ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE VIDEOJUEGO 2D PARA DISPOSITIVOS ANDROID CON UNITY

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE

JUAN CARLOS BOLAÑOS VIRACOCHA

DIRECTOR: ING. JUAN PABLO ZALDUMBIDE PROAÑO, MSC.

DMQ, septiembre 2022

CERTIFICACIONES

Yo, JUAN CARLOS BOLAÑOS VIRACOCHA declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

JUAN CARLOS BOLAÑOS VIRACOCHA

juan.bolanos@epn.edu.ec

juan.bolanos.v@gmail.com

Certifico que el presente trabajo de integración curricular fue desarrollado por JUAN CARLOS BOLAÑOS VIRACOCHA, bajo mi supervisión.

JUAN PABLO ZALDUMBIDE PROAÑO

DIRECTOR

juan.zaldumbide@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

JUAN CARLOS BOLAÑOS VIRACOCHA

DEDICATORIA

Le dedico el esfuerzo y el tiempo que invertí en este proyecto a las personas que me han apoyado siempre. Mi familia.

AGRADECIMIENTO

Quisiera agradecer a cada una de las personas que me han acompañado en esta etapa de mi vida, ya sean familiares o amigos.

También quisiera hacer una mención especial a todos los profesores de la carrera de Desarrollo de Software por su paciencia y enseñanza. Pues sin su guía, no sería posible estar en este punto de mi carrera universitaria.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

Este documento explica el proceso de elaboración de un videojuego 2D para dispositivos móviles *Android* utilizando la metodología SUM y la herramienta de creación de videojuegos *Unity*. El producto resultante de este proyecto tiene como título: *"Castillo".* Y está pensado para ser un juego multijugador de acción para 2 jugadores en el que el objetivo de cada uno es llegar a su meta a la vez que impiden que el otro jugador también lo haga.

Al ser un videojuego multijugador, utilizaremos los novedosos servicios que Unity proporciona para facilitar la conexión y jugabilidad del videojuego. También, se detallará en el documento las mecánicas de juego que se ha optado por desarrollar para que este fuese entretenido. Este proceso se describe en la sección de resultados y tiene su comienzo desde el nacimiento del concepto del videojuego hasta su correspondiente depuración y lanzamiento en una tienda de aplicaciones.

Cabe recalcar que en el documento no se proporcionará información de la parte artística tales como elaboración de personajes, animaciones o ambientación del juego. Tampoco se explicará de manera profunda el código fuente que usa Unity para el funcionamiento del videojuego. Este documento proporciona en sustento teórico que sigue los pasos de la metodología SUM y las tareas propuestas en la elaboración del *Sprint Backlog* e historias de usuario.

PALABRAS CLAVE: videojuego, SUM, Unity, multijugador, acción

ABSTRACT

This document explains the process of developing a 2D video game for Android mobile devices using the SUM methodology and the Unity video game creation tool. The resulting product of this project has the title: "*Castillo*". And it is designed to be a multiplayer action game for 2 players in which the objective of each one is to reach their goal while preventing the other player from doing so.

We will use the innovative services that Unity provides to facilitate the connection and playability of the video game. Likewise, the document details the game mechanics that have been chosen to be developed so that it is entertaining. This process is described in the results section and has its beginning from the birth of the video game concept to its corresponding debugging and launch in an application store.

It should be noted that the document will not provide information on the artistic part such as character development, animations, or game setting. Nor will the source code used by Unity for the operation of the video game be explained in depth. This document provides theoretical support that follows the steps of the SUM methodology and the tasks proposed in the development of the Sprint Backlog and user stories.

KEYWORDS: videogame, SUM, Unity, multiplayer, action.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El mundo de los videojuegos ha tenido un crecimiento importante en los últimos años gracias a plataformas de retransmisión como *Twitch*, *Youtube*, *Facebook Gaming*, entre otras. Estas páginas con miles de creadores de contenido y una audiencia mundial han popularizado a los videojuegos como un método de diversión, entretenimiento, aprendizaje y hasta rehabilitación para la salud. Ahora mismo, ya no solo la audiencia joven hace uso de los videojuegos, sino también los adultos han encontrado un gran aporte en este método de ocio [1].

En Ecuador, de acuerdo con el censo del 2013 del Instituto nacional de Estadística y Censos (INEC), el 86.4% de los hogares posee al menos un teléfono celular, 36.7 puntos más que lo registrado en 2010 [2]. Es decir, ha existido un aumento de personas que poseen teléfonos inteligentes en los últimos años que podrían fácilmente tener acceso a juegos para dispositivos móviles. En cambio, la industria del desarrollo de videojuegos en Ecuador no ha tenido un crecimiento precisamente importante. En los últimos años, los proyectos locales más exitosos han sido apenas populares, pero muy bien calificadas dentro del mundo de los videojuegos. Como, por ejemplo, "*To Leave*" desarrollado por *Freaky Creations* con un rating de 8/10 [3] o "*Secret of The Lost Keys*" desarrollado de forma independiente por José Luis Escobar con un rating de 7/10 [4]. Estos dos proyectos se diferencian en los equipos de trabajo que los conforman, ya que mientras uno fue desarrollado por un estudio dedicado al desarrollo de videojuegos, el otro fue desarrollado por una persona. La gran ventaja del desarrollador de "*Secret of The Lost Keys*" fue que su producto se desarrolló con uno de los mejores motores gráficos de la actualidad: *Unity* [5]. Este tipo de tecnología ha ayudado al desarrollo de videojuegos independientes ("*indie games*") que no cuentan con el apoyo financiero de distribuidores y usualmente tienen un presupuesto limitado o nulo para producirlos, sin embargo, los desarrolladores no tienen restricciones creativas [6].

Para apoyar al desarrollo independiente de videojuegos en Ecuador este proyecto se enfocará en el desarrollo de un videojuego 2D para dispositivos móviles utilizando *Unity*, uno de los mejores motores gráficos de videojuegos multiplataformas en la actualidad [5]. Además, para lograr este objetivo se hará uso de la metodología ágil SUM, que está basada en la conocida metodología SCRUM, pero adaptada específicamente para la creación de videojuegos [7]. El videojuego que se desarrollará será de "acción" [8]. Específicamente de lucha y peleas en el que dos usuarios tratarán de llegar a sus

respectivas metas durante una determinada cantidad de rondas. Durante estas rondas los usuarios podrán hacer uso de una determinada cantidad de disparos de proyectiles. El juego está pensado para dispositivos Android.

1.1 Objetivo general

Desarrollar un videojuego 2D del tipo Acción para dispositivos móviles, utilizando el motor de videojuegos Unity y aplicando la metodología SUM

1.2 Objetivos específicos

1. Conceptualizar la idea del videojuego.
2. Planificar los requerimientos que definen el videojuego.
3. Elaborar el videojuego utilizando las herramientas que *Unity* proporciona.
4. Entregar versiones Beta evaluando y ajustando distintos aspectos del videojuego en cada una de estas.
5. Entregar la versión final del videojuego.
6. Gestionar los riesgos durante todas las fases del proyecto.

1.3 Alcance

El presente proyecto propone el desarrollo de un videojuego desde su conceptualización, desarrollo y pruebas, hasta el lanzamiento de su versión final. El videojuego tendrá algunas versiones beta en las cuales se harán pruebas para corregir los posibles errores que se tenga durante la fase de desarrollo. Por otra parte, el proyecto será desarrollado en base a la metodología SUM, la cual está basada en Scrum. Es decir que, de igual forma tiene un sistema de entrega a través de iteraciones.

Es importante recalcar que tanto la programación y el diseño de *mockups* estarán a cargo del equipo de desarrollo. Pero, los *sprites* o imágenes que serán utilizados para la animación de personajes de combate serán adquiridos legalmente y con los derechos de *copyright* a través del sitio web *graficriver.net* [9].

El videojuego constará de 2 pantallas esencialmente. La pantalla de inicio que servirá para conectarte con otro jugador a través de un código y la pantalla de juego en donde 2 jugadores conectados se enfrentarán en un escenario que simulará un edificio. Su objetivo es llegar a la base enemiga mientras se diputan batallas cortas ente ellos usando una

determinada cantidad de proyectiles que se les dará. Las bases estarán ubicadas en las plantas más baja y alta del edificio. Además, cada vez que un jugador muera, reaparecerá después de un determinado tiempo para dificultar que el otro jugador llegue fácilmente a su base. El producto final será un videojuego multijugador para 2 personas para dispositivos Android que se podrá descargar de forma gratuita.

1.4 Marco Teórico

La ciencia avanza al aprender cómo funcionaban las cosas en el pasado y luego mejorarlas. Los videojuegos son bastante parecidos en el sentido de que aprenden cómo funcionaban las cosas en el pasado, las mejoran y luego entregan el producto final para el entretenimiento de cualquier usuario. Aunque, la mejora real de los juegos está en los ojos del espectador, pocos pueden argumentar los avances extremos que se han realizado en la tecnología de los juegos, tanto visualmente como en capacidad [10].

Actualmente los videojuegos están enfocados en los gráficos 3D que tienen una calidad que hace que se vean realistas. Sin embargo, muchos jugadores desean algo clásico. Algunos quieren una experiencia de juego accesible, sin la complejidad que implica el uso de los ejes del espacio tridimensional. Es por estos jugadores que el arte de los juegos 2D ha revivido [10].

Videojuegos 2D

Los videojuegos 2D utilizan gráficos planos, llamados *sprites*, tienen geometría bidimensional, es decir con solo 2 ejes de movimiento. Se dibujan en la pantalla como imágenes planas, y la cámara (cámara ortográfica) no tiene perspectiva [11]. Pero esto no fue así desde siempre. A finales de los 60 aparece el que es considerado por muchos como el primer videojuego de la historia: *Tennis for Two.* El cual únicamente se podía jugar en un osciloscopio [12].

Esto abrió el camino para que posteriormente a finales de los años setenta y principios de los ochenta se popularizaran los videojuegos, pero en especial los pertenecientes al género de plataformas [13]. Un género en donde el personaje debe saltar y escalar entre plataformas repartidas por el mapa que generan dificultad para el jugador [13].

Con la herramienta *Unity* se puede crear este tipo de experiencias interactivas. Los editores *Tilemap* se usarán para mosaicos cuadrados, hexagonales e isométricos, animación basada en huesos, y la posibilidad de crear fácilmente *shaders* y luces 2D [11].

Videojuegos para dispositivos móviles

Aunque los videojuegos para móviles suenen que tuvieron éxito hace relativamente poco, en 1997 Nokia lanzó su legendario juego titulado *Snake* como una característica de su teléfono celular Nokia 6110. Los usuarios quedaron fascinados con lo entretenido y simple que este título resultaba ser. Tanto, que Nokia lo hizo parte indispensable de cualquier dispositivo que lanzaran [14].

Es así como la industria *gamer* fue evolucionando y mejorando la experiencia en videojuegos para dispositivos móviles hasta la actualidad. Y, con la implementación de tecnologías como *Wireless Fidelity (WiFi),* empresas que comúnmente desarrollaban videojuegos para consolas o PC's como *Activision, Riot Games, Electronic Arts,* entre otras, han destinado grandes presupuestos para adentrarse en el mundo de los juegos para móviles. Todo esto gracias a las ganancias que ofrece un modelo de negocio *freemium*, el cual consiste en permitir la descarga gratuita a cualquier usuario y ofrecerle mejoras para optimizar la experiencia a través de micro transacciones dentro del propio videojuego [14].

Unity

Unity Technologies es una empresa de desarrollo de software destinada para la creación de Videojuegos con sede en San Francisco, y es conocida por crear *Unity,* un motor de videojuegos [15]. En el año 2004, tras el fracaso de lanzar el videojuego "*Goobal",* se decide transformar a la compañía y utilizar las herramientas que se desarrolló en el último lanzamiento para crear un motor que todos puedan utilizar por igual, a través de un entorno amigable para programadores y diseñadores [15].

Desde entonces *Unity* ha lanzado múltiples versiones. Al principio como un motor exclusivo para sistemas operativos *OSX* y soporte únicamente para *iOS.* Pero a partir de 2010 se incluye el soporte para *Android, Mecanim, DX11, Linux, Flash, WebGL, Audio Mixer,* entre otros.

El presente trabajo realizará el desarrollo de un videojuego con la tecnología de juegos 2D que nos proporciona *Unity* combinándolo con los servicios de *Gaming* que recientemente ha hecho público, tales como *Multiplayer, Analytics, DevOps o Monetization*. El videojuego por desarrollar, en principio, podrá jugarse desde cualquier dispositivo Android. Desde luego, esto puede cambiar a medida que el juego se elabore, restringiéndose a teléfonos con versiones de Android más antiguas.

2 METODOLOGÍA

2.1 Metodología de Desarrollo

Para el desarrollo del videojuego el equipo ha optado por escoger la metodología *SUM* la cual tiene como objetivo desarrollar proyectos enfocados a videojuegos en un tiempo óptimo ofreciendo calidad y bajos costos en sus procesos, inclusive se puede utilizar como la mejora continua de sus procesos para de esa forma lograr incrementar la eficiencia y eficacia. Gracias a esta metodología el equipo de desarrollo logra alta productividad en utilizar los recursos y obtener resultados predecibles para así gestionar los riesgos del proyecto [16].

Para la creación de videojuego el procedimiento es un trabajo complejo, usualmente el equipo de desarrollo tiende a emplear metodologías de software ya conocidas o generales. El desarrollo de videojuego no sigue específicamente una metodología tradicional para su creación. Sin embargo, existen varios procesos de distintas metodologías que se adaptan a facilitar su creación [17]. *SUM* emplea un método acorde a las características que debe cumplir el proyecto incluyendo los recursos que tiene a disposición, el equipo de trabajo, etc. [18].

SUM se adapta a trabajar en el mismo lugar o de manera distribuida, se ha concebido que en este tipo de metodología pueden trabajar de 2 a 7 integrantes en un equipo. *SUM* propone que para proyectos cortos la duración debe ser menor a la de un año siempre y cuando el cliente que solicito el trabajo participe en todas las actividades [16]. *SUM* se compone de la siguiente estructura: concepto, planificación, elaboración, etapa beta, y cierre [18]. Las fases de concepto, planificación y cierre solo tienen una iteración, mientras que la elaboración y beta cuentan de varias iteraciones.

Concepto: se define la temática del videojuego. Esto implica los aspectos como público, objetivo, modelo de negocio, elementos que van a contener el juego, personajes, historia, idioma y por supuesto la herramienta de desarrollo [18]. Se definirá también cómo será el sistema de lucha, cuánta vida se le quita a un jugador después de cada golpe y los ítems que se utilizarán para dar ventaja al jugador que los obtenga.

Planificación: se basa en planificar las otras fases del proyecto, se definen cronogramas los cuales van a especificar las principales actividades del proyecto, conformación del equipo para la elaboración, tercerización de tareas, las necesidades y técnicas del proyecto, y por supuesto definir el presupuesto y las características del juego [16]. Definir

las características o requerimientos funcionales y no funcionales del videojuego es una parte esencial dentro del proyecto. Algunos de los requerimientos que se discutirán son acerca de la jugabilidad, controles, tiempo de juego, menús, entre otros.

Elaboración: su objetivo es simplemente implementar el desarrollo del videojuego, se divide en 3 etapas, que son la planificación de objetivos, ejecución de las características planificadas y realización de la evaluación del estado en el que se encuentra el videojuego [18]. El videojuego utilizará una interfaz sencilla que permita iniciar al juego casi al instante de abrir la aplicación móvil.

Beta: sus objetivos son evaluar y ajustar varios aspectos en el videojuego como los errores detectados y eliminarlos. Para esta fase los desarrolladores realizan sus pruebas en distintas versiones, envían reportes con las pruebas realizadas ya sean positivas o negativas, las pruebas se las analiza para verificar si el videojuego necesita un reajuste o simplemente liberar una nueva versión [16]. En este punto el juego debe funcionar en un 80%, asegurando la jugabilidad del combate, el sistema de puntaje y la conexión multijugador online debe ser estable.

Cierre: es el último paso de la metodología. Si el equipo ha logrado llegar a esta fase es cuando se realizó un lanzamiento de la versión estable y final del videojuego. Se evalúa problemas, soluciones, éxitos, de los objetivos que se cumplieron en la elaboración de este [18]. El sistema debe tener la calidad para que se pueda implementar actualizaciones con nuevos ítems o skins de personajes de forma sencilla

Roles

La metodología de desarrollo llamada SUM que usaremos está basada en Scrum y por lo tanto varios de los roles que especifica en su arquitectura pueden parecer similares. Así entonces contamos con 4 roles los cuales son cliente, producto interno, equipo de desarrollo y verificador beta.

Cliente: Este rol te obliga a ser el representante de las personas a quienes les interesa los resultados del proyecto [19]. Entre las responsabilidades del cliente se encuentran actualizar el plan del proyecto constantemente, definir aspectos técnicos, de negocios o de cronograma, o definir objetivos y métricas. Todas estas tareas desde el día uno del proyecto hasta su correspondiente entrega final [19]. Algo parecido al cargo de “*product owner”* en Scrum.

