UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

TEMA:

VIDEOJUEGO EDUCATIVO EN 3D PARA DISPOSITIVOS MÓVILES ANDROID, ENFOCADO AL APRENDIZAJE DE LA LÓGICA DE PROGRAMACIÓN PARA USUARIOS ENTRE LOS 5 A 18 AÑOS DE EDAD.

Trabajo de Graduación Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniero en Sistemas Computacionales e Informáticos

SUBLÍNEA DE INVESTIGACIÓN:

Aplicaciones para dispositivos móviles

AUTOR: Hussein Gabriel Rahman Núñez

TUTOR: Ing. Clay Fernando Aldás Flores, Mg

Ambato - Ecuador

Mayo, 2017

APROBACIÓN DEL TUTOR

En mi calidad de Tutor del Trabajo de Investigación sobre el Tema:

"VIDEOJUEGO EDUCATIVO EN 3D PARA DISPOSITIVOS MÓVILES ANDROID, ENFOCADO AL APRENDIZAJE DE LA LÓGICA DE PROGRAMACIÓN PARA USUARIOS ENTRE LOS 5 A 18 AÑOS DE EDAD", del señor Hussein Gabriel Rahman Núñez, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el numeral 7.2 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ambato, mayo de 2017

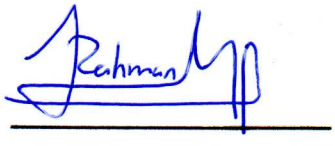
Ing. Clay Fernando Aldás Flores, Mg

EL TUTOR

AUTORÍA

El presente trabajo de investigación titulado: Videojuego educativo en 3D para dispositivos móviles Android, enfocado al aprendizaje de la Lógica de Programación para usuarios entre los 5 a 18 años de edad.. Es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, mayo de 2017



Hussein Gabriel Rahman Núñez

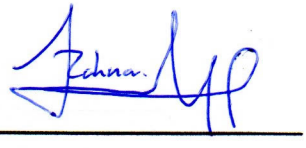
CC: 1721563599

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación, con fines de difusión pública, además autorizo su reproducción dentro de las regulaciones de la Universidad.

Ambato, mayo de 2017



Hussein Gabriel Rahman Núñez

CC: 1721563599

APROBACIÓN DEL TRIBUNAL DE GRADO

La Comisión Calificadora del presente trabajo conformada por los señores docentes Ing. David Guevara e Ing. Carlos Núñez, revisó y aprobó el Informe Final del trabajo de graduación titulado "VIDEOJUEGO EDUCATIVO EN 3D PARA DISPOSITIVOS MÓVILES ANDROID, ENFOCADO AL APRENDIZAJE DE LA LÓGICA DE PROGRAMACIÓN PARA USUARIOS ENTRE LOS 5 A 18 AÑOS DE EDAD", presentado por el señor Hussein Gabriel Rahman Núñez de acuerdo al numeral 9.1 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ing. Mg. Elsa Pilar Urrutia Urrutia

PRESIDENTE DEL TRIBUNAL

Ing. Mg. David Guevara	Ing. Mg. Carlos Núñez
DOCENTE CALIFICADOR	DOCENTE CALIFICADOR

DEDICATORIA

El presente trabajo dedico con todo mi cariño y esfuerzo.

A Dios por darme la fuerza y valentía diaria de seguir adelante en todos los momento de mi vida.

A mis padres y a mi hermano quienes fueron el eje fundamental para lograr este objetivo, dandome todo su apoyo para trazar mis metas y objetivos.

A la Facultad de Ingeniería en Sistemas Electrónica e Industrial.

A todos ellos muchas gracias.

Hussein Gabriel Rahman Núñez

AGRADECIMIENTO

Agradezco a Dios, a mis padres y a mi hermano quienes me brindaron su sabiduria y apoyo incondicional para culminar una etapa muy importante en mi vida como es mi formación profesional.

Además quiero extender un sincero agradecimiento a la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, y a todos los docentes que compartieron su conocimiento conmigo, por haberme acogido en estos años de incansable lucha para poder lograr un importante objetivo en mi vida, formándome como un profesional para seguir creciendo cada día más y más.

Hussein Gabriel Rahman Núñez

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Resumen ejecutivo

Una nueva tendencia mundial está cambiando la forma de aprender la programación, los países del primer mundo conocen la importancia de la programación para el futuro tecnológico e incentivan a los niños por medio de juegos y herramientas multimedia como el ya conocido proyecto Scratch desarrollado por el MIT (Massachusetts Institute of Technology).

En el país no se han realizado este tipo de proyectos y la malla educativa del nivel básico no contiene ninguna materia que cubra este conocimiento; es por ello que surge la necesidad de desarrollar un videojuego educativo para dispositivos móviles Android con el fin de llegar a la mayor cantidad de usuarios de esta plataforma.

El desarrollo de este videojuego se lo realizó utilizando el motor Unity3D junto con el lenguaje C#, mientras que para el diseño gráfico se utilizó Blender y Gimp principalmente.

Asimismo, para la planificación y desarrollo del proyecto se empleó la metodología DAV (Desarrollo ágil de videojuegos) propuesta por Gabriel Armendáriz estudiante de la facultad de Informática de la EPOCH (Escuela Superior Politécnica de Chimborazo).

Los resultados después de la publicación del videojuego en la tienda online GameJolt fueron muy satisfactorios, las personas que lo jugaron comprendieron el objetivo del proyecto aportando con criticas positivas y un número significativo de descargas tomando en cuenta que se trata de un videojuego independiente.

Abstract

A new global trend is changing the way we learn to program, first world countries know the importance of programming for the technological future and encourage children through multimedia games and tools such as the well-known Scratch project developed by the Massachusetts Institute of Technology (MIT).

In the country, this type of project has not been carried out and the educational network at the basic level does not contain any material that covers this knowledge; Is why the need arises to develop an educational video game for Android mobile devices in order to reach the largest number of users of this platform.

The development of this videogame was done using the Unity3D engine and C # language, while for the graphic design used Blender and Gimp mainly.

Likewise, for the planning and development of the project was used the DAV methodology (Agile video game development) proposed by Gabriel Armendáriz student of the computer science faculty of the EPOCH.

The results after the publication of the video game in the online store GameJolt were very satisfactory, the people who played it understood the objective of the project, obtained many positive reviews and a significant number of downloads considering that it is an independent game.

Glosario de términos

Assets.- Los Assets o "Activos" son los recursos que por defecto proporciona Unity y que se pueden utilizar en la creación de un nuevo proyecto. Los tipos de recursos que ofrece son numerosos, desde cámaras o personajes a efectos o sistemas de partículas por citar algunos.

DirectX.- Es una colección de API (Application Programming Interface) desarrolladas para facilitar las complejas tareas relacionadas con multimedia, especialmente programación de juegos y vídeo, en la plataforma Microsoft Windows.

Entregable.- El término entregable es utilizado en la gestión de proyectos para describir un objeto, tangible o intangible, como resultado del proyecto, destinado a un cliente, ya sea interno o externo a la organización.

Frames per seconds.- Es la velocidad (tasa) a la cual un dispositivo muestra imágenes llamadas cuadros o fotogramas. El término aplica igualmente a películas y cámaras de video, gráficos computacionales y sistemas de captura de movimiento. La tasa de refrescamiento se expresa en cuadros por segundo o por la sigla en inglés FPS (de frames per second).

Framework.- Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.

Freelance.- Se denomina freelance (o trabajador autónomo, por cuenta propia o trabajador independiente) a la persona cuya actividad consiste en realizar trabajos propios de su ocupación.

GameObject.- Es un término usado dentro de Unity para describir a un objeto dentro del proyecto.

Gameplay.- Es un término empleado en el diseño y análisis de juegos que describe la calidad del juego en términos de sus reglas de funcionamiento y de su diseño como juego.

Indentar.- Es un anglicismo de uso común en informática, que significa mover

un bloque de texto hacia la derecha insertando espacios o tabuladores.

Indie.- Es un término inglés que significa independiente o independencia.

Joystick.- Es una palanca de control que permite desplazar manualmente, y con gran rapidez, el cursor en una pantalla de computadora o videojuego; se usa especialmente en programas informáticos de juego.

Mapeado.- Es un término usado en diseño 3D que se refiere a la ubicación de la textura sobre el objeto al momento de proyectarse.

OpenGL.- (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.

PowerUps.- Es un término usado en videojuegos que se refiere al aumento de poder.

Prefab.- Es un tipo de asset que permite almacenar un objeto GameObject completamente con sus propiedades. El prefab actúa como una plantilla a partir de la cual se pueden crear nuevas instancias del objeto en la escena.

Puzzle.- Es un género de videojuegos que se caracterizan por exigir agilidad mental al jugador. Pueden involucrar problemas de lógica, matemáticas, estrategia, reconocimiento de patrones, completar palabras o hasta simple azar.

Refactorizar.- Es una técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

Rendering.- Es un término usado en para referirse al proceso de generar una imagen desde un modelo. Este término técnico es utilizado por los animadores o productores audiovisuales y en programas de diseño en 3D.

Rigging.- Es la acción o el efecto de hacer rigs. Los rigs son sistemas de cadenas de huesos y objetos de control, con o sin características interactivas, que sirven para definir deformaciones sobre un objeto geométrico.

Scripting.- Es la acción de crear scripts, un script es un documento que contiene instrucciones, escritas en códigos de programación.

Shooter.- Es un género que engloban un amplio número de subgéneros que tienen la característica común de permitir controlar un personaje que, por norma general, dispone de un arma (mayoritariamente de fuego) que puede ser disparada a voluntad.

Sprite.- Son mapas de bits en 2D que se dibujan directamente en un destino de representación sin usar la canalización de transformaciones, iluminación o efectos. Se suelen usar para mostrar información como las barras de estado, el número de vidas o texto como las puntuaciones..

Storyboard.- Es un conjunto de ilustraciones mostradas en secuencia con el objetivo de servir de guía para entender una historia, pre-visualizar una animación o seguir la estructura de una película antes de realizarse o filmarse.

Swipe.- Es un término para referirse a la acción al desplazar el dedo sobre una pantalla táctil.

Viewport.- Es una región de visualización de polígonos o gráficos tridimensionales, es un término que se utiliza comúnmente en software para modelamiento en 3D.

WebGL.- Es una especificación estándar que está siendo desarrollada actualmente para mostrar gráficos en 3D en navegadores web. El WebGL permite mostrar gráficos en 3D acelerados por hardware (GPU) en páginas web, sin la necesidad de plugins.

INTRODUCCIÓN

El presente proyecto de investigación denominado: "VIDEOJUEGO EDUCATIVO EN 3D PARA DISPOSITIVOS MÓVILES ANDROID, ENFOCADO AL APRENDIZAJE DE LA LÓGICA DE PROGRAMACIÓN PARA USUARIOS ENTRE LOS 5 A 18 AÑOS DE EDAD"; se encuentra dividido en los siguientes capítulos:

CAPÍTULO I. "EL PROBLEMA", identifica el problema a investigar y además se plantea la justificación por la cual se investiga así como los objetivos necesarios que guiarán al desarrollo del proyecto.

CAPÍTULO II. "MARCO TEÓRICO", establece el conjunto de conocimientos en los cuales se sustenta la investigación, además de presentar investigaciones previas que sirven de soporte a la investigación y se establece la propuesta de solución al problema.

CAPÍTULO III. "METODOLOGÍA", se indica la metodología que se utilizará especificando las técnicas e instrumentos para recolectar, procesar y analizar la información recabada, también se definen las etapas para el desarrollo del proyecto de investigación.

CAPÍTULO IV. "DESARROLLO DE LA PROPUESTA", en este capítulo se detalla de una manera clara el desarrollo de la propuesta, comparando diferentes frameworks y metodologías para desarrollo móvil que cumplan con los requerimientos del proyecto.

CAPÍTULO V. "CONCLUSIONES Y RECOMENDACIONES", se describe las conclusiones a las que llega el investigador luego de terminar el desarrollo del proyecto, así también se describe las recomendaciones pertinentes.

CAPÍTULO 1

El problema

1.1. Tema de investigación

Videojuego educativo en 3D para dispositivos móviles Android, enfocado al aprendizaje de la Lógica de Programación para usuarios entre los 5 a 18 años de edad.

1.2. Planteamiento del problema

La programación a través de los años ha tomado un papel muy importante para el desarrollo tecnológico de la humanidad, ya que cualquiera de los sistemas actuales no funcionarían sin alguien que lo hubiera programado con la lógica apropiada para que funcione, sin la programación muchos avances en el área tecnológica no podrían ser una realidad, desde el momento en que encienden el computador, cuando escriben algún documento, cuando realizan alguna búsqueda en Internet, hasta incluso cuando se juega o se ve algún video en la red, hacen uso de diferentes sistemas que funcionan gracias a la programación, los algoritmos se encuentran en la vida cotidiana sin que las personas se den cuenta [1].

En la vida cotidiana, se usan algoritmos en todo momento, lo que pasa es que se lo hace de forma inconsciente, por ejemplo cuando se va al colegio o universidad no se piensa: "preparo la mochila, desayuno, tomo las llaves de la puerta, abro la puerta, cierro la puerta, voy a la parada del bus, tomo el bus" sino que estas son acciones que se realizan de forma automática y sin pensarlas paso a paso, además muchas veces se olvida de estas acciones repetitivas muy similares a las realizadas en un algoritmo repetitivo. Para todos los seres humanos la programación es algo que debería ser muy importante e indispensable, ya que la única manera de conseguir exactamente lo que se quiere es llevando a cabo una serie de pasos específicos y ordenados. La programación es un factor fundamental en todos los aspectos de la cotidianidad, seguir un orden específico de pasos (aunque estos se hagan por inercia) se convierte en más que una opción, una necesidad para lograr satisfacer las necesidades que se presentan a diario [1].

Una de las grandes tendencias educativas globales ahora mismo consiste en acercar el lenguaje de programación a los niños, desde los cuatro años en adelante. En el pasado quedaron las viejas lecciones de Informática, con protocolos incomprensibles. Hoy en día, algunas empresas e instituciones públicas y privadas de todo el mundo se han tomado tan en serio el objetivo convertir la escritura de código en una tarea tan cotidiana como escribir una redacción, por lo que proliferan en la red toda clase de juegos para dar los primeros pasos [2].

A nivel mundial en Estados Unidos existen iniciativas como *code.org* ofrece a los niños cursos de iniciación de 20 horas adaptados para distintos tramos de edad, el más temprano de cuatro a seis años. De una naturaleza similar es el proyecto *Hour of Code (La Hora del Código)* que ofrece tutoriales de una hora para que cualquiera pueda organizar un evento en cualquier lugar del mundo en el que niños a partir de cuatro años descubran que pueden programar. Por otro lado los genios del *MIT (Massachusetts Institute of Technology)* invitan a los más pequeños a adentrarse en el mundo de la informática mediante si juego *Scratch*, y lo mejor es que existe una versión disponible en español. Para los niños desde los 13 años de edad, la *Universidad de Stanford* ha diseñado clases especiales de aproximación a la programación disponibles en *Youtube*, por otro lado *EdSurge* publica hasta 50 herramientas con esta declaración de intenciones: "Hemos enseñado a los niños a cultivar plantas, construir una casa, forjar una espada o soplar un cristal delicado, cocinar pan, crear un soufflé, escribir una historia o lanzar aros. Ahora les estamos enseñando código" [2].

En Inglaterra se a creado un nuevo enfoque a la enseñanza de la informática en niños, dejando atrás al viejo modelo educativo en el cual solo se enseñaba programas de ofimatica durante toda su vida escolar. Hoy en día el plan educativo de Inglaterra es la enseñanza de la programación en niños desde los 5 hasta los 14 años de edad, no quieren crear desarrolladores, quieren que los niños empiecen a ser creativos [3].

En el Ecuador no existen actividades que incentiven al aprendizaje de la lógica de programación en edades tempranas, y para completar este panorama la malla curricular del nivel de educación básica no cuenta con materias relacionadas con la *Lógica de Programación*. El perfil de salida del estudiante en el país [4] no se enfoca ni un poco al área de la informática, durante años se ha venido enseñando lo mismo una y otra vez (como manejar Word, como manejar Excel, etc).

En las diferentes Universidades del país incluida la Universidad Técnica de Ambato no se han desarrollado videojuegos enfocados al aprendizaje de la lógica de programación, solo la Escuela Politécnica del Ejercito en la carrera de Sistemas e Informática ha realizado proyectos afines enfocándose principalmente en el aprendizaje del razonamiento.

Todo lo anteriormente descrito da como resultado la falta de una herramienta didáctica apropiada que incentive a los niños y niñas al aprendizaje de la *Lógica de Programación.*

1.3. Delimitación

Área Académica: Software
Línea de investigación: Desarrollo de Software
Sublíneas de investigación: Aplicación móvil
Delimitación Espacial: Este videojuego no está delimitado para un área geográfica específica, el objetivo es publicar el proyecto en una tienda virtual como Google PlayStore o Gamejolt, por lo que la delimitación espacial del proyecto es a nivel global.
Delimitación Temporal: La presente investigación se desarrollará durante los 6 meses posteriores a la aprobación del proyecto por parte del Consejo Directivo de la Facultad.

1.4. Justificación

Las necesidades cada vez más crecientes del sistema educativo que obliga a los estudiantes a ser parte de esta nueva era de cambio, en donde ya no solo deben dedicarse a estudiar, deben ser investigadores y empezar a innovar para llegar a ser competitivos.

La Programación sigue siendo uno de los pilares fundamentales para el desarrollo de la tecnología actual, además como se detalló en el planteamiento del problema, tanto las escuelas como los colegios no brindan la posibilidad del aprendizaje de la Lógica de Programación como una asignatura en su malla curricular, asignatura que actualmente es muy importante para el desarrollo del razonamiento lógico en niños y adolescentes.

Los niños y adolescentes aprenden de mejor manera cuando se divierten, el modelo tradicional de aprendizaje muchas veces es aburrido, tedioso y mecánico, por lo que un videojuego es la mejor forma de captar la atención y que el propio niño se convierta en el constructor de su aprendizaje.

1.5. Objetivos

1.5.1. General

- Desarrollar un videojuego educativo en 3D para dispositivos móviles Android, enfocado al aprendizaje de la Lógica de Programación para usuarios entre los 5 a 18 años de edad.

1.5.2. Específicos

- Realizar un estudio de los principales Frameworks existentes en el mercado tecnológico para el desarrollo de videojuegos para dispositivos móviles.
- Comparar diferentes motores de juegos tanto de tipo comercial como Open Source para el desarrollo de aplicaciones sobre plataformas móviles.
- Seleccionar una metodología ágil aplicada al campo del desarrollo de videojuegos móviles que facilite el desarrollo e implementación del proyecto planteado.
- Implementar el videojuego educativo en 3D para dispositivos móviles Android enfocado al aprendizaje de la Lógica de Programación.

CAPÍTULO 2

Marco Teórico

2.1. Antecendentes investigativos

Los videojuegos son juegos digitales interactivos, se trata de softwares ejecutables en dispositivos electrónicos diversos tales como computadoras, teléfonos móviles, consolas, tabletas entre otros. En general los videojuegos recrean entornos y situaciones en los que el jugador (o varios jugadores) pueden controlar personajes e interactuar con el entorno para alcanzar un objetivo determinado. Un juego puede resultar tanto o más enriquecedor que cualquier otra actividad siempre y cuando el mismo ofrezca oportunidades de aprendizaje y sea abordado con la orientación y guía necesarias para aprovechar su valor educativo [5].

Realizando un estudio sobre los videojuegos móviles existentes en el mercado global enfocados al aprendizaje de la lógica de programación se encontraron los siguientes:

MIT MEDIA LAB desarrolló un lenguaje de programación visual diseñado para introducir las habilidades de programación en niños de 5-7 años de edad. Mediante la creación de proyectos con ScratchJr, los niños pequeños pueden aprender a pensar de forma creativa y razonar de forma sistemática. Está disponible como una aplicación gratuita para iOS, Android y Chromebook. ScratchJr es un derivado del lenguaje Scratch, que ha sido utilizado por más de 10 millones de personas en todo el mundo. La codificación en Scratch requiere habilidades básicas de lectura, sin embargo, los creadores vieron una necesidad de otro lenguaje de programación que proporcionaría una forma simplificada de codificación para aprender a una edad más joven y sin ningún tipo de lectura obligatoria [6].

Danny Yaroslavski realizó el videojuego educativo Lightbot para el aprendizaje de conceptos de programación de software. Este videojuego se ha jugado 7 millones de veces, y es muy valorado en iTunes y Google Play Store, está disponible como un juego flash en línea, y una aplicación para teléfonos móviles Android y iOS. Lightbot se ha construido con Flash y OpenFL. El objetivo de Lightbot es mover

a un pequeño robot para navegar por un laberinto usando diferentes comandos que se basan en conceptos de programación básicos y así poder encender unas luces y avanzar al siguiente nivel [7].

Elias Cisneros en su proyecto de Tesis: *"Videojuego Educativo como apoyo a la enseñanza de la algoritmia para los estudiantes del programa nacional de formación en Sistemas e Informática"* desarrolla un videojuego en 3D para el aprendizaje de conceptos de la programación, su proyecto lo delimita para los estudiantes de la carrera de Informática del Instituto Superior Politécnico José Antonio Echeverría en Cuba [8].

A nivel del país no se han desarrollado videojuegos móviles enfocados al aprendizaje de la programación, aunque existen proyectos afines realizados por la Escuela Politécnica del Ejercito en la carrera de Sistemas e Informática, los autores de los proyectos que se citan a continuación mencionan la importancia de los videojuegos lúdicos para mejorar el aprendizaje en niños y jóvenes.

Karla Albuja en su proyecto: *"Análisis, diseño y desarrollo de un juego didáctico de razonamiento abstracto en 3d, para ayudar al desarrollo del pensamiento de niños entre 4 y 8 años, utilizando un game engine con c# y aplicando la metodología OOHDM"* habla sobre las ventajas y desventajas del uso de la tecnología en los niños y plantea una aplicación que permitirá tener las bases teóricas, prácticas, habilidades y destrezas en la formación integral de los niños provocando en ellos un interés y un aprendizaje fácil mediante el uso juegos educativos interactivos en 3D [9].

Margaritha Zambrano en su proyecto: *"Diseño y desarrollo de un video-juego educativo con técnicas de inteligencia artificial para plataforma Android utilizando metodología OOHDM. Caso de estudio: Laberinto 3D"* enfatiza que los videojuegos lúdicos o educativos son uno de los tipos de videojuegos que más se aplican como parte del proceso de enseñanza/aprendizaje en los niños para el desarrollo del pensamiento y el desarrollo psicomotriz de los mismos [10].

Marcelo Torres en su proyecto: *"Aplicación de la Metodología OOHDM y Técnicas de Inteligencia Artificial en la Solución del Desarrollo de un videojuego, enfocado a niños de 6 a 10 años, utilizando la Tecnología gdi+ basado en c# y Wiimote, para su aplicación en la Empresa Virtual Learni"* propone un videojuego edu-

cativo para ser vendido por una empresa comercial y su planteamiento se basa principalmente en rompecabezas didácticos hechos con animales ecuatorianos autóctonos con el propósito que los niños generen un poco de conciencia hacia el país [11].

Además de los autores citados anteriormente, el Dr. Christopher Thomas Miller autor del libro *"Games Purpose and Potential in Education"* habla sobre como los profesores de centros educativos de educación básica actualmente ven como una herramienta para el desarrollo de aprendizaje a los juegos, cuando antes la palabra juego y educación no podían estar en el mismo contexto, además enfatiza que usar juegos como material de aprendizaje en niños de temprana edad ayuda a desarrollar más rápidamente su razonamiento y pensamiento lógico [12].

Finalmente mediante una revisión de las tesis realizadas en la Facultad de Ingeniería en Sistemas Electrónica e Industrial de la Universidad Técnica de Ambato no se encontraron investigaciones que tengan relación con el tema en estudio.

2.2. Fundamentación teórica

2.2.1. Concepto de videojuego

Dentro del mundo del entretenimiento electrónico, un videojuego normalmente se suele asociar a la evolución, entendida desde un punto de vista general, de uno o varios personajes principales o entidades que pretenden alcanzar una serie de objetivos en un mundo acotado, los cuales están controlados por el propio usuario [13].

De este modo, existe una interacción explícita entre el jugador o usuario de videojuegos y el propio videojuego, el cual plantea una serie de retos al usuario con el objetivo final de garantizar la diversión y el entretenimiento. Además de ofrecer este componente emocional, los videojuegos también suelen tener un componente cognitivo asociado, obligando a los jugadores a aprender técnicas y a dominar el comportamiento del personaje que manejan para resolver los retos o puzzles que los videojuegos plantean [13].

2.2.2. Videojuegos "Indies"

Los videojuegos "Indies" nacieron como una filosofía (o necesidad) de crear videojuegos en una habitación con poco o ningún presupuesto, 2 o 3 personas dejando su alma para crear un videojuego sin seguir las normas impuestas de los productores. Los juegos independientes empezaron como una corriente alternativa en la que la originalidad no iba de la mano con el éxito. Apuestas arriesgadas que no tenían porqué ser populares y que descubrieron una inquietud en muchos de los jugadores empachados de súper producciones. Los productores no se arriesgan, no deja de ser un negocio y lo que quieren es ganar dinero de forma segura (o todo lo seguro que puede ser este mercado) [14].

Crear un videojuego nunca había sido tan fácil (desde la época de los 8 bits con Basic como estandarte) ya que existen cientos de herramientas que reducen el duro camino del desarrollo. Programas como *Game Maker* o *Unity 3D* permiten hacer cosas increíbles con un menor esfuerzo. Además con la era de la información en la que se vive, es fácil aprender cualquier cosa por medio del internet [14].

2.2.3. Videojuegos educativos

En los últimos años aumentó sensiblemente la oferta de videojuegos educativos, que se presentan como una alternativa a los videojuegos tradicionales. Este incremento viene motivado por varios factores, como se extrae de los estudios de Pérez Martín, entre los que destaca la madurez de las empresas desarrolladoras españolas, lo que implica productos de gran calidad con buenos guiones y acabados, entrada en la cadena de distribución y actividades de promoción con el fin de ser conocidos por el público, la apuesta por el mercado del PC (Personal Computer) y, en la actualidad, por el de las consolas junto con la expansión de las capacidades multijugador de los videojuegos y del hardware [15].

Los videojuegos favorecen los reflejos, la psicomotricidad, la iniciativa y la autonomía, pudiéndose introducir en la educación con una finalidad didáctica, para contribuir al logro de determinados objetivos educativos. Existen diferentes aspectos que permiten desarrollar los videojuegos como son:

1. **Aspectos cognitivos:** memorización de hechos; observación hacia los detalles; percepción y reconocimiento espacial; descubrimiento inductivo; capacidades lógicas y de razonamiento; comprensión lectora y vocabulario; conocimientos geográficos, históricos, matemáticos, resolución de problemas

y planificación de estrategias [15].

2. **Destrezas y habilidades:** autocontrol y autoevaluación; implicación y motivación, instinto de superación; inversión de esfuerzo que es reconocido de forma inmediata; habilidades motrices, de reflejos y respuestas rápidas; percepción visual, coordinación óculo-manual, y percepción espacial; curiosidad e inquietud por probar y por investigar [15].

3. **Aspectos socializadores:** aumenta la autoestima, proporcionan un sentido de dominio, control y cumplimiento, debido en gran parte a que existen recompensas personalizadas; interacción con amigos de manera no jerárquica (presencial o a distancia)[15].

