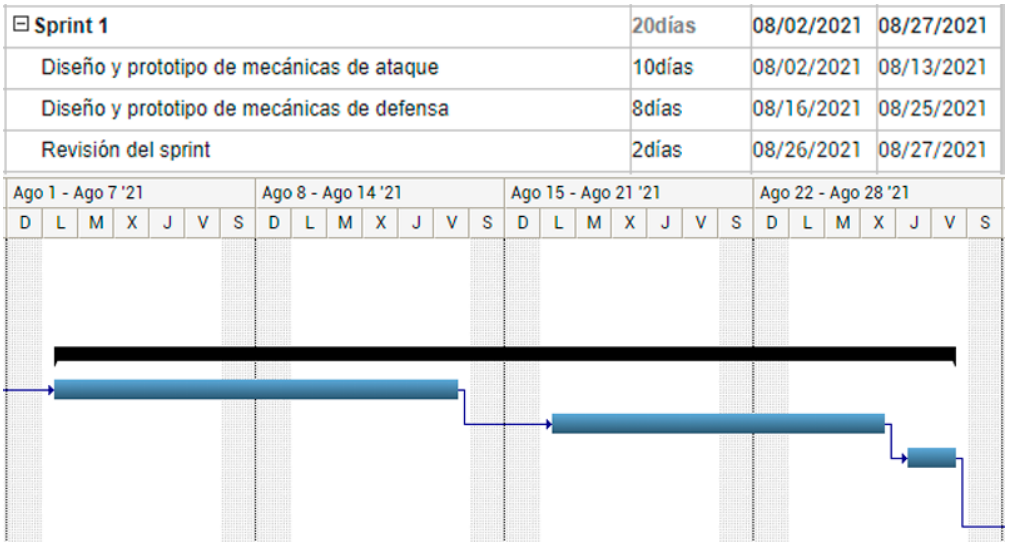

Figura 26: Cronograma del sprint 1.

Fuente: Elaboración propia.

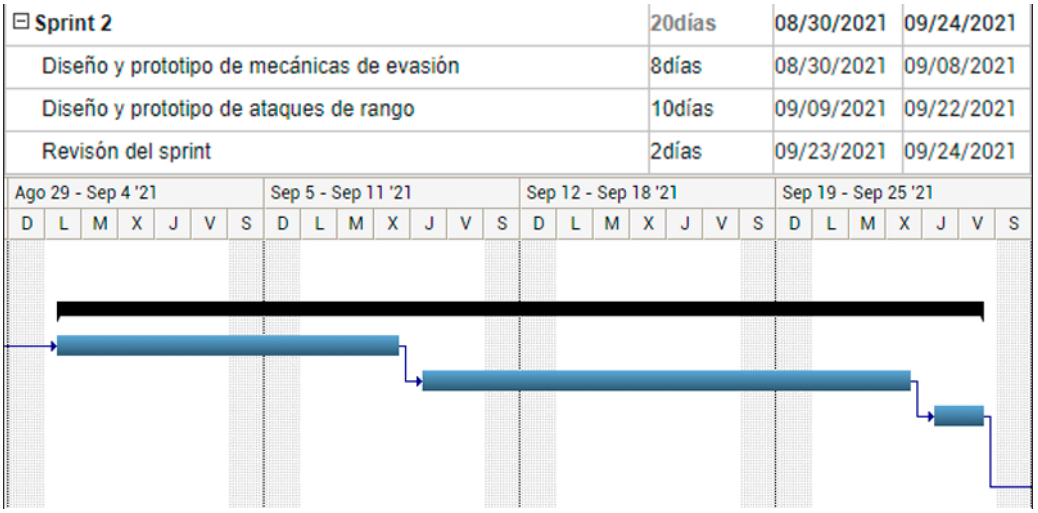

Figura 27: Cronograma del sprint 2.

Fuente: Elaboración propia.

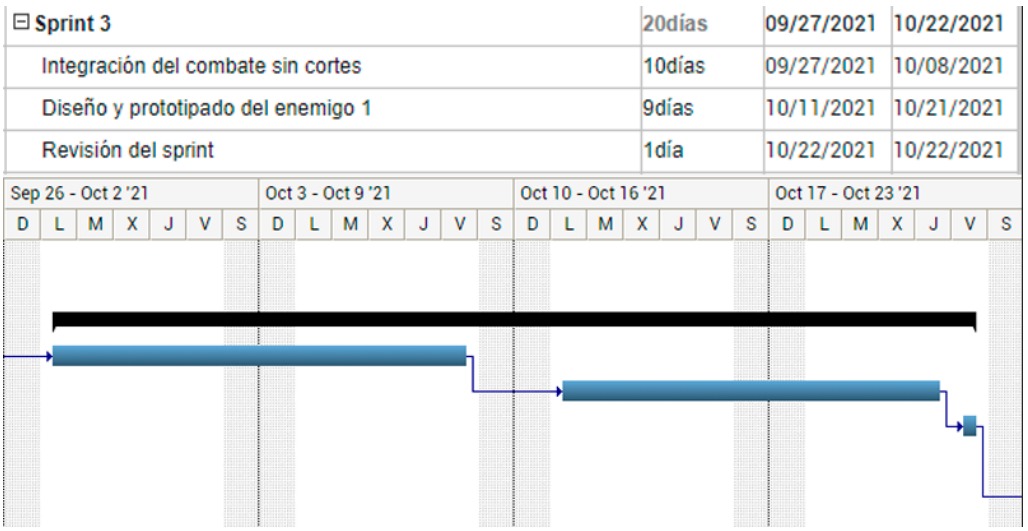

Figura 28: Cronograma del sprint 3.

Fuente: Elaboración propia.

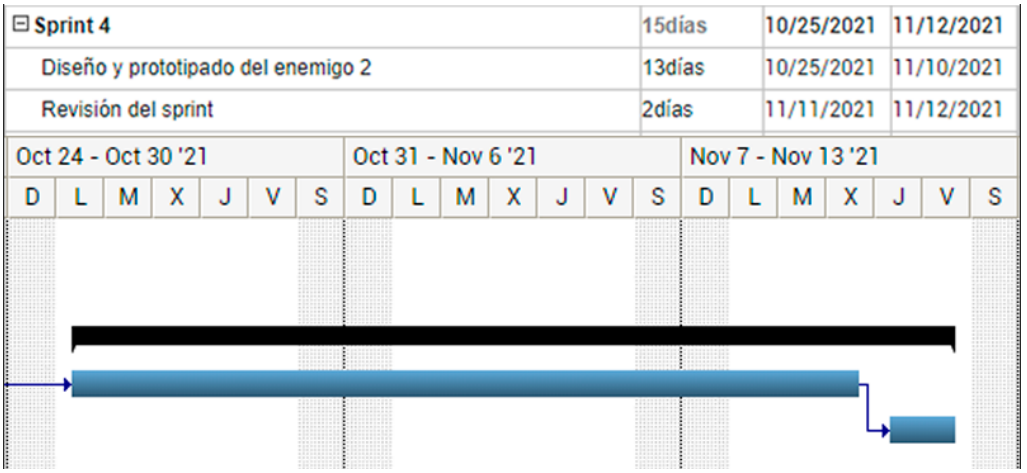

Figura 29: Cronograma del sprint 4.

Fuente: Elaboración propia.