Productor Interno: Esta persona guiará el desarrollo del videojuego además de ser el intermediario entre el cliente y el equipo de desarrollo. Dentro de las tareas que debe

cumplir el productor interno se encuentran: definir los aspectos del juego, presupuestos, calendarios; evaluar de forma constante el estado del videojuego incluso después de su entrega final; monitorear riesgos en cada iteración; entre muchos otros [19]. Este rol en Scrum es el conocido como *Scrum Máster.*

Equipo de desarrollo: Debido a que un videojuego no se construye con solo un equipo de programadores, SUM subdivide al equipo de desarrollo que dependiendo de su tamaño se asignaran más o menos personas en cada grupo. Estas subcategorías dentro del equipo de desarrollo son 4: el diseñador de juego, quien es el que diseña todos los elementos relacionados con la experiencia del jugador como mapas, jugabilidad, personajes o niveles; programadores, los cuales son los encargados de implementar el software con el que se construye el juego; artista sonoro, que se ocupa de ambientar el juego con efectos o música; y artista gráfico, cuyo trabajo es el de crear el arte del juego, es decir, la estética en los modelos 2D o 3D, así como en animaciones o texturas [19].

Verificador Beta: Dentro del mundo de los videojuegos, a este grupo de personas se los conoce como *testers* porque precisamente por su traducción al español son los encargados de probar y detectar errores que puedan arruinar la experiencia del jugador [20]. Además, especialmente importantes en la etapa Beta del videojuego [19].

La TABLA I muestra las personas y los roles que cumplen dentro de este proyecto

TABLA I: Roles del Proyecto

Rol	**Encargado**
Cliente	Juan Bolaños
Productor Interno	Juan Pablo Zaldumbide
Equipo de desarrollo	Juan Bolaños
Verificador Beta	Juan Bolaños

Artefactos

Pese a que SUM está basado prácticamente en Scrum. La metodología que utilizamos no es rigurosa con los artefactos al momento de planificar y desarrollar nuestro proyecto. En cambio, incentiva a adaptar la documentación según sus necesidades, poniendo a elección si utilizar o no los artefactos que propone la metodología Scrum [21].Los Artefactos producidos para este proyecto son los siguientes:

Concepto del Juego

El presente documento describe aspectos del juego como la jugabilidad, diseño artístico, ambientación, forma de conexión multijugador. Dependiendo del tipo de proyecto y de si existe un propósito monetario o de beneficios se puede añadir en el documento planes de modelo de negocio o de marketing. En general, el concepto del videojuego busca orientar o informar a todos los que estén relacionados con el proyecto con el fin de asegurar su calidad.

Historias de Usuario

Una historia de usuario es una forma de describir una función puntual que se requiere que tenga el software y que necesita de ser desarrollada para el usuario final. Además, son un componente clave dentro de las metodologías de desarrollo ágil ya que hacen que el usuario final esté en el centro de la conversación al momento de construir el videojuego.

La TABLA II muestra una de las historias de usuario del proyecto como ejemplo

TABLA II: Historia de Usuario No. 1

	Historia de Usuario
Identificador (ID): HU001	**Usuario:** Jugador
Nombre de Historia: Ejecutar Juego	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Responsable: Juan Bolaños	
Descripción: El jugador podrá ejecutar el juego en su dispositivo Android en forma de una aplicación que será proporcionada junto con este proyecto.	
Observaciones:	

Plan del Proyecto

La TABLA III muestra parte del plan del proyecto elaborado.

TABLA III: Formato del Plan de Proyecto Utilizado

Plan de Proyecto				
HU-ID	**Actividad**	**Iteración**	**Estado**	**Prioridad**
TODAS	Conceptualizar el Videojuego	1	En progreso	Alta
N/A	Diseñar el arte del juego	1	En progreso	Alta
HU008	Construir el escenario de combate	1	En progreso	Alta

N/A	Conseguir Sprites de movimiento para los personajes	1	Finalizado	Alta
HU010, HU012, HU013, HU014, HU015	Codificar movimientos de personaje	1	Pendiente	Alta

Sprint Backlog

La TABLA IV muestra el formato utilizado con el primer sprint del proyecto.

TABLA IV: Sprint Backlog de la Iteración No. 1

Sprint Backlog					
SB-ID	**Nombre**	**HU-ID**	**Historia de Usuario**	**Tareas**	**Tiempo Estimado**
SB001	Conceptualización, Diseño y Codificación de Mecánicas Básicas	N/A	Todas	• Configurar ambiente de desarrollo • Familiarización con la herramienta Unity	3 semanas
		Todas	Todas	• Conceptualizar el videojuego	
		N/A	N/A	• Diseñar escenarios, menús, personajes • Diseñar la estética del juego	
		HU008, HU009, HU010, HU012,	Escenario de Combate, Generación	• Codificar mecánicas básicas de un personaje	

		HU013, HU014, HU015	de personajes en el escenario de combate, Saltar, Correr, Usabilidad del joystick, Botón de Disparar, Botón de saltar	• Codificar Interfaz de Usuario para las mecánicas básicas	

2.2 Diseño de interfaces (mockups)

Herramienta utilizada para el diseño

Para el presente proyecto se decidió utilizar herramientas fáciles de utilizar, pero también aprender. Estas herramientas son*: Figma* y *Adobe Photoshop.* Los Mockups y la tipografía de las diferentes pantallas fueron diseñadas con la herramienta *Figma*, que es un sitio web que se utiliza como editor de gráficos y generación de prototipos. Es de uso libre y se puede utilizar tanto en macOS como en Windows. En cambio, el escenario de combate y algunos elementos de este fueron diseñados en Adobe Photoshop, tomando como referencia imágenes de Google y retocadas con este software que no es gratuito y se necesita una licencia de *Adobe Systems Incorporated.*

Diseño de Pantallas de Menú y Opciones

La mayoría de los juegos en la actualidad constan de un menú principal en el que el usuario puede elegir entre varias opciones de configuración para mejorar la experiencia del jugador. En este caso y por el tiempo en el que este proyecto debe ser desarrollado los menús tratan de ser lo más simple y minimalistas posibles.

El menú principal consta de 3 opciones: jugar, ir a la pantalla de créditos y salir del juego. Estas funciones estarán implementadas en botones que redirigirán a nuevas pantallas. La opción de “Jugar” redirigirá al jugador a una pantalla en donde elegirá entre dos opciones:

“Crear una partida” o “Unirse a una partida”. Cada una de estas desplegara una nueva pantalla en donde se otorgará o se pedirá que se ingrese un código de 6 dígitos según el corresponda.

Fig. 1: Ilustración menú principal

Diseño de escenario de combate

Como ya hemos mencionado, en este proyecto se desarrollará un videojuego de Acción. Es decir, existirá un combate entre dos jugadores. Entonces, el escenario está pensado para que uno de los dos sea el ganador en partidas relativamente cortas pero dinámicas. Tal y como indica el título del juego, el escenario de combate intenta imitar lo que sería un castillo. Consta de pisos por los que los jugadores deberán avanzar si desean ganar la partida.

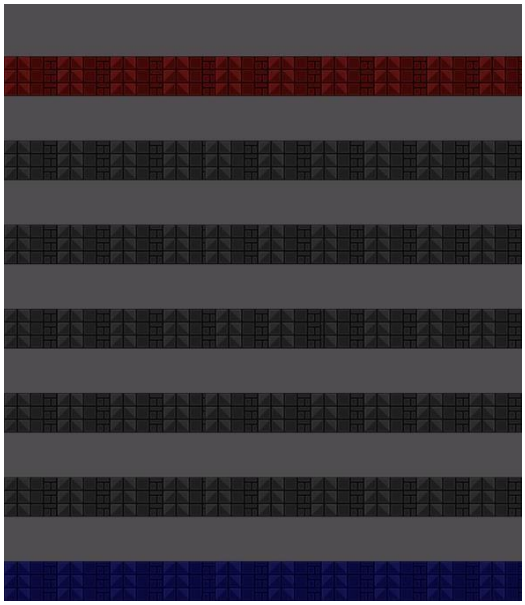

Fig. 2: Escenario de Combate sin decoraciones

Diseño de Personajes

Debido al poco tiempo que se dispone para el proyecto, los personajes que se utilizarán para combatir son simples y sin ninguna personalización en especial. También son llamados popularmente como *stickman.* Y los *sprites* que utilizaremos cumplen con los requisitos para animar movimientos como correr, saltar, morir, disparar o descansar.

Como ya he mencionado en el Alcance del proyecto, estos *stickman* y sus *sprites* de movimiento no fueron diseñados por el integrante del proyecto. Fueron adquiridos a través de una tienda en línea [9].

Fig. 3: Sprite de Correr-1

2.3 Diseño de la arquitectura

Assets

Un *asset* es la representación de cualquier ítem u objeto que pueda ser utilizado en la compilación del videojuego. Podría ser un *sprite*, un archivo de audio, un video o cualquier otro tipo de formato que sirva para su implementación en el juego. Pero también pueden ser otro tipo de archivos que manejen el comportamiento de los diferentes objetos existentes en el proyecto, tales como controladores o scripts.

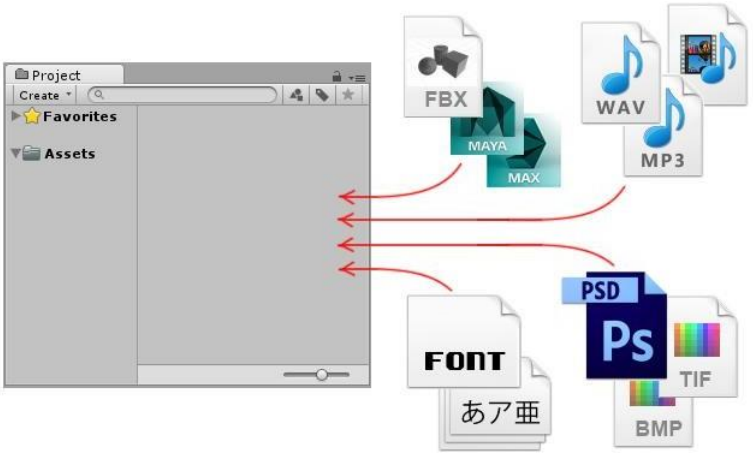

Fig. 4: Ejemplos de Archivos que podrían ser Assets

Escenas

Contienen los objetos o *GameObjects* de nuestro juego. Pueden usarse para navegar entre pantallas o niveles dependiendo de cómo se configuren. Cada escena debe cargarse completamente para desplegarse y puede tardar un buen tiempo según la complejidad o la cantidad de objetos que tenga. Es por eso por lo que comúnmente se agrega una mini pantalla de carga cuando se cambia de escena.

En nuestro caso tendremos 3 Escenas. 2 de tipo menú y 1 que será la escena de combate.

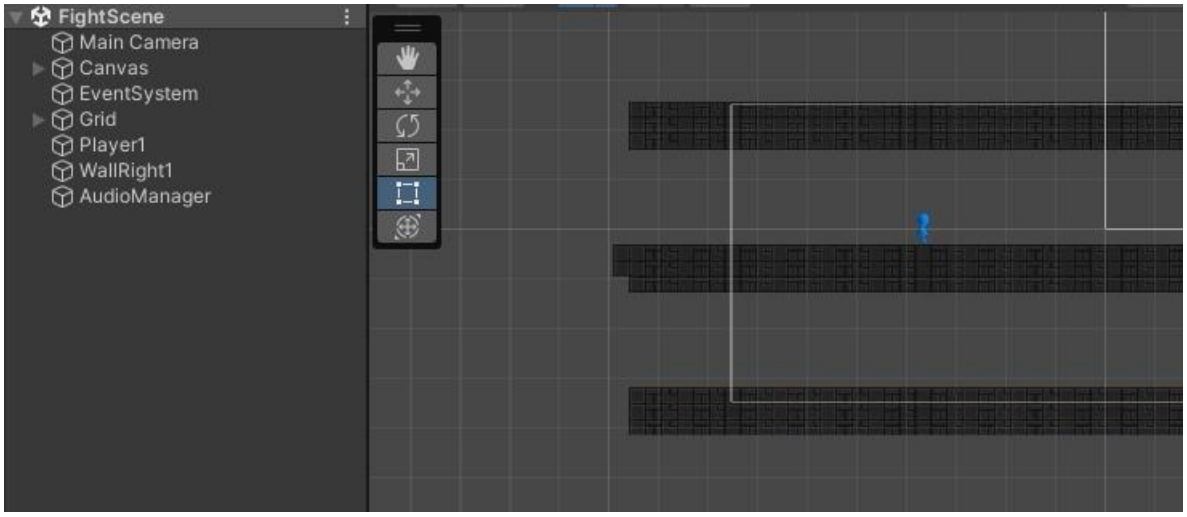

Fig. 5: Escena de combate y sus GameObjects

Game Objects

Un *Game Object* es cualquier objeto que sirva para una representación gráfica dentro de una Escena. Un *Game Object* sin ninguna propiedad no realiza ninguna acción, pero al funcionar como contenedores para componentes se le pueden agregar propiedades como que pueda colisionar, tenga físicas de movimiento y por lo tanto gravedad, o que cumpla con una animación determinada, entre muchas otras.

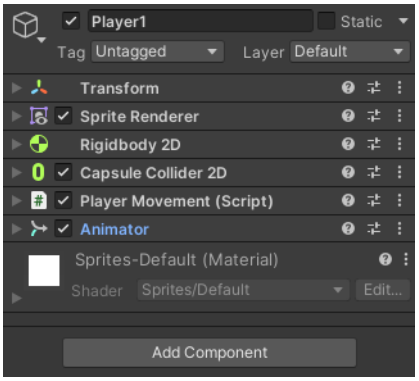

Fig. 6: Propiedades del GameObject Player1

Aplicación Móvil

Unity cuenta con la posibilidad de exportar tu juego en más de 25 plataformas distintas. En nuestro caso queremos exportarla en un ambiente para dispositivos Android. Para esto debemos configurar las *Build Settings* adecuadamente

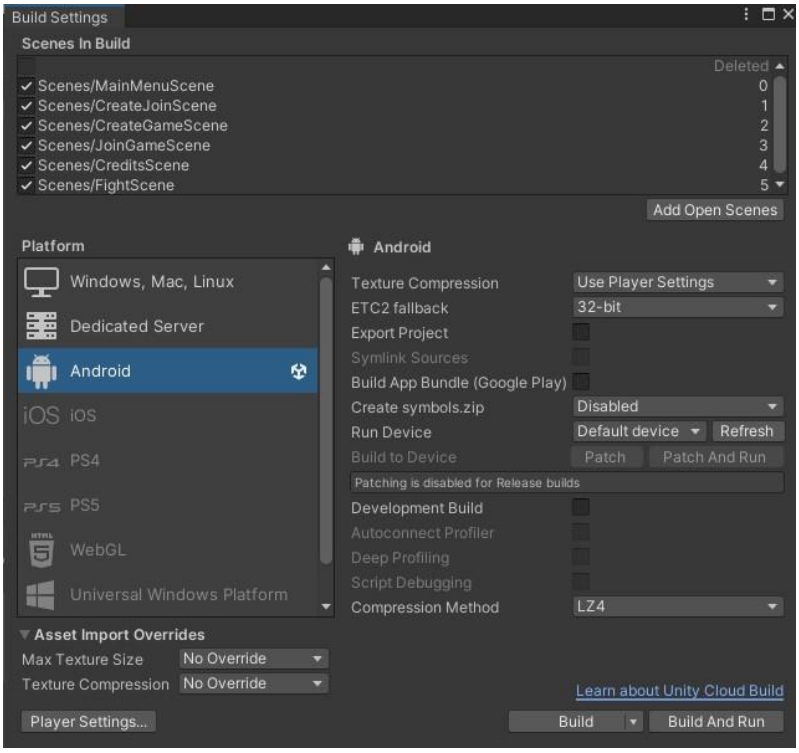

Fig. 7: Pantalla de Build Settings para Android

2.4 Herramientas de desarrollo

A continuación, se presentan las herramientas que se utilizó para el desarrollo del presente proyecto.

TABLA V: Herramientas para el desarrollo del videojuego

Herramienta	Justificación
Figma	Es una herramienta fácil de utilizar y aprender. Cuenta con una interfaz de usuario intuitiva y la capacidad de modificar pantallas u objetos con las características que comúnmente presentan elementos de la mayoría de los sitios web en el mundo. Aunque no está especialmente diseñada para videojuegos, en este proyecto se lo utiliza para incluir elementos como botones o texto como parte de la Interfaz de Usuario.
Adobe Photoshop	Aunque no es una herramienta tan intuitiva como lo es *Figma*, Adobe Photoshop nos da una gran cantidad de

	opciones al momento de editar una imagen y darle el aspecto que se requiere. En este proyecto se lo utilizo para modificar imágenes de uso libre de internet e implementarlas en el escenario de combate o en los menús.
Unity	Un programa que está diseñado para crear videojuegos y cuenta con una interfaz complicada al principio pero que conforme vas avanzando en el proyecto se vuelve más fácil de utilizar. Además, cuenta con la capacidad de desplegar aplicaciones tanto de escritorio como de móviles. Por lo que es perfecto para este proyecto. Dentro de la arquitectura de trabajo, nos encontramos con la carpeta de *Assets*, el cual contiene toda la información para el desarrollo del juego
Visual Studio Community 2019	Dentro de la carpeta *Assets*, se encuentra una carpeta llamada Scripts, el cual contiene todos los *Scripts* de comportamiento de los diferentes objetos del videojuego. Visual Studio nos permite codificar de forma rápida y sencilla ya que el lenguaje que utilizamos es C#.

3 RESULTADOS

El desarrollo de este proyecto se dividió en iteraciones o *Sprints* como está especificado en la metodología *SUM*, a continuación, se describen las actividades realizadas en cada uno de estos *Sprints* en concordancia con el *Sprint Backlog* y los resultados obtenidos luego de realizarlas.