2.2.4. Motor de juego

El término motor de juego surgió a mediados de los años 90 con la aparición del famoso juego de acción en primera persona Doom, desarrollado por la compañía *id Software* bajo la dirección de J*ohn Carmack*. Esta afirmación se sustenta sobre el hecho de que *Doom* fue diseñado con una arquitectura orientada a la reutilización mediante una separación adecuada en distintos módulos de los componentes fundamentales, como por ejemplo el sistema de renderizado gráfico, el sistema de detección de colisiones o el sistema de audio, y los elementos más artísticos, como por ejemplo los escenarios virtuales o las reglas que gobernaban al propio juego [13].

Arquitectura de un motor de juego

Como ocurre con la gran mayoría de sistemas software que tienen una complejidad elevada, los motores de juegos se basan en una arquitectura estructurada en capas. De este modo, las capas de nivel superior dependen de las capas de nivel inferior, pero no de manera inversa [13].

- **Gestor de recursos.-** Esta capa es la responsable de proporcionar una interfaz unificada para acceder a las distintas entidades software que conforman el motor de juegos, como por ejemplo la escena o los propios objetos 3D [13].

- **Motor de rendering.-** El motor de renderizado es una de las partes más complejas de cualquier motor de juego. En la capa de rendering sólo se

dibujan las primitivas que están dentro del campo de visión de la cámara, es decir, dentro del viewport, es posible aplicar más optimizaciones que simplifiquen la complejidad de la escena a renderizar, obviando aquellas partes de la misma que no son visibles desde la cámara [13].

- **Motor de física.-** La detección de colisiones en un videojuego y su posterior tratamiento resultan esenciales para dotar de realismo al mismo. Sin un mecanismo de detección de colisiones, los objetos se traspasarían unos a otros y no sería posible interactuar con ellos. Un ejemplo típico de colisión está representado en los juegos de conducción por el choque entre dos o más vehículos [13].

- **Interfaces de usuario.-** En cualquier tipo de juego es necesario desarrollar un módulo que ofrezca una abstracción respecto a la interacción del usuario, es decir, un módulo que principalmente sea responsable de procesar los eventos de entrada del usuario. Típicamente, dichos eventos estarán asociados a la pulsación de una tecla, al movimiento del ratón o al uso de un joystick, entre otros [13].

- **Audio.-** Tradicionalmente, el mundo del desarrollo de videojuegos siempre ha prestado más atención al componente gráfico. Sin embargo, el apartado sonoro también tiene una gran importancia para conseguir una inmersión total del usuario en el juego. Por ello, el motor de audio ha ido cobrando más y más relevancia [13].

2.2.5. Metodologías tradicionales de desarrollo de software

Las metodologías tradicionales son denominadas, a veces, de forma peyorativa, como metodologías pesadas, centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto. Otra de las características importantes dentro de este enfoque, son los altos costes al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil. Las metodologías tradicionales (formales) se focalizan en la documentación, planificación y procesos (plantillas, técnicas de administración, revisiones, etc.)[16] .

Entre las principales metodologías tradicionales tenemos los ya conocidos RUP (RATIONAL UNIFIED PROCESS) y MSF (MICROSOFT SOLUTION FRAMEWORK) entre otros, que centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido

todo esto, en la fase inicial del desarrollo del proyecto.

- **RUP.-** Es un proceso formal: Provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando cronograma y presupuesto). Fue desarrollado por *Rational Software*, y está integrado con toda la suite *Rational* de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Es guiado por casos de uso y centrado en la arquitectura, y utiliza UML como lenguaje de notación [17].

- **MSF.-** Es un enfoque personalizable para entregar con éxito soluciones tecnológicas de manera más rápida, con menos recursos humanos y menos riesgos, pero con resultados de más calidad. MSF ayuda a los equipos a enfrentarse directamente a las causas más habituales de fracaso de los proyectos tecnológicos y mejorar así las tasas de éxito, la calidad de las soluciones y el impacto comercial [18].

2.2.6. Metodologías ágiles

El desarrollo ágil de software se refiere a métodos de Ingeniería del Software basados en el desarrollo iterativo e incremental, estas metodologías son imprescindibles en un mundo en el que nos exponemos a cambios recurrentemente. Siempre hay que tener en cuenta como programadores que lo que es la última tendencia hoy puede que no exista mañana y por esto existe la metodología ágil donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto organizados y multidisciplinarios [19].

Según el estudio de la Universidad de la República de Uruguay, las empresas que realizan videojuegos en equipos pequeños y de corta duración no tienen una metodología de desarrollo formalizada. Las metodologías que utilizan siguen principios de las metodologías ágiles que se adaptan con éxito para el desarrollo de videojuegos a nivel mundial y aplican a realidades similares. En particular se registran casos de éxito con adaptaciones de SCRUM () y XP (Extreme Programming), aunque estas tampoco se encuentran formalizadas [20].

- **SCRUM. -** El SCRUM es un proceso de la Metodología Ágil que se usa para minimizar los riesgos durante la realización de un proyecto, pero de manera colaborativa. Entre las ventajas se encuentran la productividad, calidad y que se realiza un seguimiento diario de los avances del proyecto, logrando que los integrantes estén unidos, comunicados y que el cliente vaya viendo los avances. La profundidad de las tareas que se asignan en SCRUM tiende a ser incremental, caso que coincide exactamente con el devenir normal de un desarrollo [19].

- **XP. -** La "programación extrema" es un proceso de la Metodología Ágil que se aplica en equipos con muy pocos programadores quienes llevan muy pocos procesos en paralelo. Consiste entonces en diseñar, implementar y programar lo más rápido posible, hasta en casos se recomienda saltar la documentación y los procedimientos tradicionales. Se fundamenta en la capacidad del equipo para comunicarse entre sí y las ganas de aprender de los errores propios inherentes en un programador [19].

La tendencia a utilizar metodologías ágiles para videojuegos tomó fuerza en los últimos años por existir varios casos de empresas en la industria que logran adaptar estas metodologías, entre las empresas con casos de éxito documentados se encuentran *Crytek* y *DICE (EA Digital Illusions Creative Entertainment)* que utilizan SCRUM, *Titus Interactive Studios* que utiliza XP y *High Moon Studios* que utiliza ambas. A pesar de los beneficios que reportan, ninguna de estas adaptaciones está especificada formal y públicamente [20].

2.2.7. Lógica de programación

Consideraciones algorítmicas sobre el pensamiento humano

Los algoritmos informales.- son aquellos que no se realizan para ser ejecutados por una computadora, sino se diseñan para ser ejecutados por el ser humano. Todos los días se ejecutan algoritmos informales para realizar actividades: al prepararse para ir al trabajo, al vestirse, al comer, al regresar a casa, etc [21].

Los algoritmos computacionales.- son aquellos que son ejecutados por una computadora. Se aprovecha la velocidad de procesamiento del ordenador para obtener un resultado mucho más confiable, por ejemplo el cálculo de una raíz cuadrada [21].

Analizando desde muchos ángulos del pensamiento humano y teniendo en cuenta los conceptos de algoritmo informal y algoritmo computacional se llegó a la conclusión de que dicho pensamiento se mueve entre tres estructuras básicas [21]:

- Secuencias
- Decisiones
- Ciclos

Secuencias

Para escribir una secuencia de ordenes o acciones todo lo que tiene que hacer es colocar una nueva orden o una nueva acción después de la última que haya colocado. De esta manera se entiende la secuencialidad y la ordinalidad en la ejecución de esas acciones [21].

El siguiente algoritmo permite asomarse por la ventana, pero asumiendo que la ventana ya está abierta y que no existe mucha distancia hacia la ventana. Por lo tanto el algoritmo quedaría de la siguiente manera:

```
1 Algoritmo para asomarnos a la ventana
2 Inicio
3   Ubicar la ventana por la que nos queremos asomar
4   Levantarnos del lugar en donde estemos sentados
5   Avanzar hacia la ventana Llegar hasta tener la ventana muy muy cerquita
6   Asomarnos por la ventana
7 Fin
```

Figura 2.1: Algoritmo de secuencias
Fuente: [21]

En el ejemplo dado se puede ver que cada acción está antes de una y después de otra (excepto por supuesto la primera y la última). También puede notar que para que este algoritmo permite asomarse por la ventana, todo lo hay que hacer es realizar cada acción en el orden en que están planteadas y sencillamente realizar una a la vez. Eso permite lograr el objetivo propuesto.

Decisiones

Siempre que se necesite tomar una decisión o, mas bien, siempre que se utilice la estructura de Decisiones será necesaria una condición. La condición es lo que

permite elegir que camino lógico tomar [21].

A continuación se muestra el mismo algoritmo de asomarse a la ventana pero esta vez se añadirán unas líneas de decisión que permitirán crear la estructura de Decisiones, así :

```
1  Algoritmo para asomarnos a la ventana
2  Inicio
3    Ubicar la ventana por la que nos queremos asomar
4    Si estamos sentados
5      Levantarnos del lugar en donde estemos sentados
6      Orientarnos hacia la ventana
7    Sino
8      Orientarnos hacia la ventana
9      Avanzar hacia la ventana
10     Llegar hasta tener la ventana muy muy cerquita
11     Si esta cerrada Abrirla
12     Asomarnos por la ventana
13 Fin
```

Figura 2.2: Algoritmo de decisiones
Fuente: [21]

1. Las palabras **Si** que aparecen son exclusivamente condicionales y no afirmativas como pudiera pensarse en algunos casos.

2. Después de cada **Si** condicional va una condición que es la que permite que se haga una cosa u otra. La condición regula las acciones que vienen después y que dependen del **Si** condicional inicial. En la decisión.

Ciclos

Se va a suponer que un supervisor de una fábrica a lo largo de todo el día, debe estar vigilando determinada acción a través de una ventana. El algoritmo para cumplir su objetivo que es el de Vigilar (como supervisor de la fábrica) parte de una tarea muy sencilla la cual es **"asomarse"** por una ventana. En palabras sencillas el supervisor tendrá que asomarse por una ventana mientras no termine el día. De esta forma, la estructura para este algoritmo es la siguiente [21]:

```
1 Algoritmo para Vigilar desde una ventana
2 Inicio
3   Llegar puntual a la hora de inicio de la jornada laboral
4   Ubicarnos en nuestro escritorio
5   Sentarnos cerca de la ventana
6   Mientras no sea el fin del día
7     Asomarse por la ventana
8   Fin_Mientras
9 Fin
```

Figura 2.3: Algoritmo de ciclos
Fuente: [21]

Varios factores nuevos entran en este algoritmo:

1. La palabra **Mientras** establece en relación con una condición el inicio de un conjunto de acciones que se repiten precisamente mientras esa condición lo permita

2. Todo **Mientras** (por efectos de clarificación del algoritmo) debe tener un finalizador que indique hasta donde llega el bloque de acciones que debemos repetir.

3. La indentación o lo que corrientemente se conoce como el **"Sangrado"** del texto es decir el hecho de que algunas acciones estén mas adentro de la hoja que otras, representa que existen bloques de acciones que tienen una característica.

   - Las acciones contenidas entre el **Inicio** y el **Fin** indican que son las acciones que conforman el algoritmo en mención.
   - Las acciones comprendidas entre **Mientras** no sea **Fin del día** y su correspondiente **Fin_Mientras** son el conjunto o bloque que se debe repetir (o iterar) precisamente mientras la condición sea **Verdadera** o sea **Mientras** no sea fin del día.

4. Cada ciclo de acciones que se inicie con **Mientras** deberá tener un **Fin_Mientras** asociado y a su vez cada **Fin_Mientras** deberá finalizar uno y solo un ciclo iniciado con Mientras.

CAPÍTULO 3

Metodología

3.1. Modalidad básica de la investigación

El presente trabajo tiene las siguientes modalidades de investigación:

- **Modalidad bibliográfica o documentada.-** Se considera esta modalidad ya que se recurre a diferentes fuentes obtenidas de libros, artículos científicos, tesis desarrolladas en Universidades para profundizar enfoques con respecto al tema de la investigación.
- **Modalidad aplicada.-** Por la utilización de los conocimientos adquiridos a lo largo de la carrera universitaria.

3.2. Recolección de información

La información que se necesitará para la realización del proyecto se obtendrá mediante el uso de libros digitales y fuentes online de autores confiables sobre los temas a investigarse.

3.3. Procesamiento y análisis de datos

El primer paso será recopilar información sobre frameworks, motores de juegos y metodologías ágiles apropiadas para el desarrollo del proyecto, con los resultados obtenidos se procederá a investigar el funcionamiento del framework, motores de juego y la metodología seleccionada.

3.4. Desarrollo del proyecto

- Selección del framework y motor de juego.
- Determinación de la metodología ágil para desarrollo de videojuegos.
- Desarrollo de la propuesta del videojuego.
- Publicación del videojuego.
- Análisis de los resultados obtenidos posterior a la publicación.

CAPÍTULO 4

Desarrollo de la propuesta

4.1. Estudio de los principales Frameworks existentes en el mercado tecnológico para el desarrollo de videojuegos para dispositivos móviles.

4.1.1. Frameworks

Un framework es una estructura de soporte en el cual el desarrollo de software puede ser organizado y encarado con mayor simplicidad. Un buen framework es aquel que resuelve los problemas técnicos más complejos, como por ejemplo, el manejo de transacciones, permitiendo a los desarrolladores concentrarse en el diseño y desarrollo de la aplicación sin llegar a ser intrusivos para el programador. Los frameworks suelen incluir: soporte de programas , bibliotecas, lenguaje de scripting , software para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas. [22].

Esta generación ha sido hogar de una enorme cantidad de videojuegos, los cuales fueron desarrollados sobre motores gráficos muy capaces y altamente superiores a lo que se ha visto en otras épocas. También fue una de las primeras veces en que las grandes empresas empezaron a licenciar sus motores gráficos, como los populares *Unreal Engine* y el *id Tech* , dichos motores se han convertido en frameworks completos, ofreciendo un conjunto de herramientas especificas para el desarrollo de juegos, lo que facilita la incursión de pequeñas empresas y desarrolladores independientes [23].

Unity3D

Es un framework y motor de juego para el desarrollo de videojuegos multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows y OS X,permite crear juegos para Windows, OS X, Linux, Xbox 360, PlayStation 3, PlayStation Vita, Wii, Wii U, iPad, iPhone, Android y Windows Phone [24].

Unity 3D provee un editor visual muy útil y completo donde mediante unos pocos clicks se puede importar modelos 3D, texturas, sonidos, etc. para después ir trabajando con ellos. Además incluye la herramienta de desarrollo MonoDevelop con la que podremos crear scripts en JavaScript, C# y un dialecto de Python llamado Boo con los que extender la funcionalidad del editor, utilizando las API que provee la cual se encuentra documentada junto a tutoriales y recursos en su web oficial. Además si el desarrollador no domina el diseño en 3D o necesita recursos para su juego, en la propia aplicación se puede acceder a una tienda como si de la App Store se tratase, donde se encontrará multitud de recursos gratuitos y de pago. Incluso se puede extender la herramienta mediante plugins que se obtendrán en dicha tienda.

Un desarrollador puede empezar con la licencia gratuita, pero tiene ciertos límites. Incluye obligatoriamente el logotipo de Unity en la carga inicial de tu juego y solo puede usarse si la facturación total de tu empresa no supera los $ 100.000 anuales. El precio de la licencia de Unity Pro es de 1.500 $ por persona más impuestos. Permite el uso de todas las prestaciones de Unity Pro en hasta 2 ordenadores (de la misma persona) [24].

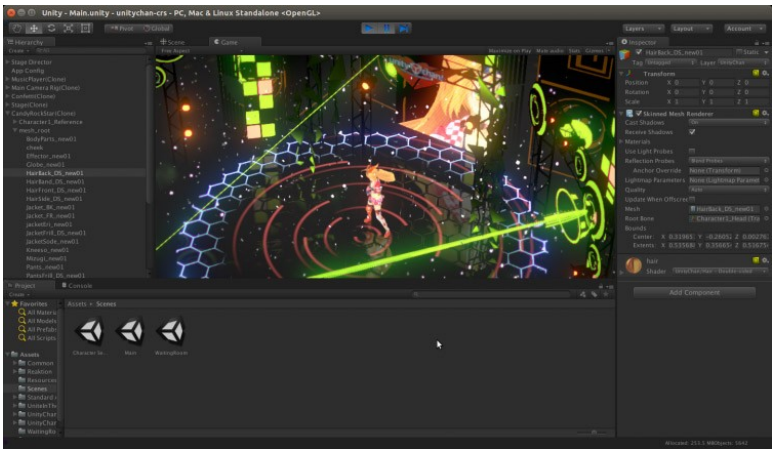


Figura 4.1: Unity3D
Fuente: [24]

Unreal Engine 4

Unreal Engine es un framework y motor de juego de PC y consolas creados por la compañía Epic Games. Implementado inicialmente en el shooter en primera persona llamado Unreal en el año 1998.

La versión actual está programada en C++ y es compatible tanto con OpenGL

como DirectX 11 y 12, siendo compatible con varias plataformas como PC (Microsoft Windows, GNU/Linux), Apple Macintosh (Mac OS X) y la mayoría de consolas (Xbox One y PlayStation 4). Unreal Engine también ofrece varias herramientas de gran ayuda para diseñadores y artistas, facilitando la visualización de entornos o de construcciones.

Unreal Engine 4 ha sido reescrito desde cero en C++ . Esto solo tiene un inconveniente: imposible mudar de UE3 (Unreal Engine 3) a UE4 (Unreal Engine 4) sin exportar a formatos universales y volver a importar los assets e imposible mudar la programación gráfica o tradicional. UE4 directamente fusila el pseudo lenguaje que se inventó Epic Games para UE3 llamado unrealscript (en parte pensado para que el público no pudiera acceder al código fuente del engine) y abre directamente todo el código fuente de Unreal Engine 4 al público y por si fuera poco, lo pone de acceso público en Github para que cualquiera pueda revisarlo y aportar mejoras.

Por otra parte, uno de los mejores elementos de Unreal Engine 3, Kismet, una especie de entorno para programar sin tener que usar código, evoluciona para convertirse en un entorno de programación gráfica con el que se pueden crear hasta clases de cualquier tipo. Ahora sí se puede decir; si quieres puedes construir juegos desde cero sin tener que escribir una sola línea de código. Tan solo uniendo "flechitas". Eso sí, no hay que pensar que la programación de toda la vida queda relegada a un segundo plano; el código C++ aun es necesario para realziar tareas complejas de programación [23].

Figura 4.2: Unreal Editor
Fuente: [23]

Godot Engine

Godot es un framework y motor multiplataforma de juego 2D y 3D con licencia MIT de código abierto desarrollado por la comunidad Godot Engine y se utiliza internamente por varias empresas en América Latina antes de ser evaluado y publicado el código del motor . El framework de desarrollo se ejecuta en Windows, OS X y Linux (32 y 64 bits).

Godot soporta múltiples plataformas. Dentro de un proyecto, los desarrolladores tienen control para desplegar en móviles, web, PC, y consolas. Godot también deja especificar la compresión de textura y encuadres de resolución para cada plataforma. Actualmente las plataformas soportadas son Windows, OS X, Linux, Android, iOS, BlackBerry 10, HTML5, PlayStation 3, PlayStation Vita y Nintendo 3DS.

Los videojuegos en Godot son codificados en el lenguaje de programación C++, o en el lenguaje GDScript. GDScript, es un lenguaje de programación de alto nivel, muy similar a Python que fue creado especialmente para Godot, por lo que añade funcionalidades y optimización. Tiene un editor con auto indentación, resaltado de sintaxis, autocompletado de código y depurador que soporta breakpoints y ejecución paso a paso. [25].

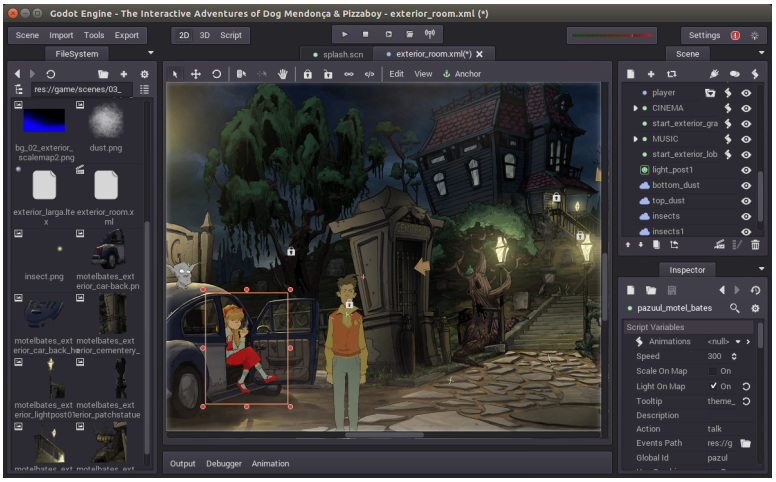


Figura 4.3: Godot Editor
Fuente: [25]

4.2. Comparación de diferentes motores de juegos tanto de tipo comercial como Open Source para el desarrollo de aplicaciones sobre plataformas móviles.

Después del estudio realizado en la **sección 4.1** se llego a la conclusión que los frameworks para el desarrollo de videojuegos incluyen su propio motor de juego, por lo tanto desde ahora en adelante al hablar de un **framework** se estará hablando del framework y del motor como un solo conjunto de herramientas.

Para la evaluación de los frameworks se utilizó la escala de evaluación por puntos donde 0 = no cumple, 1 = cumple parcialmente y 2 cumple totalmente. A continuación se describen los criterios a evaluar.

- **Licenciamiento gratuito.-** Esta característica se refiere a licencias gratuitas otorgadas a desarrolladores freelance que no obtienen ganancias por la publicación de su producto.
- **Varias plataformas de desarrollo.-** Esta característica se refiere a los sistemas operativos en los cuales se puede instalar el entorno de desarrollo.
- **Requisitos mínimos para desarrolladores.-** Característica que indica requisitos de hardware accesibles para los desarrolladores.
- **Lenguaje de programación de alto nivel.-** Indica los lenguajes de programación de alto nivel aceptados por el framework.
- **Orientación 2D y 3D.-** Esta característica se refiere a la capacidad del motor para desarrollo de videojuegos en 2 y 3 dimensiones.
- **Publicación multiplataforma.-** Característica que indica la posibilidad de publicar el juego en más de una plataforma.
- **Herramientas extras para desarrollo móvil.-** Indica si el framework posee herramientas específicas para el desarrollo de videojuegos móviles.
- **Integración con otras herramientas.-** Se refiere a la posibilidad de integrarse con otras herramientas como software de diseño 3D.
- **Suficiente documentación.-** Esta característica indica la cantidad de documentación disponible en la propia página junto con el contenido de ayuda creado por la comunidad.

Unity3D

Característica	**Detalles**	
Licenciamiento gratuito	La licencia gratuita de Unity es para aquellos desarrolladores freelance y empresas de desarrollo de juegos que facturen menos de 100.000 dólares al año	1
Varias plataforma de desarrollo	Windows, Mac	1
Requisitos mínimos para desarrolladores	OS (Operative System): Windows 7 sp1. GPU (Graphics Processing Unit): Soporte mínimo DX9	2
Lenguaje de programación de alto nivel	C#, Javascript, Boo	2
Orientación 2D/3D	2D y 3D	2
Publicación multiplataforma	Windows, Linux, Android, iOS, Windows Phone, WebGL,PS4,PSVita,Xbox 360, XboxOne, WiiU, 3DS, Oculus Rift, Hololens.	2
Herramientas para desarrollo móvil	Posee herramientas específicas para el desarrollo de videojuegos móviles, principalmente para la creación de interfaces de usuario.	2
Integración con otras herramientas	Visual Studio, 3d max, Blender, Maya	2
Suficiente documentación	Su página web provee una gran serie de tutoriales, videos y ejemplos y una gran comunidad de ayuda.	2
Total		16

Tabla 4.1: Evaluación de las características de Unity3D

Elaborado por: Hussein Rahman
Fuente: [24]

Unreal Engine 4

Característica	**Detalles**	
Licenciamiento gratuito	Suscripción: 5 % de regalías cuando las ventas del juego superan los $3000, caso contrario el desarrollador para 0 % de regalías.	1
Varias plataforma de desarrollo	Windows, Mac, Linux	2
Requisitos mínimos para desarrolladores	OS: Windows 7 64 bits sp1. GPU: Soporte mínimo DX11. RAM: 8gb CPU: QuadCore	0
Lenguaje de programación de alto nivel	C++	2
Orientación 2D/3D	2D y 3D	2
Publicación multiplataforma	Windows PC, PlayStation 4, Xbox One, Mac OS X, iOS, Android, Vive, Oculus Rift, PlayStation VR, SteamOS, and HTML5L.	2
Herramientas para desarrollo móvil	No posee herramientas específicas, ya que su enfoque es principalmente para juegos de PC y consolas.	0
Integración con otras herramientas	Visual Studio,3d max, Blender, Maya	2
Suficiente documentación	Existe bastante documentación en su página web, pero al ser una nueva tecnología la comunidad de ayuda aún no es tan grande.	1
Total		12

Tabla 4.2: Evaluación de las características de Unreal Engine 4

Elaborado por: Hussein Rahman
Fuente: [23]

Godot Engine

Característica	**Detalles**	
Licenciamiento gratuito	MIT License	2
Varias plataforma de desarrollo	Windows, Mac, Linux	2
Requisitos mínimos para desarrolladores	OS: Windows 7 sp1 o Linux GPU: Soporte mínimo OpenGL 2.0	2
Lenguaje de programación de alto nivel	GDScript	1
Orientación 2D/3D	2D y 3D	2
Publicación multiplataforma	Windows, OS X, Linux, FreeBSD, Android, iOS, BlackBerry 10, HTML5, PlayStation 3, PlayStation Vita y Nintendo 3DS.	2
Herramientas para desarrollo móvil	No posee herramientas especificas para desarrollo móvil, ya que su enfoque es para juegos de PC.	0
Integración con otras herramientas	Blender	1
Suficiente documentación	Poca documentación existente sobre el manejo del lenguaje y entorno de desarrollo.	1
Total		13

Tabla 4.3: Evaluación de las características de Godot Engine

Elaborado por: Hussein Rahman
Fuente: [25]

4.2.1. Resultado de la evaluación

Los resultados de la evaluación de los frameworks se muestran a continuación en la **Figura 4.4**.

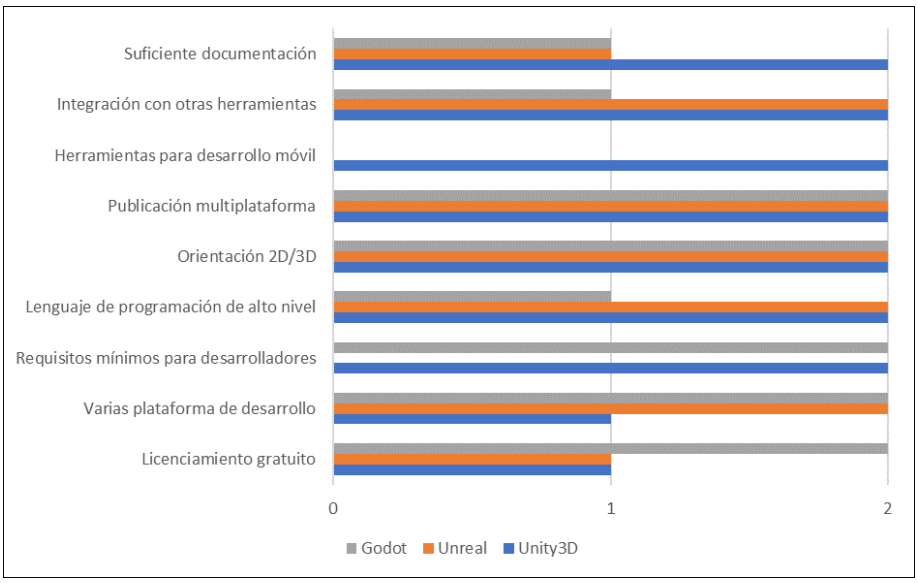


Figura 4.4: Resultados evaluación frameworks
Elaborado por: Hussein Rahman

A continuación de detalla un breve análisis de los resultados del gráfico anterior.

- **Licenciamiento gratuito.-** Unity3D y Unreal obtienen 1 punto ya que su licencia gratuita tiene condiciones de regalías, por lo que Godot Engine gana en este apartado con 2 puntos al poseer una licencia abierta.
- **Varias plataformas de desarrollo.-** En este criterio gana Unreal y Gotot con 2 puntos cada uno ya que poseen soporte para Windows, Linux y Mac, Unity se queda con un punto ya que no posee soporte para Linux.
- **Requisitos mínimos para desarrolladores.-** Unity3D y Godot empatan en este criterio ya que sus requisitos de hardware son bastante accesibles en comparación de los requisitos necesarios para ejecutar el editor de Unreal, por lo que este obtiene 0 puntos.
- **Lenguaje de programación de alto nivel.-** Los tres productos poseen lenguajes de alto nivel, por lo que obtienen 2 puntos.
- **Orientación 2D y 3D.-** Los tres productos están orientados a los gráficos 2D y 3D, por lo que obtienen 2 puntos.
- **Publicación multiplataforma.-** Los tres productos tienen la posibilidad de publicar en múltiples plataformas, por lo que obtienen 2 puntos.
- **Herramientas extras para desarrollo móvil.-** Solo Unity3D posee herramientas especificas para el desarrollo móvil por lo que obtiene 2 puntos, mientras Unreal y Godot 0 puntos.

- **Integración con otras herramientas.-** Unity3D y Unreal obtienen 2 puntos por su integración con varias herramientas de diseño, mientras que Godot obtiene 1 debido a que solo se integra con una herramienta de diseño.

- **Suficiente documentación.-** Unity3D y Unreal obtienen 2 puntos ya que poseen suficiente documentación tanto oficiales como por su comunidad, mientras que Godot obtiene 1 por su insuficiente documentación.

En base a los resultados mostrados, Unity3D obtiene 16 puntos, Unreal 12 puntos y Godot 13 por lo tanto para este proyecto se ha seleccionado **Unity3D** como framework y motor de juego apropiado para el desarrollo del videojuego.

4.2.2. Descripción de los componentes de Unity3D

Editor Unity

La interfaz del editor de Unity después de su instalación y ejecución se puede observar en la **Figura 4.5**, cada una de las zonas del editor permiten realizar una tarea específica, la zona central será el lugar en el cual se muestren todos los elementos del juego en ejecución y los botones para la ejecución del juego se encuentran en la parte superior.

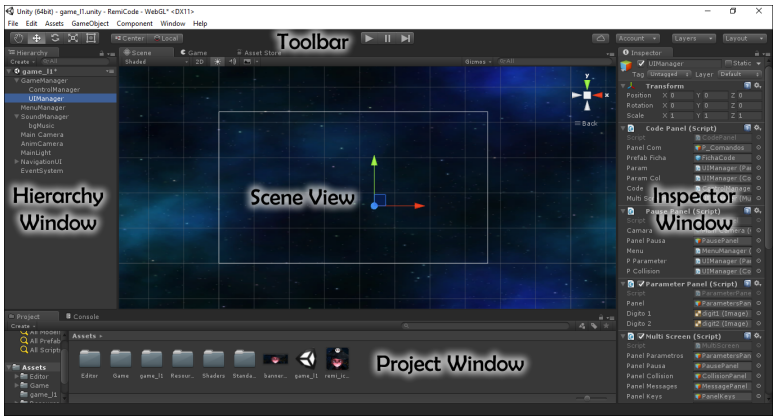


Figura 4.5: Partes de la interfaz del editor de Unity3D
Elaborado por: Hussein Rahman
Fuente: [24]

- **Project Window.-** Muestra los assets de librería que están disponibles para ser usados.

- **Scene View.-** Permite una navegación visual y editar la escena. La scene view puede mostrar una perspectiva 2D o 3D dependiendo en el tipo de proyecto en el que esté trabajando.

- **Hierarchy Window.-** Es una representación de texto jerárquico de cada objeto en la escena; cada elemento en la escena tiene una entrada en la jerarquía, por lo que las dos ventanas están inherentemente vinculadas. La jerarquía revela la estructura de cómo los objetos están agrupados el uno al otro.

- **Inspector Window.-** Permite visualizar y editar todas las propiedades del objeto actualmente seleccionado, cada objeto tiene diferentes propiedades por lo tanto el contenido de la ventana va a variar.

- **Toolbar.-** Proporciona un acceso a las características más esenciales para trabajar; en la izquierda contiene las herramientas básicas para manipular la scene view y los objetos dentro de esta; en el centro están los controles de reproducción, pausa, y pasos; los botones a la derecha dan acceso a servicios de Unity Cloud y una cuenta de Unity, seguido por un menú de visibilidad de capas, y finalmente el menú del layout del editor (que proporciona algunos diseños alternativos para la ventana del editor, y le permite a usted guardar sus propios layouts personalizados).

MonoDevelop

MonoDevelop es un entorno de desarrollo integrado o entorno de desarrollo interactivo, en inglés Integrated Development Environment (IDE) proporcionado con Unity. Un IDE combina la operación familiar de un editor de texto con características adicionales para depurar y gestionar otras tareas del proyecto. Este IDE esta diseñado primordialmente para C#, Javascript y Boo [24].

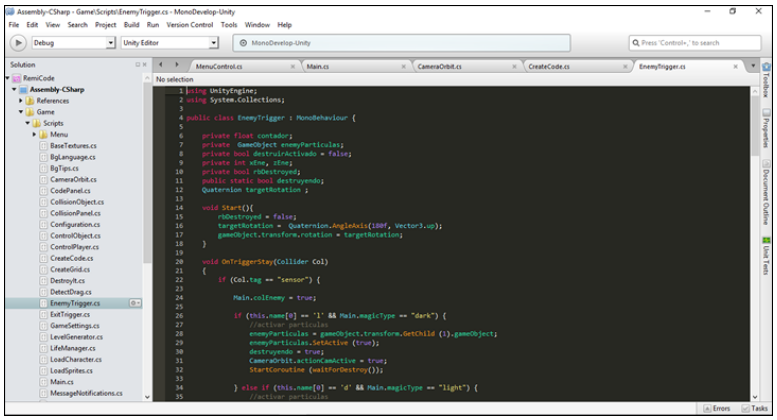


Figura 4.6: MonoDevelop
Fuente: [24]

Arquitectura del motor

La estructura del motor Unity se divide en las siguientes capas como se muestra en la **Figura 4.7**:

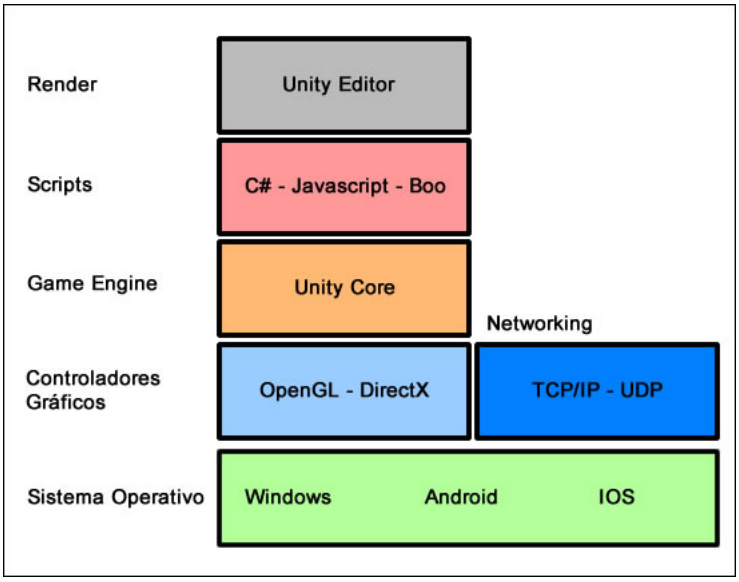


Figura 4.7: Arquitectura de Unity3D
Elaborado por: Hussein Rahman
Fuente: [24]

- **Sistema Operativo.-** Es la capa en la cual se ejecutará el videojuego, ya sean plataformas móviles como Android e IOS o plataformas de escritorio como Windows [24].
- **Controladores Gráficos.-** Según la plataforma en la que se ejecute el videojuego, requerirá un controlador gráfico como OpenGL para Android o DirectX para Windows [24].
- **Networking.-** Esta capa es usada en juegos con características de multijugador utilizando el protocolo TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet) y UDP (Protocolo de datagrama de usuario) [24].
- **Game Engine.-** Es el núcleo de toda la arquitectura, el cual contiene los diferentes componentes que se utilizarán para la creación del juego, componentes como físicas, colisiones, animaciones, materiales, texturas, etc [24].
- **Scripts.-** La programación y la lógica del juego se realiza aquí utilizando lenguajes como C#, javascript y Boo equivalente a Python [24].

- **Render.-** La ultima capa en la cual se muestra el mundo/nivel de forma jugable desde el Editor de Unity [24].

4.3. Selección de una metodología ágil aplicada al campo del desarrollo de videojuegos móviles que facilite el desarrollo e implementación del proyecto planteado.

Desarrollar un proyecto de software requiere de una metodología apropiada para una mejor organización y planificación de las diferentes tareas que se llevarán a cabo por el equipo de desarrollo, lo que permitirá tener un control de lo que se está haciendo y lo que aún falta hacer, evitando el fracaso del proyecto al tener estimaciones de tiempo totalmente incorrectas o recursos mal utilizados.

Un videojuego es también un proyecto de software en el cual interviene un equipo multidisciplinario con conocimientos específicos en áreas como: programación, diseño gráfico y multimedia, efectos de sonidos, etc., por lo tanto el uso de una buena metodología determinará el éxito o fracaso del proyecto.

Como se habló anteriormente en la ***subsección 2.2.6***, las pequeñas empresas desarrolladoras de videojuegos con equipos realmente pequeños no cuentan con una metodología formalizada y recurren principalmente al uso de metodologías ágiles como SCRUM y XP adaptándolas a sus necesidades.

Estas metodologías se han transformado en metodologías ágiles específicamente para el desarrollo de videojuegos, y es aquí donde nacen nuevas metodologías como SUM [20] basándose en la estructura de SCRUM [19]; aparece también XGD (Extreme Game Development) [26] la cual es una adaptación de eXtreme Programming (XP) para videojuegos; adicionalmente a estas dos metodologías aparece una tercera y muy interesante que es la combinación entre las características de SCRUM y XGD, llamada metodología DAV (Desarrollo Ágil de Videojuegos) realizada por estudiantes de la ESPOCH [27].

4.3.1. Metodología SUM

La metodología SUM para videojuegos tiene como objetivo desarrollar videojuegos de calidad en tiempo y costo, así como la mejora continua del proceso para incrementar su eficacia y eficiencia. Pretende obtener resultados predecibles, ad-

ministrar eficientemente los recursos y riesgos del proyecto, y lograr una alta productividad del equipo de desarrollo. SUM fue concebida para que se adapte a equipos multidisciplinarios pequeños (de tres a siete integrantes que trabajan en un mismo lugar físico o estén distribuidos), y para proyectos cortos (menores a un año de duración) con alto grado de participación del cliente [20].

Roles

La metodología define cuatro roles: equipo de desarrollo, productor interno, cliente y verificador beta. El productor interno y el cliente se relacionan en forma directa con los roles de SCRUM Master y Product Owner de SCRUM respectivamente. El equipo de desarrollo tiene las características del Scrum team, pero a diferencia de SCRUM se definen subroles dentro del equipo, estos son equivalentes a los usados por la industria local tales como: programador, artista gráfico, artista sonoro y diseñador de juego. Es necesario esta definición ya que se requiere una alta especialización para satisfacer las distintas disciplinas que involucra del desarrollo de videojuegos, aspecto no contemplado en Scrum [20].

El rol de verificador beta no está presente en Scrum pero si se detecta su existencia en la industria del videojuego. Su responsabilidad es realizar la verificación funcional del videojuego y comunicar su resultado [20].

Los objetivos de cada uno de los roles son los siguientes:

1. **Cliente.-** Define y valida el concepto del juego, aprueba los planes de proyecto y determina los hitos, prioriza las características y tareas que dan más valor al videojuego en cada momento, evalúa el cumplimiento de las tareas y el producto obtenido al finalizar cada iteración, y participa de la evaluación del proyecto. Prioriza los errores a corregir buscando la mejor calidad posible de acuerdo a sus intereses y valida las versiones del producto [20].

2. **Productor Interno.-** Es responsable de la planificación y la ejecución del proyecto, participa en la definición de los objetivos y hace seguimiento del proyecto ayudando a resolver los impedimentos que ocurren en el proyecto. Lleva a cabo las acciones para la mejora continua coordinando la comunicación con el cliente y mantiene al equipo enfocado en los objetivos del proyecto [20].

3. **Equipo de Desarrollo.-** Representa a todos los miembros encargados de construir el videojuego [20].
    - **Diseñador de juego.-** Se encarga de diseñar el gameplay, la historia, ambientación, personajes, niveles y todos los elementos que hacen a la experiencia del jugador. Todos estos factores determinarán que tan divertido será el juego. Para asegurar la diversión debe mantener balanceada la dificultad del juego y el aprendizaje del jugador [20].
    - **Programador.-** El programador tiene como principal responsabilidad implementar el software que compone al juego. Además deberá realizar el diseño de software necesario para poder realizar el desarrollo y posteriormente verificarlo. Por lo tanto el desarrollador de videojuegos debe tener conocimientos de diseño de software, implementación y verificación [20].
    - **Artista sonoro.-** Los artistas sonoros deben tener buen oído para mezclar los sonidos, los efectos de sonido deben ser diseñados de forma que se correspondan con la interacción del jugador. El sonido da vida a la escena y complementa la experiencia del jugador [20].
    - **Artista gráfico.-** El arte y la animación son gran parte del trabajo requerido para el desarrollo del videojuego. Las habilidades necesarias para un artista varían según los requerimientos del juego en particular. De cualquier forma requieren conocimientos sobre las últimas herramientas gráficas, creatividad y talento [20].
4. **Verificador Beta.-** Son los designados para realizar la verificación funcional del videojuego. Participan, fundamentalmente, en la etapa beta del proyecto, ya que es cuando se obtiene la primer versión completa del juego. Dependiendo del proyecto es posible que participen verificadores beta un poco antes de dicha etapa [20].

Ciclo de vida

El ciclo de vida se divide en fases iterativas e incrementales que se ejecutan en forma secuencial con excepción de la fase de gestión de riesgos que se realiza durante todo el proyecto. Las cinco fases secuenciales son: concepto, planificación, elaboración, beta y cierre, como se aprecia en la Figura. Las fases de concepto, planificación y cierre se realizan en una única iteración, mientras que elaboración y beta constan de múltiples iteraciones [20].

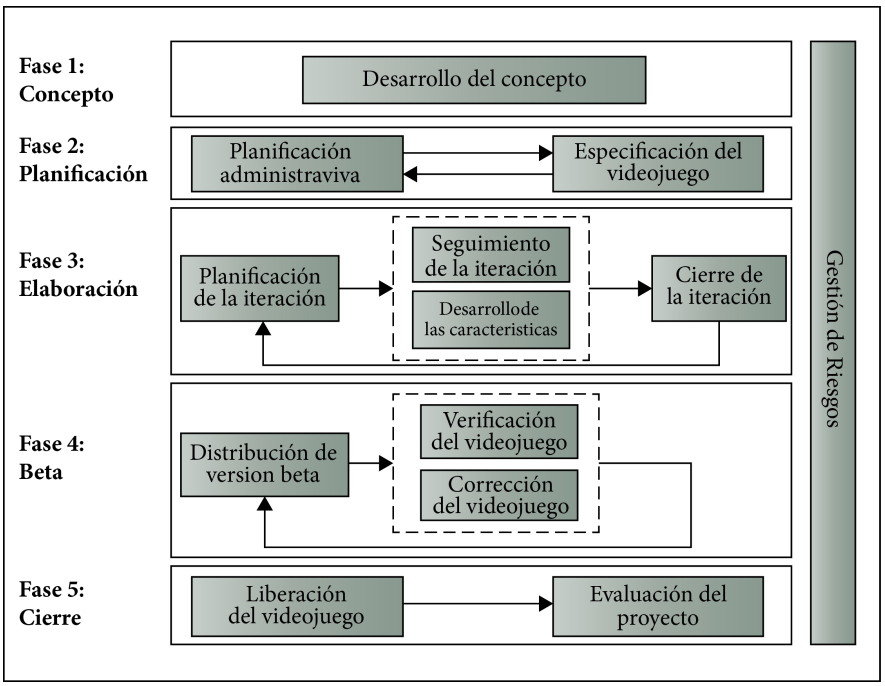


Figura 4.8: Ciclo de vida de la metodología SUM
Elaborado por: Hussein Rahman
Fuente: [28]

1. **Fase del concepto**

   Tiene como objetivo principal definir el concepto del videojuego lo que implica definir aspectos de negocio (publico objetivo, modelo de negocio), de elementos de juego (principales características, gameplay, personajes e historia entre otros) y técnicos (lenguajes y herramientas para el desarrollo). El concepto del videojuego se construye a partir de ideas y propuestas de cada rol involucrado sobre los aspectos a definir. Las propuestas se refinan a través de reuniones y se analiza su factibilidad con pruebas de concepto. Esta fase finaliza cuando se tiene el concepto validado entre todas las partes involucradas [20].

2. **Fase de la planificación**

   La fase tiene como objetivo principal planificar las restantes fases del proyecto. Para ello es necesario definir el cronograma del proyecto junto con sus principales hitos, conformar el equipo para la fase de elaboración de acuerdo a las necesidades técnicas del proyecto, determinar y tercerizar las tareas que el equipo no pueda cumplir, definir el presupuesto y especificar las características. Esto último consiste en describir, estimar y priorizar cada una de las características funcionales y no funcionales que definen el videojuego [20].

3. **Fase de elaboración**

   El objetivo de esta fase es implementar el videojuego, para ello se trabaja en forma iterativa e incremental para lograr una versión ejecutable del videojuego al finalizar cada iteración. Estas se dividen en tres etapas, en la primera se planifican los objetivos a cumplir, las métricas a utilizar en el seguimiento, las características a implementar y las tareas necesarias para ello. En la segunda se desarrollan las características planificadas a través de la ejecución de las tareas que la componen [20].

4. **Fase beta**

   La fase tiene como objetivos evaluar y ajustar distintos aspectos del videojuego como por ejemplo gameplay, diversión, curva de aprendizaje y curva de dificultad, además de eliminar la mayor cantidad de errores detectados. Se trabaja en forma iterativa liberando distintas versiones del videojuego para verificar. Para ello primero se distribuye la versión beta del videojuego a verificar y se determinan los aspectos a evaluar. Mientras esta se verifica, se envían reportes con los errores o evaluaciones realizadas [20].

5. **Fase de cierre**

   Esta fase tiene como objetivo entregar la versión final del videojuego al cliente según las formas establecidas y evaluar el desarrollo del proyecto. Para la evaluación se estudian los problemas ocurridos, los éxitos conseguidos, las soluciones halladas, el cumplimiento de objetivos y la certeza de las estimaciones. Con las conclusiones extraídas se registran las lecciones aprendidas y se plantean mejoras a la metodología. [20].

6. **Fase de gestión de riesgos**

   Esta fase se realiza durante todo el proyecto con el objetivo de minimizar la ocurrencia y el impacto de problemas. Esto se debe a que distintos riesgos pueden ocurrir en cualquiera de las fases, por lo cual siempre debe existir un seguimiento de los mismos. Para cada uno de los riesgos que se identifican se debe establecer la probabilidad y el impacto de ocurrencia, mecanismos de monitoreo, estrategia de mitigación y plan de contingencia. [20].

4.3.2. Metodología DAV

DAV (Desarrollo Ágil de Videojuegos), es una metodología que nace con la unión de las características de SCRUM y XGD, con la finalidad de proporcionar una estructura que permita la creación de videojuegos de forma técnica y sencilla. Esta metodología se compone de los siguientes elementos: valores, prácticas, roles, reuniones, fases, artefactos y herramientas [27].

Prácticas

Esta metodología utiliza prácticas tanto de SCRUM como de XGD para integrar en la nueva metodología DAV, a continuación se describe cada una de ellas [27]:

- **Diseño incremental:** Las tareas de un videojuego deben ser de la forma más sencilla posible, que funcionen correctamente.
- **Historias de usuario:** Se trata de una breve descripción de las funcionalidades del videojuego, por lo general descrito por el Dueño del Producto.
- **Ciclos semanales:** El proyecto está organizado y planificado para ser ejecutado en ciclos de corta duración.

- **Entregas pequeñas:** Producir rápidamente versiones del videojuego estables, aunque no cuenten con toda la funcionalidad del videojuego. Esta versión ya constituirá un resultado de valor para el negocio.

- **Pruebas:** El videojuego será probado en varios momentos por diferentes personas, dando como resultado una información valiosa que podrá ser tomada como base para decisiones futuras.

Roles

Los roles de DAV se basan únicamente en SCRUM, añadiendo al equipo DAV el rol de tester, siendo estos necesarios y suficientes para ser aplicados en equipos pequeños, clasificándose en dos tipos de roles [27]:

- **Dueño del Producto.-** El Dueño del Producto es la única persona autorizada para decidir sobre cuáles funcionalidades y características funcionales tendrá el videojuego. Es quien representa al cliente y todas aquellas partes interesadas en el videojuego, además es quien maneja y prioriza las funcionalidades a desarrollar y las coloca en la Reserva de Producto [27].

- **DAV Máster.-** El DAV Máster no es un líder típico, sino un auténtico servidor neutral, que será el encargado de enseñar la metodología DAV a cada integrante implicado en el proyecto, preocupándose de poner la metodología en práctica de modo que se encuentre dentro de la cultura de la organización y así entregue las ventajas previstas, asegurándose de que cada uno siga las reglas y prácticas de DAV [27]:.

- **Equipo DAV.-** Es un equipo multidisciplinario y auto-organizado, su función principal es construir el videojuego que el Dueño del Producto especifica. Los miembros del equipo pueden ser de 3 a 9 personas, entre los cuales tenemos [27]:

  - **Diseñador.-** Realiza el diseño conceptual, define las reglas y la mecánica, escribe el guión y diseña los niveles.

  - **Programador.-** Implementa la funcionalidad pedida en el diseño, programa la lógica del juego, gráficos (efectos), inteligencia artificial y audio y conoce de herramientas/motores (infraestructura).

  - **Artista.-** Construye los objetos, personajes, niveles y anima los personajes.

- **Sonidista.-** Crea los sonidos, estilos, efectos y ambientes sonoros para el videojuego.
- **Tester.-** Verifican que el contenido, funcionalidad del videojuego son correctos, completos y evalúan la jugabilidad.

- **Interesados (Clientes)** Se refiere a la gente que hace posible el proyecto y para quienes el videojuego producirá el beneficio acordado que justifica su producción. Sólo participan directamente durante las revisiones de Iteración.

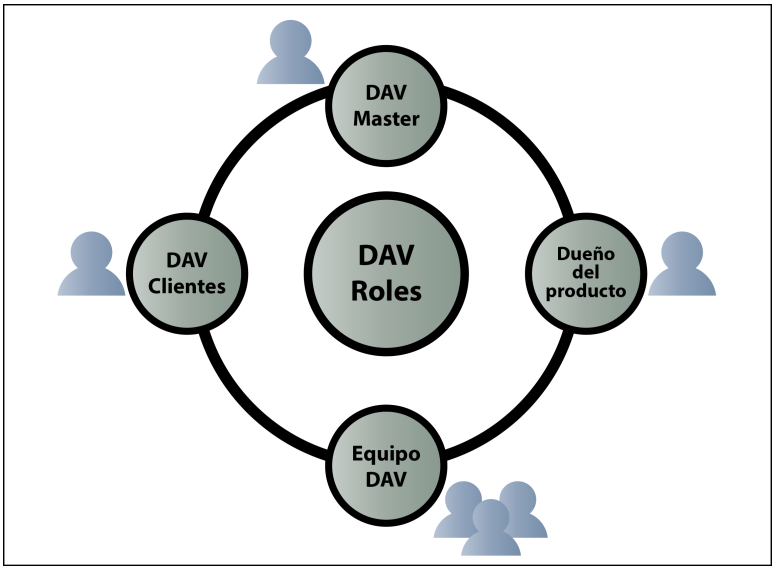


Figura 4.9: Roles de DAV
Elaborado por: Hussein Rahman
Fuente: [27]

Fases

La división de trabajo en el desarrollo de videojuegos no se distribuye por igual en sus iteraciones; para apoyar esta realidad DAV se basa únicamente en las fases de SCRUM: pre-juego, juego y post-juego [27].

1. **Fase pre-juego**

   En esta fase se realiza la concepción de la idea del juego del dueño del producto con ayuda de los miembros del equipo, es decir, los aspectos fundamentales que conformarán el videojuego [27]:

   - **Género:** Se debe especificar el género o géneros al que pertenece el videojuego para establecer las características básicas que tendrá en su posterior diseño.

- **Historia:** Se debe realizar un bosquejo de la trama o historia del videojuego a desarrollar, indicando qué se quiere contar y cómo se quiere contar.
- **Bocetos:** Se crean bocetos o diseños preliminares de los personajes y en dónde trascurrirá la acción del videojuego, ya sean decorados, ambientaciones, ropaje, música, movimientos, etc.
- **Aspecto:** A partir de los bocetos se define el aspecto gráfico y artístico del videojuego, colores, temas dominantes, musicalidad, técnicas de diseño 3D o 2D, posiciones de cámaras, etc.
- **Interfaz de usuario:** Se define la manera de cómo interactuará el jugador con el videojuego y con qué mecanismos contará para ello.
- **Objetivos:** Cuáles son las metas del videojuego, de acuerdo a la historia de éste.
- **Reglas:** Qué cosas podemos hacer y cómo vamos a dejar que se hagan.
- **Características:** Especificación de las principales características de cada personaje del videojuego y de los elementos que intervienen en éste.
- **Jugabilidad:** Es aquí donde se definirá cómo se va a jugar, de qué manera se va a jugar, qué cosas podemos hacer en el juego y cómo va reaccionar el entorno del juego a las acciones del jugador a través del personaje. A su vez se debe establecer cómo será la curva de aprendizaje del jugador. Todo esto sin entrar en detalles gráficos, sonoros o de historia.
- **Diseño de niveles:** Describir qué niveles, según la historia o dificultad, se tendrá, cómo serán éstos, cuántos serán, y qué dificultad y retos se plantearán en cada uno de ellos.
- **Requerimientos técnicos:** Establecer los requerimientos técnicos del equipo que necesitará el videojuego para poder ejecutarse.
- **Marketing:** Parte fundamental, debido a que muchos videojuegos de inversiones millonarias han ido al traste por una mala campaña de publicidad, para evitar esto hay que establecer las líneas de publicidad para el videojuego.

La fase pre-juego reduce la necesidad de encontrar ese elemento elusivo de diversión durante la ***etapa de juego*** y permite al equipo centrarse en la ejecución del videojuego, en lugar de experimentar con él.

2. **Fase juego**

   En esta fase se tiene bien definido el alcance del proyecto, por lo tanto ya existe una idea clara de lo que realmente es el videojuego y lo que se debe hacer [27].