Sprint 1. Conceptualización, Diseño y Codificación de Mecánicas Básicas

Para este proyecto, la configuración del ambiente de desarrollo no es muy complicada. Por los que se decidió unir el Sprint 0 con las tareas del primer Sprint. En esta iteración se manejarán tareas como las siguientes:

- Configurar ambiente de desarrollo
- Familiarización con la herramienta Unity
- Conceptualizar el videojuego
- Diseñar escenarios, menús, personajes
- Codificar mecánicas básicas de un personaje
- Codificar Interfaz de Usuario para las mecánicas básicas

Configurar ambiente de desarrollo

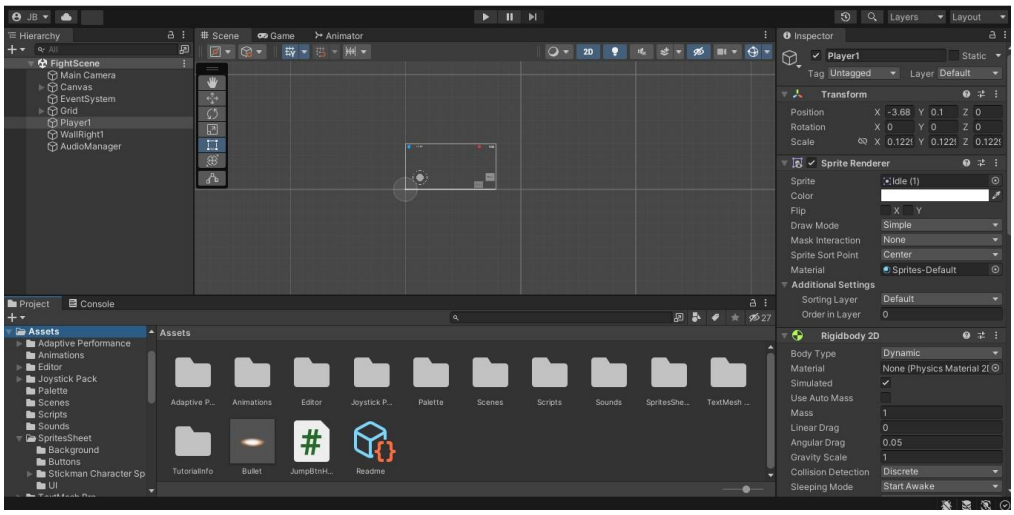

Fig. 8: Pantalla inicial de Unity

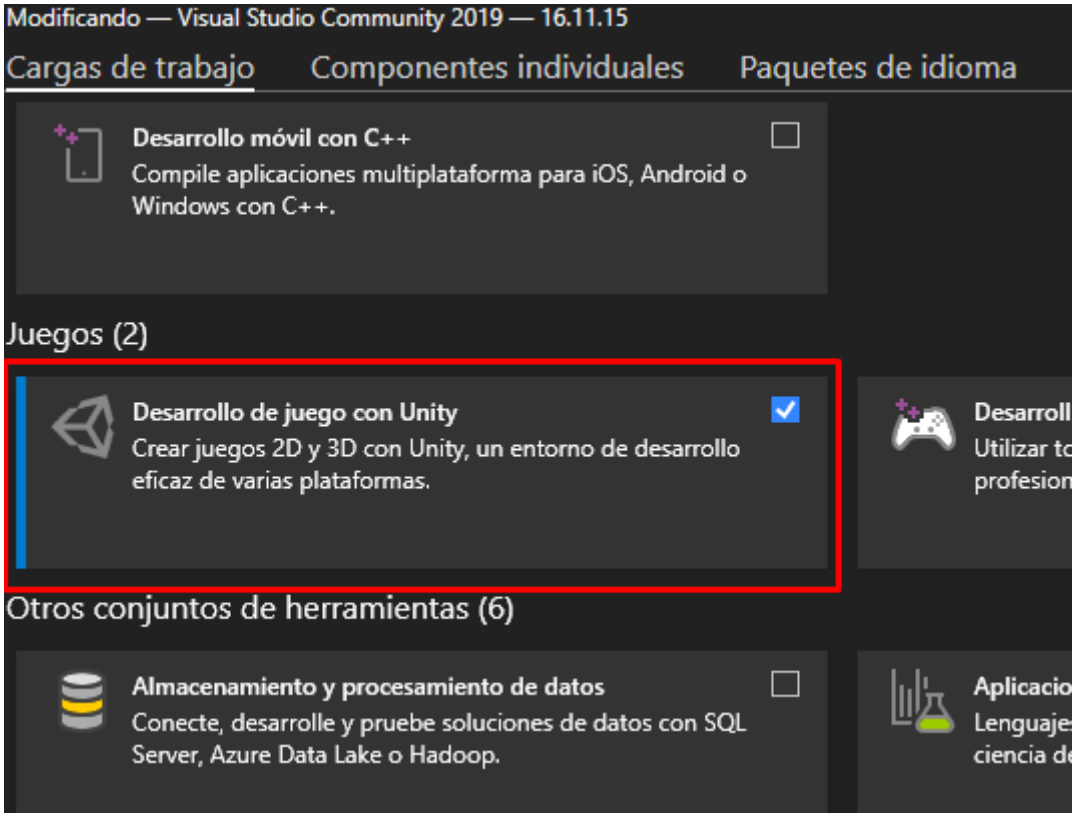

Fig. 9: Configuración de Visual Studio 2019 para trabajar con Unity

Familiarización con la herramienta Unity

No hay nada mejor para aprender a usar una herramienta que tener una o varias guías que detallen su uso y lo que se puede lograr con la misma. Para dar nuestros primeros pasos en este proyecto se tomó como referencia unos cuantos videos tutoriales gratuitos que se pueden encontrar en *Youtube* y por supuesto la documentación oficial del equipo de *Unity.*

Esta es una lista de los tutoriales que se tomaron en cuenta:

- Tutorial COMPLETO Unity 2D desde Cero [22]
- Joystick para Móviles en Unity/Mobile Input/Control táctil/Thumbstick/Virtual Joystick [23]
- How To Make A Game With Unity Multiplayer Netcode | Relay Service Setup [24]

Conceptualizar el videojuego

En la fase previa al desarrollo en sí del videojuego es importante tener claro que es lo que queremos representar o codificar. Es por eso por lo que la conceptualización del

videojuego nos ayuda a tener claras nuestras expectativas en cuanto a cómo queremos que sea el producto final.

Escenario:

Este compuesto por 7 pisos con la estética de un castillo. Cada piso servirá como campo de batalla para los jugadores. También posee dos metas en los pisos inferior y superior que los jugadores tienen como misión alcanzar para ganar la partida

Personajes:

- **Jugador 1 –** Es un *stickman* de color azul que puede saltar, correr y disparar. Su meta es llegar al piso inferior de color azul.
- **Jugador 2 –** Es un *stickman* de color rojo que puede saltar, correr y disparar. Su meta es llegar al piso superior de color rojo.

Gameplay:

El jugador controlará a uno de los dos personajes y su objetivo es llegar al piso inferior o superior según corresponda el color. El jugador tendrá una vida de 100 unidades y su contrincante también. Además, poseerá 100 proyectiles que se recargarán luego de 3 segundos si se terminan.

- Saltar – Mientras saltas no podrás disparar proyectiles, pero te servirá para esquivar los del rival recibiendo menos daño y por tanto resistir para poder atacar de mejor forma a tu contrincante.
- Disparar – Mientras disparas no podrás saltar. Además, si mantienes pulsado el botón disparar, los proyectiles se lanzarán de forma consecutiva para otorgar más daño a tu rival en menos tiempo.

Para poder avanzar por los pisos del castillo el jugador primero debe matar a su contrincante disparándole proyectiles hasta quitarle todas sus unidades de vida. Si su contrincante lo mata primero habrá un tiempo de espera para reaparecer. Si no matas primero a tu contrincante no se te permitirá avanzar a otro piso. Esto será indicado por un aviso en la parte superior central de la pantalla.

El jugador ganará la partida cuando llegue a su correspondiente meta

Diseñar escenarios, menús, personajes

Con la ayuda de las Herramientas de diseño mencionadas en la Sección 2, fue posible crear el arte del videojuego, así como también el escenario de combate. En este caso *Figma* ayudó a la creación de los menús y elementos interactivos del mismo, tales como botones o texto.

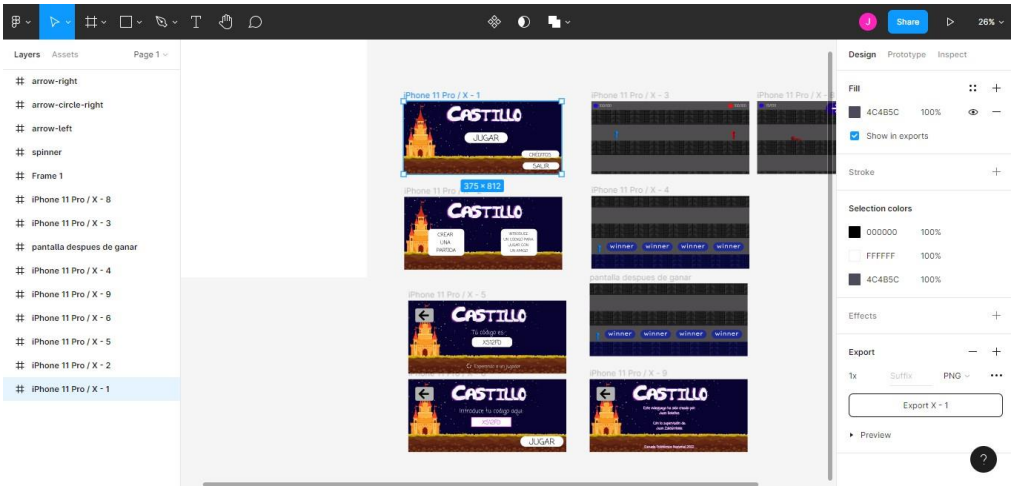

Fig. 10: Pantalla de la herramineta Figma

Por otro lado, para la creación del escenario de combate se utilizó la herramienta de Adobe Photoshop para utilizar una imagen encontrada en Google Imágenes y modificarla para crear el mapa en el que los jugadores pelearán.

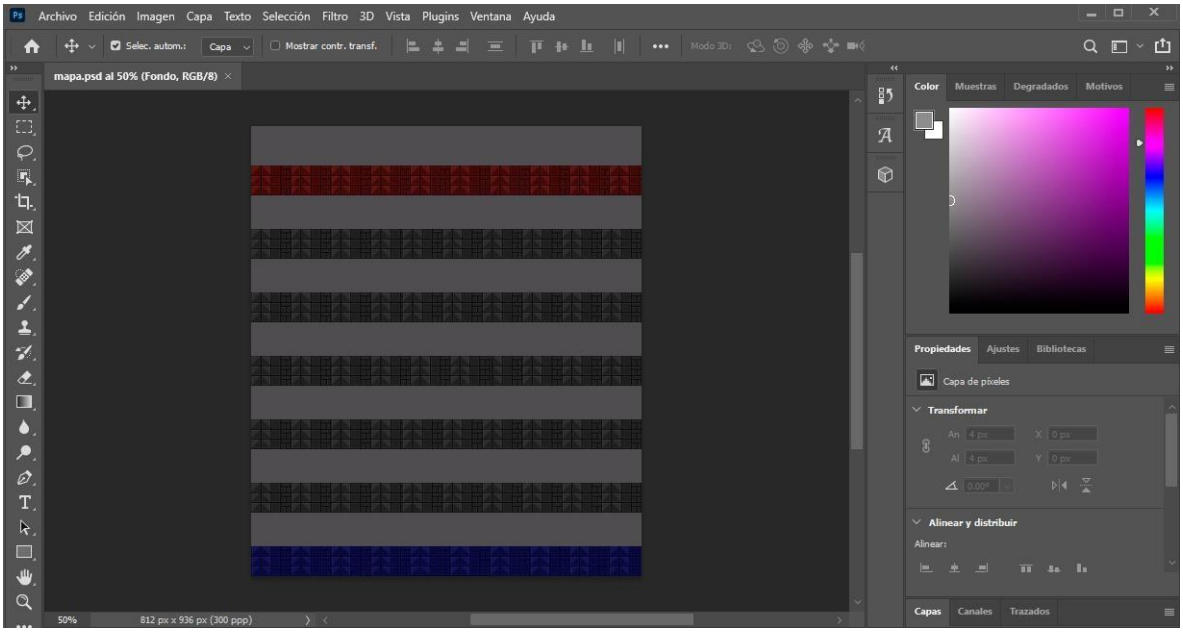

Fig. 11: Pantalla de Adobe Photoshop para creación del Escenario de Combate

Codificar mecánicas básicas de un personaje

A pesar de que muchas de los requerimientos que fueron pensados en la etapa de conceptualización del videojuego puede que cambien a medida que se va avanzando en el proyecto. Las mecánicas de movimiento para un jugador cuando el combate ha iniciado son básicas y sirven para avanzar por el mapa. De acuerdo con las historias de usuario un personaje debe tener la capacidad de: correr, saltar, y disparar.

Correr. - Para implementar la función de correr a nuestro personaje hay que tener en cuenta que cualquier objeto de *Unity* posee la Propiedad *transform,* la cual nos permite saber en qué posición del mapa se encuentra situado el objeto. En este caso, nuestro personaje. Es por esto por lo que para cambiar de posición y dar la ilusión de movimiento lo que se hace es modificar la posición en cada *frame*. Es decir, dentro de la función *update()* que se repite 60 veces por segundo.

```
//Movimiento horizontal
horizontalMove = joystick.Horizontal * runSpeedHorizontal;
transform.position += new Vector3(horizontalMove, verticalMove , 0) * Time.deltaTime * runSpeed;

if (horizontalMove < 0.0f) transform.localScale = new Vector3(-0.12f, 0.12f, 0.12f);
else if (horizontalMove > 0.0f) transform.localScale = new Vector3(0.12f, 0.12f, 0.12f);
```

Fig. 12: Implementación de movimiento horizontal del personaje

Saltar. – Para saltar, primero implementamos una nueva propiedad a nuestro personaje. RigidBody2D es el componente de Unity que nos permite incluir físicas a nuestros objetos. La gravedad y la rotación son elementos que podemos controlar con este interesante componente. Pero para complementar su uso hay que darle la capacidad de que nuestro personaje pueda chocar con otros objetos. Para esto agregamos el componente *CapsuleCollider2D,* que le permite poder interactuar de forma física con otros elementos del mapa. Ahora bien, para ejecutar el salto simplemente cada vez que el salto sea pulsado modificamos la posición en vertical del personaje. El resto del movimiento lo hará los componentes que agregamos.

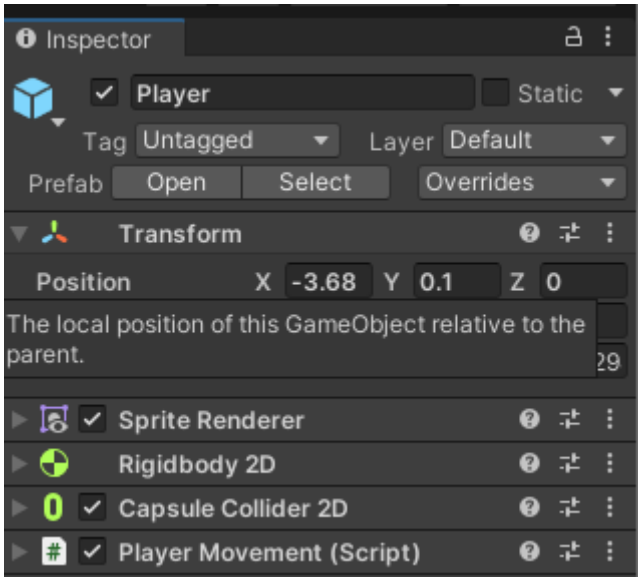

Fig. 13: Componentes del objeto "Player"

Disparar. – Esta acción es un poco peculiar ya que primero debemos crear la bala o proyectil, dándole las características que deseamos. Esto implica también agregarle unas ciertas físicas como al personaje. Como cada bala interviene en la escena, cada bala que se dispara será un objeto con el que se puede interactuar. El problema con esto es que si un jugador dispara 100 balas habrá 100 objetos que si no chocaron con nada seguirán existiendo y provocando que el rendimiento del juego no sea bueno. Es por esto por lo que se planifico hacer que la bala se destruya luego de cierto tiempo. Para lograrlo, hacemos que la bala sea un *Prefab,* el cual actúa como una plantilla y así no codificar por separado cada una de las 100 balas que puede disparar un jugador. El próximo paso es instanciar el *Prefab* de una bala cada que el usuario le dé al botón de disparar.

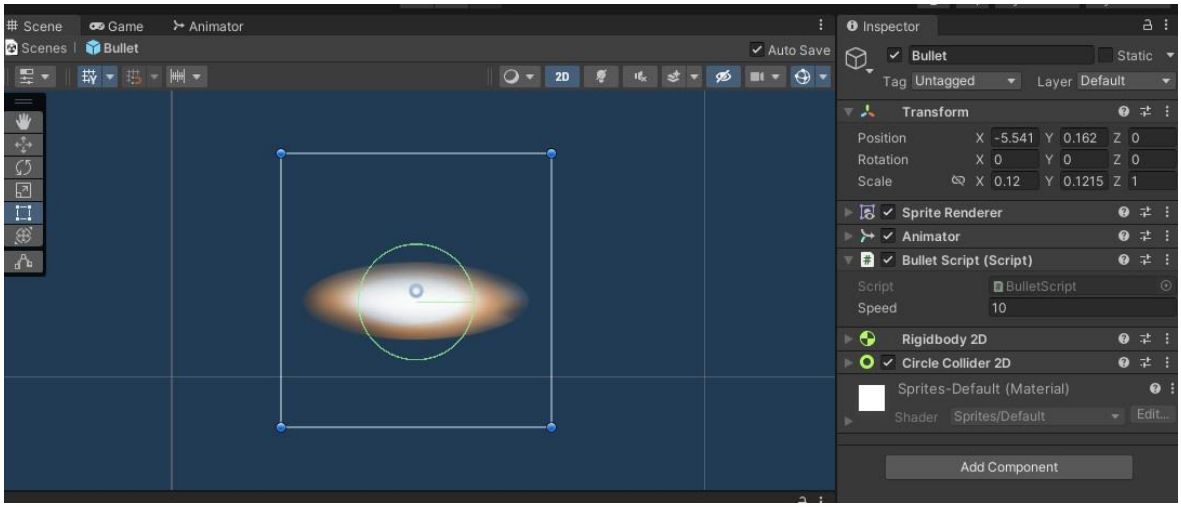

Fig. 14: Prefab del Objeto "Bullet"

Codificar Interfaz de Usuario para las mecánicas básicas

Este videojuego está pensado únicamente para dispositivos móviles Android. Por lo cual se debería jugar utilizando el *touch* de las pantallas de nuestros *smartphones.* Entonces, en la etapa de conceptualización del juego se proyectó que los controles sean un joystick digital en la parte izquierda inferior de la pantalla y dos botones en la parte inferior derecha. Cada uno con un objetivo específico.

Para implementar este tipo de Interfaz de usuario (UI), agregamos un objeto tipo *Canvas* en nuestra escena. Aquí importaremos un *Asset* en la propia tienda de *Assets* de *Unity*. El objeto que importaremos es un Joystick Digital.

Fig. 15: Joystick digital disponible en la Asset Store de Unity

Después agregaremos un par de botones que servirán para saltar y disparar. Los botones se agregan como un objeto dentro del *canvas*. Adicionalmente en la parte de superior se colocará la vida de cada uno de los personajes. Esto lo hacemos mediante imágenes y texto.

Así, el resultado final de nuestra Interfaz de Usuario es la siguiente:

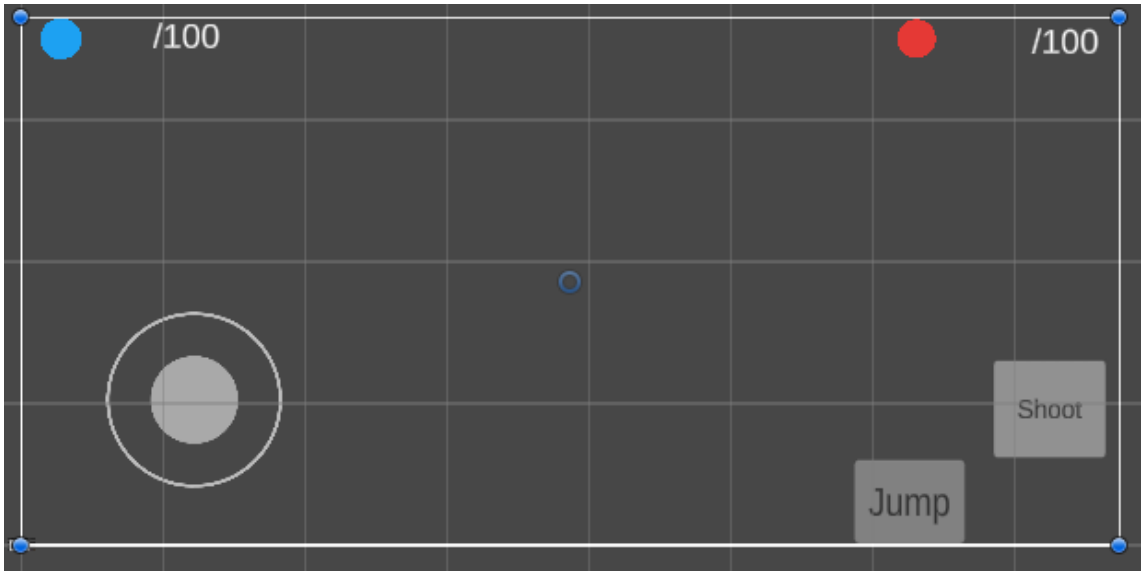

Fig. 16: Interfaz de Usuario Final

Sprint 2. Implementación de Interfaz de Usuario y Menús

Una vez que hemos dado los primeros pasos en la creación de este videojuego, es tiempo de adecuarlo más para que el resultado final sea el que se esperaba en la etapa de conceptualización del juego. Los menús y las pantallas de opciones en general son muy comunes en cualquier videojuego. Por esta razón este sprint corto está dedicado al diseño tanto visual como funcional de los siguientes escenas y paneles:

- Escena de menú principal
- Escena de créditos
- Panel de crear o unirse a partida
- Panel de crear partida
- Panel de unirse a una partida

Escena de menú principal

Esta escena esta encargada de ser la primera en cargarse cuando el jugador inicia la aplicación en su dispositivo inteligente. De manera funcional, estará compuesto de 3 botones que nos servirán para iniciar el juego, salir del juego y ver los créditos.

En cuanto a su diseño, gracias a las herramientas de *Photoshop* y *Figma* se pudo representar el concepto del juego al colocar el título del videojuego y como fondo de pantalla una imagen que hace referencia a su título: "*Castillo*".

Fig. 17: Escena de menú principal

Escena de créditos

Una escena simple pero que sirve para darle el reconocimiento a los que lograron desarrollar el videojuego. De forma práctica esta escena solo tiene un botón que sirve para regresar al menú principal.

Fig. 18: Escena de créditos

Panel de crear o unirse a partida

Este panel se despliega apenas se cargue la escena de combate. Está compuesto de 3 botones: el botón regresar que se usa para regresar al menú principal; el botón para crear una partida y; el botón para unirse a una partida previamente creada. Estos dos últimos desplegarán otro panel según corresponda.

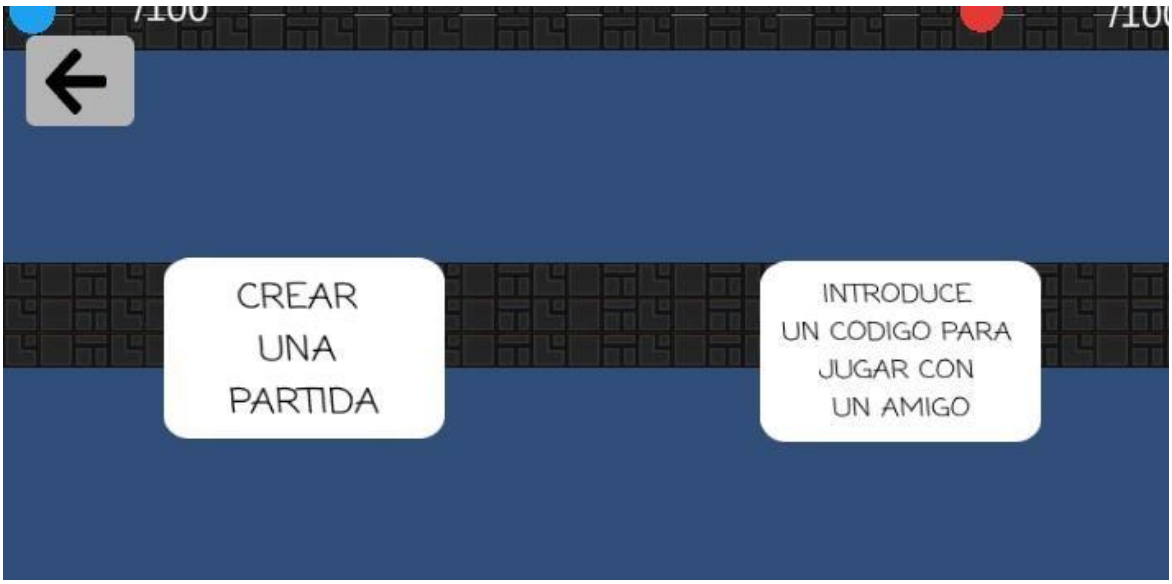

Fig. 19: Panel de crear o unirse a una partida

Panel de crear partida

Este panel contiene información importante para el correcto funcionamiento del juego. Se le proporcionará un código que deberá compartir con el otro jugador para lograr la conexión multijugador. También tiene un botón que sirve para iniciar el juego y a su vez ocultar los paneles para que la interfaz de usuario de combate se active.

Fig. 20: Panel de creación de partida

Panel de unirse a una partida

El panel sirve para unirse a una partida previamente creada y contiene dos elementos: un campo de texto (*textfield*) para que el usuario ingrese el código que el jugador que creo la partida le debió compartir y; un botón para mandar la petición de unirse al juego.

Como en el anterior panel, este deshabilitará todos los paneles y activará la interfaz de combate, dando así inicio a la pelea.

Fig. 21: Panel de unirse a una partida

Sprint 3. Implementación de conexión multijugador

La oferta de videojuegos multijugador online se ha visto incrementada en los últimos años gracias al triunfo que algunas empresas como *Riot Games* o *Activision* con el lanzamiento de títulos como *League of Legends, Fortnite* o *Call of Duty* [14]. Y la conexión que deben soportar este tipo de juegos es masiva. Por lo que es necesario que cuenten con servidores especializados y distribuidos en regiones del mundo para proporcionar una cierta calidad a la hora de conectarse a un servidor para jugar.

Para este proyecto no se ha usado un servidor dedicado a este videojuego. En cambio, hemos utilizado uno de los más recientes servicios que *Unity* ofrece en cuanto a conexión multijugador: *Relay Service.*

Relay funciona de forma que un jugador toma el papel de *host* y es el que permite la conexión con un servidor *Relay.* Los demás usuarios o jugadores se conectan al mismo como clientes e intercambian mensajes con el *host* y con los otros clientes a través del protocolo de mensajes de *Relay* [25].

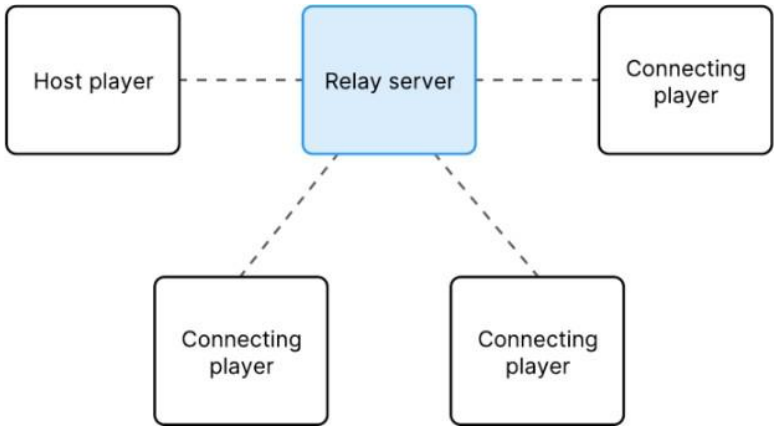

Fig. 22: Como funciona un relay server

Este sprint está destinado a toda la lógica detrás de que un jugador se conecte a una partida con un código de 6 dígitos. Los siguientes puntos son los apartados que se siguió para poder hacer que dos jugadores se conecten al juego de forma simultánea.

- Introducción a *Unity Gaming Services*
- Configuración del proyecto para trabajar con el servicio *Relay.*
- Codificación para usar un *Relay Server*
- Codificación para que un cliente se conecte al servidor

Introducción a *Unity Gaming Services*

Los servicios de videojuegos de *Unity* nos proporcionan soluciones para crear, administrar y potenciar nuestro juego. Para usar este tipo de servicios es necesario iniciar sesión y entrar al *dashboard* de *Unity Gaming Services* y crear un nuevo proyecto. En este caso, usamos el nombre "*Castillo".* Una vez que tenemos un proyecto podremos visualizar los servicios relacionados con el multijugador a los que podemos acceder.

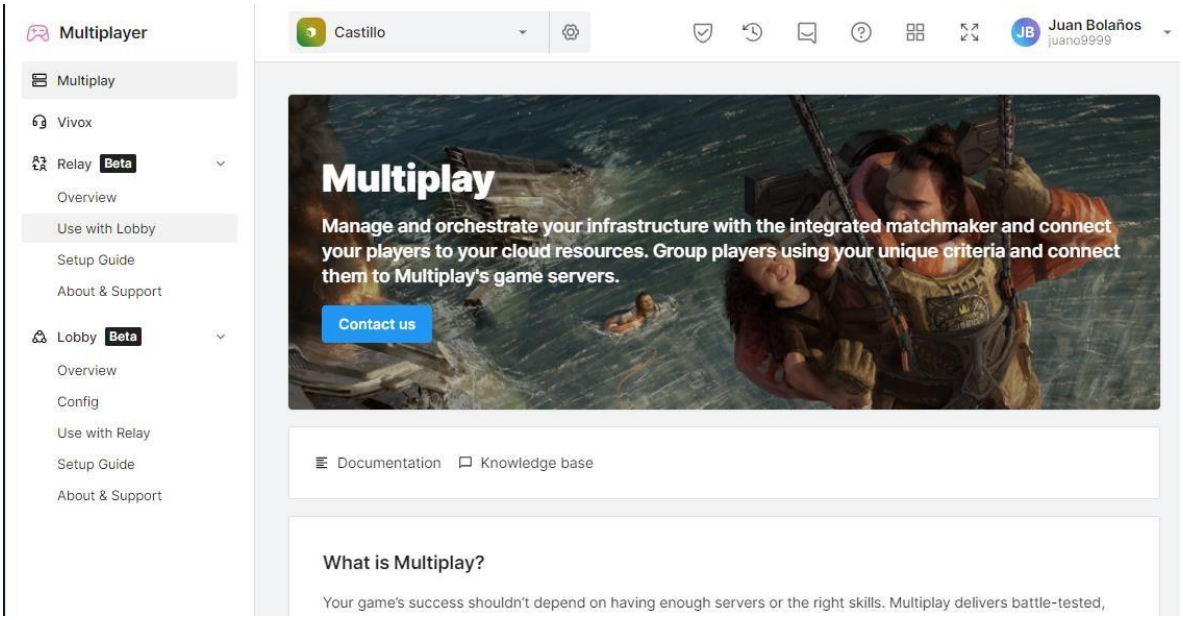

Fig. 23: Ventana de servicios de multijugador del proyecto Castillo

Una vez dentro, nos ubicamos en el apartado del servicio *Relay* que se encuentra en fase Beta. Y para activarlo debemos seguir la guía ubicada en el apartado "*Setup Guide*" y completar los cuatro pasos. Si nuestro proyecto está habilitado para usar *Relay* lo veremos en el apartado "*Overview*" en el cual nos indicara que el servicio está encendido como se indica en la siguiente figura:

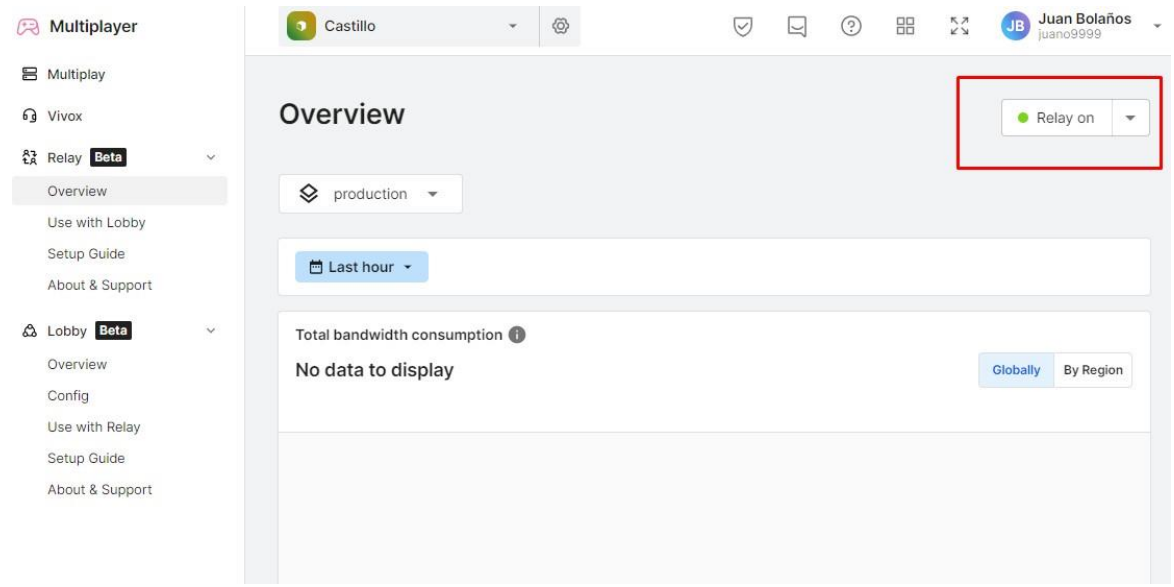

Fig. 24: Comprobación de que el servicio relay está encendido

Configuración del proyecto para trabajar con el servicio *Relay.*

Para iniciar a trabajar con *Relay* dentro de nuestro editor de *Unity*, hay que instalar los paquetes *SDK* que los servicios multijugador nos proporcionan. En este caso se seleccionó los siguientes:

- *Authenticaction*
- *Relay*
- *Netcode for Game Objects (NGO)*

Posteriormente generamos los *snippets* para pegarlos en nuestro *manifest.json* que se encuentra en la carpeta "*Packages"* de nuestro proyecto.