   - **Iteraciones.-** Esta fase que suele llamarse iteración, consta de ciclos que pueden durar de 1 a 4 semanas (por lo general 1-2 semanas), donde se desarrollan las funcionalidades del videojuego, su duración debe ser constante en cada ciclo. Sin embargo, este requisito no es estricto debido a que a veces el tiempo de cada iteración es diferente; una Iteración consta de las siguientes actividades:

     - **Elaborar:** Se desarrollan las funcionalidades o requisitos del videojuego que se encuentran en la reserva de iteración.
     - **Integrar:** Las funcionalidades desarrolladas son continuamente integradas en la nueva versión del videojuego.
     - **Revisar:** Se examinan las funcionalidades desarrolladas en la iteración para corregir posibles errores y probar que su funcionalidad sea correcta.
     - **Ajustar:** Se adecua las funcionalidades del videojuego desarrolladas encajándolas correctamente, además se realiza la refactorización del código de las mismas.

     Estas actividades no tienen una secuencia. A veces un elemento de la Reserva del Producto se tiene que desarrollar, integrar, y revisar cuando otros sólo debe ser revisado o ajustado. En la **Figura 4.10** se puede observar como es el proceso de las iteraciones.

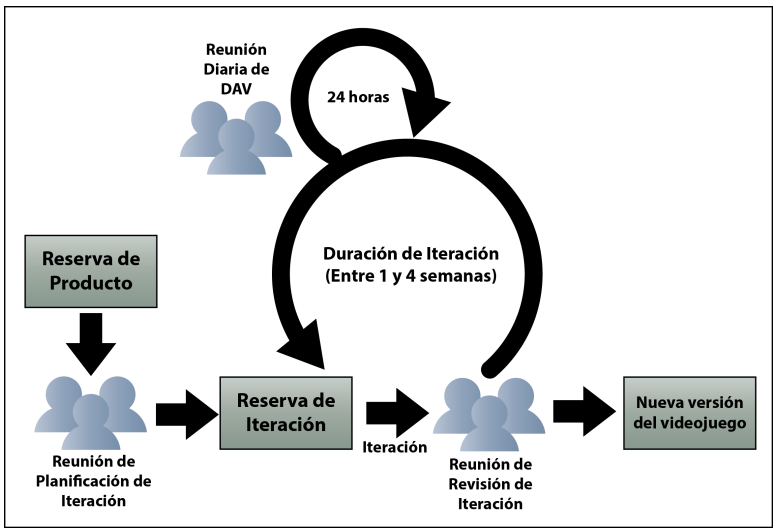


Figura 4.10: Proceso de DAV
Elaborado por: Hussein Rahman
Fuente: [27]

Al comienzo de cada Iteración se realiza su planificación, donde se describen, priorizan y estiman las funcionalidades que se van a desarrollar, también se efectúan reuniones diarias para mejorar la comunicación del Equipo DAV y al concluir cada Iteración se evalúa su resultado [27].

3. **Fase post-juego**

   Esta fase comienza cuando el equipo DAV ha desarrollado con éxito todas las Historias de Usuario y el Dueño del Producto ya no tiene más funcionalidades para implementar. Aquí se realiza el cierre del proyecto, donde se prepara la liberación del videojuego, se verifican las versiones a entregar, se genera la documentación final y se realiza el pre-lanzamiento y el lanzamiento [27].

Artefactos

En el caso de los artefactos, se han constituido en base a una selección mayoritaria de los artefactos de Scrum complementados con los de XGD, para mantener organizado el proyecto. Estos artefactos, ayudan a planificar y revisar cada uno de las Iteraciones, aportando medios ineludibles para efectuar cada una de las reuniones, los mismos se detallan a continuación [27]:

Historia de usuario

Una Historia de Usuario es la definición de un requisito (normalmente una funcionalidad) que el Dueño del Producto quiere, con total claridad y sin ninguna ambigüedad, la cual el Equipo DAV deberá desarrollar como parte de una Iteración. Son el resultado de escuchar al Dueño del Producto y ayudarle a resumir el requisito en una sola frase. Algo muy importante es que están escritas con el vocabulario del cliente, no con vocabulario técnico [27]. En la ***Tabla 4.5*** se muestra la estructura de una historia de usuario:

ID	**Como**	**Quiero.../Quiero que...**	**De modo que.../Para**
HU01	Jugador	Visualizar las instrucciones del juego	Saber cómo jugar
HU02	Jugador	La pala se encuentre en la parte inferior y pueda moverse en forma horizontal	La bola no se caiga y revotaría para romper los bloques
HU03	Jugador	Al romper los bloques con al bola se escuche un sonido de "golpe o ruptura"	Sea divertido y le dé realismo

Tabla 4.5: Historia de usuario de DAV

Elaborado por: Hussein Rahman
Fuente: [27]

Prueba de aceptación

Las Pruebas de Aceptación validan el grado de satisfacción del Dueño del Producto, es decir confirman que una Historia de Usuario ha sido implementada correctamente. Este tipo de pruebas son comúnmente realizadas por el Dueño del Producto y supervisadas por el Tester, informando de todas las deficiencias o errores que se encuentre antes de dar por aprobado el requisito definitivamente. En la siguiente tabla se muestra el formato de la Prueba de Aceptación [27].

Prueba de Aceptación	
Identificador: PA001	**Historia de Usuario (Nro. y Nombre):** HU001-Visualizar las Instrucciones del videojuego
Nombre: Visualizar las instrucciones de usuario.	
Descripción: Visualizar en una ventana las instrucciones del videojuego, además mediante un botón de color amarillo debe permitir regresar al menú principal.	
Condiciones de ejecución: Debe ejecutarse en el dispositivo móvil, además debe estar terminado el menú inicio.	
Entrada/Pasos de ejecución: - Escoger la opción "Instrucciones" en el Menú Principal. - En la pantalla que aparece las instrucciones del videojuego, para regresar al Menú Principal dar clic en el botón Regresar.	
Resultado esperado: Mostrar las instrucciones del videojuego en una ventana, además tener un botón que permita salir de la ventana y regresar al menú principal.	
Evaluación de la prueba: Correcto	

Tabla 4.6: Prueba de aceptación de DAV

Elaborado por: Hussein Rahman

Para cada ítem, es necesario especificar:

- **Identificador:** Es un identificador único de la Prueba de Aceptación para futuras referencias.
- **Historia de usuario (Nro. Y Nombre):** El identificador y nombre de la Historia de Usuario a la cual pertenece la Prueba de Aceptación.
- **Nombre:** La Prueba de Aceptación debe tener un nombre entendible para cualquier persona, facilitando la comprensión tanto del propósito de la prueba como del campo de aplicación.
- **Descripción:** Contiene una breve descripción del propósito de la prueba y la funcionalidad que examina.
- **Condiciones de ejecución:** Contiene información acerca de hardware o software necesario para poder ejecutar la prueba.
- **Entrada / pasos de ejecución:** Pasos a realizar para completar la prueba.
- **Resultado esperado:** Contiene una descripción de lo que el analista debería ver tras haber completado todos los pasos de la prueba.
- **Evaluación de la prueba:** Indica el resultado cualitativo de la ejecución de la prueba, a menudo con un Correcto/Fallido.

Reserva del producto

La Reserva del Producto es un listado de alto nivel, dinámico y públicamente visible para todos los involucrados en el proyecto. En ella, el Dueño de Producto, mantiene una lista actualizada de requerimientos funcionales para el videojuego. Esta lista, representa "qué es lo que se pretende hacer" pero sin mencionar "cómo hacerlo", debido a que esta última, será tarea del Equipo DAV. La Reserva del Producto es creada y modificada únicamente por el Dueño de Producto. Durante la reunión de planificación, el Equipo DAV obtendrá los ítems del videojuego, que deberá desarrollar durante la Iteración. En la siguiente tabla se muestra el formato de la Reserva del Producto [27].

PRI	Nro. HU	Ítem	Estimación	Estado	PA
1	HU01	Como Jugador quiero visualizar las instrucciones del juego, para saber cómo jugar.	3	Pendiente	PA01
2	HU02	Como jugador quiero visualizar el puntaje de los bloques que voy rompiendo, de modo que pueda conocer cuántos puntos voy acumulando.	5	Pendiente	PA01

Tabla 4.7: Reserva producto de DAV

Elaborado por: Hussein Rahman
Fuente: [27]

Para cada ítem, es necesario especificar:

- **El grado de prioridad (PRI).-** Los ítems de la Reserva del Producto, deben guardar un orden de prioridad, en base a la pérdida o costo que demande posponer la implementación.
- **Esfuerzo que demanda.-** DAV, propone la estimación de esfuerzo y complejidad que demanda el desarrollo de las funcionalidades, solo para aquellas cuyo orden sea prioritario. Estas estimaciones, no se efectúan sobre ítems poco prioritarios ni tampoco sobre aquellos donde exista un alto grado de incertidumbre. De esta manera, se evita la pérdida de tiempo en estimaciones irrelevantes, postergándolas para el momento en el cual realmente sea necesario comenzar a desarrollarlas.

- **Prueba de aceptación (PA).-** Es necesario especificar para cada ítem de la Reserva del Producto, la o las Pruebas de Aceptación que debe superar, para considerar cumplido el requisito.

Tarjeta de tarea

Las Tarjetas de Tarea se usan para describir las tareas que se realizarán en el proyecto y representar las responsabilidades asignadas a cada miembro del Equipo DAV. Estas tareas pueden ser de desarrollo, corrección o mejora, contienen número y nombre de la tarea, Historia de Usuario a desarrollar, fecha de inicio y fin de la tarea, se nombra al miembro del equipo responsable y presenta una descripción de la tarea como se muestra en la siguiente tabla [27].

Tarjeta de tarea	
Número de Tarea:	**Historia de Usuario (Nro. y Nombre):**
Nombre Tarea:	
Tipo de Tarea:	**Puntos Estimados:**
Fecha de inicio:	**Fecha fin:**
Miembro responsable:	
Descripción:	

Tabla 4.8: Tarjeta de tarea de DAV

Elaborado por: Hussein Rahman
Fuente: [27]

4.3.3. Evaluación de las metodologías ágiles SUM y DAV

Para la evaluación de las metodología ágiles para desarrollo de videojuegos se utilizó la escala de evaluación por puntos donde 0 = no cumple, 1 = cumple parcialmente y 2 cumple totalmente. A continuación se describen los criterios a evaluar.

- **Flexibilidad.-** Esta característica se refiere a la flexibilidad para cambios durante el desarrollo del proyecto.
- **Equipos multidiciplinarios.-** Esta característica indica la posibilidad de trabajar en equipos con especialistas de diferentes áreas.
- **Documentación.-** La metodología posee una documentación clara y con ejemplos prácticos.

- **Corto tiempo.-** La metodología permite tener versiones entregables del proyecto en corto tiempo.
- **Comunicación.-** La metodología permite tener una comunicación transparente con todos los miembros del equipo, retroalimentando constantemente el estado del proyecto

Metodología SUM

Características	**Metodología SUM**	
Flexibilidad	Al ser una metodología basada unicamente en SCRUM las tareas terminadas y revisadas no se vuelven a tocar.	0
Equipos multidisciplinarios	Esta metodología posee un rol de equipo multidisciplinario adaptándose al desarrollo de videojuegos.	2
Documentación	Aunque posee una amplia documentación en su página web, no existen ejemplos prácticos para un mejor entendimiento.	1
Corto tiempo	Permite tener entregables del proyecto en poco tiempo para la revisión y aprobación del cliente	2
Comunicación	Posee una comunicación transparente gracias a las reuniones diarias y la retroalimentación del estado del proyecto.	2
Total		7

Tabla 4.9: Evaluación de la metodología SUM

Elaborado por: Hussein Rahman
Fuente: [27]

Metodología DAV

Características	Metodología DAV	
Flexibilidad	Al basarse en XGD toma las características de esta metodología, permitiendo que las tareas siempre estén susceptibles a cambios aunque ya estén finalizadas y aprobadas.	2
Equipos multidisciplinarios	Esta metodología posee un rol de equipo multidisciplinario adaptándose al desarrollo de videojuegos.	2
Documentación	Posee una amplia documentación y ejemplos prácticos para un mejor entendimiento del proceso de la metodología.	2
Corto tiempo	Permite tener una versión del proyecto en corto tiempo para la revisión del cliente	2
Comunicación	Existe comunicación transparente dentro de todos los miembros del equipo gracias a las reuniones diarias y la retroalimentación del estado del proyecto.	2
Total		10

Tabla 4.10: Evaluación de la metodología SUM

Elaborado por: Hussein Rahman
Fuente: [27]

Resultado de la evaluación de metodologías

A continuación de detalla un breve análisis de los resultados obtenidos durante la evaluación de la metodología SUM y DAV:

- **Flexibilidad.-** DAV obtiene 2 puntos debido a que permite realizar cambios aunque la tarea este finalizada y aprobada, algo muy útil por los cambios constantes en el desarrollo de videojuegos, por otra parte SUM obtiene 0 puntos ya que no tiene este grado e flexibilidad en tareas finalizadas.

- **Equipos multidiciplinarios.-** Tanto SUM como DAV obtienen 2 puntos ya que permiten la integración de equipos multidiciplinarios.

- **Documentación.-** SUM obtiene 1 punto por la falta de ejemplos prácticos en su metodología, mientras que DAV obtiene 2 puntos por la cantidad de ejemplos documentados que facilitan el uso de esta metodología, tomando en cuenta que ambas metodologías no son usadas comercialmente la documentación es muy importante para aplicar estas herramientas apropiadamente.

- **Corto tiempo.-** Tanto SUM como DAV obtienen 2 puntos ya que permiten realizar entregas del videojuego en corto tiempo.

- **Comunicación.-** Tanto SUM como DAV obtienen 2 puntos ya que permiten una comunicación transparente por todos los miembros del equipo.

En base a los resultados mostrados, SUM obtiene 7 puntos, y DAV 10 puntos, destacándose esta ultima en su flexibilidad a la hora de realizar cambios a funciones ya terminadas y su documentación ejemplificada algo muy importante en metodologías poco conocidas, por lo tanto para este proyecto se ha seleccionado **DAV** como metodología de desarrollo ágil para el desarrollo del videojuego.

4.4. Implementación del videojuego educativo en 3D para dispositivos móviles Android enfocado al aprendizaje de la Lógica de Programación.

De acuerdo al resultado de la ***sección 4.3.3***, se ha seleccionado a la metodología DAV como la más apropiada para el desarrollo de la propuesta de videojuego. A continuación se explicará todo el proceso de desarrollo dividido en tres fases principales según como lo indica la metodología.

4.4.1. Fase pre-juego

Descripción del videojuego

El videojuego ha desarrollar se llamará ***"Hello Remi"***, el nombre esta inspirado en la clásica frase que se imprime en pantalla al aprender un nuevo lenguaje de programación por primera vez *"Hello World"*, el modo de juego se basa en el concepto de *"LightBot"* [7].

El mundo será desarrollado totalmente en 3 dimensiones tanto los personajes como los escenarios, la vista del mundo tendrá dos perspectivas; la primera será una vista aérea que permitirá visualizar todo el mapa del juego; la segunda perspectiva será una vista en tercera persona, permitiendo tener una visualización más cercana de los movimientos y acciones del personaje.

"Remi" será el personaje principal y único personaje jugable, mientras que los personajes NPC (non playable character) serán dos; el primer NPC será un "gato" el cual deberá ser rescatado por el jugador, el segundo NPC será el o los "monstruos" que deberán ser eliminados por el jugador.

El jugador podrá mover a ***"Remi"*** unicamente por las casillas del mapa usando ***acciones*** predefinidas (mover, girar, volar, aterrizar, usar portal, usar magia).

Las acciones se deberán agrupar dentro de ***bloques de acciones***, existen 3 tipos de bloques:

- **bloque GO**.- Las acciones dentro de este bloque se realizarán una sola vez;
- **bloque R**.- Las acciones dentro de este bloque se realizarán n veces;
- **bloque IF** .- Las acciones de este bloque se realizarán solo si una condición resulta verdadera.

Existirán ***3 niveles de dificultad*** (fácil, medio y difícil), dependiendo de la dificultad el mapa tendrá una mayor cantidad de obstáculos y enemigos, el mapa del juego estará compuesto de columnas que se generarán aleatoriamente al empezar la partida.

Aspectos fundamentales

1. **Género**

   Este videojuego entra en la categoría de los ***puzzles*** y la ***estrategia***, el jugador deberá utilizar su orientación, razón espacial, razonamiento geométrico y rapidez para determinar el camino correcto a seguir.

2. **Historia**

   La historia empieza con Remi y su gato descansando tranquilamente en el bosque, de repente un extraño ruido los asusta, el gato corre hacia el lugar del cual se produjo el ruido, rápidamente aparece una extraña mano monstruosa que se lleva al gato dentro de un oscuro pozo. La misión de la protagonista será entrar al extraño mundo donde se encuentra secuestrado su gato eliminar los monstruos y rescatarlo.

3. **Bocetos**

   Los bocetos o diseños preliminares son dibujos realizados a mano de los elementos principales que conformarán el videojuego, hay que tener en cuenta que estos diseños pueden sufrir cambios en la versión final del videojuego por lo tanto deben ser tomados en cuenta como una referencia para el diseñador del juego el cual podrá realizar cambios mínimos si lo requiere necesario.

   En las ***Figuras 4.11, 4.12, 4.13, 4.14*** se pueden observar los bocetos del personaje principal, los NPC's y el mundo del juego respectivamente.

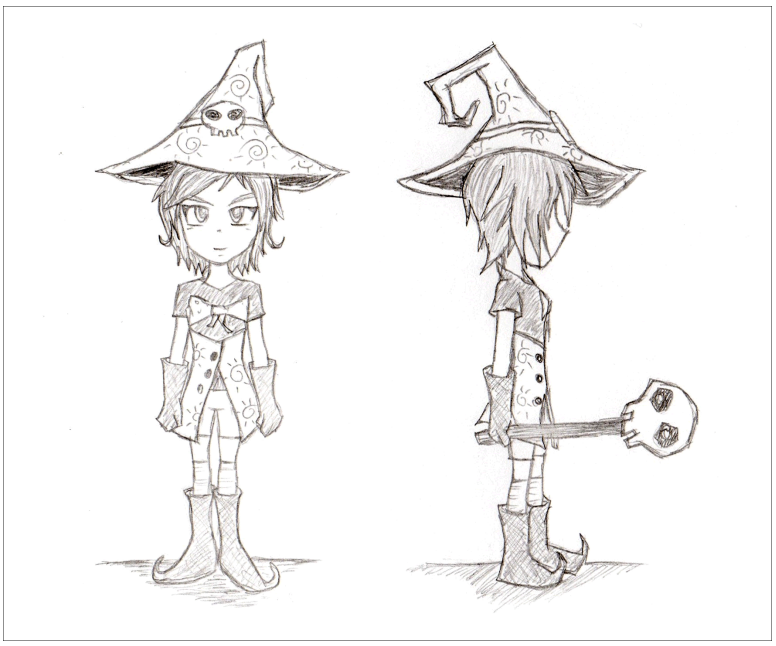


Figura 4.11: Boceto del personaje principal
Elaborado por: Hussein Rahman

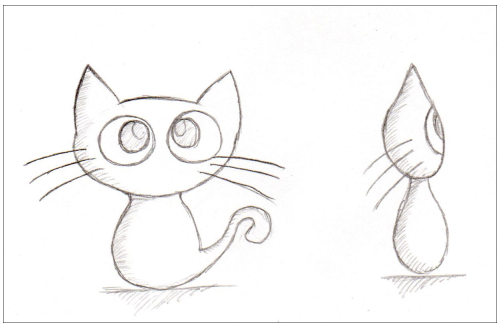


Figura 4.12: Boceto NPC (gato)
Elaborado por: Hussein Rahman

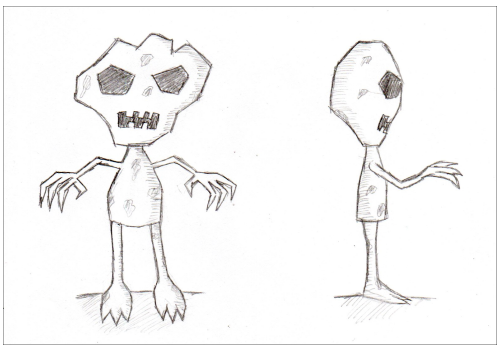


Figura 4.13: Boceto NPC (enemigo)
Elaborado por: Hussein Rahman

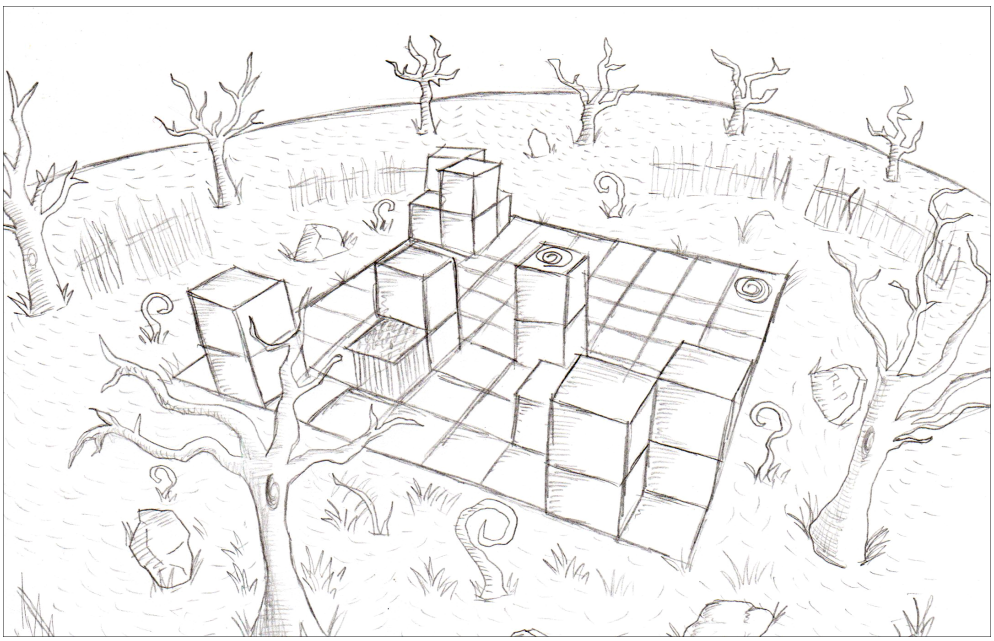


Figura 4.14: Boceto del mapa del juego
Elaborado por: Hussein Rahman

4. **Aspectos gráficos**

   El juego será desarrollado totalmente en 3D constará de 2 cámaras y 4 vistas; la cámara principal mostrará una vista aérea; en tercera persona con rotación manual y una vista 360º con rotación automática la cual se habilitará con ciertas acciones del jugador, la cámara secundaria mostrará el mapa del juego con una visión de 360º rotando automáticamente, esta cámara se activará únicamente al inicio y al final de la partida.

5. **Interfaz de usuario**

   Todas las acciones se realizan desde la interfaz de usuario (UI: User Interface) que será totalmente táctil, y se dividirá en 5 interfaces como se describen a continuación:

   - **UI Logo.-** Esta será la primera interfaz en mostrarse, cuyo único propósito es mostrar el logo del creador del juego.
   - **UI Menú principal.-** El menú principal del juego tendrá 4 botones principales: iniciar juego, salir del juego, configuración y créditos.

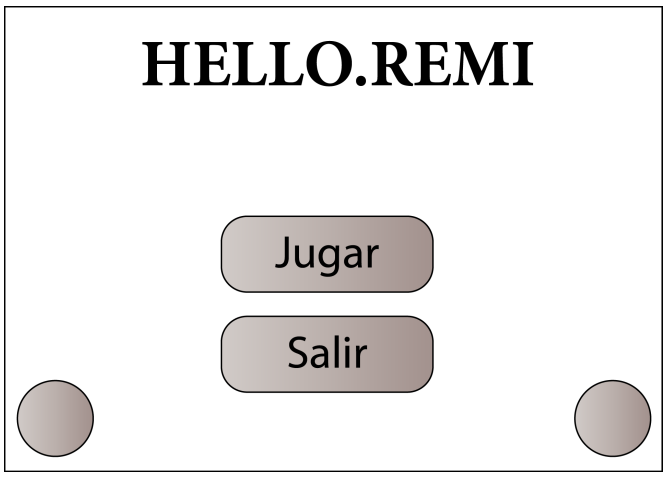


Figura 4.15: UI Menú principal
Elaborado por: Hussein Rahman

- **UI StoryBoard.-** Esta interfaz se encargará de mostrar un pequeño cómic en forma de storyboard con la introducción de la historia del juego.

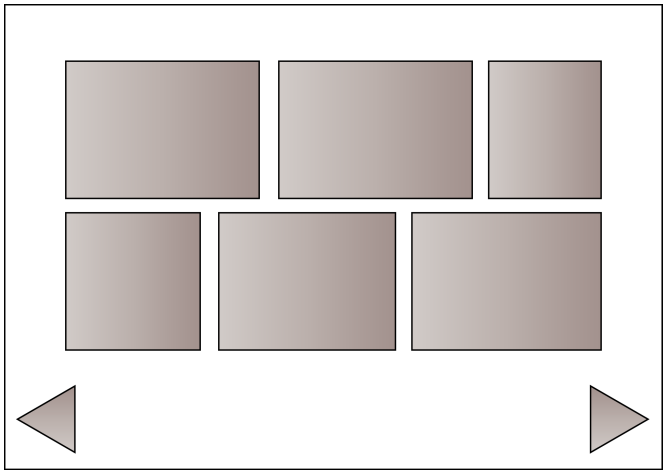


Figura 4.16: UI StoryBoard
Elaborado por: Hussein Rahman

- **UI Selección dificultad.-** El propósito de esta interfaz será mostrar un menú para seleccionar la dificultad y generar el mundo.

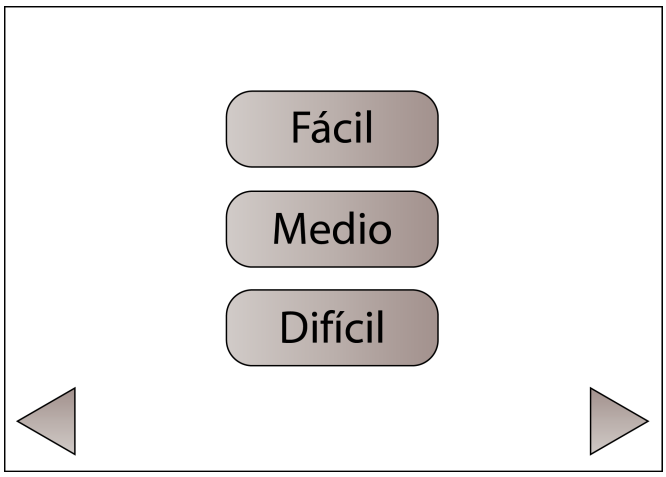


Figura 4.17: UI Selección niveles
Elaborado por: Hussein Rahman

- **UI Juego.-** Este interfaz tendrá dos paneles principales, el panel de acciones, el cual contendrá todas las acciones disponibles; el panel de ejecución ubicado a la derecha mostrará los comandos que están siendo ingresados por el jugador, incluye un botón para ejecución y un botón para borrar el ultimo comando ingresado; en la parte superior central se encontrará el botón pausa y en la parte inferior central el botón para cambiar la perspectiva de la cámara.

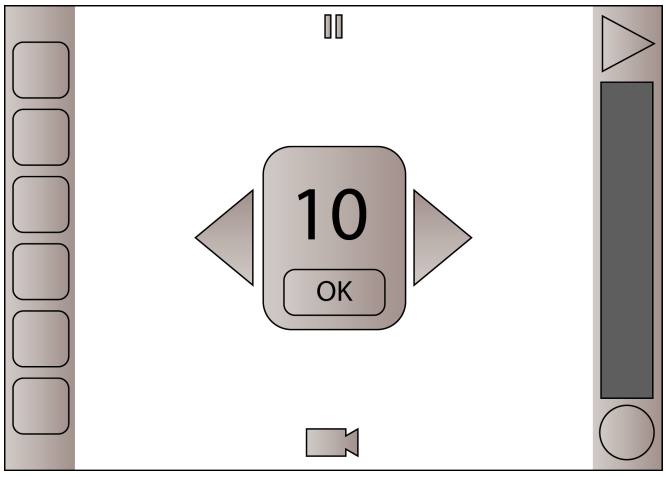


Figura 4.18: UI Juego
Elaborado por: Hussein Rahman

6. **Características de los elementos del videojuego**

   El la ***Tabla 4.11*** se describe de forma resumida las características de los principales elementos del juego.

Elementos	Descripción
Personaje principal (Remi)	Aparece al inicio de la partida, ubicada en la casilla inicial del mapa, se moverá de acuerdo a los comandos que utilice el jugador..
NPC (gato)	El NPC del juego no es un personaje controlable, y su función es la de permitir al jugador terminar la partida
NPC (enemigos)	Existirán dos tipos de enemigos, negros y blancos, los cuales tratarán de impedir que el jugador pueda avanzar libremente por el mata, disminuyendo su vida en un punto si el jugador colisiona con ellos.
PowerUps (calaveras)	Los powerUps son ítems que permiten obtener el poder de magia negra y blanca, algo necesario para derrotar a los enemigos.
Magia (blanca/negra)	Existirá dos tipos de magia, negra y blanca, este poder se obtendrá con 3 PowerUps, permitiendo eliminar enemigos, usando la magia negra, para enemigos blancos y la magia blanca para enemigos negros.
Mundo/Niveles	Cada nivel se generará aleatoriamente.

Tabla 4.11: Características de los elementos del juego

Elaborado por: Hussein Rahman

7. **Objetivo del videojuego**

   El objetivo del juego es mover al personaje principal del juego ***"Remi"*** para obtener ***3 powerUps*** y eliminar a todos los ***NPC (monstruos)*** que se encuentran en el escenario, luego de eliminarlos se desbloqueará la ubicación del ***NPC (gato)*** que el jugador deberá recuperar. Se deberá terminar el nivel en el menor tiempo posible para obtener una mejor puntuación, ademas dispondrá de un número de intentos limitados según el nivel de dificultad escogido.

   El videojuego busca divertir al jugador mientras mejora su razonamiento y rapidez mental con el uso de diferentes conceptos que se utilizan en la lógica de la programación pero de forma amigable y divertida.

8. **Reglas**

   - El juego inicia con 3 vidas (corazones) , 0 powerUps (calaveras) y las acciones de magia deshabilitadas.

- Cada nivel (fácil, medio o difícil) tiene 3 intentos.
- Al dar clic en el botón ejecutar se consumirá un intento.
- Si el jugador colisiona con un monstruo perderá una vida.
- Si el jugador colisiona con un objeto del mapa perderá un intento.
- Si el jugador ingresa una cantidad de pasos que excede al número de casillas perderá un intento.
- Si las vidas del jugador llegan a 0 perderá la partida y deberá empezar de nuevo.
- Si los intentos del jugador llegan a 0 sin haber llegado a la meta, perderá la partida y deberá empezar de nuevo.
- Cuando el jugador colisione con un powerUp sus powerUps se incrementaran en 1.
- Cuando el jugador obtenga 3 powerUps se habilitarán las acciones magia negra/blanca.
- La magia es la única acción que elimina a los monstruos.
- Existen dos tipos de magia y dos tipos de enemigos, blancos y negros.
- Los enemigos blancos se destruyen con magia negra, mientras que los negros se destruyen con magia blanca.
- Existirán enemigos especiales que intercambiarán su color entre negro y blanco.
- Cuando el jugador elimine a todos los enemigos se desbloqueará la ubicación del personaje NPC gato.
- Una vez que el jugador recupere al NPC gato se completará el nivel y se mostrará el puntaje obtenido en base a los intentos utilizados.
- Terminar el nivel con un intento equivale a 3 puntos, dos intentos son 2 puntos y tres intentos son 1 punto.
- Jugar en dificultad fácil multiplicará los puntos por uno, nivel medio por 2 y nivel difícil por 3.
- Los puntos obtenidos permiten desbloquear nuevos trajes para el personaje.

Aspectos técnicos

1. **Requerimientos técnicos**

   Los requisitos técnicos, tanto de hardware como de software son los siguientes:

Hardware	**Software**
■ CPU ARMv7 (Cortex) con tecnología NEON o CPU Atom; OpenGL ES 2.0 o posterior. ■ 1GB Ram	Android OS 2.3.3 o superior

Tabla 4.12: Requisitos técnicos

Elaborado por: Hussein Rahman
Fuente: [24]

2. **Arquitectura del videojuego**

   La arquitectura del videojuego utiliza MVC (Modelo Vista Controlador) aplicado específicamente a este proyecto y al motor Unity, como se muestra a continuación:

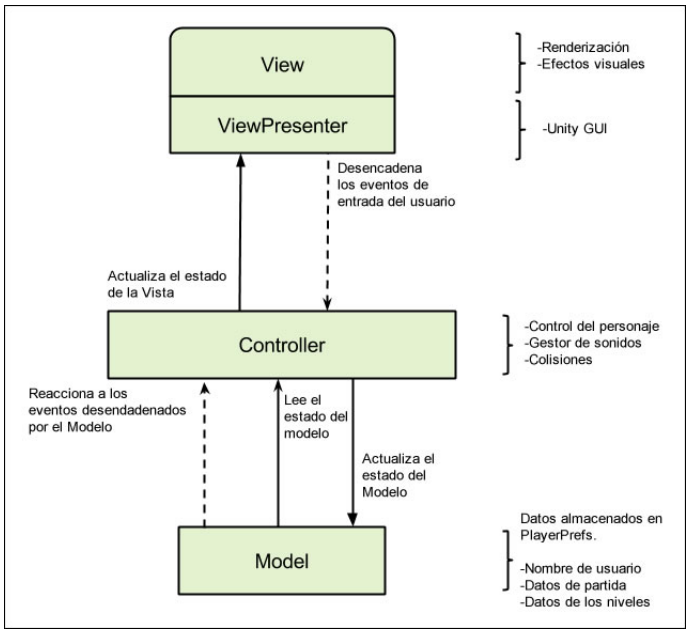


Figura 4.19: MVC del videojuego
Elaborado por: Hussein Rahman

- **Model.-** Esta capa se encarga de gestionar los datos que serán usados durante la ejecución del videojuego utilizando la clase ***PlayerPrefs*** de Unity, encargada de guardar y mostrar datos.

- **Controller.-** Esta capa se encarga de leer los datos almacenados en el Modelo para actualizar el estado de la Vista, gestionando otros componentes del videojuego como el control del personaje, la gestión de los sonidos y las colisiones.
- **ViewPresenter.-** Esta capa muestra la interfaz de usuario GUI (Interfaz gráfica de usuario), y desencadena los eventos de entrada del usuario que son enviados hacia la capa del Controlador.
- **View.-** La ultima capa se encarga de la renderización de los elementos del videojuego, así como la gestión de efectos visuales.

3. **Diseño conceptual**

   En la ***Figura 4.20*** se indica el diseño conceptual que contiene las diferentes interfaces de usuario y su forma de navegación.

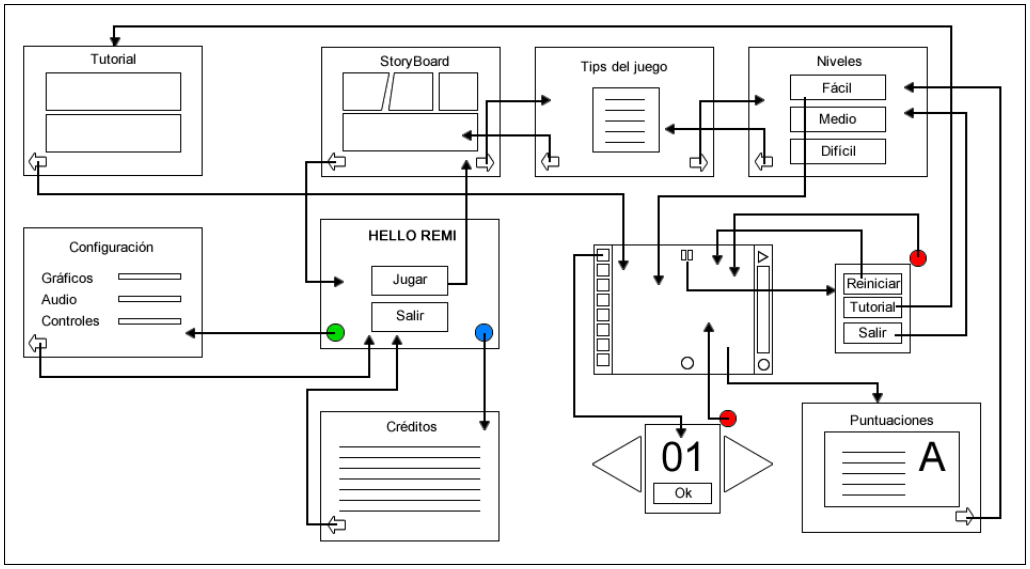


Figura 4.20: Diseño conceptual
Elaborado por: Hussein Rahman

Herramientas a utilizarse

Para el diseño de la parte gráfica se utilizarán las siguientes herramientas:

- **Diseño gráfico**
  - **Blender.-** Es un programa informático multiplataforma, dedicado especialmente al modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales. El programa fue inicialmente distribuido de forma gratuita pero sin el código fuente, con un manual disponible para la venta, aunque posteriormente pasó a ser software

libre. Actualmente es compatible con todas las versiones de Windows, Mac OS X, GNU/Linux (Incluyendo Android), Solaris, FreeBSD e IRIX [29].

- **Gimp.-** Es un programa de edición de imágenes digitales en forma de mapa de bits, tanto dibujos como fotografías, es un programa libre y gratuito. GIMP tiene herramientas que se utilizan para el retoque y edición de imágenes, dibujo de formas libres, cambiar el tamaño, recortar, hacer fotomontajes, convertir a diferentes formatos de imagen, y otras tareas más especializadas. Se pueden también crear imágenes animadas en formato GIF (Graphics Interchange Format) e imágenes animadas en formato MPEG (Moving Picture Experts Group) usando un plugin de animación [30].
- **InkScape.-** Es un editor de gráficos vectoriales gratuito y de código libre. InkScape puede crear y editar diagramas, líneas, gráficos, logotipos, e ilustraciones complejas. El formato principal que utiliza el programa es SVG (Scalable Vector Graphics) [31].
- **Audacity.-** Es una aplicación informática multiplataforma libre, que se puede usar para grabación y edición de audio, distribuido bajo la licencia GPL (General Public License) [32].

- **Desarrollo**

  Para el desarrollo del videojuego se ha seleccionado Unity 3D como framework y motor de juego en base a un análisis y evaluación previa como se describe en la ***sección* ??** ademas se han elegido las siguientes herramientas complementarias para cumplir con los objetivos del proyecto.

  - **Unity 3D.-** Es un motor de videojuego multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows, OS X y Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas. Desde su versión 5.4.0 ya no soporta el desarrollo de contenido para navegador a través de su plugin web en su lugar se utiliza WebGL. Unity tiene dos versiones: Unity Professional (pro) y Unity Personal [24].
  - **Visual Studio 2015.-** Es un programa de desarrollo para sistemas operativos Windows desarrollado y distribuido por Microsoft Corporation. Soporta varios lenguajes de programación tales como Visual

C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros [33].

- **Android SDK.-** El SDK (Software Development Kit) de Android, incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. El SDK es indispensable para poder compilar desde Unity hacia Android [34].

Roles del equipo

Para el desarrollo del videojuego se asignaron los siguientes roles mostrados en la ***Tabla 4.13***.

Persona	**Contacto**	**Rol**
Hussein Rahman	husseingabriel@outlook.com	DAV Master
		Dueño del Producto
		Equipo DAV (Sonidista)
		Equipo DAV (Diseñador)
		Equipo DAV (Programador)

Tabla 4.13: Roles

Elaborado por: Hussein Rahman

Historias de usuario

Las siguientes historias de usuario presentan las funcionalidades que tendrá el videojuego "Hello Remi".

ID	**Como**	**Quiero.../Quiero que...**	**De modo que.../Para**
HU01	Jugador	Acceder al juego, salir, configurar y ver los créditos	Acceder al juego, salir, configurar y ver los créditos
HU02	Jugador	Antes de iniciar el juego ver la historia y las motivaciones del los personajes.	Motivar a empezar el juego y causar diversión.
HU03	Jugador	Tener la posibilidad de cambiar la calidad gráfica del juego	Mejorar el rendimiento del juego en caso de que el dispositivo no sea muy potente.
HU04	Jugador	Visualizar las instrucciones del juego por medio de algún tutorial	Entender la mecánica y controles del juego.
HU05	Jugador	Visualizar una pantalla de créditos.	Conocer quienes fueron los que desarrollaron el juego.

ID	Como	Quiero.../Quiero que...	De modo que.../Para
HU06	Jugador	Visualizar un menú para seleccionar los mundos y niveles.	Acceder a los mundos y niveles disponibles
HU07	Jugador	El personaje tenga diferentes atuendos o vestimentas.	Evitar la monotonía y tener mas variedad dentro del juego
HU08	Jugador	Visualizar una pantalla que muestre las puntuaciones obtenidas al finalizar el nivel.	Motivar al jugador ha rejugar el nivel para mejorar su puntaje.
HU09	Programador	Cada casilla del mapa se identifique con una coordenada única.	Facilitar el movimiento del personaje dentro del mapa
HU10	Programador	El personaje tenga diferentes acciones ejecutables.	Facilitar el desplazamiento y la interacción con el mundo.
HU11	Jugador	Existan enemigos variados dentro del mapa.	Ofrecer una mejor experiencia jugable con mayor retos para el jugador
HU12	Jugador	Un mapa con elevaciones y una estructura variada.	Brindar un mayor desafió a la hora de resolver el nivel
HU13	Jugador	La cámara posea una vista aérea y en tercera persona.	Se tenga una mejor visualización del tablero del juego desde diferentes perspectivas.
HU14	Programador	Tener comandos para ejecutar las acciones del personaje.	Facilitar la ejecución de las acciones por medio de un lenguaje común.
HU15	Diseñador	El personaje sea divertido y amigable.	Crear un personaje divertido y amigable para el jugador
HU16	Jugador	El juego tenga al menos dos idiomas para escoger.	Facilitar la comprensión del del juego a personas que no dominen el idioma español.
HU17	Jugador	El personaje posea acciones avanzadas y pueda tomar decisiones.	Mejorar la experiencia jugable con una mayor capacidad de estrategia dentro del juego.
HU18	Programador	Tener comandos avanzados para ejecutar las acciones avanzadas del personaje.	Facilitar la ejecución de las acciones por medio de un lenguaje común.
HU19	Diseñador	La decoración del mundo este acorde a los personajes y la historia del juego.	Crear un ambiente divertido para el jugador.
HU20	Jugador	El personaje principal cambie de animación según la acción que realice.	Crear un personaje con movimientos naturales y divertidos.

ID	Como	Quiero.../Quiero que...	De modo que.../Para
HU21	Programador	Controlar las colisiones del jugador con los objetos del mundo.	Crear realismo entre el personaje y los objetos del mundo, reaccionando de diferente manera según el objeto con el que se colisione.
HU22	Jugador	Visualizar una pantalla de pausa que permita reiniciar o salir de la partida.	El jugador pueda reiniciar o salir de la partida rápidamente.
HU23	Programador	Mostrar una notificación en la pantalla si los comandos se ingresan de forma incorrecta.	Evitar errores internos que impidan el correcto funcionamiento del juego.
HU24	Jugador	Visualizar los comandos que serán ejecutados.	Retroalimentar al usuario al visualizar los comandos que serán ejecutados.
HU25	Jugador	El personaje tenga powerUps para mejorar sus habilidades.	Mejorar la experiencia jugable y aumentar la diversión del juego.
HU26	Jugador	Visualizar la salida o meta del juego.	La meta permitirá dar por terminado el nivel del juego.
HU27	Jugador	El personaje pierda el nivel si sus vidas llegan a 0.	Aumentar la dificultad del juego y mejorar la experiencia.
HU28	Sonidista	Efectos de sonidos y música de fondo para el videojuego	Retroalimentar al jugador cuando escuche los sonidos de determinadas acciones.

Elaborado por: Hussein Rahman

Tabla 4.14: Historias de usuario del videojuego

Pruebas de aceptación

Estas pruebas permiten validar las diferentes funcionalidades del videojuego una vez desarrolladas, de esta forma se garantizará que cumplan con lo requerido y pasarán al estado de completado.

Prueba de Aceptación	
Identificador: PA01	**Historia de Usuario (Nro. y Nombre):** HU01- Como jugador quiero visualizar el menú principal del videojuego
Nombre: Realizar el menú principal del juego	
Descripción: Crear una pantalla con botones que permitan entrar al juego, salir y cambiar las configuraciones del juego.	
Condiciones de ejecución: Debe ejecutarse en el dispositivo móvil, adaptarse a cualquier pantalla y funcionar correctamente.	
Entrada/Pasos de ejecución: - Presentar el menú principal con sus respectivos botones.	
Resultado esperado: Acceder a las respectivas pantallas según el botón seleccionado.	
Evaluación de prueba: Correcto	

Tabla 4.15: Prueba de aceptación "PA01"

Elaborado por: Hussein Rahman

Reserva del producto

En la ***Tabla 4.16*** se observan las ***historias de usuario*** con sus respectivas pruebas de aceptación ***(PA)***, puntos de prioridad ***(P)*** y estimación ***(E)*** según el dueño del producto.

P	Nro. HU	Ítem	E	Estado	PA
4	HU01	Como Jugador, quiero visualizar el menú principal del videojuego	1	Pendiente	PA01
4	HU02	Como jugador quiero antes de iniciar el juego ver la historia y las motivaciones del los personajes.	2	Pendiente	PA02
4	HU03	Como jugador quiero tener la posibilidad de cambiar la calidad gráfica del juego	2	Pendiente	PA03
4	HU04	Como jugador quiero visualizar las instrucciones del juego por medio de algún tutorial	1	Pendiente	PA04
4	HU05	Como jugador quiero visualizar una pantalla de créditos.	1	Pendiente	PA05
4	HU06	Como jugador quiero visualizar un menú para seleccionar los mundos y niveles.	1	Pendiente	PA06
4	HU07	Como jugador quiero que el personaje tenga diferentes atuendos o vestimentas.	2	Pendiente	PA07
4	HU08	Como jugador quiero visualizar una pantalla que muestre las puntuaciones obtenidas al finalizar el nivel.	2	Pendiente	PA08

P	Nro. HU	Ítem	E	Estado	PA
1	HU09	Como programador quiero que cada casilla del mapa se identifique con una coordenada única.	2	Pendiente	PA09
1	HU10	Como jugador quiero que el personaje tenga diferentes acciones ejecutables.	1	Pendiente	PA10
2	HU11	Como jugador quiero que existan enemigos variados dentro del mapa.	1	Pendiente	PA11
2	HU12	Como jugador quiero un mapa con elevaciones y una estructura variada..	1	Pendiente	PA12
2	HU13	Como jugador quiero que la cámara posea una vista aérea y en tercera persona.	2	Pendiente	PA13
1	HU14	Como programador quiero tener comandos para ejecutar las acciones del personaje.	2	Pendiente	PA14
1	HU15	Como jugador quiero que el personaje sea divertido y amigable.	1	Pendiente	PA15
4	HU16	Como jugador quiero que el juego tenga al menos dos idiomas para escoger	1	Pendiente	PA16
3	HU17	Como jugador quiero que el personaje posea acciones avanzadas y pueda tomar decisiones.	1	Pendiente	PA17
3	HU18	Como programador quiero tener comandos avanzados para ejecutar las acciones avanzadas del personaje.	1	Pendiente	PA18
3	HU19	Como diseñador quiero que la decoración del mundo este acorde a los personajes y la historia del juego.	1	Pendiente	PA19
1	HU20	Como diseñador quiero que el personaje principal cambie de animación según la acción que realice.	3	Pendiente	PA20
2	HU21	Como programador quiero controlar las colisiones del jugador con los objetos del mundo.	1	Pendiente	PA21
2	HU22	Como jugador quiero visualizar una pantalla de pausa que permita reiniciar o salir de la partida.	1	Pendiente	PA22
2	HU23	Como programador quiero mostrar una notificación en la pantalla si los comandos se ingresan de forma incorrecta.	2	Pendiente	PA23
1	HU24	Como jugador quiero visualizar los comandos que serán ejecutados.	1	Pendiente	PA24
3	HU25	Como jugador quiero que el personaje tenga powerUps para mejorar sus habilidades.	2	Pendiente	PA25
1	HU26	Como jugador quiero visualizar la meta o salida del juego	3	Pendiente	PA26

P	Nro. HU	Ítem	E	Estado	PA
3	HU27	Como jugador quiero que el personaje pierda el nivel si sus vidas llegan a 0.	1	Pendiente	PA27
3	HU28	Como sonidista quiero efectos de sonidos y música de fondo para el videojuego	1	Pendiente	PA28

Elaborado por: Hussein Rahman

Tabla 4.16: Reserva del producto del videojuego

Determinación de las fechas de entrega

Toda la ***reserva del producto*** se ha dividido en 4 iteraciones o entregas, con una duración de 2 semanas cada una, en la ***Tabla 4.17*** se muestra las fechas de inicio y fin así como también las fechas de entrega de cada iteración.

Iteración	Fecha de inicio	Fecha de fin	Fecha de entrega
1	04 de julio del 2016	16 de julio del 2016	17 de julio del 2016
2	18 de julio del 2016	30 de julio del 2016	31 de julio del 2016
3	1 de agosto del 2016	13 de agosto del 2016	14 de agosto del 2016
4	15 de agosto del 2016	27 de agosto del 2016	28 de agosto del 2016

Tabla 4.17: Fechas de entrega de cada iteración

Elaborado por: Hussein Rahman

4.4.2. Fase juego

Una vez que se ha definido el alcance, las funcionalidades y un cronograma de las fechas de entrega de cada iteración del videojuego se empezará el desarrollo de cada una de las iteraciones.

La duración de cada iteración será de 12 días es decir de lunes a sábado, los días lunes se realizarán las reuniones de planificación de iteración y las revisiones de las mismas se efectuarán los días domingos, de esta manera los otros días serán usados para desarrollar las funcionalidades de cada entrega.

Desarrollo de la iteración 1

El objetivo de esta iteración es obtener la primera versión del videojuego, la fecha de revisión de la iteración será el día 17 de julio de 2016, en esta iteración se desarrollará las siguientes historias de usuario.

ID	Descripción
HU09	Como programador quiero que cada casilla del mapa se identifique con una coordenada única.
HU10	Como jugador quiero que el personaje tenga diferentes acciones ejecutables.
HU14	Como programador quiero tener comandos para ejecutar las acciones del personaje.
HU15	Como jugador quiero que el personaje sea divertido y amigable.
HU20	Como diseñador quiero que el personaje principal cambie de animación según la acción que realice.
HU24	Como jugador quiero visualizar los comandos que serán ejecutados.
HU26	Como jugador quiero visualizar la salida o meta del juego

Tabla 4.18: Historias de usuario a implementarse en la Iteración 1

Elaborado por: Hussein Rahman

Se han generado ***tarjetas de tareas*** en base a las ***historias de usuarios*** seleccionadas para la ***iteración 1***, las tareas fueron auto-asignadas al miembro del equipo DAV, el contenido de las tarjetas se encuentran en el ***Anexo A.***

A continuación se resumen las tareas y subtareas realizadas para la iteración 1.

T01: Generación de las casillas del mapa

El objetivo de esta tarea es generar el mapa base del juego con la creación de casillas las mismas que deben tener coordenadas únicas entre ellas. La clase que se encargará de realizar esto se llamará **"CreateGrid.cs"**. Para el cumplimiento de esta tarea se han identificado las siguientes subtareas:

- **Crear las casillas del mapa.**- Para la creación de la casilla es necesario crear una instancia de un ***GameObject***, y el mismo debe contener al objeto ***"tile"*** (Objeto 3D) la cual se clonara las veces que sean necesarias para la creación del plano.

  El método ***"CreateTile"*** es el encargado de la clonación del objeto ***"tile"*,** pero ademas se encarga también de cambiar la posición, nombre, escala, material y convertirlo en un objeto hijo de otro objeto.

En la ***Figura 4.21*** se puede observar la funcionalidad del método.

```
1  public void CreateTile(float x, float y, float z, float scaleY, string name,
       GameObject plane, Color color)
2  {
3    GameObject tile = (GameObject)Instantiate (prefabTile);
4    tile.transform.position = new Vector3(x,y,z);
5    tile.name = name;
6    tile.transform.localScale = new Vector3 (1, scaleY, 1);
7    tile.GetComponent<Renderer> ().material.mainTexture = txtBase.texture (b);
8
9    tile.GetComponent<Renderer> ().material.color = color;
10   //añadir casilla hija al padre plano
11   tile.GetComponent<RectTransform>().SetParent(plane.transform);
12 }
```

Figura 4.21: Método "CreateTile"
Elaborado por: Hussein Rahman

- **Crear el eje del mundo**.- El eje del mundo es un objeto no visible necesario para tener un punto de referencia central del mundo y servirá como eje para la rotación de la cámara aérea que se implementará en las siguientes versiones del videojuego.

- **Crear el mapa**.- El mapa será un conjunto de casillas en las cuales se podrá mover el personaje, para su creación fue necesario entender como funciona el sistema de coordenadas en Unity 3D, en la ***Figura 4.22*** se ilustra el espacio tridimensional de Unity con los ejes ***X***, ***Y*** y ***Z*** respectivamente.

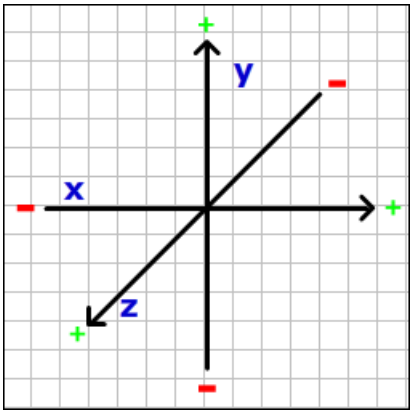


Figura 4.22: Ejes de coordenadas Unity
Elaborado por: Hussein Rahman

Como se puede observar en la ***Figura 4.22***, los ejes ***X*** y ***Z*** representarán el piso de nuestro mundo, por lo tanto serán los ejes con los que se trabajará

para la creación del mapa y la asignación de coordenadas en cada casilla clonada.

El método ***"CreateMap"*** mostrado en la ***Figura 4.23*** es el encargado de la clonación y asignación de coordenadas, utilizando un ***For*** externo para las filas y un ***For*** interno para las columnas, ademas se puede observar que el parámetro ***Y*** del método ***"CreateTile"*** se encarga de la posición en el eje ***Y*** tiene un valor de ***0*** lo que permite que las casillas se posicionen a una altura con valor ***0*** respecto al piso del mundo.

Finalmente la asignación de coordenadas únicas es realizado por el método ***"CreateTile"*** con su parámetro ***"name"*** se asigna el valor de la fila y columna de la casilla actual que esta siendo clonada.

```
1  public void CreateMap(int rows, int columns)
2  {
3    CreatePlane ();
4    int[,] mat;
5    mat = new int[rows,columns];
6    for(int f = 0; f < mat.GetLength(0); f++){
7      for (int c = 0; c < mat.GetLength(1); c++){
8      //f=coorX, c=coorZ, ejm: 0,0 => coordenada x=0 y z=0
9      CreateTile(f,0,c,tileHeight,f+""+c,plane,color);
10     }
11   }
12   CreateAxisWorld ();
13 }
```

Figura 4.23: Método "CreateMap"
Elaborado por: Hussein Rahman

En la ***Figura 4.24*** se puede observar el plano de ***n*** x ***n*** casillas, donde cada casilla tiene una coordenada única que permitirá mas adelante establecer el movimiento del personaje y la creación de objetos dentro del mapa según estas coordenadas.