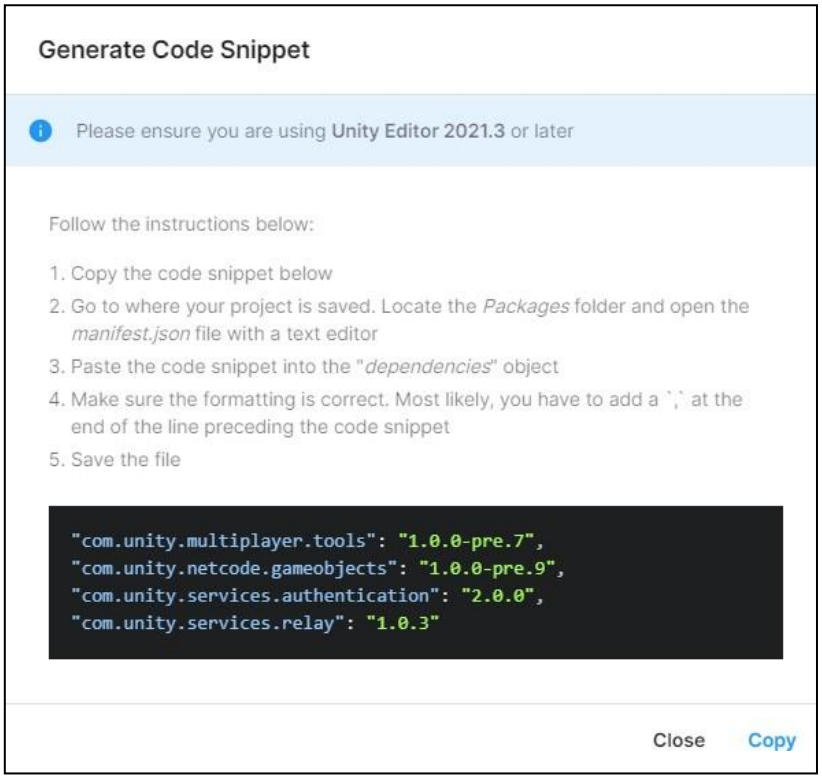

Fig. 25: Snippets generados en la pagina de packetes de Unity Services

Dentro del editor de *Unity* se compilará los nuevos paquetes que introducimos y se habilitaran las nuevas opciones para utilizar *Netcode*. Para activar finalmente los servicios multijugador necesitamos ligar nuestro proyecto en *Unity* con el proyecto en la nube de *Unity Services.* Para ello entramos a las configuraciones de proyecto en nuestro editor y seleccionamos el proyecto que creamos en *Unity Dashboard.* En este caso se llama Castillo.

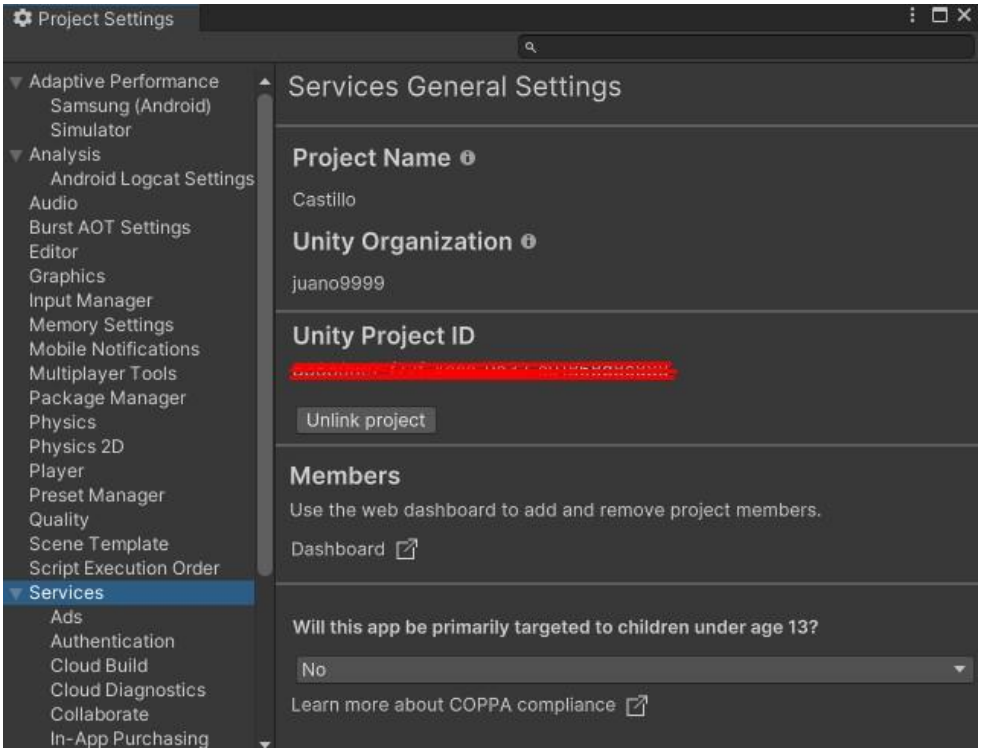

Fig. 26: Vinculacion del proyecto con Unity Gaming Services

Una vez hecho esto, estamos listos para usar el servicio de *Relay* junto con *Netcode for GameObjects.*

Codificación para usar un *Relay Server*

Para usar este servicio crearemos un *script* en la carpeta “Scripts” llamado “RelayManager.cs” el cual servirá para administrar las conexiones de jugadores tanto en el rol de *host* como en el rol de *client.*

El primer paso es implementar un objeto que llamamos “NetworkManager”, que será un objeto vacío en nuestra escena. Dentro de este objeto vacío agregamos el componente “NetworkManager” propio de la librería de *Netcode.* Este servirá para administrar la conexión a la red mediante las configuraciones que le hagamos. Para que la configuración sea correcta hemos usado un *Prefab* de jugador que aparecerá cada vez que un cliente se conecte al servidor.

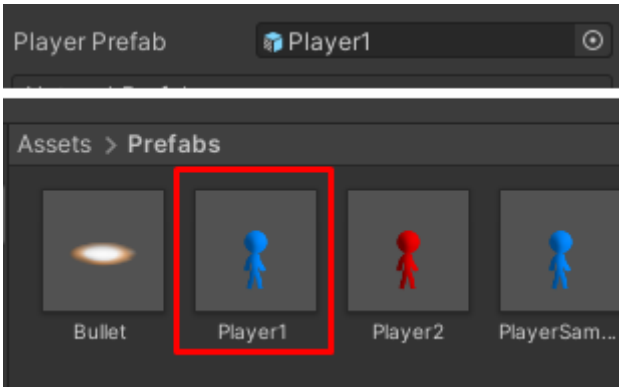

Fig. 27: Prefab de Player1 en la carpeta Prefabs

El segundo paso es incluir el método de transporte que queremos que *Unity* utilice. Para esto agregamos un nuevo componente en el mismo objeto que estamos trabajando. El componente llamado “Unity Transport” es una biblioteca orientada al desarrollo de juegos multijugador y que nos proporciona una capa de abstracción basada en conexiones UDP con funcionalidades como confiabilidad, fragmentación y ordenación. Esto significa que será de mucha ayuda al momento de transportar datos o enviar mensajes al servidor. Es importante recalcar que el protocolo que hemos usado para este proyecto es el “Relay Unity Transport” y no el protocolo “Unity Transport” por defecto.

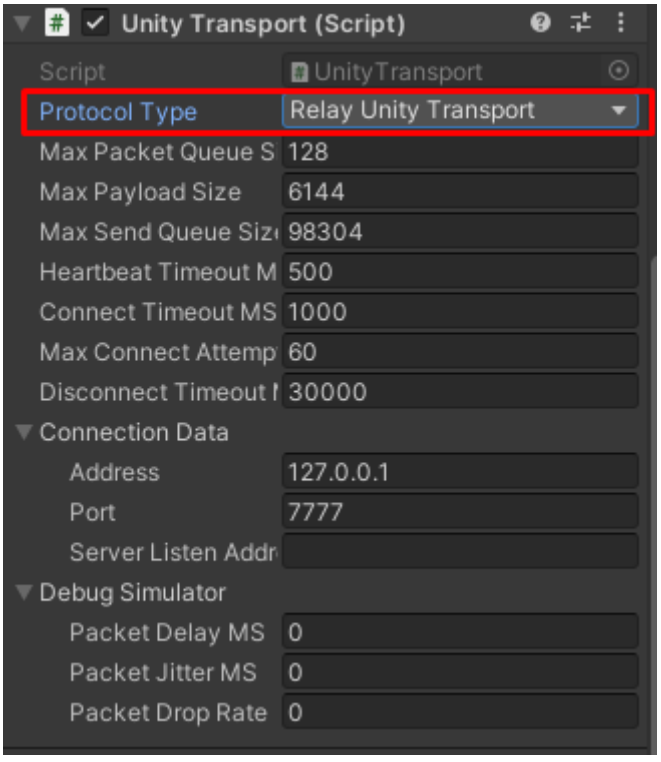

Fig. 28: Componente Unity Transport con protocolo Relay Unity Transport

Por último, hemos accedido a un servidor *Relay* ubicado en una región especifica que automáticamente es determinada por la propia biblioteca de *Relay.* Para acceder como host y obtener un código que nos servirá para después compartirlo con el otro jugador codificamos las siguientes acciones para que se ejecuten al pulsar el botón “CREAR UNA PARTIDA”:

1. Inicializamos los Servicios Multijugador de *Unity.*
2. Iniciamos sesión de forma anónima.
3. Creamos una asignación para acceder a un servidor *Relay* en alguna región disponible.
4. Obtenemos y mostramos el código en pantalla.
5. Guardamos la información de la asignación.
6. Con la información guardada configuramos los datos del *Relay Host* (en este caso nosotros).
7. Iniciamos el Host con el método “*StartHost()”* de la biblioteca de *NetworkManager.*

Ahora si ejecutamos el juego y creamos una partida podemos ver que el jugador aparece y lo podemos controlar. Además, en nuestro resumen general del servicio de *Relay* en la página web de *Unity Services* se muestra todas las conexiones y hosts que se han alojado en los servidores de manera global o por regiones.

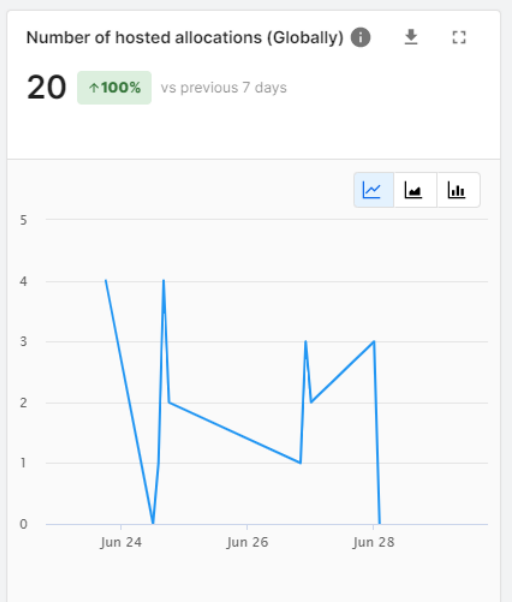

Fig. 29: Numero de asignaciones alojadas en los últimos 7 días

Codificación para que un cliente se conecte al servidor

Para que el juego desarrollado realmente sea un multijugador, otro jugador en cualquier parte del mundo debería poder conectarse simultáneamente con el servidor y por lo tanto con el host. El servicio *Relay* también nos brinda una forma de conectarse a un servidor de forma fácil con un código de 6 dígitos.

Para poder acceder al servidor y conectarnos con el usuario que sirve de host como clientes se realiza los siguientes pasos:

1. Escribimos el código en la parte de la Interfaz de Usuario correspondiente
2. Inicializamos los servicios de *Unity.*
3. Iniciamos sesión de forma anónima.
4. Nos unimos al servidor asignado utilizando el código de 6 dígitos previamente proporcionado.
5. Guardamos los datos de la asignación
6. Con la información guardada configuramos los datos del *Relay Client*
7. Iniciamos el cliente con el método *"StartClient()"* de la biblioteca *NetworkManager*

Si ejecutamos el juego, introducimos el código de una partida anteriormente creada y pulsamos el botón Jugar. Se iniciará el cliente y un nuevo jugador aparecerá en la escena de combate.

Fig. 30: Conexión de un cliente al escenario de combate utilizando el codigo

Sprint 4. Codificación de mecánicas de lucha

Después de que hemos logrado que dos jugadores en diferentes partes del mundo puedan conectarse al mismo servidor e intercambiar mensajes, es el momento de desarrollar la parte interesante del juego: el combate. Aunque previamente ya hemos implementado las mecánicas básicas y componentes para que nuestros jugadores puedan, saltar, correr y disparar, ahora la misión es hacer que estas acciones sirvan para realizar un progreso en la partida para tratar de ganarla.

Para ello, en este sprint se desarrolla todo aquello que tiene relación con el combate entre los dos jugadores y también la lógica que hay que seguir para que uno de los dos sea el ganador de la partida. Los siguientes puntos son la lista de tareas que se han seguido para que los jugadores puedan interactuar de forma correcta:

- Ajustes de controles para un multijugador
- Daño del proyectil en jugadores
- Reaparición de jugadores después de morir
- Progreso en los pisos de la escena de combate
- Un jugador gana

Ajustes de controles para un multijugador

En el *sprint 1* trabajamos y codificamos los movimientos y mecánicas básicas de un jugador de forma local. Dándole características de comportamiento en solitario (en *Unity* se les define como *Monobehaviour*). Cuando implementamos las conexiones multijugador tanto de host como de cliente es necesario cambiar al jugador para que ahora sea un objeto de red (*Network Object)* [26]. Y, por consiguiente, también tenga un comportamiento en la red (en *Unity* se le define como *NetworkBehaviour*).

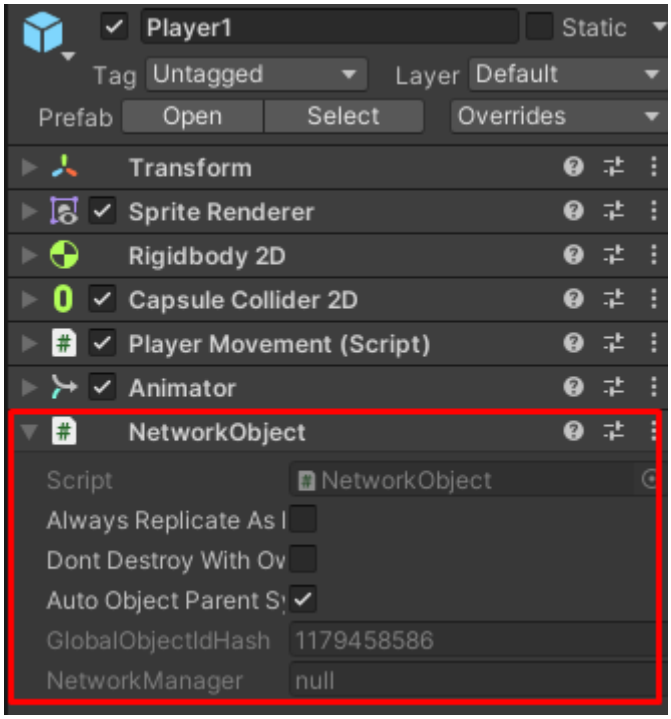

Fig. 31: Se añade el component NetworkObject al prefab del Player 1

Como se mencionó, añadir el componente de *NetworkObject* no es suficiente para usarlo en la red. Además, en el *script* que maneja los movimientos de un jugador, es decir "*PlayerMovement.cs"* modificamos la clase base del script de *Monobehaviour* a *NetworkBehaviour.* Esto nos abre la posibilidad de utilizar métodos y características fundamentales a la hora de trabajar en este caso con los jugadores y las mecánicas de cada uno.

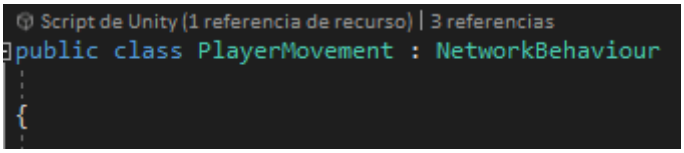

Fig. 32: Modificación en la clase base de PlayerMovement

A partir de este cambio, se cambia ligeramente la lógica para que los movimientos de un jugador sean correctos. Se limita a que los controles y animaciones solo sean ejecutadas por el propietario o el cliente que ha generado el objeto *PlayerMovement*. En resumen, si un cliente no es dueño del personaje no podrá modificar otros valores o acceder a las funciones de otro jugador.

Y, como eliminamos la opción de que el otro jugador se mueva en nuestra perspectiva, es necesario comunicar al servidor la posición del otro jugador, para que podamos verlo moverse por el mapa. Para esto, se ha incluido desde una biblieçoteca externa el

componente “*Client Network Transform”,* al cual encargaremos que comparta con el otro jugador la posición en los ejes “x” & “y”.