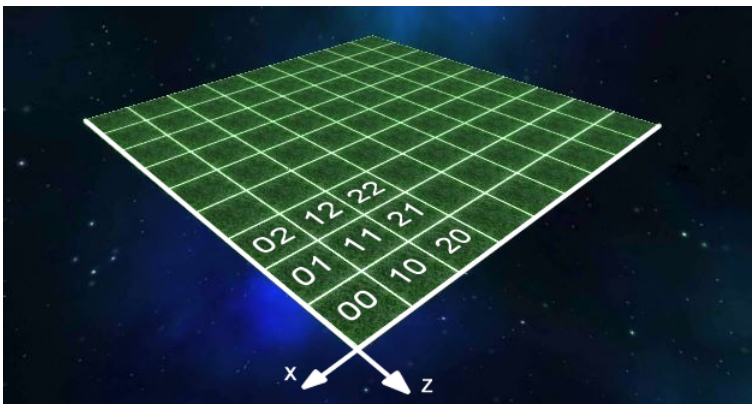


Figura 4.24: Casillas generadas para el mapa
Elaborado por: Hussein Rahman

T02: Acciones básicas para el personaje

El objetivo de esta tarea es otorgar al personaje del juego la capacidad para realizar diferentes tipos de acciones para que pueda moverse e interactuar con el mapa. La clase que se encargará de esto se llamará ***"ControlPlayer.cs"***.

Antes de mostrar el funcionamiento de cada acción es importante entender como se ubica el personaje con respecto a los ángulos del mundo.

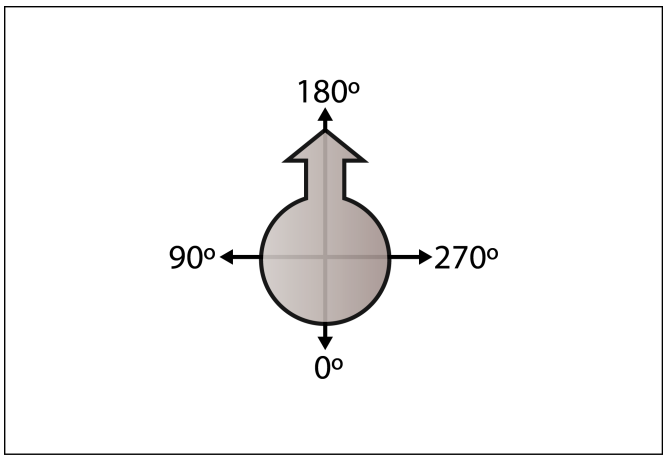


Figura 4.25: Ángulos del personaje con respecto al mundo
Elaborado por: Hussein Rahman

Se han determinado las siguientes acciones que podrá realizar el personaje:

- **Acción mover.-** La acción mover utiliza dos métodos, el método ***"MoveInCoordinates"*** encargado de mover al personaje a una coordenada especifica

del mapa, siendo necesarios los datos de la posición actual del personaje para realizar el movimiento.

Cuando los datos de las coordenadas ***X*** y ***Z*** origen son iguales a las coordenadas del destino, se finaliza el movimiento y se suma +1 a la variable ***"action"*** la cual indicará el fin de la acción actual y el comienzo de la siguiente.

```
1  public void MoveInCoordinates(int x,int z){
2    coordinates = x + "" + z;
3    target = GameObject.Find (coordinates);
4    x0 = target.transform.position.x;
5    y0 = jugador.transform.position.y;
6    z0 = target.transform.position.z;
7    destination = new Vector3 (x0, y0, z0);
8    step = speed * Time.deltaTime;
9    player.transform.position =
10       Vector3.MoveTowards (player.transform.position, destination, step);
11   if (player.transform.position.x == target.transform.position.x &&
12     player.transform.position.z == target.transform.position.z) {
13     action += 1;
14   }
15 }
```

Figura 4.26: Método "MoveInCoordinates"
Elaborado por: Hussein Rahman

Aunque con el método anterior ya es posible desplazar al personaje a cualquier posición dentro del mapa, este no es el objetivo de la acción mover, el objetivo es mover al personaje hacia adelante con pasos entre ***1*** a ***n*** (número máximo de casillas), para esto es necesario el método ***"Move"***.

El método ***"Move"*** tiene como único parámetro el número de pasos que el personaje recorrerá, para obtener la coordenada a la que se debe desplazar se obtiene primeramente la coordenada ***X*** y ***Z*** actual del personaje, luego dependiendo del ángulo actual del personaje se sumará el número de pasos a la coordenada con referencia al ángulo actual. Por ejemplo si el ángulo es ***180º*** el frente del personaje esta hacia las ***Z+*** por lo que se sumará a las ***Z*** el número de pasos tal y como se muestra en la ***Figura 4.27***.

```
public void Move(int step){
  x = (int)Mathf.Round(player.transform.position.x);
  z = (int)Mathf.Round(player.transform.position.z);
  if (playerAngle == 180) {
    z += step;
  } else if (playerAngle == 270) {
    x += step;
  } else if (playerAngle == 0) {
    z -= step;
  } else if (playerAngle == 90) {
    x -= step;
  }
  //si las coordenadas son positivas y no son mayores al numero de filas/
      columnas -1
  if (x >= 0 && z >= 0 && x <=main.nMatrix-1 && z <=main.nMatrix-1) {
    MoveInCoordinates (x, z);
  }
}
```

Figura 4.27: Método "Move"
Elaborado por: Hussein Rahman

- **Acción rotar.-** Esta acción permite rotar en cualquiera de los 4 ángulos mostrados en la ***Figura 4.25***, el método ***"Turn"*** utiliza un valor entero como parámetro de entrada entre los rangos de 1 y -1.

  Cuando el valor es ***1*** el giro será en sentido horario incrementando ***90º*** al valor del ángulo actual del personaje, por el contrario si el valor es ***-1*** el giro será antihorario restando ***90º*** al ángulo actual del personaje

  En la ***Figura 4.28*** se muestra la estructura del método ***"Turn"***.

```
public void Turn(int value) //1 = giro derecha, -1 giro izquierda
{
  finalAngle = playerAngle + (90 * value);
  if (finalAngle > 270) {
    finalAngle = finalAngle - 360;
  } else if (finalAngle < 0) {
    finalAngle = 270;
  }
  step = speed * Time.deltaTime;
  targetRotation =  Quaternion.AngleAxis(finalAngle, Vector3.up);
  player.transform.rotation =
    Quaternion.Lerp (player.transform.rotation, targetRotation , step);
  playerAngle = Mathf.Round(player.transform.eulerAngles.y);
  if (finalAngle == playerAngle) {
    action += 1;
  }
}
```

Figura 4.28: Método "Turn"
Elaborado por: Hussein Rahman

- **Acción volar.-** Esta acción permite el desplazamiento del personaje sobre

el eje ***Y***, es decir el desplazamiento vertical sobre el mapa simulando un vuelo, para realizar esto se creará el método ***"Fly"***, con un valor entero como parámetro de entrada que determina la distancia de desplazamiento vertical.

La gravedad del mundo al igual que en la realidad es una fuerza que atrae los cuerpos hacia la tierra, para evitar que la gravedad afecte al personaje durante el vuelo, se desactiva mientras se realiza la acción y se activa cuando la acción termina.

En la ***Figura 4.29*** se muestra la estructura del método ***"Fly"***:

```
1  public void Fly(int y){
2    x0 = player.transform.position.x;
3    y0 = player.transform.position.y;
4    z0 = player.transform.position.z;
5    rb.useGravity = false;
6    destination = new Vector3 (x0, y0 + y, z0);
7    step = speed * Time.deltaTime;
8    player.transform.position =
9      Vector3.MoveTowards (player.transform.position, destination, step);
10
11   if (player.transform.position.y==destination.y) {
12     Physics.gravity = new Vector3 (0f, -1.3f, 0f);
13     rb.useGravity = true;
14     action += 1;
15   }
16 }
```

Figura 4.29: Método "Fly"
Elaborado por: Hussein Rahman

T03: Comandos básicos

El objetivo de esta tarea es crear una nomenclatura fácil de entender para el manejo de las acciones del personaje. La clase que se encargará de esto se llamará ***"CreateCode.cs"***. Para el cumplimiento de esta tarea se han identificado las siguientes subtareas:

- **Nomenclatura de los comandos básicos.-** Para facilitar el uso de las acciones dentro del juego es necesario formalizar una nomenclatura de comandos que será utilizada internamente para la ejecución de las acciones desarrolladas en la Tarea T02.

Los comandos tendrán una longitud de 2 a 6 caracteres, se añadirá un punto y el valor si requiere de un parámetro. En la ***Tabla 4.19*** se detalla la nomenclatura para los comandos básicos, los parámetros ***"right"*** y ***"left"*** se pueden acortar con las letras ***"r"*** y ***"l"*** respectivamente.

Comando	**Significado de los comandos**
go	El comando **go** indica el inicio de un **bloque GO**
repeat.n	El comando **repeat** indica el inicio de un **bloque R** repitiendo las acciones del bloque según el valor del parámetro n, ejm: repeat.3 (las acciones del bloque se repetirán 3 veces)
end	El comando **end** indica el final de un bloque de acciones.
move.n	El comando **move** ejecuta la acción mover, utilizando el parámetro n, ejm: move.2 (El personaje se moverá 2 pasos)
turn.right turn.left	El comando **turn** ejecuta la acción rotar, utilizando el parámetro n, ejm: turn.left (El personaje rotará 90º hacia la izquierda)
fly.n	El comando **fly** ejecuta la acción volar, desplazándose horizontalmente, utilizando el parámetro n, ejm: fly.2 (El personaje se volará 2 casillas hacia arriba)

Tabla 4.19: Nomenclatura de comandos básicos

Elaborado por: Hussein Rahman

- **Bloques de código.-** Un bloque de código permite agrupar comandos de acciones, para esta iteración se han creado 2 bloques principales, el ***bloque GO*** que ejecutará el grupo de acciones una sola vez y el ***bloque R*** que ejecutará ***n*** veces el grupo de acciones.

En la ***Figura 4.30*** muestra la estructura del bloque de código y su funcionamiento.

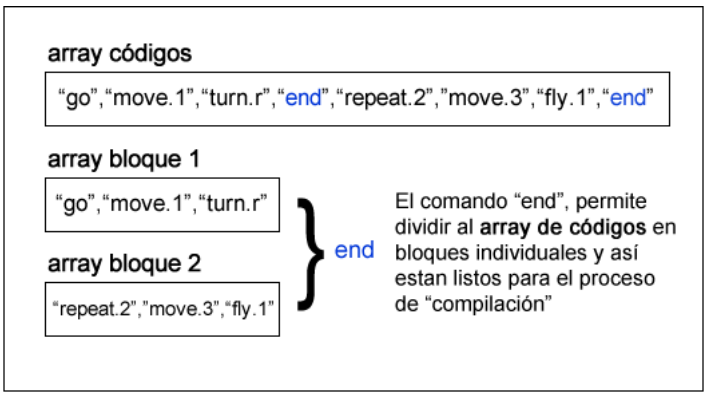


Figura 4.30: Estructura del bloque de códigos
Elaborado por: Hussein Rahman

- **Compilación de códigos.-** El proceso de compilación permite convertir los ***comandos*** en ***acciones*** que el personaje realizará sobre el mundo, el método encargado de realizar este proceso se llama ***"CompileCode"*** y esta dividido en dos partes principales.

  La primera parte del método se encarga de extraer los comandos de cada bloque y ejecutar cada una de las acciones que contiene.

```
1  public void CompileCode(int num)
2  {
3    if (control.Action != block [num].Length) {
4      if (control.Action < control.Action + 1) {
5        t = block [num] [control.Actioncion].Split ('.');
6        if (t [0] == "go") {
7          strCom = true; //inicio del bloque GO
8          repeat = 1;
9          control.Action += 1;
10       } else if (t [0] == "repeat") {
11         strCom = true; //inicio del bloque R
12         repeat = int.Parse (t [1]);
13         control.Action += 1;
14       }
15       //Aqui van mas acciones...
16     }
17   }
```

Figura 4.31: Método "CompileCode" primera parte
Elaborado por: Hussein Rahman

En la segunda parte del método ***"CompilarCodigo"*** verifica si existen repeticiones en el bloque, caso contrario da por finalizada la ejecución del bloque actual**.**

```
1    else {
2    //Si existe repetición realizar esto
3      if (endRepeat < repeat) {
4        control.Action = 0;
5        endRepeat += 1;
6      } else {
7        t = null;
8        strCom = false;
9        currentBlock += 1;
10       control.Action = 0;
11       endRepeat = 1;
12     }
13   }
14 }
```

Figura 4.32: Método "CompileCode" segunda parte
Elaborado por: Hussein Rahman

T04: Diseño y texturización del personaje principal

El objetivo de esta tarea es diseñar al personaje del videojuego para su posterior animación, el encargado de esta tarea es el diseñador y para su cumplimiento se han identificado las siguientes subtareas:

- **Modelado del personaje.-** El propósito de esta subtarea es obtener un modelo tridimensional a partir del boceto realizado en la fase pre-juego, la cantidad de polígonos del modelo determinara la calidad del mismo y será un aspecto fundamental para el redimiendo del juego.

  Al tratarse de un videojuego para plataformas móviles, la optimización del juego es muy importante, la cantidad de recursos necesarios para el funcionamiento del videojuego dependerá en gran parte de la cantidad de polígonos que el CPU (Unidad central de procesamiento) y GPU (Unidad de procesamiento gráfico) del dispositivo móvil tengan que procesar, por lo tanto para el diseño de los modelos poligonales se utilizó una cantidad de polígonos apropiada sin perder la calidad del producto final como se puede observar en la ***Figura 4.33***.

Figura 4.33: Modelado del personaje
Elaborado por: Hussein Rahman

- **Mapeado UV (unwrapping).-** El mapeado UV permite texturizar al modelo para darle un aspecto mas realista y profesional, primeramente se realiza el mapeado UV desde blender, lo que genera una imagen que servirá de plantilla para crear la textura.

  Utilizando Gimp se diseña la textura y finalmente se la agrega al modelo obteniendo el resultado que se muestra en la ***Figura 4.34***.

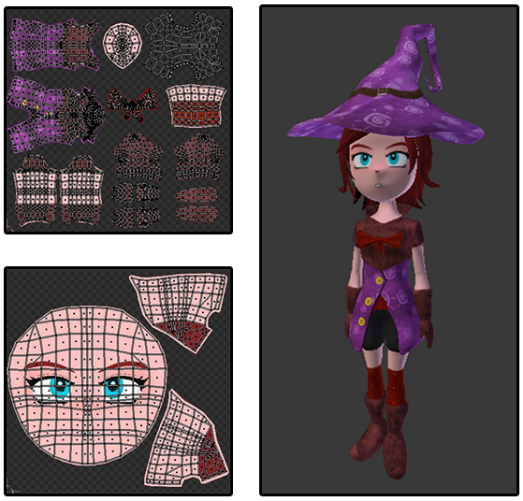


Figura 4.34: Texturización del personaje
Elaborado por: Hussein Rahman

- **Rigging.-** Con el personaje creado y texturizado, el siguiente paso consiste

en añadir los “huesos” al modelo para luego crear los movimientos y las animaciones del personaje, en la ***Figura 4.35*** se muestra el resultado de esta tarea.

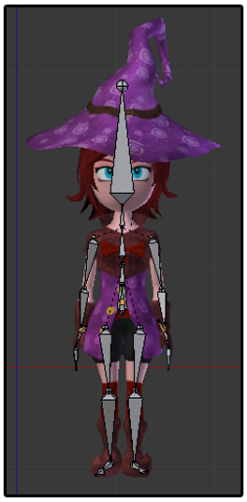


Figura 4.35: Rigging del personaje
Elaborado por: Hussein Rahman

T05: Animaciones del personaje

El objetivo de esta tarea es crear animaciones para el personaje diseñado en la tarea anterior, se crearán un total de 5 animaciones para el movimiento de los huesos y 2 animaciones faciales (ojos abiertos y pestañeo).

En la ***Figura 4.36*** se puede observar las animaciones creadas para el personaje.

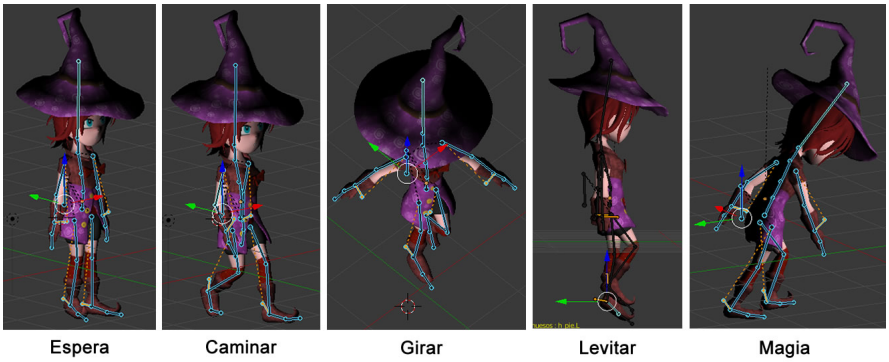


Figura 4.36: Animaciones del personaje
Elaborado por: Hussein Rahman

La clase encargada de ejecutar estas animaciones se llama ***"PlayerAnimations.cs"***, utilizando el componente ***"Animation"*** y el método ***"CrossFade"*** es posible reproducir las animaciones obteniendo una transición suave entre el final de una animación y el inicio de otra, todo esto se realiza desde el método ***"Update"*** propio de *Unity* 3D, el cual se ejecuta en cada ***frame*** del juego.

Adicionalmente se creó el método ***"OpenCloseEyes"*** para animar el rostro del personaje, en la ***Figura 4.37*** se puede observar todo lo anteriormente descrito.

```
1  void Update(){
2    remiMesh.GetComponent<Animation> () [anim].speed = 1f;
3    if (numAnim == 1){
4      anim = "espera";
5      remiSubmeshWings.SetActive (true);
6    }else if (numAnim == 2){
7      anim = "caminar_normal";
8    }
9    //mas animaciones...
10   remiSubmeshWings.SetActive (true);
11 }
12   remiMesh.GetComponent<Animation> ().CrossFade (anim);
13   OpenCloseEyes ();
14 }
```

Figura 4.37: Método update de la clase "PlayerAnimations"
Elaborado por: Hussein Rahman

T06: Interfaz de comandos

El objetivo de esta tarea es crear una interfaz que permita ingresar, visualizar y ejecutar los comandos desde la pantalla. Para el cumplimiento de esta tarea se han creado 2 subtareas.

- **Diseñó de la interfaz de comandos.-** Esta interfaz esta dividida en 2 paneles, el de la derecha es el panel de comandos encargado de mostrar al jugador los comandos disponibles, mientras que a la izquierda se encuentra el panel de ejecución dividiéndose en 3 partes:
  - **Botón Run.-** Encargado de ejecutar las acciones ingresadas. en el subpanel consola
  - **Subpanel consola.-** En este subpanel se visualizará los comandos que fueron ingresados por el jugador y están listos para ser ejecutados.
  - **Botón Borrar.-** Permite borrar el ultimo comando ingresado en la consola.

En la ***Figura 4.38*** se puede visualizar la estructura y los componentes de la interfaz de comandos.

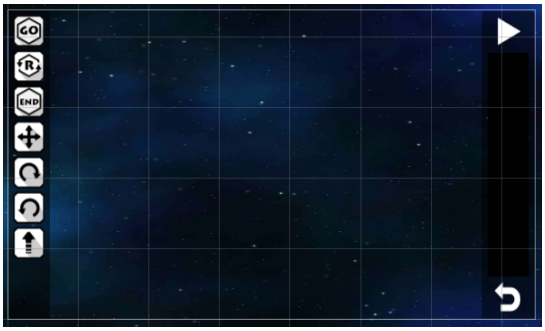


Figura 4.38: Interfaz de comandos
Elaborado por: Hussein Rahman

- **Programación de la interfaz de comandos.-** La clase encargada de relacionar la interfaz creada con los comandos de la clase ***"CreateCode.cs"*** se llama ***"CodePanel.cs"*** y su método principal es ***"AddFichaPrefab"*** encargado de crear un instancia de un objeto prefab para visualizarlo en el subpanel consola según el comando ingresado.

  En la ***Figura 4.39*** se muestra el método antes mencionado y su funcionamiento.

```
1  public void AddTokenPrefab(GameObject panel, Sprite sp, string action, string
       value){
2    GameObject token = (GameObject)Instantiate (prefabToken);
3    token.transform.GetChild(0).GetComponent<Text>().text = value;
4    token.GetComponent<Image> ().sprite = sp;
5    if (spFor==true) {
6      token.GetComponent<GridLayoutGroup> ().childAlignment = TextAnchor.
         MiddleCenter;
7    } else {
8      token.GetComponent<GridLayoutGroup> ().childAlignment = TextAnchor.
         UpperRight;
9    }
10   token.GetComponent<RectTransform>().SetParent(panel.transform);
11   code.AddAction (action);
12 }
```

Figura 4.39: Método "AddTokenPrefab"
Elaborado por: Hussein Rahman

T07: Meta del juego

El objetivo de esta tarea es crear un objeto dentro del mapa que establezca la meta temporal para el juego cuando el jugador colisione con este objeto, para tal

propósito se diseño un modelo del NPC (gato) mostrado según el boceto diseñado en la fase pre-juego.

Al NPC (gato) contendrá un script con la clase ***"ExitTrigger.cs"*** junto con el método ***"OnTriggerEnter"*** se encargará de realizar una determinada acción cuando un objeto colisione con el NPC (gato), por lo tanto cuando el personaje colisione con el NPC (gato) se indicará que el nivel fue completado exitosamente. En la ***Figura 4.40*** se detalla el funcionamiento del método ***"OnTriggerEnter"***.

```
1 void OnTriggerEnter(Collider Col)
2 {
3   if (Col.tag == "Player" && LevelGenerator.numAlmas<=0){
4     LevelGenerator.exit = true;
5     print ("level complete");
6     Destroy (gameObject); //Se destruye la meta
7   }
8 }
```

Figura 4.40: Método "OnTriggerEnter"
Elaborado por: Hussein Rahman

Entrega funcional de la iteración 1

Al finalizar el desarrollo de la iteración 1 todas las tareas fueron completadas entregando la primera versión de videojuego el día 17 de julio de 2016 tal como se había planificado.

El resultado del videojuego en ejecución se puede apreciar en la ***Figura 4.41***, el personaje se encuentra ubicado en el mapa, la interfaz inicial es totalmente funcional con la posibilidad de mover al personaje utilizando los comandos de la izquierda junto con los bloques Go y R, el personaje puede colisionar con la meta para indicar el final del nivel.

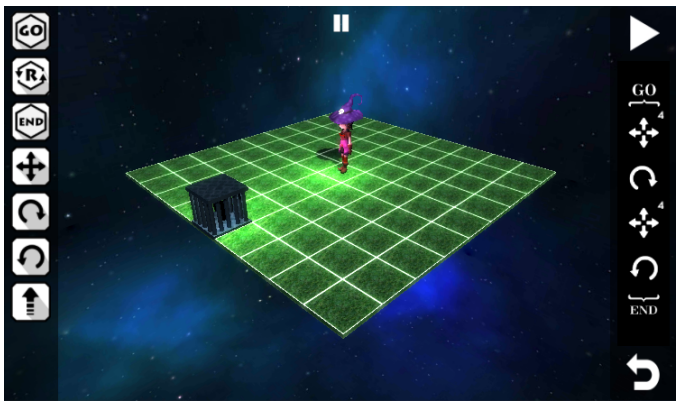


Figura 4.41: Entrega de la primera versión del juego
Elaborado por: Hussein Rahman

Desarrollo de la iteración 2

El objetivo de esta iteración es obtener la segunda versión del videojuego, la fecha de revisión de la iteración será el día 31 de julio de 2016, en esta iteración se desarrollarán las siguientes historias de usuario.

ID	Descripción
HU11	Como jugador quiero que existan enemigos variados dentro del mapa.
HU12	Como jugador quiero un mapa con elevaciones y una estructura variada.
HU13	Como jugador quiero que la cámara posea una vista aérea y en tercera persona.
HU21	Como programador quiero controlar las colisiones del jugador con los objetos del mundo.
HU22	Como jugador quiero visualizar una pantalla de pausa que permita reiniciar o salir de la partida.
HU23	Como programador quiero mostrar una notificación en la pantalla si los comandos se ingresan de forma incorrecta.

Tabla 4.20: Historias de usuario a implementarse en la Iteración 2

Elaborado por: Hussein Rahman

Se han generado ***tarjetas de tarea***s en base a las ***historias de usuarios*** seleccionadas para la ***iteración 2***, las tareas fueron auto-asignadas al miembro del equipo DAV, el contenido de las tarjetas se encuentran en el ***Anexo A.***

A continuación se resumen las tareas y subtareas realizadas para la iteración 2.

T08: Diseño del enemigo

El objetivo de esta tarea es diseñar al enemigo en base a los bocetos realizados en la fase de pre-juego, se realizó el mismo proceso de la tarea ***T04***. En la ***Figura 4.42*** se puede observar el resultado final.

Figura 4.42: Diseño del enemigo
Elaborado por: Hussein Rahman

T09: Generación de niveles

El objetivo de esta tarea es crear elevaciones en las casillas del mapa para crear niveles mas diversos y con un mayor grado de dificultad, para lo cual se ha creado la clase ***"LevelGenerator.cs"***.

El método principal de esta clase se llama ***"CloneTile"*** encargado de clonar una casilla según su nombre de coordenada, este método es la base para crear columnas, obstáculos, ubicar enemigos, ubicar powerUps, etc.

Para crear elevaciones se utiliza el método ***"CreateColumn"*** utilizando casillas clonadas con una elevación mayor, de esta forma se crean columnas con diferentes alturas.

En la ***Figura 4.43*** se muestra la estructura del método ***"CreateColumn"***, el parámetro ***col*** de tipo string con longitud de 3 caracteres determina la coordenada X, Z y su respectiva altura por ejemplo si se ingresa el string "023", el valor de cada carácter seria: 0 = coordenada X, 2 = coordenada Z y 3 = altura.

```
private void CreateColumn(string col){
  coordinate = col[0].ToString() + "" + col[1].ToString();
  columnHeight = int.Parse(col [2].ToString());
  for (int i = columnHeight; i >= 1; i--) {
    CloneTile (SearchTile (coordinate),"", "column",i, 1f,1f,1f,RandomTexture
        ());
  }
  //Destruir la casilla original
  if (SearchTile (coordinate).transform.localScale == new Vector3 (1f, 0.1f,
      1f)) {     Destroy(SearchTile (coordenada));
  }
}
```

Figura 4.43: Método "CreateColumn"
Elaborado por: Hussein Rahman

Para complementar esta tarea es necesaria la generación aleatoria de las columnas, para lograr esto se creó la clase ***"WorldGenerator.cs"*** la cual usará todos los métodos creados en la clase ***"LevelGenerator.cs"*** con la ayuda de un valor aleatorio entre rangos de valores específicos. En la ***Figura 4.44*** se muestra uno de varios métodos que permitirán la creación aleatoria de los niveles con 3 grados de dificultad.

```
public void RandomColumns(int level){
  int[,] mat;
  mat = new int[main.nMatrix,main.nMatrix];
  for(int f = 0; f < mat.GetLength(0); f++) {
    for (int c = 0; c < mat.GetLength(1); c++){
      randomCreate = Random.Range (0, level);
      if (randomCreate == 0) {
        h = Random.Range (1, 3); //columnas de 1 a 3 bloques de altura
        cod_columns += f+""+c+""+h+".";
      }
    }
  }
  cod_columns = cod_columns.TrimEnd ('.');
}
```

Figura 4.44: Método "RandomColumns"
Elaborado por: Hussein Rahman

T10: Movimiento de cámaras

El objetivo de esta tarea es mover la cámara con vista aérea y en tercera persona utilizando el touch del dispositivo móvil, es decir desplazamientos del dedo sobre la pantalla táctil, para este propósito se creó la clase ***"CameraOrbit.cs"*** y el método ***"RotateWithMouse"*** con este método se logra que el juego detecte el movimiento del mouse en caso de tratarse de un PC o el movimiento swipe en caso de un dispositivo táctil, como se muestra en la ***Figura 4.45*** .

```
1  private void RotateWithMouse(int value){
2    if (Input.GetMouseButton (value)) {
3      x += Input.GetAxis ("Mouse X") * xSpeed * 0.02f;
4      y -= Input.GetAxis ("Mouse Y") * ySpeed * 0.02f;
5      rot = Quaternion.Euler (y, x, 0);
6      pos = rot * new Vector3 (0.0f, 0.0f, -distance) + target.position;
7      mainCam.transform.rotation = rot;
8      mainCam.transform.position = pos;
9    }
10 }
```

Figura 4.45: Método "RotateWithMouse"
Elaborado por: Hussein Rahman

T11: Control de colisiones del personaje

Para controlar las colisiones del personaje con los diferentes objetos del mundo, se han creado 3 sensores como se puede observar en la ***Figura 4.46***, el sensor frontal encargado de detectar objetos ubicados frente al personaje; el sensor del piso encargado de detectar si el personaje esta sobre el piso; y el sensor de vidas permite detectar si el personaje esta colisionando con un enemigo. Ademas se ha creado un método ***"CollisionDetected"*** dentro de la clase ***"ControlPlayer.cs"*** este método permite detectar una colisión y mostrar una alerta en pantalla tal y como se muestra en la ***Figura 4.47***.

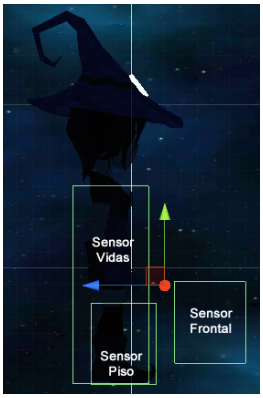


Figura 4.46: Sensores del personaje
Elaborado por: Hussein Rahman

```
1  public void CollisionDetected(){
2    wrongMotion = true;
3    colPlayer.GetComponent<BoxCollider> ().isTrigger = false;
4    Initialize ();
5    code.Initialize ();
6    error.ActivatePanel (3);
7    StartCoroutine (waitforBoxTriggerTrue());
8  }
```

Figura 4.47: Método "CollisionDetected"
Elaborado por: Hussein Rahman

T12: Panel flotante de pausa

El objetivo de esta tarea es crear un panel flotante que se visualice cuando el juego este pausado, desde el panel se podrán realizar acciones como: reiniciar el juego, visualizar la ayuda y salir al menú principal.

Se creó la clase ***"PausePanel.cs"*** y el método ***"ActivatePausePanel"*** encargado de mostrar y ocultar el panel, en la ***Figura 4.48*** se puede visualizar el diseño del panel.

Figura 4.48: Panel flotante de pausa
Elaborado por: Hussein Rahman

T13: Notificación de errores

Para el cumplimiento de esta tarea se creó la clase ***"MessageNotifications.cs"*** y el método ***"TypeMessage"*** el cual se encarga de mostrar una notificación el pantalla con el tipo de error que se ha generado deteniendo momentáneamente la ejecución del juego. Existen 3 tipos de errores y estos son:

- **Error de movimiento.-** Es causado cuando el jugador quiere moverse a un numero de casilla superior al máximo del mapa.
- **Error de acción.-** Cuando se ingresan incorrectamente las acciones, por ejemplo: olvidar iniciar con GO para un bloque de acciones .
- **Error de colisión.-** Si el jugador colisiona contra una columna o un enemigo.

  En la **Figura 4.49** se puede observar con mas detalle la estructura del método ***"TypeMessage"***.

```
1  public void TypeMessage(int value){
2    if (value == 1) {
3      imgTitle.sprite = sp.LoadImageTitle("mov_error");
4      TypeButtons (1);
5    } else if (value == 2) {
6      imgTitle.sprite = sp.LoadImageTitle("action_error");
7      TypeButtons (1);
8    } else if (value == 3) {
9      imgTitle.sprite = sp.LoadImageTitle("coll_error");
10     TypeButtons (1);
11   }
12 }
```

Figura 4.49: Método "TypeMessage"
Elaborado por: Hussein Rahman

Entrega funcional de la Iteración 2

Al finalizar el desarrollo de la iteración 1 todas las tareas fueron completadas, entregando la segunda versión de juego el día 31 de julio de 2016 tal como se había planificado.

El resultado del videojuego en ejecución se puede apreciar en la ***Figura 4.50***, con elevaciones aleatorias sobre el mapa,el diseño de los enemigos y el movimiento de las cámaras, etc.

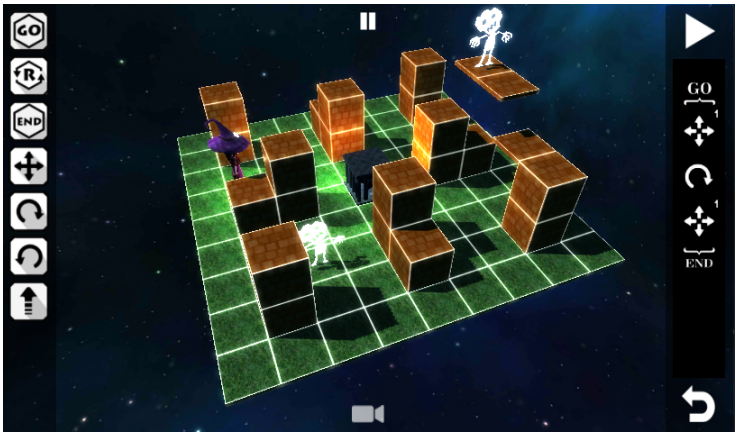


Figura 4.50: Entrega de la segunda versión del juego
Elaborado por: Hussein Rahman

Desarrollo de la iteración 3

El objetivo de esta iteración es obtener la tercera versión del videojuego, la fecha de revisión de la iteración será el día 14 de agosto de 2016, en esta iteración se desarrollará las siguientes historias de usuario.

ID	Descripción
HU17	Como jugador quiero que el personaje posea acciones avanzadas y pueda tomar decisiones.
HU18	Como programador quiero tener comandos avanzados para ejecutar las acciones avanzadas del personaje.
HU19	Como diseñador quiero que la decoración del mundo este acorde a los personajes y la historia del juego.
HU25	Como jugador quiero que el personaje tenga powerUps para mejorar sus habilidades.
HU27	Como jugador quiero que el personaje pierda el nivel si sus vidas llegan a 0.
HU28	Como sonidista quiero efectos de sonidos y música de fondo para el videojuego

Tabla 4.21: Historias de usuario a implementarse en la Iteración 4

Elaborado por: Hussein Rahman

Se han generado ***tarjetas de tareas*** en base a las ***historias de usuarios*** seleccionadas para la ***iteración 3***, las tareas fueron auto-asignadas al miembro del equipo DAV, el contenido de las tarjetas se encuentran en el ***Anexo A.***

T14: Acciones avanzadas del personaje

En la tarea ***T02*** se crearon acciones básicas para el personaje ***(mover, rotar y volar)*** las cuales permitieron un desplazamiento básico dentro del mapa. En esta tarea se crearán acciones avanzadas de desplazamiento e interacción del personaje con el mundo, estas nuevas acciones son:

- **Portal.-** Esta acción permite desplazarse entre 2 portales del mismo color ubicados en las casillas del mapa, tanto de ida como de vuelta. En la ***Figura 4.51*** se puede apreciar la ubicación de los portales dentro del mapa.

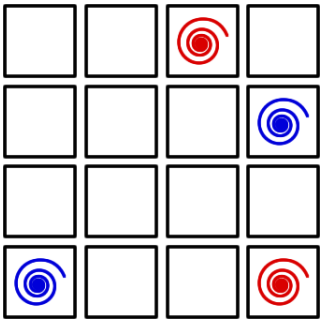


Figura 4.51: Portales
Elaborado por: Hussein Rahman

Para la interacción de los portales se utiliza el sensor frontal de personaje, realizando un cambio de posición entre el portal de origen-destino, En la ***Figura 4.52*** se muestra un fragmento método ***"PortalMotion"*** ubicado dentro de la clase ***"Main.cs"***.

```
1 if (nPortal[0]=='a' && control.PortalActivated==true){
2   endPortal = GameObject.Find ("portalB_"+nPortal[1]);
3   destination = endPortal.transform.position;
4   control.player.transform.position =
5     new Vector3 (destination.x,destination.y,destination.z);
6 }
```

Figura 4.52: Fragmento del método "PortalMotion"
Elaborado por: Hussein Rahman

- **Magia.**- Con la aparición de enemigos dentro del juego, la magia será la única acción para eliminarlos, permitiendo al jugador utilizar dos tipos de magia: light y dark. La magia light eliminará enemigos oscuros, mientras que la magia dark eliminará enemigos de luz. En al ***Figura 4.53*** se muestra el diseño de los botones asociados a estas dos nuevas acciones

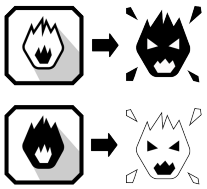


Figura 4.53: Magia y enemigos
Elaborado por: Hussein Rahman

El método encargado de activar la magia se encuentra en la clase ***"CreateCode.cs"*** la misma que contiene todas las acciones del personaje, en la ***Figura 4.54*** se muestra el método ***"UseMagic"***.

```
public void UseMagic(int tipo){
  if (type == 0 && Main.colEnemy && powerActive) {
    Main.magicType = "light";
  } else if (type == 1 && Main.colEnemy && powerActive) {
    Main.magicType = "dark";
  } else if (Main.magicType == "none"){
    action += 1;
  }
}
```

Figura 4.54: Método "UseMagic"
Elaborado por: Hussein Rahman

Adicionalmente fue necesario crear una clase que detecte si el enemigo se encuentra frente al personaje y determinar si la magia usada es la correcta para eliminarlo, la clase para realizar esta tarea es ***"EnemyTrigger.cs"***.

T15: Comandos avanzados

Los comandos avanzados permitirán que el personaje posea más opciones para interactuar con el escenario, se agregarán comandos para usar portales y magia, ademas un nuevo bloque de código llamado ***bloque if***.

Los comandos avanzados tendrán una longitud de 3 a 6 caracteres, se añadirá un punto y el valor si requiere de un parámetro. En la ***Tabla 4.22*** se detalla la nomenclatura de los comandos avanzados, los parametros ***"dark"*** y ***"light"*** se pueden acordar con las letras ***"d"*** y ***"l"*** respectivamente.

Comando	Significado de los comandos
use.portal	El comando **use.portal** activará el portal de origen y permitirá desplazarse al portal de destino.
magic.dark magic.light	El comando **magic.dark/magic.light** permite activar magia de tipo dark o light.
if	El comando **if** establece el inicio de un bloque if.
enemy.dark enemy.light	El comando **enemy.dark/enemy.light** es una condición que determina si el personaje esta frente a un enemigo tipo negro o blanco.
then	El comando **then** ejecuta las acciones si la condición es verdadera.
else	El comando **else** ejecuta las acciones si la condición es falsa.
endif	El comando **endif** establece el final del bloque if.
null	El comando **null** es un comando no jugable que no realiza ninguna acción.

Tabla 4.22: Nomenclatura de comandos avanzados

Elaborado por: Hussein Rahman

- **Bloque if.-** Este nuevo bloque permitirá tomar uno de dos posibles resultados según una condición planteada, por ejemplo si un enemigo cambia entre el tipo negro y blanco cada cierto tiempo el jugador no sabrá con exactitud que tipo de magia utilizar en ese instante.

  Con el **bloque if** se podrá detectar el tipo de enemigo se encuentra frente al jugador y realizar una u otra acción según el cumplimiento de la condición "colisión con enemigo". En versiones futuras del juego se pretende crear mas condiciones para expandir el uso de este bloque, por ejemplo una condición que determine si una puerta esta abierta o cerrada para el uso de una llave. En la ***Figura 4.55*** se puede observar la estructura del bloque if con mas detalle

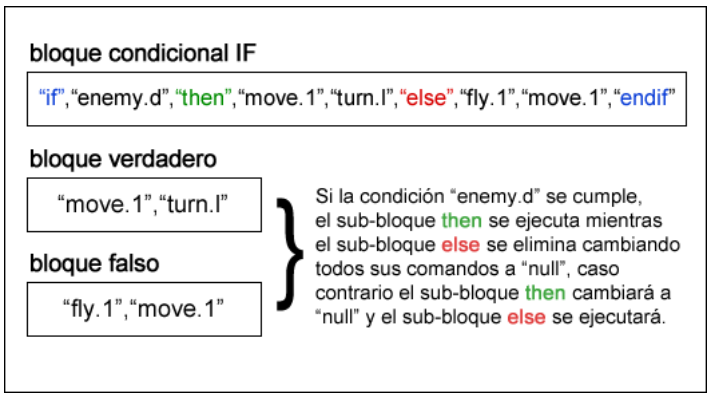


Figura 4.55: Estructura del bloque if
Elaborado por: Hussein Rahman

El método encargado de ejecutar al ***bloque if*** se encuentra en la clase ***"CreateCode.cs"*** y dentro del método ***"CompileCode"***. En la ***Figura 4.56*** se muestra un extracto de la primera comprobación que realiza cuando el enemigo es de tipo blanco.

```
1 //Si el jugador se encuentra frente a un enemigo tipo Light
2 else if (t [0] == "then" && ifCom==true && eLight == true) {
3   thenCom = true;
4   if (Main.colLight == true) {
5     maxIndex = blocks [num].Length;
6     minIndex = Array.IndexOf (blocks [num],"else")+1;
7     while (minIndex < maxIndex) {
8       blocks[num][minIndex]="null";
9       minIndex = minIndex + 1;
10    }
11    control.Action += 1;
12    minIndex = 0;
13    maxIndex = 0;
14  }
15 }
```

Figura 4.56: Fragmento del bloque IF primera parte
Elaborado por: Hussein Rahman

Si la primera condición no se cumple quiere decir que el enemigo es de tipo negro lo que anula los comandos de la primera condición reescribirlos con la acción ***null*** tal y como se observa en la ***Figura 4.57***.

```
1   } else if (Main.colLight == false) {
2     minIndex = Array.IndexOf (blocks [num],"then");
3     maxIndex = Array.IndexOf (blocks [num],"else");
4     while (minIndex < maxIndex) {
5       blocks[num][minIndex]="null";
6       minIndex = minIndex + 1;
7     }
8     control.Action += 1;
9     minIndex = 0;
10    maxIndex = 0;
11  }
12  ifCom = false;
13  eLight = false;
14 }
```

Figura 4.57: Fragmento del bloque IF segunda parte
Elaborado por: Hussein Rahman

En el código se muestra dos resultados posibles según la condición, el resultado que no se cumpla se reescribirá las acciones a ***null*** lo que impedirá su ejecución, llegándose a ejecutar solo uno de los resultados de la condición.

T16: Decoración del mundo

La decoración del mundo consta de objetos 3D e imágenes 2D para crear el escenario en base a los bocetos realizados en la fase ***pre-juego***. El uso de imágenes 2D con transparencias ayuda a simular objetos decorativos y principalmente optimizar el juego ahorrando el uso de recursos. En la ***Figura 4.58*** se muestra el resultado final de la decoración.

Figura 4.58: Decoración del mundo
Elaborado por: Hussein Rahman

T17: Gestión de powerUps

Los ***powerUps*** son objetos recolectables que permitirán al personaje desbloquear el arma para eliminar a los enemigos del juego, con el arma desbloqueada se podrá utilizar las acciones de magia ***dark*** y ***light***. La clase encargada de esta tarea se llama ***"PowerUp.cs"***.

En la ***Figura 4.59*** se muestra el diseño de los objetos powerUps.

Figura 4.59: Diseño de powerUps
Elaborado por: Hussein Rahman

T18: Gestión de vidas

El objetivo de esta tarea es gestionar las vidas del personaje, se iniciará con 3 vidas; si se colisiona con un enemigo se perderá una vida; y si las vidas llegan a 0 se terminará la partida. La clase encargada de visualizar las vidas restantes en la pantalla se llama ***"LifeManager.cs"*** y la disminución de vidas se lo hace desde la clase ***"EnemyTrigger.cs"*** tal y como se muestra en la ***Figura 4.60***.

```
1 void OnTriggerEnter(Collider Col){
2   if(Col.tag == "lifeSensor"){
3     LifeManager.lifeCount -= 1;
4     LifeManager.loseLife = true;
5   }
6 }
```

Figura 4.60: Clase "EnemyTrigger"
Elaborado por: Hussein Rahman

T19: Música de fondo

El objetivo de esta tarea es añadir música de fondo para el juego. Se ha creado la clase ***"SoundManager.cs"*** junto con el método ***"SelectTrack"*** para la selección de las pistas al inicio del juego, el método encargado de la reproducción aleatoria se llama ***"RandomPlay"***, ambos métodos se pueden observar en las ***Figuras 4.61***

y ***4.62*** respectivamente.

```
1  public void SelectTrack(int num){
2    if (num == 1) {
3      backgroundAudio.GetComponent<AudioSource> ().clip = LoadMusic ("sonidos/
           music_1");
4    } else if (num == 2) {
5      backgroundAudio.GetComponent<AudioSource> ().clip = LoadMusic ("sonidos/
           music_2");
6    } else if (num == 3) {
7      backgroundAudio.GetComponent<AudioSource> ().clip = LoadMusic ("sonidos/
           music_3");
8    } else if (num == 4) {
9    backgroundAudio.GetComponent<AudioSource> ().clip = LoadMusic ("sonidos/
         music_4");
10   }
11 }
```

Figura 4.61: Método "SelectTrack"
Elaborado por: Hussein Rahman

```
1  public void RandomPlay(){
2    backgroundAudio.GetComponent<AudioSource> ().Stop ();
3    randomNum = Random.Range (1, 4);
4    print ("track: " + randomNum);
5    SelectTrack (randomNum);
6    backgroundAudio.GetComponent<AudioSource> ().Play ();
7  }
```

Figura 4.62: Método "RandomPlay"
Elaborado por: Hussein Rahman

Entrega funcional de la Iteración 3

Al finalizar el desarrollo de la iteración 3, todas las tareas fueron completadas entregando la tercera versión de juego el día 14 de agosto de 2016 tal como se había planificado.

El resultado de la tercera versión del juego funcional se puede apreciar en la Figura ***4.63***, con la inclusión de comandos avanzados, el ***bloque if***, la gestión de ***vidas***, gestión de ***powerUps*** y la decoración del mundo, etc.

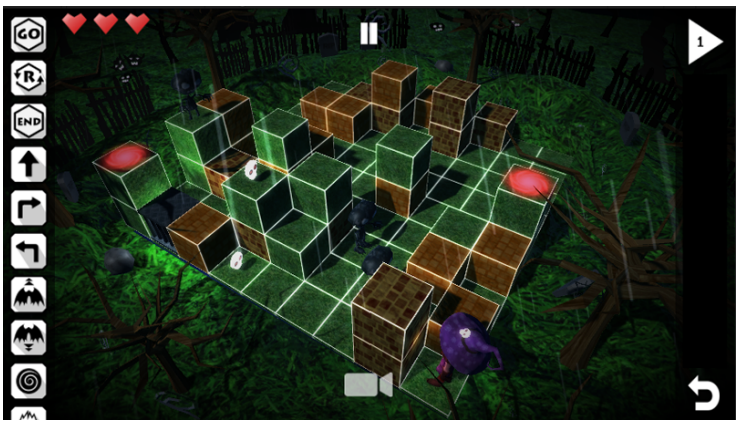


Figura 4.63: Entrega de la tercera versión del juego
Elaborado por: Hussein Rahman

Desarrollo de la iteración 4

El objetivo de esta iteración es obtener la cuarta versión del videojuego, la fecha de revisión de la iteración será el día 28 de agosto de 2016, en esta iteración se desarrollará las siguientes historias de usuario.

ID	Descripción
HU04	Como jugador quiero visualizar las instrucciones del juego por medio de algún tutorial
HU06	Como jugador quiero visualizar un menú para seleccionar los mundos y niveles.
HU08	Como jugador quiero visualizar una pantalla que muestre las puntuaciones obtenidas al finalizar el nivel.
HU16	Como jugador quiero que el juego tenga al menos dos idiomas para escoger
HU01	Como Jugador, quiero visualizar el menú principal del videojuego
HU02	Como jugador quiero antes de iniciar el juego ver la historia y las motivaciones del los personajes.
HU05	Como jugador quiero visualizar una pantalla de créditos.
HU07	Como jugador quiero que el personaje tenga diferentes atuendos o vestimentas.
HU03	Como jugador quiero tener la posibilidad de cambiar la calidad gráfica del juego

Tabla 4.23: Historias de usuario a implementarse en la Iteración 3

Elaborado por: Hussein Rahman

Se han generado ***tarjetas de tareas*** en base a las ***historias de usuarios*** seleccionadas para la ***iteración 4***, las tareas fueron auto-asignadas al miembro

del equipo DAV, el contenido de las tarjetas se encuentran en el ***Anexo A.***

T20: Interfaz de ayuda

El objetivo de la siguiente tarea es crear una pantalla de ayuda que permita al jugador entender la mecánica, controles y objetivos del juego. Para cumplir con esta tarea se utiliza la clase ***"MenuControl.cs"*** la cual contiene métodos para activar y desactivar las diferentes interfaces de usuario.

En la ***Figura 4.64*** se puede observar la interfaz de ayuda con los diferentes tópicos que el jugador puede consultar.

Figura 4.64: Pantalla de ayuda
Elaborado por: Hussein Rahman

T21: Interfaz gráfica de selección de niveles

El objetivo de esta tarea es crear una interfaz de usuario que permita seleccionar el nivel de dificultad, el método ***"ShowLevels"*** se encarga de activar y desactivar esta interfaz, se encuentra dentro de la clase ***"MenuControl.cs"***.

Para seleccionar el nivel de dificultad se utiliza el método ***"NumLevel"*** encargado de almacenar el número de dificultad (1 = fácil, 2 = medio y 3 difícil) utilizando ***PlayerPrefs*** , el número almacenado solo cambiará cuando se seleccione otro nivel de dificultad. En la ***Figura 4.65*** se puede observar el funcionamiento del método.

```
public void NumLevel(int num){
  PlayerPrefs.SetInt ("numLevel",num);
  menu.ActivarLoading();
}
```

Figura 4.65: Método "NumLevel"
Elaborado por: Hussein Rahman

T22: Interfaz gráfica de puntuaciones

El objetivo de esta tarea es crear una interfaz que muestre la puntuación obtenida al completar el nivel, evaluando el número de ***intentos*** usados. Para este propósito se creó la clase ***"StartEndGame.cs"***, la cual contiene el método ***"ActivatePumkims"*** encargado de activar o desactivar los sprites que representan los puntos ganados tal y como se muestra en ***Figura 4.66***.

El segundo método se llama ***"ShowScore"*** encargado de mostrar el puntaje obtenido de acuerdo al número de ***intentos*** usados y al nivel de dificultad actual, del nivel dependerá el factor multiplicativo (fácil x1, medio x2 y difícil x3), este factor multiplicará el puntaje total obtenido. Un fragmento de este método se puede observar en la ***Figura 4.67***.

```
public void ActivarPumkims(bool p1, bool p2, bool p3){
  temp = scorePanel.transform.GetChild (0).gameObject;
  temp.transform.GetChild (0).gameObject.SetActive (p1);
  temp.transform.GetChild (1).gameObject.SetActive (p2);
  temp.transform.GetChild (2).gameObject.SetActive (p3);
}
```

Figura 4.66: Método "TotalTime"
Elaborado por: Hussein Rahman

```
public void ShowScoreTime(){
  factorMultiplicador = PlayerPrefs.GetInt ("numLevel");
  score.GetFactorValue (factorMultiplicador);
  if (Main.intentos == 0) {
    ActivarPumkims (true,false,false);
  }else if (Main.intentos == 1){
    ActivarPumkims (true,true,false);
  }else if (Main.intentos == 2){
    ActivarPumkims (true,true,true);
  }
  puntosPartida = (Main.intentos + 1) * factorMultiplicador;
}
```

Figura 4.67: Método "ShowScore"
Elaborado por: Hussein Rahman

T23: Interfaz gráfica de selección de idioma

El objetivo de esta tarea es permitir al jugador seleccionar el idioma (ingles o español) al iniciar el juego, para tal propósito se creó la clase ***"LoadLanguage.cs"*** con dos métodos como se puede observar en la ***Figura 4.68***, estos métodos permiten cargar los **sprites** de los títulos, los mensajes y tutoriales en el idioma seleccionado mediante un código (en=ingles, es=español) que se guarda en la variable ***LANG***.

```
1 public Sprite LoadImageTitle(string name){
2   lang = PlayerPrefs.GetString ("LANG");
3   return LoadSP ("titulos/"+lang+"/"+name+"");
4 }
5 public Sprite LoadSP(string path){
6   return (Sprite) Resources.Load (path, typeof(Sprite));
7 }
```

Figura 4.68: Método "LoadImageTitle"
Elaborado por: Hussein Rahman

T24: Interfaz gráfica del menú principal

El objetivo de esta tarea es crear la pantalla inicial del juego, la misma que tendrá botones para iniciar el juego, salir del juego,configurar la calidad gráfica y ver los créditos. En la ***Figura 4.69*** se muestra el método ***"ShowMenu"*** encargado de activar el menú principal.

```
1 public void ShowMenu(){
2   uiMenu.SetActive(true);
3   uiLevels.SetActive(false);
4   uiStory.SetActive (false);
5   cam.GetComponent<BlurOptimized> ().enabled = false;
6 }
```

Figura 4.69: Método "ShowMenu"
Elaborado por: Hussein Rahman

T25: Interfaz gráfica del storyboard

El objetivo de esta tarea es crear una interfaz para visualizar el storyboard del juego diseñado en forma de cómic para una mejor presentación. El método para activar esta pantalla tiene similitud con el método de la tarea ***T24***.

T26: Interfaz gráfica de créditos e información del juego

Mediante esta interfaz se mostrará información de los miembros del equipos involucrados en el desarrollo del juego, así como referencia de material externo utilizado para el desarrollo como la música. El método para activar esta pantalla tiene similitud con el método de la tarea ***T24***.

T27: Selección del atuendo del personaje

El objetivo de esta interfaz es permitirle al usuario seleccionar el atuendo o vestimenta del personaje antes de iniciar la partida. Se creó la clase ***"LoadCharacter.cs"*** junto con los métodos ***"SelectCharacter"*** y ***"CharTextures"*** ambos encargados de cargar al personaje y sus respectivas texturas, en la ***Figura 4.70*** se muestra un pequeño fragmento del primer método.

```
1  public void SelectCharacter(int value){
2    temp = GameObject.Find ("remi_player");
3    if (temp != null) {
4      Destroy (temp);
5    }
6    if (value == 1) {
7      temp = LoadPrefabCharacter ("characters/remi_witch");
8    } else if (value == 2) {
9      temp = LoadPrefabCharacter ("characters/remi_christmas");
10   }
11   remi = (GameObject)Instantiate (temp);
12   remi.name = "remi_player";
13   typeClothes = value;
14 }
```

Figura 4.70: Fragmento del método "SelectCharacter"
Elaborado por: Hussein Rahman

T28: Interfaz gráfica de configuración del juego

Mediante esta interfaz se podrá cambiar la calidad gráfica del juego permitiendo establecer una relación calidad/rendimiento al momento de iniciar el juego. Se creó la clase ***"GameSettings.cs"*** con dos métodos para guardar la configuración gráfica del juego como se puede observar en la ***Figura 4.71***.

```
1 public void ChangeQuality(){
2   qualityValue = (int)qualitySlider.value;
3 }
4 public void SaveChanges(){
5   QualitySettings.SetQualityLevel (qualityValue, true);
6   PlayerPrefs.SetInt ("qvalue",qualityValue);
7   PlayerPrefs.SetFloat ("bgVolume",bgVolumeValue);
8   PlayerPrefs.SetString("newSettings","true");
9 }
```

Figura 4.71: Métodos de la clase “GameSettings.cs”
Elaborado por: Hussein Rahman

Entrega funcional de la Iteración 4

Al finalizar el desarrollo esta iteración todas las tareas fueron completadas entregando la cuarta versión de juego el día 18 de agosto de 2016 tal como se había planificado.