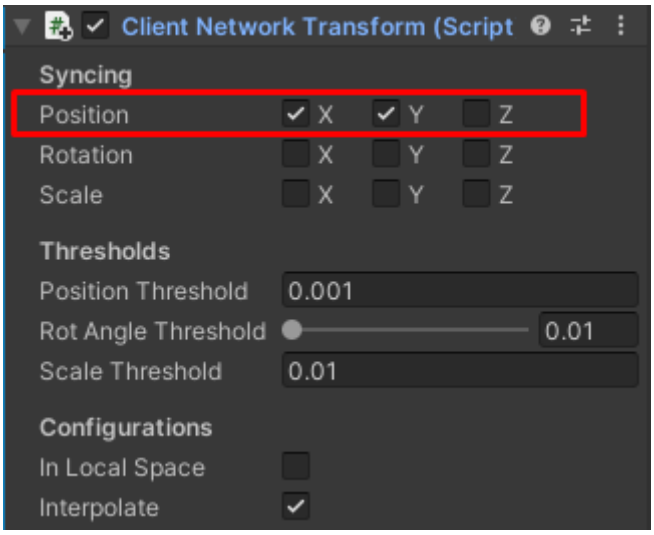

Fig. 33: Componente Client Network Transform

Daño del proyectil en jugadores

Como se había planteado en la fase de conceptualización del videojuego, parte esencial del videojuego planteado en el proyecto es el combate con el otro jugador, así que el daño que un jugador proporcione a otro es fundamental para ganar ventaja durante la partida. En este caso el personaje lanza proyectiles que rebajan la vida del oponente en una unidad. Los personajes que pierden toda su vida mueren y después de 3 segundos reaparecen. Además, un jugador solo tiene la posibilidad de disparar una determinada cantidad de veces seguidas. Luego de ejecutar estos disparos tendrá que esperar un tiempo de recarga de proyectiles. Nuevamente, para hacer que un disparo se vea representado tanto en el lado del cliente que lo ejecuta como en el cliente que lo recibe hemos realizado un proceso con variables de la red (*Network Variables*) y peticiones al servidor (*ClientRpc* y *ServerRpc)* utilizando scripts de comportamiento para la red (*Network Behaviour*).

El proceso para que un jugador dispare un proyectil se desarrolla en la siguiente figura:

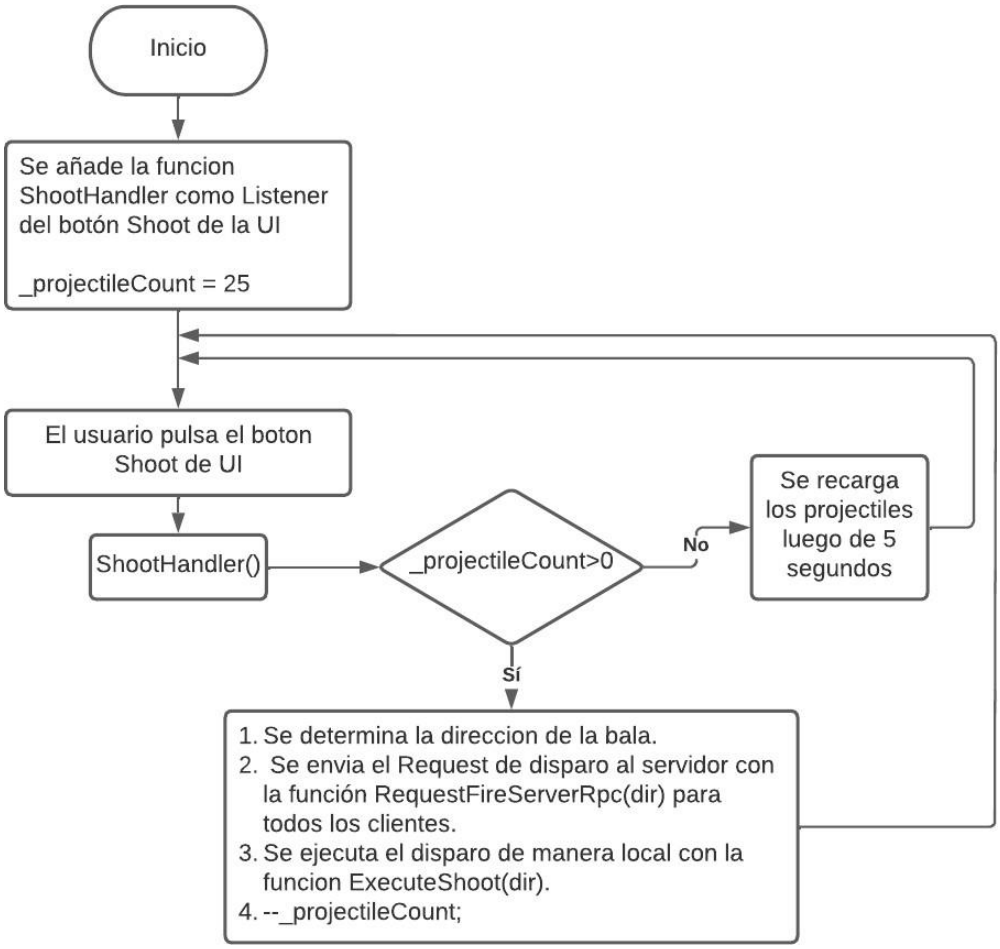

Fig. 34: Diagrama de flujo para disparar un proyectil

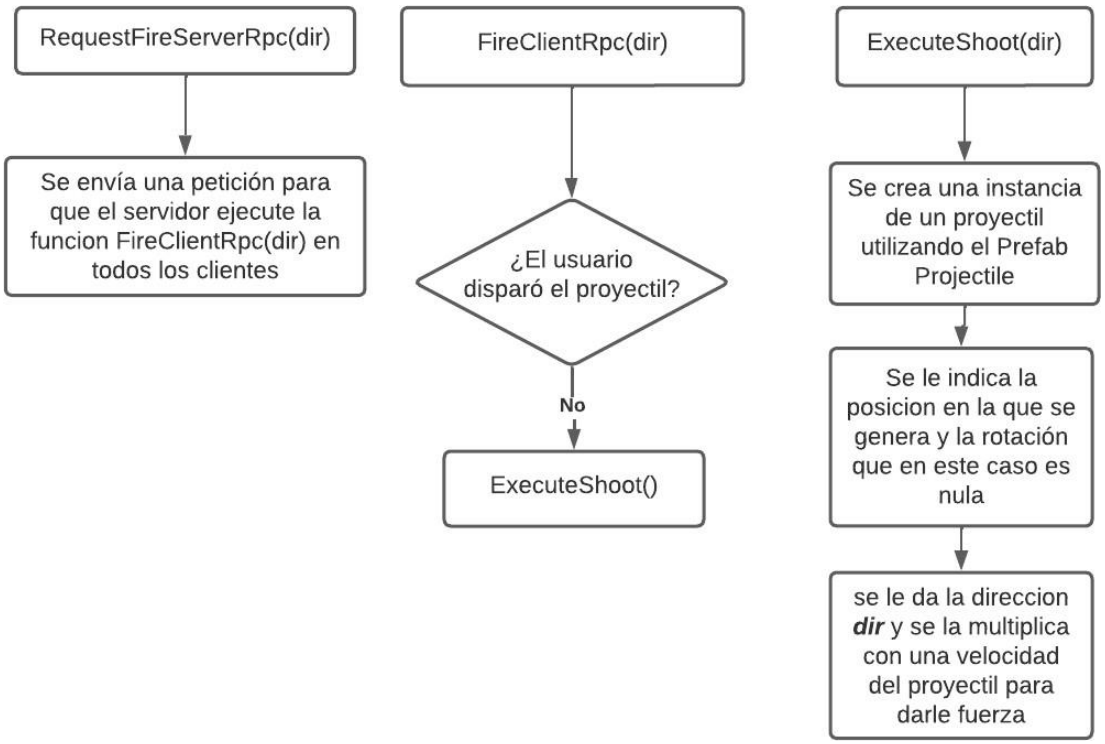

Fig. 35: Funciones adicionales para que un proyectil se dispare

Nota: En la función *FireClientRpc(dir)* solo se ejecuta en los clientes que no dispararon el proyectil porque se busca que se represente el disparo de forma gráfica para el que no disparó el proyectil.

Reaparición de jugadores después de morir

Si un disparo es efectivo al jugador se le restara una unidad de vida de una determinada cantidad. Y si la vida llega a 0 el jugador instantáneamente muere y reaparece después de 3 segundos.

Para lograr esto hemos agregado una nueva *Network Variable* llamada *IsDead,* la cual es de tipo *Bool* y que nos indicará si un jugador está muerto o no. Tomando esto en cuenta cada vez que la variable *Health* llegue a cero se ejecutara la función *Respawn()* luego de 3 segundos de haber muerto.

El siguiente diagrama de flujo especifica el comportamiento del *Respawn* de un jugador:

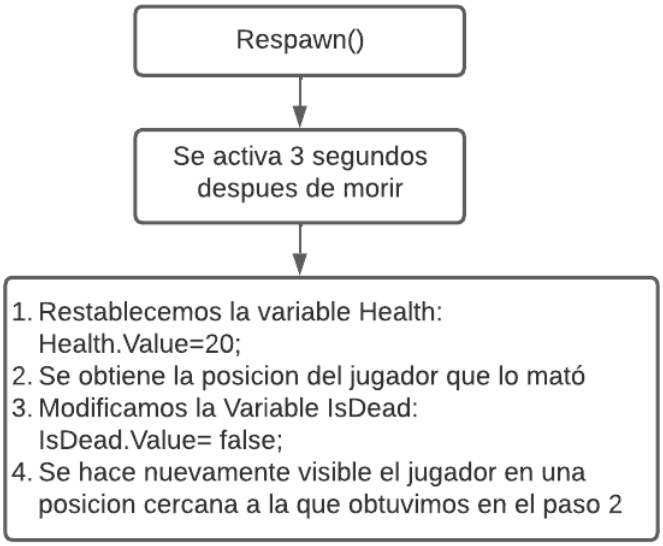

Fig. 36: Diagrama de flujo para que un jugador reaparezca en la escena despues de morir

Nota: Luego de que un jugador muere deja de ser visible para si mismo y para los clientes. Nunca desaparece como un *Network Object.*

Progreso en los pisos de la escena de combate

Una vez que hemos implementado las mecánicas de enfrentamiento entre los dos jugadores, es hora de encaminarse a cómo uno de ellos gana una partida. Para ser el ganador de una partida, hay que avanzar entre los pisos del castillo hasta llegar a tu meta matando a tu contrincante las veces que sea necesario.

La forma en la que se ha pensado que un jugador avance al siguiente piso según corresponda es que después de matar al contrincante tenga a favor la ventaja de la partida y esto hace que se habilite el poder acceder a una puerta que lo llevara al siguiente piso. Para esto se ha modificado el escenario de combate ligeramente agregando un par de elementos:

Las murallas: que son las que se activan y desactivan dependiendo de quien tenga la ventaja en la partida.

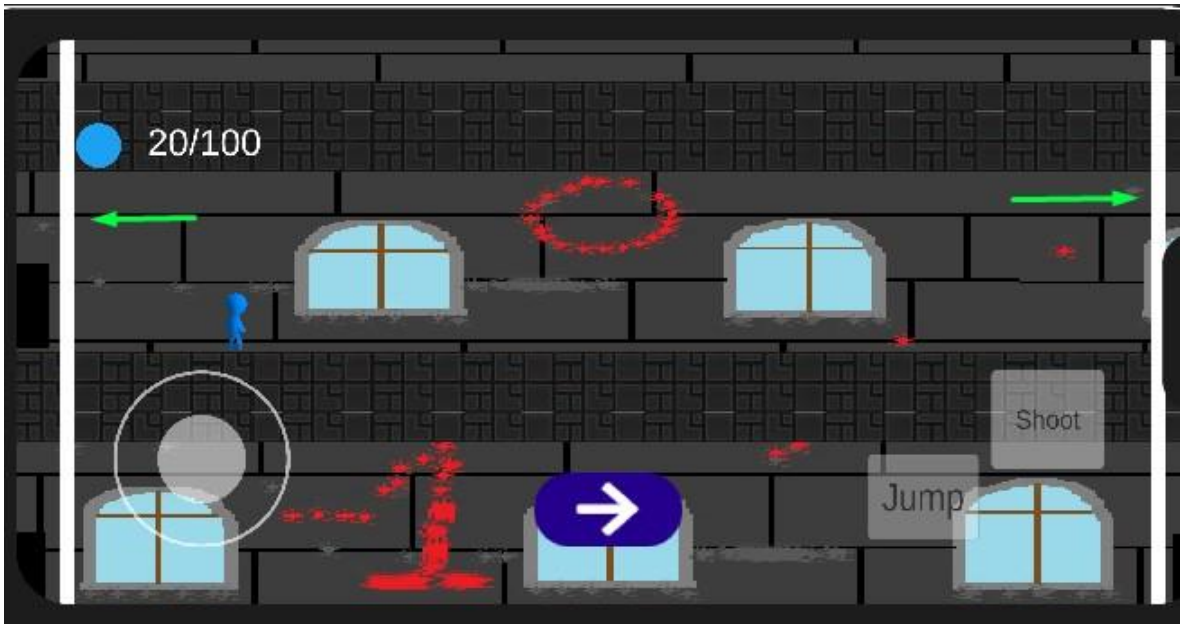

Fig. 37: Murallas que impiden el avance a siguientes pisos

La alerta para avanzar: Esta se activa al matar a un jugador e indica que puedes avanzar al siguiente nivel:

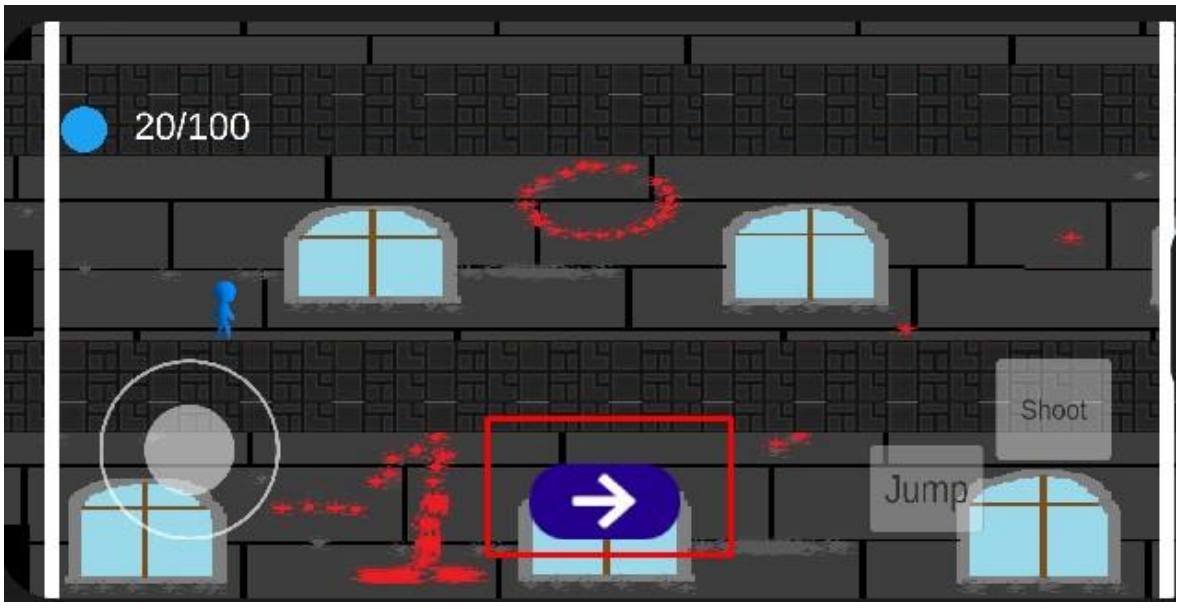

Fig. 38: Alerta para avanzar

Un jugador gana

Luego de disputar repetidas batallas que buscan liberar el camino y que cada jugador llegue a su correspondiente meta. Finalmente, los jugadores llegan al piso en donde podrán obtener la victoria de la partida.

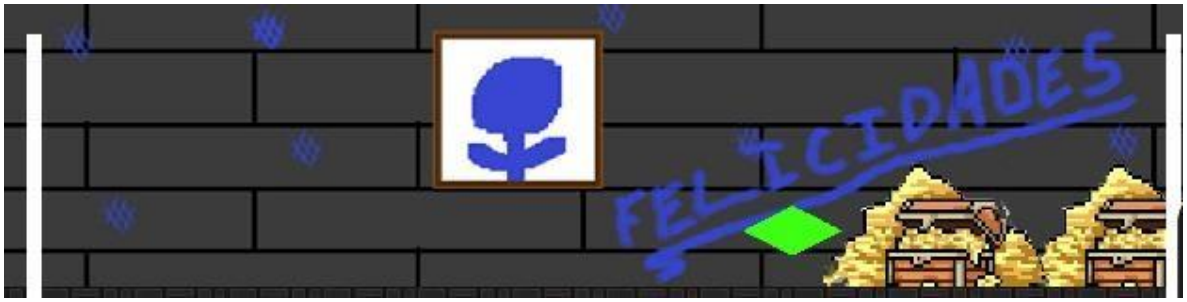

Fig. 39: Diseño de piso de victoria de jugador azul

Fig. 40: Diseño de piso de victoria de jugador rojo

Cuando lleguen a los pisos que se indican en las figuras. El jugador debe acercarse a los tesoros para conseguir la victoria. Al entrar en contacto con este item se lanza una petición de tipo *ClientRpc* para que se ejecute en los dos clientes y decirles quien ha ganado la partida. Inmediatamente, también se activa un pequeño cartel que indica que ha ganado la partida.

Fig. 41: Aviso de victoria de jugador azul

Sprint 5. Pruebas, depuración y lanzamiento

Esta iteración se ha tratado de depurar el juego para lograr que tenga una calidad tanto de software como de videojuego. Para esta fase de la metodología SUM se ha planeado implementar pruebas de aceptación debido al tiempo que se dispone para la misma. Dichas pruebas nos ayudarán a probar el juego completo, reportando errores o mecanismos a mejorar. Tales como, la conexión multijugador, el *gameplay*, o hasta el sistema de combate.

Por lo tanto, en este *sprint* se ha propuesto probar y mejorar el videojuego de la siguiente forma:

- Distribución de Versión *Alpha*
- Reporte de errores y desarrollo de posibles requerimientos de la versión *Alpha*
- Distribución de Versión *Beta*
- Reporte de errores y desarrollo de posibles requerimientos de la Versión *Beta*
- Lanzamiento de Versión final

Distribución de Versión *Alpha*

La versión Alpha del videojuego se ha liberado hacia un total de 3 personas incluyendo al desarrollar del mismo. El principal objetivo de esta versión no es probar a detalle los mecanismos de combate o la conexión, sino detectar posibles fallas en la interfaz de usuario o en la conexión multijugador.

De forma general, tratar de detectar errores comunes que podrían entorpecer el correcto desarrollo antes y después de iniciar el combate.

Reporte de errores y desarrollo de posibles requerimientos de la versión *Alpha*

Luego de 3 días de que se haya liberado la versión *Alpha* del juego y que se hayan hecho las pruebas de aceptación por las pocos *testers* a los que se les encargo esta tarea. Se ha detectado los siguientes errores:

- Botón de salir no funciona
- Si creas una partida y vuelves atrás al menú principal, ya no se puede crear una nueva partida
- Cuando creas una partida se debería ocultar o bloquear los controles hasta que se conecte el otro jugador

- Al introducir el código de 6 dígitos, te permite ingresar en minúsculas o mayúsculas.
- Al terminar el juego, regresar al menú principal y volver a crear una partida, ya no funciona el juego.

Distribución de Versión *Beta*

Tras la primera semana de pruebas de la versión Alpha y habiendo corregido los errores simples y comunes que fueron reportados en esta subetapa de este *sprint,* se dispuso a entregar la versión Beta a un total de 10 personas para que la puedan probar entre sí.

Las pruebas de aceptación para esta versión del juego son pensadas para calificar y retocar tanto funcionalidad de menús o conexión como del propio *gameplay* mecánicas para que el videojuego sea lo más equilibrado y divertido para el combate multijugador.

Reporte de errores y desarrollo de posibles requerimientos de la Versión *Beta*

Después de casi 2 semanas de compartir el videojuego en formato ".apk" con un total de 10 personas se reportó algunos *bugs* más y recomendaciones para hacer que el combate sea lo más fluido posible. Algunos de estos son:

- No avanzar al siguiente piso si el jugador está muerto.
- Rebajar la vida a 20 unidades.
- No restringir el uso de proyectiles.
- Bloquear el joystick y botones de interfaz de usuario en el combate hasta que el otro jugador se conecte

Lanzamiento de versión final

Luego de casi 3 semanas de corrección de errores. Finalmente se ha conseguido una versión estable del videojuego casi en su totalidad. Como ya se mencionó, *Unity* nos facilita la exportación de nuestros videojuegos en varias plataformas. En este caso lo hemos realizado para la plataforma Android.

Para exportar nuestra aplicación como "*.apk"* nos dirigimos a la pestaña "File", y en el submenú clicamos en "*Build Settings*". Esto nos abrirá una pantalla como la siguiente:

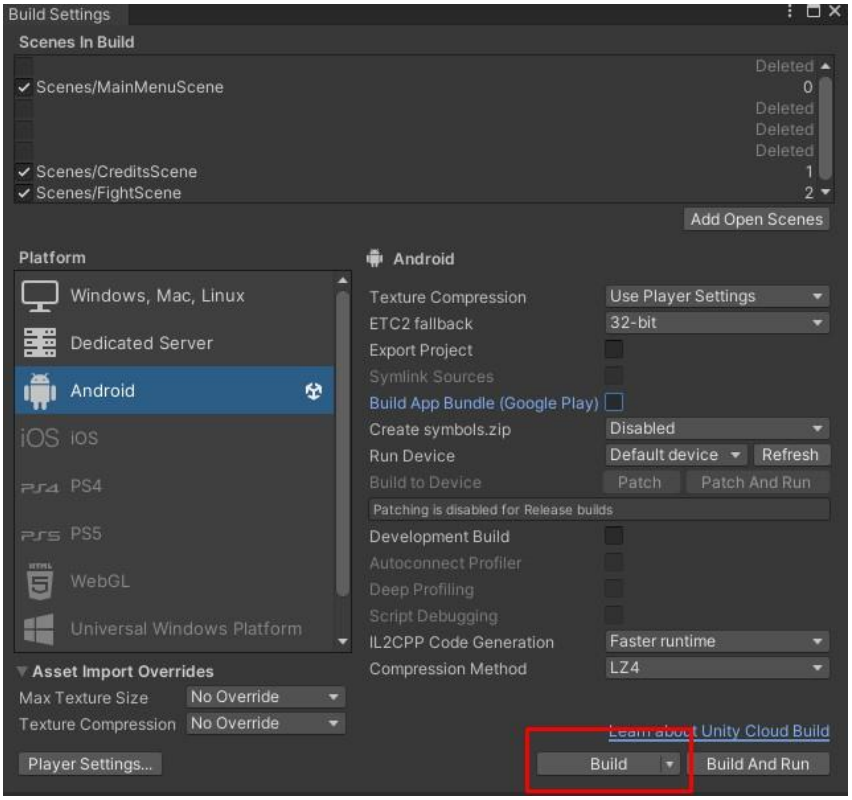

Fig. 42: Build Settings para exportar la aplicación como .apk

Por último, pulsamos el botón “*Build*” y comenzará el proceso de compilación de la aplicación con el nombre que se nos ha pedido que le demos al archivo previamente.

4 CONCLUSIONES

- Siguiendo las etapas de la metodología de SUM, la fase de conceptualización del videojuego es muy importante al momento de entrar a la fase de desarrollo de este. Si tenemos muy claro desde el principio todo lo que nuestro videojuego va a tener, no perderemos tiempo en su programación replanteándonos nuevas tareas o requerimientos. Aunque muchas veces esto es inevitable. En este proyecto se pudo mantener y casi no variar en la parte del *gameplay* como tal.
- La fase de planificación de requerimientos para una aplicación o, en este caso un videojuego, es de los pasos más útiles para poder entregar al usuario final un producto entretenido y de calidad. Cualquier interfaz de usuario debe ser pensada con anticipación teniendo en cuenta si al jugador le parecerá excesiva o no, la cantidad de botones u opciones que se pueda proporcionar.
- Las herramientas que proporciona *Unity*, como su motor o incluso los servicios de *multiplayer* fueron indispensables para la creación de este juego. A pesar de que llevan más de 15 años desde su aparición, siguen brindándole a desarrolladores la posibilidad de plasmar ideas y exportarlas al resto del mundo.
- Aunque si se lanzó versiones Beta del videojuego, no se pudo implementar en su totalidad todas las correcciones o ajustes necesarios para que sea un juego completamente estable. Esto debido al tiempo de desarrollo que se nos dio para este proyecto. A pesar de este contratiempo, los requerimientos propuestos en historias de usuario fueron cumplidos y se espera seguir perfeccionando el juego incluso después de que este proyecto culmine.
- En la última etapa del desarrollo, es decir la depuración, se modificaron cosas pequeñas pero importantes para llegar a la versión del videojuego final. Esta fue entregada con la menor cantidad de errores posibles.
- Durante todo el proceso se debe gestionar los posibles riesgos que el proyecto podría llegar a tener. Desde fallos fatales para el videojuego, hasta la computadora con la que estas trabajando en el mismo. Gestionar los riesgos es una obligación que nos permite entregar al usuario un software de calidad, que es lo que cualquier desarrollador desea.

5 RECOMENDACIONES

- Si desea recrear o construir un videojuego parecido al de este documento se recomienda iniciar con el Sprint No. 2. Debido a la facilidad y lo corto que es servirá

como un pequeño tutorial que ayudará a comprender algunos elementos de *Unity Engine* para aquellos que es su primera vez desarrollando un videojuego.

- Este proyecto se planteo desde el principio como un multijugador online muy básico, pero para introducirse al mundo de los videojuegos se recomienda desarrollar videojuegos que no tengan este componente de dificultad de lograr la conexión entre varios jugadores a través de la red, pues hay muchos conceptos que hay que entender antes de desarrollar una aplicación de este tipo.
- Se recomienda Unity como motor para videojuegos ya que es una herramienta versátil y muy cómoda para desarrolladores que están empezando en la programación de videojuegos. El único requisito es que sepa fundamentos de programación básicos y el lenguaje “C#”. Por lo demás, ayudarse de tutoriales en plataformas como *Youtube* para dominar la herramienta es muy recomendable.
- Trabajar en el desarrollo de un juego de forma completamente solitaria puede llegar a ser frustrante. Es por eso por lo que se recomienda tener un equipo de trabajo de al menos 2 personas para poder desarrollar un producto de forma rápida y eficaz.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] C. López, "UDG Virtual," 2016. [Online]. Available: http://www.udgvirtual.udg.mx/apertura/index.php/apertura/article/view/825. [Accessed 11 Julio 2021].

[2] INEC, "ecuadorencifras," 2013. [Online]. Available: https://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_Sociales/TIC/Resultados_principales_140515.Tic.pdf. [Accessed 12 Julio 2021].

[3] STEAM, "Steam," Valve Corporation, [Online]. Available: https://store.steampowered.com/app/896340/To_Leave/?l=spanish&curator_clanid=6599272. [Accessed 12 Julio 2021].

[4] STEAM, "Steam," Valve Corporation, [Online]. Available: https://store.steampowered.com/app/1424500/Secret_of_The_Lost_Keys_Episodio_I_El_ataque_en_Disred/?l=spanish. [Accessed 12 Julio 2021].

[5] Rosepac, "Ciberninjas," 14 Enero 2020. [Online]. Available: https://ciberninjas.com/motores-videojuegos/. [Accessed 26 Julio 2021].

[6] M. I. Edgar Peña, "Estudios preliminares de una propuesta de un modelo de procesos para el desarrollo de videojuegos independientes," Ciudad de México, 2015.

[7] J. Chero, "ESTUDIO DE USABILIDAD DE VIDEOJUEGOS WEB UTILIZANDO LA METODOLOGÍA DE DESARROLLO SUM," UTMACH, Machala, 2019.

[8] "Euronics," Euronics.es, [Online]. Available: https://www.euronics.es/blog/que-tipos-de-videojuegos-existen-clasificacion-y-diferencias/. [Accessed 26 Julio 2021].

[9] pasilan, "graphicriver.net," Envato Market, 10 Diciembre 2015. [Online]. Available: https://graphicriver.net/item/stickman-character-sprites-159/13980089. [Accessed 6 Junio 2022].

[10] C. Dave, "Unity 2D Game Development," Packt Publishing, Birmingham, 2014.

[11] "Unity," Unity Technologies, 2021. [Online]. Available: https://unity.com/es/how-to/difference-between-2D-and-3D-games. [Accessed 26 Julio 2021].

[12] Redacción Canal Trece, "canaltrece.com.co," Canal Trece, 18 Agosto 2021. [Online]. Available: https://canaltrece.com.co/noticias/videojuegos-historia-linea-de-tiempo-nintendo-sega-atari/. [Accessed 6 Junio 2022].

[13] P. Díaz, "33bits.net," 33bits, 25 Enero 2019. [Online]. Available: https://portal.33bits.net/los-plataformas-bidimensionales/. [Accessed 6 Junio 2022].

[14] Volk Games Noticias, "volkgames.com," VOLK, 5 Mayo 2021. [Online]. Available: https://www.volkgames.com/videojuegos-moviles-una-evolucion-en-la-industria-gamer/. [Accessed 6 Junio 2022].

[15] D. Candil, "vidaextra.com," Webedia, 21 febrero 2014. [Online]. Available: https://www.vidaextra.com/industria/unity-el-motor-de-desarrollo-capaz-de-partir-la-historia-de-los-videojuegos-en-dos. [Accessed 25 Julio 2022].

[16] N. Aceranza, G. M. Ariel Coppes, A. Viera and E. Fernández, "Una metodología para desarrollo de videojuegos," PEDECIBA Informática, Montevideo, 2009.

[17] J. Pareja and R. Rivera, "Evaluación de Procesos de Software," Escuela Politécnica Nacional, Quito, 2011.

[18] X. Murillo and A. Gutierrez, "Implementación de la metodología SUM modificada para el desarrollo de videojuegos orientados al aprendizaje en Bolivia," Universidad Católica Boliviana "San Pablo", La Paz, 2018.

[19] "gemserk," Eclipse, 2008. [Online]. Available: http://www.gemserk.com/sum/#:~:text=La%20metodolog%C3%ADa%20SUM%20par. [Accessed 16 Mayo 2022].

[20] M. Borja, «DESARROLLO DE VIDEOJUEGO DE PLATAFORMAS 2D,» Escuela Politécnica Nacional, Quito, 2022.

[21] A. Alvear and E. Vargas, "Repositorio Digital - EPN," 1 Junio 2021. [Online]. Available: https://bibdigital.epn.edu.ec/handle/15000/21674. [Accessed 16 Mayo 2022].

[22] JLPM, "Youtube," Google Inc., 31 Enero 2021. [Online]. Available: https://www.youtube.com/watch?v=GbmRt0wydQU&list=WL&index=7. [Accessed 16 Junio 2022].

[23] LuisCanary, "Youtube," Google Inc., 15 Abril 2020. [Online]. Available: https://www.youtube.com/watch?v=8C9h4CCoC2E&t=822s. [Accessed 16 Junio 2022].

[24] D. Valecillos, "Youtube," Google Inc., 25 Noviembre 2021. [Online]. Available: https://www.youtube.com/watch?v=82Lbho7S0OA&list=WL. [Accessed 16 Junio 2022].

[25] Unity, "Unity Documentation," Unity technologies, [Online]. Available: https://docs.unity.com/relay/introduction.html. [Accessed 16 Junio 2022].

[26] Unity, "docs-multiplayer.unity3d.com," Unity Technologies, 15 Junio 2022. [Online]. Available: https://docs-multiplayer.unity3d.com/netcode/current/basics/networkobject/index.html. [Accessed 30 Junio 2022].

[27] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.

7 ANEXOS

ANEXO I. Turnitin porcentaje máximo 12%.

ANEXO II. Manual técnico

ANEXO III. Manual de usuario

ANEXO IV. Manual de instalación

ANEXO I

ANEXO II

1 DOCUMENTO DE CONCEPTO

Este artefacto se originó en la fase de conceptualización del videojuego y se ha ido modificando mientras se desarrolló

Concepto del Juego

Título: Castillo

Género: Juego de acción/plataformas 2D

Escenario:

Este compuesto por 7 pisos con la estética de un castillo. Cada piso servirá como campo de batalla para los jugadores. También posee dos metas en los pisos inferior y superior que los jugadores tienen como misión alcanzar para ganar la partida

Personajes:

- Jugador 1 – Es un stickman de color azul que puede saltar, correr y disparar. Su meta es llegar al piso inferior de color azul.
- Jugador 2 – Es un stickman de color rojo que puede saltar, correr y disparar. Su meta es llegar al piso superior de color rojo.

Gameplay:

El jugador controlará a uno de los dos personajes y su objetivo es llegar al piso inferior o superior según corresponda el color. El jugador tendrá una vida de 100 unidades y su contrincante también. Además, poseerá 100 proyectiles que se recargarán luego de 3 segundos si se terminan.

- Saltar – Mientras saltas no podrás disparar proyectiles, pero te servirá para esquivar los del rival recibiendo menos daño y por tanto resistir para poder atacar de mejor forma a tu contrincante.
- Disparar – Mientras disparas no podrás saltar. Además, si mantienes pulsado el botón disparar, los proyectiles se lanzarán de forma consecutiva para otorgar más daño a tu rival en menos tiempo.

Para poder avanzar por los pisos del castillo el jugador primero debe matar a su contrincante disparándole proyectiles hasta quitarle todas sus unidades de vida. Si su contrincante lo mata primero habrá un tiempo de espera para reaparecer. Si no matas

primero a tu contrincante no se te permitirá avanzar a otro piso. Esto será indicado por un aviso en la parte superior central de la pantalla.

El jugador ganará la partida cuando llegue a su correspondiente meta

Aspectos Adicionales:

Se tendrá un menú principal el cual tendrá las opciones de salir del juego o jugar. Cuando se elige jugar se desplegará otro menú en donde podrás crear una partida o unirte a una con un código de 6 dígitos.

2 HISTORIAS DE USUARIO

A continuación, se encuentran enlistadas las historias de usuario que usamos para este proyecto.

Tabla I Historia de Usuario #1

Historia de Usuario	
Identificador (ID): HU001	**Usuario:** Jugador
Nombre de Historia: Ejecutar Juego	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Responsable: Juan Bolaños	
Descripción: El jugador podrá ejecutar el juego en su dispositivo Android en forma de una aplicación que será proporcionada junto con este proyecto.	
Observaciones:	

TABLA II Historia de usuario #2

Historia de Usuario	
Identificador (ID): HU002	**Usuario:** Jugador
Nombre de Historia: Pantalla de Inicio	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Baja
Iteración Asignada: 2	
Responsable: Juan Bolaños	
Descripción: Luego de que la aplicación se ejecute en el dispositivo Android, inmediatamente se podrá visualizar una pantalla de menú con los botones de jugar, salir del juego o configuraciones.	
Observaciones: Otros botones con diferentes funcionalidades pueden o no ser incluidos en esta pantalla. Dependiendo de si en el proceso de desarrollo lo requiera	

TABLA III Historia de usuario #3

Historia de Usuario	
Identificador (ID): HU003	**Usuario:** Jugador
Nombre de Historia: Pantalla de Conexión con otro jugador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración Asignada: 2	
Responsable: Juan Bolaños	
Descripción: Cuando se ejecuta la opción de jugar, se desplegará una pantalla que indicará dos opciones. La primera para crear una partida y la segunda para unirte a una partida ya creada mediante un código de 6 dígitos.	
Observaciones:	

TABLA IV Historia de usuario #4

Historia de Usuario	
Identificador (ID): HU004	**Usuario:** Jugador
Nombre de Historia: Botón Jugar	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Baja
Iteración Asignada: 2	
Responsable: Juan Bolaños	
Descripción: Al presionar se redirigirá al jugador a la pantalla de conexión con otro jugador	
Observaciones:	

TABLA V Historia de usuario #5

Historia de Usuario	
Identificador (ID): HU005	**Usuario:** Jugador
Nombre de Historia: Botón salir del Juego	
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Baja
Iteración Asignada: 2	
Responsable: Juan Bolaños	
Descripción: Al presionar el botón salir del Juego, la aplicación se cerrará automáticamente.	
Observaciones:	

TABLA VI Historia de usuario #6

Historia de Usuario	
Identificador (ID): HU006	**Usuario:** Jugador
Nombre de Historia: Botón Créditos	
Prioridad en Negocio: Baja	**Riesgo en Desarrollo:** Baja
Iteración Asignada: 2	
Responsable: Juan Bolaños	
Descripción: Al presionar este botón se redirigirá al jugador a una pantalla en donde se visualizará los créditos de los implicados en el videojuego.	

Observaciones:

TABLA VII Historia de usuario #7

	Historia de Usuario
Identificador (ID): HU007	**Usuario:** Jugador
Nombre de Historia: Pantalla de Crear una partida	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 2	
Responsable: Juan Bolaños	
Descripción: Se proporcionará un código de 6 dígitos para que lo compartas con la persona que quieras jugar	
Observaciones:	

TABLA VIII Historia de usuario #8

	Historia de Usuario
Identificador (ID): HU008	**Usuario:** Jugador
Nombre de Historia: Escenario de Combate	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Media
Iteración Asignada: 1	
Responsable: Juan Bolaños	
Descripción: Será dividido por pisos(plataformas) y tendrá un punto de meta para cada uno de los jugadores	
Observaciones:	

TABLA IX Historia de usuario #9

	Historia de Usuario
Identificador (ID): HU009	**Usuario:** Jugador
Nombre de Historia: Generación de personajes en el escenario de combate	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Responsable: Juan Bolaños	
Descripción: Instantáneamente después de establecer la conexión con el otro jugador aparecerán en el mismo piso, pero en el lado opuesto a su contrincante.	
Observaciones:	

TABLA X Historia de usuario #10

	Historia de Usuario
Identificador (ID): HU010	**Usuario:** Jugador
Nombre de Historia: Saltar	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Responsable: Juan Bolaños	

Descripción: El personaje podrá saltar y esto servirá para esquivar los ataques del contrincante.
Observaciones:

TABLA XI Historia de usuario #11

	Historia de Usuario
Identificador (ID): HU011	**Usuario:** Jugador
Nombre de Historia: Disparar al otro jugador	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 4	
Responsable: Juan Bolaños	
Descripción: Los jugadores dispararán proyectiles que quitarán vida a su oponente hasta matarlo.	
Observaciones:	

TABLA XII Historia de usuario #12

	Historia de Usuario
Identificador (ID): HU012	**Usuario:** Jugador
Nombre de Historia: Correr	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Responsable: Juan Bolaños	
Descripción: El personaje podrá correr para avanzar. Para esto utilizará el joystick	
Observaciones:	

TABLA XIII Historia de usuario #13

	Historia de Usuario
Identificador (ID): HU013	**Usuario:** Jugador
Nombre de Historia: Usabilidad del joystick	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 1	
Responsable: Juan Bolaños	
Descripción: El joystick estará ubicado en la parte inferior izquierda de la pantalla del dispositivo y se usará para realizar movimientos como correr en diferentes direcciones	
Observaciones:	

TABLA XIV Historia de usuario #14

	Historia de Usuario
Identificador (ID): HU014	**Usuario:** Jugador
Nombre de Historia: Botón de Disparar	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo

Iteración Asignada: 1
Responsable: Juan Bolaños
Descripción: Estará ubicado en la parte inferior derecha de la pantalla y servirá para disparar mientras no camines o saltes
Observaciones:

TABLA XV Historia de usuario #15

	Historia de Usuario
Identificador (ID): HU015	**Usuario:** Jugador
Nombre de Historia: Botón de saltar	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo
Iteración Asignada: 1	
Responsable: Juan Bolaños	
Descripción: Servirá para realizar un salto y combinado con el movimiento del joystick indicar la dirección del salto y saltar no solo de forma vertical sino también en diagonal.	
Observaciones:	

TABLA XVI Historia de usuario #16

	Historia de Usuario
Identificador (ID): HU016	**Usuario:** Jugador
Nombre de Historia: Avanzar pisos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Iteración Asignada: 4	
Responsable: Juan Bolaños	
Descripción: Un jugador solo puede avanzar al siguiente piso solo si mato con anterioridad al contrincante. Si no lo mata no podrá avanzar.	
Observaciones: Cada jugador avanzará pisos hacia arriba o abajo correspondientemente	

TABLA XVII Historia de usuario #17

	Historia de Usuario
Identificador (ID): HU017	**Usuario:** Jugador
Nombre de Historia: Reaparecer	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Iteración Asignada: 4	
Responsable: Juan Bolaños	
Descripción: Una vez que te maten tendrás una cuenta hacia atrás para reaparecer. Pero si el jugador avanza un piso, inmediatamente reaparecerás en el lado contrario del piso.	
Observaciones:	

TABLA XVIII Historia de usuario #18

Historia de Usuario	
Identificador (ID): HU018	**Usuario:** Jugador
Nombre de Historia: Ganar la partida	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alta
Iteración Asignada: 4	
Responsable: Juan Bolaños	
Descripción: Un jugador ganará la partida cuando llegue a su meta correspondiente después de avanzar por los pisos necesarios.	
Observaciones:	

3 PLAN DEL PROYECTO

La Tabla XIX muestra el plan del proyecto con las actividades que se realizaron durante el proyecto

TABLA XX Plan del Proyecto

Plan del Proyecto				
HU-ID	**Actividad**	**Iteración**	**Estado**	**Prioridad**
TODAS	Conceptualizar el Videojuego	1	Finalizado	Alta
N/A	Diseñar el arte del juego	1	Finalizado	Alta
HU008	Construir el escenario de combate	1	Finalizado	Alta
N/A	Conseguir Sprites de movimiento para los personajes	1	Finalizado	Alta
HU010, HU012, HU013, HU014, HU015	Codificar movimientos de personaje	1	Finalizado	Alta
HU002, HU003, HU004, HU005, HU006, HU007	Implementar la Interfaz de Usuario	2	Finalizado	Media
HU004, HU005,	Implementar cambios de escena y transiciones	2	Finalizado	Media

HU006, HU007				
N/A	Implementar la conexión de multijugador	3	Finalizado	Alta
HU007	Implementar la Interfaz de usuario para conexión multijugador	3	Finalizado	Media
HU011, HU016, HU017, HU018	Jugabilidad con los dos jugadores en partida	4	Finalizado	Alta
TODAS	Realizar pruebas de jugabilidad	5	Finalizado	Media
TODAS	Realizar pruebas de conexión	5	Finalizado	Alta
TODAS	Lanzar versión Beta	5	Finalizado	Alta
TODAS	Corregir errores detectados en Beta	5	Finalizado	Alta
TODAS	Lanzar versión final	5	Finalizado	Alta

4 SPRINT BACKLOG

TABLA XXI Sprint Backlog

Sprint Backlog					
SB-ID	**Nombre**	**HU-ID**	**Historia de Usuario**	**Tareas**	**Tiempo Estimado**
SB001	Conceptualización, Diseño y Codificación de Mecánicas Básicas	N/A	Todas	• Configurar ambiente de desarrollo • Familiarización con la herramienta Unity	3 semanas
		Todas	Todas	• Conceptualizar el videojuego	
		N/A	N/A	• Diseñar escenarios, menús, personajes • Diseñar la estética del juego	
		HU008, HU009, HU010, HU012, HU013, HU014, HU015	Escenario de Combate, Generación de personajes en el escenario de combate, Saltar, Correr, Usabilidad del joystick, Botón de Disparar, Botón de saltar	• Codificar mecánicas básicas de un personaje • Codificar Interfaz de Usuario para las mecánicas básicas	
SB002	Implementación de Interfaz de Usuario y Menús	HU002, HU003, HU007	Pantalla de Inicio, Pantalla de Conexión con otro jugador, Pantalla de Crear una partida	• Implementar las escenas o pantallas diseñadas de acuerdo con los prototipos	1 semana

		HU004, HU005, HU006	Botón Jugar, Botón salir del Juego, Botón Créditos,	• Diseñar los botones y su funcionalidad para cambiar las escenas correspondientes	
SB003	Implementación de conexión multijugador	N/A	N/A	• Implementar la conexión con otro jugador mediante un código de 6 dígitos	3 semanas
SB004	Codificación de mecánicas de lucha	HU011, HU016, HU017	Disparar al otro jugador, Avanzar pisos, Reaparecer	• Codificar sistema de combate y daño	3 semanas
		HU018	Ganar la partida	• Implementar la función de ganar una partida	
SB005	Pruebas, depuración y lanzamiento	N/A	N/A	• Lanzar versión BETA	2 semanas
		Todas	Todas	• Probar menús y botones • Probar mecánicas básicas de los personajes • Probar mecánicas de lucha de los personajes • Probar conexión multijugador • Verificar que un jugador gane una partida	
		HU001	Ejecutar juego	• Ejecutar el juego como una aplicación de un dispositivo Android	

5 ASSETS

En este apartado se muestran los objetos o imágenes usadas para la creación del videojuego que no son parte de la interfaz gráfica. Solo se muestran los elementos que tienen impacto en la parte del *gameplay* como tal y no en la interfaz de usuario.

5.1 Sprites

A continuación, se muestran figuras que se utilizaron para decorar o completar el diseño artístico del juego.

Figura 1: Rótulo de ganador del jugador 2

Figura 2: Background del piso en el que el jugador 1 gana

Figura 3: Rótulo de ganador del jugador 1

Figura 4: Background del menú principal

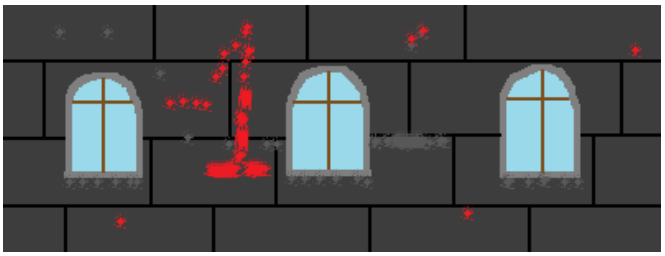

Figura 5: Background del piso -1

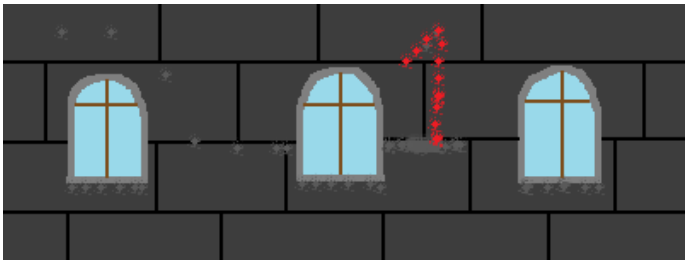

Figura 6: Background del piso 1

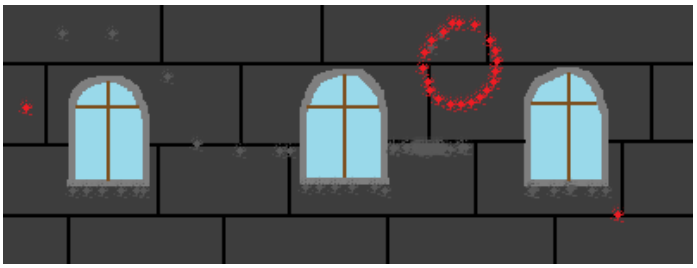

Figura 7: Background del piso 0

Figura 8: Background del piso en el que gana el jugador 2

En las figuras 9 a 12 se muestran ejemplos de los *sprites* que se utilizó para la animación de correr de los personajes.

Figura 9: Sprite del jugadro 1 corriendo

Figura 10: Sprite del jugador 1

Figura 12: Sprite del jugador 2 corriendo

Figura 11: Sprite del jugador 2

6 PRUEBAS

En esta sección se detallan las diferentes pruebas de aceptación realizadas en el sprint 5. Estas evaluaciones fueron hechas por el propio desarrollador y algunos colaboradores.

TABLA XXII Prueba de Aceptación #1

PRUEBA DE ACEPTACIÓN	
Identificador de prueba (ID): PA001	**Identificador de Historia de Usuario:** HU002, HU003, HU004, HU005, HU006, HU007,
Nombre de la prueba de aceptación: Probar Menús e Interfaz de Usuario	
Descripción: El jugador puede navegar por las pantallas de menú principal, créditos y combate libre mente hasta establecer conexión con otro jugador e iniciar la partida.	
Condiciones de la prueba: • Ejecutar el juego • Tener salida de audio • No iniciar una partida	
Pasos de Ejecución: • Iniciar el juego • Pulsar en el botón créditos • Pulsar el botón regresar • Pulsar el botón jugar • Pulsar el botón Crear Partida • Pulsar el botón regresar • Pulsar el botón Unirse a una partida • Pulsar el botón regresar • Pulsar el botón regresar a menú Principal • Pulsar el botón Salir	
Resultado Deseado: Todas las escenas y pantallas hasta iniciar la partida con otro jugador deben ser navegables por el usuario. También, la aplicación debe cerrarse con el botón Salir	
Evaluación de la prueba: El resultado obtenido es igual al deseado. Aprobación: 100%	

TABLA XXIII Prueba de Aceptación #2

PRUEBA DE ACEPTACIÓN	
Identificador de prueba (ID): PA002	**Identificador de Historia de Usuario:** HU009
Nombre de la prueba de aceptación: Probar conexión multijugador con el código	
Descripción: El jugador debe poder conectarse con su contrincante luego de recibir un código de 6 dígitos y compartirlo.	
Condiciones de la prueba:	

• Ejecutar el juego • Estar en la pantalla de Jugar
Pasos de Ejecución: • Pulsar el botón Crear una Partida • Copiar o memorizar el código de 6 dígitos proporcionado • Compartir el código con el jugador 2 • El jugador 2 pulsa el botón Unirse a una partida • Ingresa el código de 6 dígitos • Pulsa el botón Jugar
Resultado Deseado: Los personajes azul y rojo de cada jugador deben aparecer en la escena de combate y estar listos para pelear usando la interfaz de Usuario.
Evaluación de la prueba: Si el proceso es el descrito no existen observaciones. Pero si por alguna razón se ingresa mal el código, el juego no funcionaría correctamente. Se puede mejorar, pero por el tiempo de desarrollo no se logró optimizar. Aprobación: 85%

TABLA XXIV Prueba de Aceptación #3

PRUEBA DE ACEPTACIÓN	
Identificador de prueba (ID): PA003	**Identificador de Historia de Usuario:** HU011, HU016, HU017
Nombre de la prueba de aceptación: Probar mecánicas básicas del combate	
Descripción: En el combate, el jugador puede correr, saltar y disparar a otro jugador. Además de avanzar los pisos del castillo para ganar la partida.	
Condiciones de la prueba: • Ejecutar el juego • Iniciar la conexión con otro jugador	
Pasos de Ejecución: • Ingresar a la escena de combate con el código de 6 dígitos • Utilizar el joystick digital y los botones para atacar a tu oponente • Disparar a tu rival • Matar a tu rival • Avanzar según corresponda al siguiente piso	
Resultado Deseado: El combate debe ser fluido y en todo momento saber la vida tuya y de tu oponente.	
Evaluación de la prueba: Como es un multijugador depende de la conexión de internet y el tipo de teléfono que se use. Al utilizar el servicio de Unity gratuito el transporte de datos no es fluido y a veces suele perderse la conexión. Aprobación: 80%	

TABLA XXV Prueba de Aceptación #4

PRUEBA DE ACEPTACIÓN	
Identificador de prueba (ID): PA004	**Identificador de Historia de Usuario:** HU018
Nombre de la prueba de aceptación: Probar la finalización de la partida	
Descripción: Una vez llegado a su meta el jugador debe acercarse a los tesoros para acabar la partida.	
Condiciones de la prueba: • Ejecutar el juego • Un jugador debe llegar al piso en donde se encuentra su meta	
Pasos de Ejecución: • En el piso donde está su meta debe acercarse al tesoro • Una vez llegue el juego se acaba • Aparece un cartel que avisa a ambos quien ganó • Pulsar el botón “Regresar el Menú Principal”	
Resultado Deseado: Cuando el jugador se acerque a los tesoros se mostrará un cartel que indique a todos los jugadores quien fue el ganador, a la vez que aparezca una opción de ir al menú principal	
Evaluación de la prueba: El proceso para ganar una partida y volver al menú principal para jugar otra está completo. Aprobación: 100%	

ANEXO III

En el siguiente enlace se puede visualizar el manual de usuario en la plataforma de *Youtube* indicando como utilizar la aplicación.

https://youtu.be/oZulXsU-wY8

ANEXO IV

En el siguiente enlace se puede visualizar el manual de Instalación para jugadores y desarrolladores. El vínculo te redirigirá a una página de *Github* en donde se detallará los pasos para la instalación en cada uno de los casos.

juano999/castillo: Trabajo de Integración Curricular 2022-A (github.com)