El resultado de la cuarta y ultima versión del juego es totalmente funcional con todas las características que se planificaron durante las historias de usuario tal y como se puede apreciar en la ***Figura 4.72***.

Figura 4.72: Entrega de la cuarta versión del juego
Elaborado por: Hussein Rahman

4.4.3. Fase Post-juego

Una vez terminado el desarrollo de todas las iteraciones tal y como se planificó en las fases anteriores, se obtuvo un videojuego totalmente funcional y listo para ser publicado en una tienda online. Estará disponible de forma gratuita para llegar a un mayor número de jugadores, lo que permitirá recolectar opiniones, criticas y valoraciones para el mejoramiento continuo del juego en futuras versiones.

Evaluación de la tienda online

Actualmente existen muchos sitios webs para publicar videojuegos independientes, entre los principales para la plataforma móvil Android se encuentra la ***PlayStore*** de ***Google***, y en segundo lugar ***Amazon AppStore***, aunque con menos popularidad que la primera tienda. Adicionalmente a estas dos tiendas se analizará una tercera tienda menos conocida, llamada ***Gamejolt*** la cual apareció a principios del 2004 empezando como un servicio gratuito de hosting para juegos independientes.

Para la evaluación de la tienda online se utilizó la escala de evaluación por puntos donde 0 = no cumple, 1 = cumple parcialmente y 2 cumple totalmente. A continuación se describen los criterios a evaluar.

- **Pago no requerido.-** Criterio sobre la posibilidad de publicar el videojuego sin un pago por publicación.
- **Regalías para el desarrollador.-** La tienda ofrece al desarrollador un porcentaje de ganancia en caso de venderse el producto.
- **Multiplataformas.-** La posibilidad de publicar el mismo videojuego en diferentes plataformas.

PlayStore

Característica	**Detalles**	
Inscripción	Requiere de un pago único inicial de $25	0
Regalías	■ Si el juego se vende, el 70 % de la ganancia se queda el desarrollador y 30 % el patrocinador “Google”. ■ Existe restricción por parte de Google para la comercialización de videojuegos en Ecuador.	1
Plataformas	Android	1
Total		2

Tabla 4.24: Evaluación de las características de PlayStore

Elaborado por: Hussein Rahman
Fuente: [35]

Amazon AppStore

Característica	**Detalles**	
Inscripción	No requiere de un pago para la publicación de juegos	2
Regalías	■ Si el juego se vende, el 70 % de la ganancia se queda el desarrollador y 30 % el patrocinador “Amazon”. ■ Solo se puede comercializar los videojuegos en Estados Unidos.	1
Plataformas	Android	1
Total		4

Tabla 4.25: Evaluación de las características de Amazon AppStore

Elaborado por: Hussein Rahman
Fuente: [36]

Gamejolt

Característica	Detalles	
Inscripción	No se requiere de un pago para publicar los juegos	2
Regalías	■ El patrocinador "Gamejolt" no cobra un porcentaje si el juego se vende, gana por la publicidad en la página del juego dándole un 30 % al desarrollador. ■ Aunque Ecuador no esta dentro de los países habilitados para la comercialización, el desarrollador puede utilizar una cuenta de PayPal para recibir el 30 % de la ganancia por publicidad.	2
Plataformas	Android, Windows, Linux, MacOS, Html5, Flash	2
Total		6

Tabla 4.26: Evaluación de las características de Gamejolt

Elaborado por: Hussein Rahman
Fuente: [37]

Resultados de la evaluación de la tienda online

A continuación se detalla un breve análisis sobre la evaluación a las tiendas online (***PlayStore***, ***Amazon AppStore*** y ***Gamejolt***), se descartó la tienda de ***Google*** y ***Amazon*** por razones muy especificas, el primer caso debido al pago de la inscripción pierde sentido si posteriormente el juego no podrá comercializarse en el país, en el caso de ***Amazon*** aunque la publicación es gratuita, no es un buen lugar para que los desarrolladores independientes empiecen a surgir por la falta de apoyo del patrocinador muchos buenos proyectos quedan en el olvido, algo muy distinto ocurre con ***Gamejolt***.

Gamejolt aunque no es tan conocido destaca en varios aspectos frente a sus competidores como se puede observar en la **Tabla 4.26**, las regalías por publicidad sin importar el país, la posibilidad de publicar el mismo juego en múltiples plataformas, pero principalmente por el apoyo que reciben los nuevos desarrolladores al darles un espacio en su pagina principal. Motivo por el cual se a tomado la decisión de publicar el videojuego **Hello Remi** en la tienda antes mencionada.

Publicación del videojuego en la tienda gamejolt

Publicar el juego ***Gamejolt*** es realmente sencillo, solo es necesario crear una cuenta de tipo desarrollador y configurar aspectos básicos de la misma, terminada la creación de la cuenta empieza el proceso de publicación del videojuego.

A continuación se detalla todo el proceso que realizó para la publicación del videojuego:

- **Detalle.-** En esta sección se escribe el nombre del juego, la categoría y el motor de juego utilizado para el desarrollo.

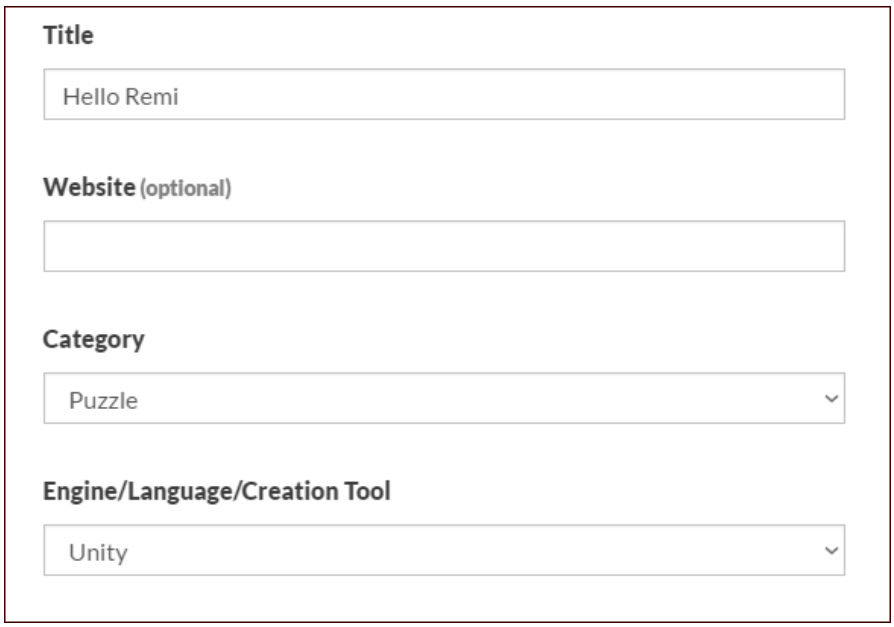


Figura 4.73: Detalle del videojuego en Gamejolt
Elaborado por: Hussein Rahman
Fuente: [38]

- **Descripción.-** En esta sección se escribe una descripción detallada del juego, con la posibilidad de añadir imágenes, gifs animados, videos, etc.

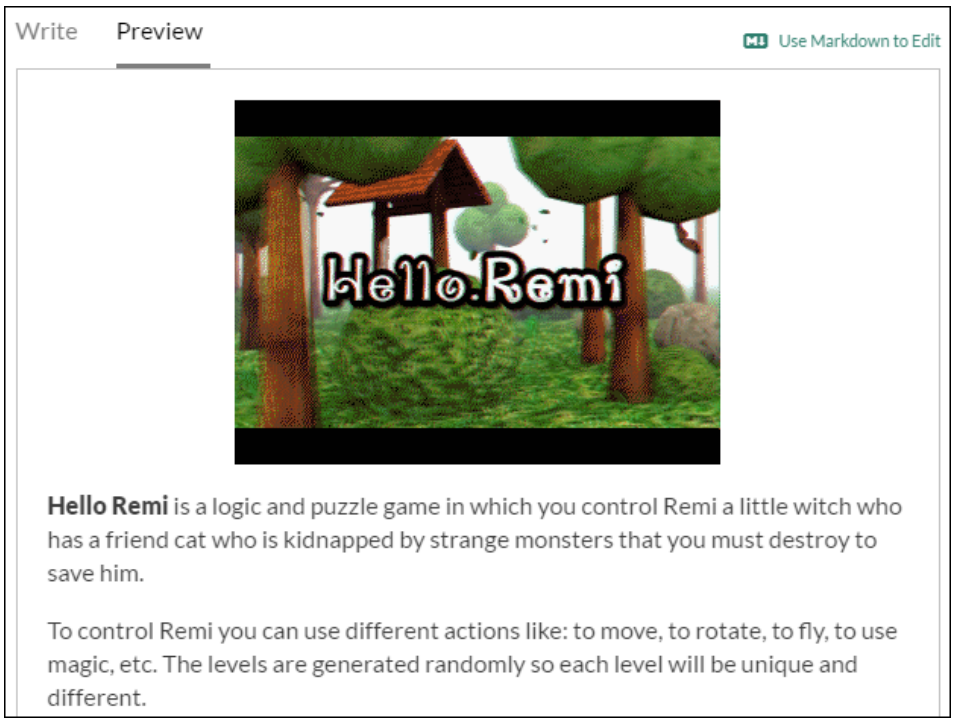


Figura 4.74: Descripción del videojuego en Gamejolt
Elaborado por: Hussein Rahman
Fuente: [38]

- **Público objetivo.-** En esta sección se indica las edades recomendadas para el juego y el tipo de contenido.

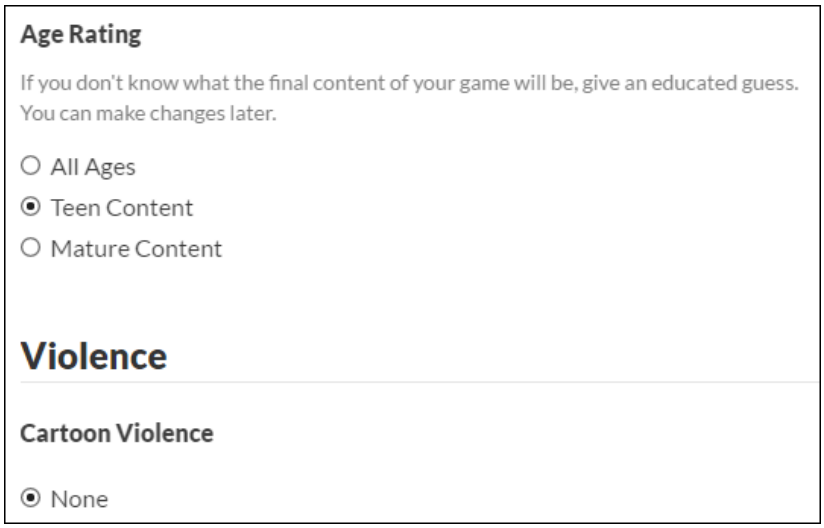


Figura 4.75: Público objetivo del videojuego en Gamejolt
Elaborado por: Hussein Rahman
Fuente: [38]

- **Capturas del videojuego.-** La página recomienda publicar la mayor

cantidad de imágenes, gameplays junto con cualquier material que permita promocionar el juego.

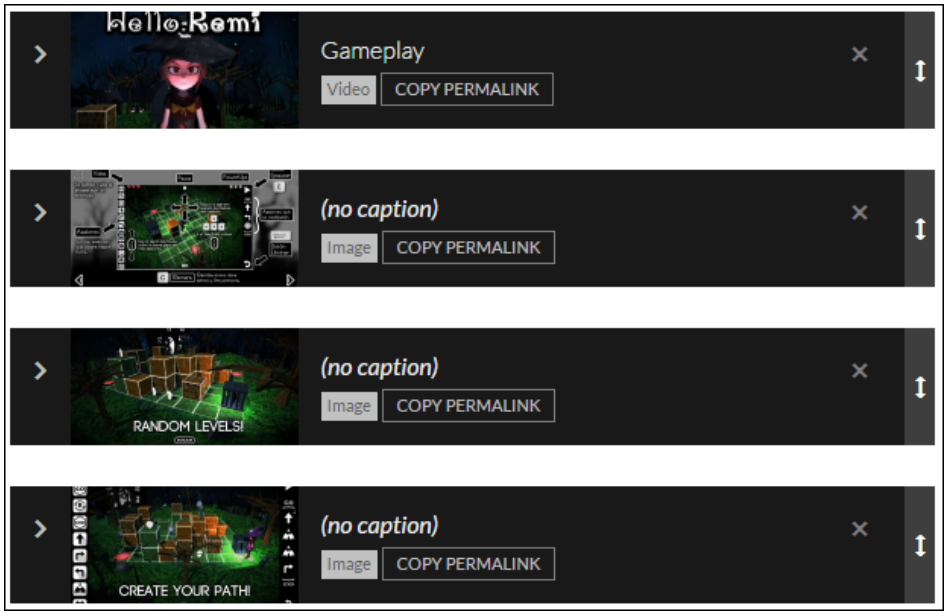


Figura 4.76: Capturas del videojuego en Gamejolt
Elaborado por: Hussein Rahman
Fuente: [38]

- **Paquetes.-** En esta sección se sube el o los archivos ejecutables del juego dentro de un ***Package*** el cual contendrá varios ***Releases*** que representan las diferentes versiones publicadas, cada ***Release*** puede contener diferentes ***Builds*** y cada ***Build*** representa una plataforma diferente, como por ejemplo: La ***Release v1.2.1*** contiene una ***Build*** para Android y otra para Windows.

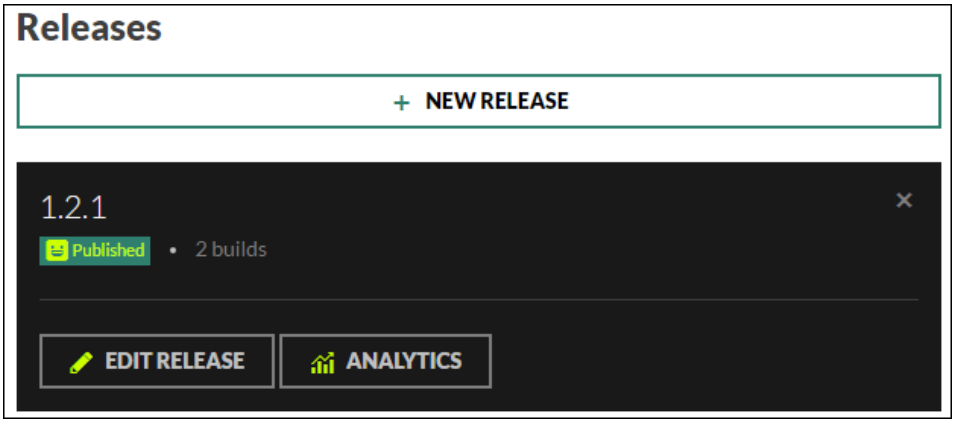


Figura 4.77: Paquetes del videojuego en Gamejolt
Elaborado por: Hussein Rahman

Análisis de los resultados obtenidos posterior a la publicación

El videojuego ***"Hello Remi"*** fue publicado el 18 de octubre del 2016, un mes después se realiza este análisis para observar el resultado obtenido por el videojuego, los datos han sido extraídos de la propia página de ***Gamejolt*** la cual genera un reporte mensual de datos como: número de descargas, visitas, rating, etc.

La ***Figura 4.78*** muestra el total de visitas, descargas, número de personas han calificado y la cantidad de seguidores del videojuego.

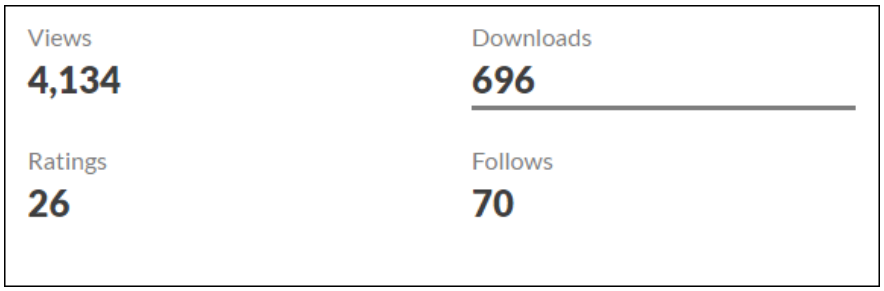


Figura 4.78: Resumen de la publicación del videojuego
Elaborado por: Hussein Rahman
Fuente: [38]

La cantidad de descargas por país se muestra en la ***Figura 4.79***, destacándose Estados Unidos y Argentina como los países con mas descargas, mientras que Ecuador se encuentra en 5to lugar con un 4.17 % del total de descargas.

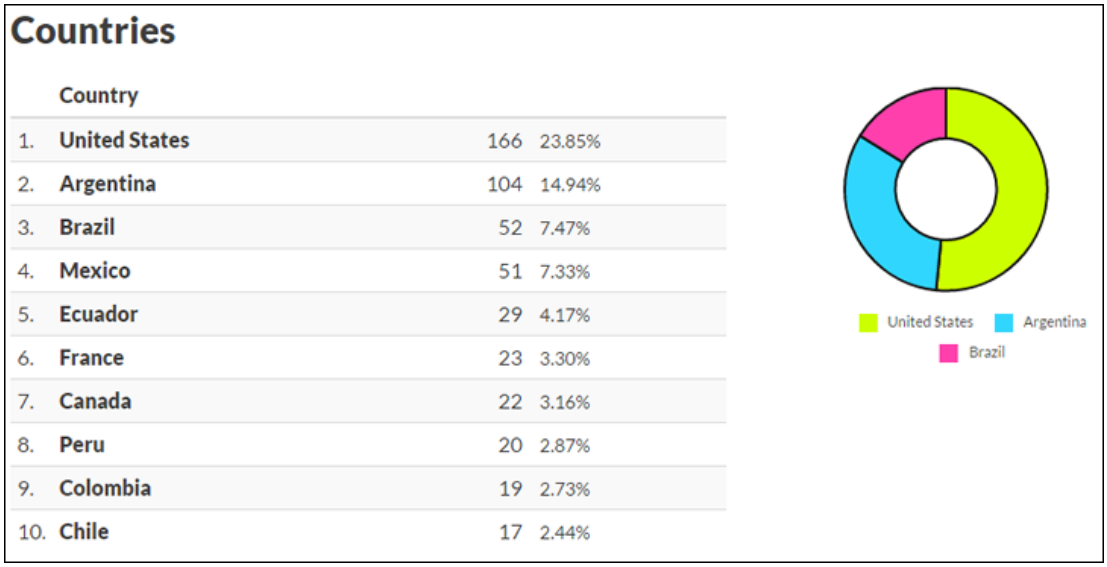


Figura 4.79: Descargas del videojuego por país
Elaborado por: Hussein Rahman
Fuente: [38]

El sistema operativo usado para las descargas del archivo .apk se puede observar en la ***Figura 4.80***, destaca Windows y el sistema other (Android).

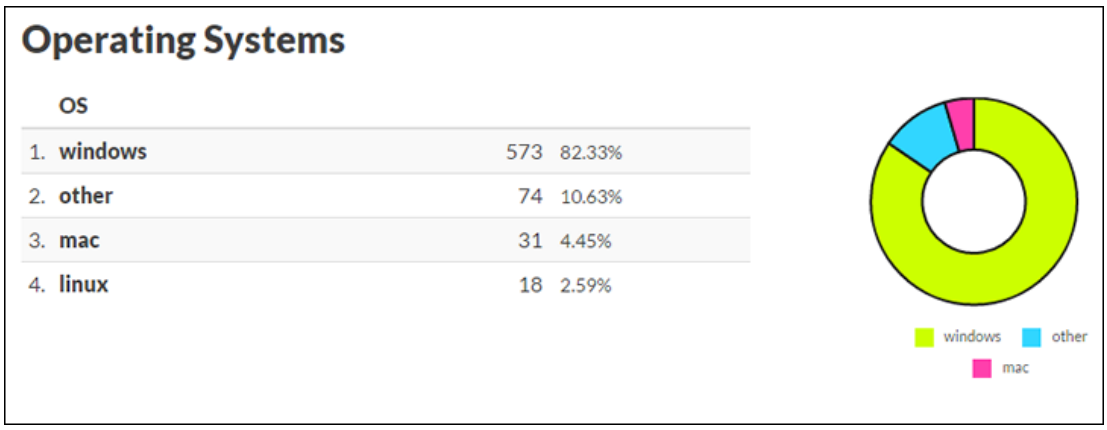


Figura 4.80: Descargas del videojuego por sistema operativo
Elaborado por: Hussein Rahman
Fuente: [38]

Finalmente en la ***Figura 4.81*** se puede observar la calificación de los usuarios entre un rango de 1 a 5, el 57.69 %, es decir mas de la mitad de las personas que calificaron al juego le dieron una calificación de 5.

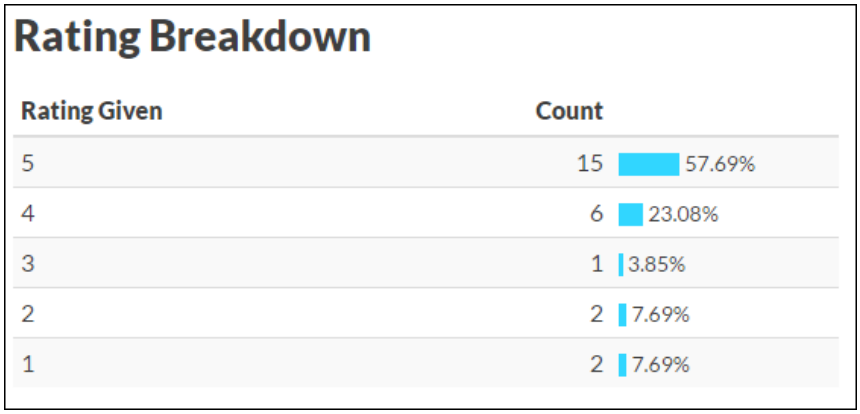


Figura 4.81: Rating del videojuego
Elaborado por: Hussein Rahman
Fuente: [38]

El resultado obtenido indica un logro aceptable al tratarse de un videojuego independiente, llegando a la conclusión de que es mucho mas fácil distribuir un software de este tipo fuera del país.

CAPÍTULO 5

Conclusiones y Recomendaciones

5.1. Conclusiones

- El uso de un framework y motor de juego apropiado ayudó a organizar de una mejor manera el desarrollo del proyecto, pero hay que destacar la complejidad existente al tratarse de un software muy diferente al que se ha venido desarrollando en las diferentes asignaturas de la carrera, se requiere de gran dedicación, tiempo y esfuerzo para integrar los diferentes elementos que componen el videojuego junto con las mejores practicas de programación aprendidas.

- Dentro del desarrollo de videojuegos a nivel personal no existe una metodología ágil establecida formalmente, pero se han creado proyectos como la metodología DAV la cual se basa en SCRUM y XGD. Esta metodología permitió un desarrollo planificado y ordenado en cada una de las fases, añadiendo bastante flexibilidad al momento de realizar cambios en tareas que ya finalizaron. Las iteraciones ayudaron bastante a tener en corto tiempo una versión funcional del juego para analizar el estado del producto y las características que aun faltan desarrollar.

- La publicación del videojuego se la realizo sin inconvenientes, después de una evaluación entre diferentes tiendas online se eligió a GameJolt por la facilidad que brindan a los nuevos desarrolladores independientes. Gracias a las facilidades de la tienda se lograron muy buenos resultados tanto en críticas por parte de usuarios como en numero de descargas después de realizar la publicación del videojuego.

5.2. Recomendaciones

- El videojuego ha sido optimizado para una gran número de dispositivos Android, es recomendable revisar los requisitos de hardware y software necesarios que se encuentran en la página de publicación y en el manual de usuario.

- Antes de empezar una partida del juego es recomendable revisar la ayuda inicial del videojuego la cual brinda ayuda sobre los controles, comandos, errores, objetivos y todos los elementos que conforman el videojuego.

- Aunque dentro del proyecto el videojuego esta limitado a un rango especifico de edades (5 a 18 años) el videojuego puede ser jugado por personas de cualquier edad por la simplicidad de los controles y la fácil comprensión del objetivo principal del videojuego. No es necesario tener ningún conocimiento sobre lógica de programación para jugarlo, ya que el contenido educativo se encuentra oculto dentro de la mecánica del juego.

- Para mayor información sobre nuevas versiones, nuevas funcionalidades, nuevas plataformas y para reportar errores se lo puede hacer directamente desde web donde se encuentra publicado en videojuego dicha url se encuentra en el Anexo B (manual de usuario).

Bibliografia

[1] SchoolControl.com, "La importancia de la programación en los pequeños," 2015. [Online] Available: https://goo.gl/kbJkzk.

[2] ElMundo.es, "Enseñar a niños a programar es lo más 'in': Las mejores herramientas," 2015. [Online] Available: https://goo.gl/lI9pZA.

[3] TheGuardian.com, "Coding at school: a parent's guide to england's new computing curriculum," 2014. [Online] Available: https://www.theguardian.com/technology/2014/sep/04/coding-school-computing-children-programming.

[4] Educacion.gob.ec, "El perfil de salida de los estudiantes de educación general básica," 2015. [Online] Available: https://educacion.gob.ec/wp-content/uploads/downloads/2012/08/Perfil_Salida_EGB.pdf.

[5] A. Rossaro, "Los videojuegos y su potencial educativo," 2012. [Online] Available: https://goo.gl/ElyJFq.

[6] Cbsnews.com, "Coding for kindergarteners: App teaches kids computer basics," 2014. [Online] Available: https://goo.gl/C0KZhi.

[7] J. Biggs, "Light-Bot Teaches Computer Science With A Cute Little Robot And Some Symbol-Based Programming," 2015. [Online] Available: https://goo.gl/IRPkmR.

[8] E. Cisneros, *Videojuego Educativo como apoyo a la enseñanza de la Algoritmia para los estudiantes del Programa Nacional de Formación en Sistemas e Informática.* PhD thesis, Instituto Superior Politécnico José Antonio Echeverría (Caracas), 2010.

[9] K. Albuja, *Análisis, diseño y desarrollo de un juego didáctico de razonamiento abstracto en 3d, para ayudar al desarrollo del pensamiento de niños entre 4 y 8 años, utilizando un game engine con c# y aplicando la metodología OOHDM.* PhD thesis, Escuela Politecnica del Ejercito, 2012.

[10] M. Zambrano, *Diseño y desarrollo de un video-juego educativo con técnicas de inteligencia artificial para plataforma Android utilizando metodología*

OOHDM. Caso de estudio: Laberinto 3D. PhD thesis, Escuela Politecnica del Ejercito, 2014.

[11] M. Torres, *Aplicación de la Metodología Oohdm y Técnicas de Inteligencia Artificial en la Solución del Desarrollo de un videojuego, enfocado a niños de 6 a 10 años, utilizando la Tecnología gdi+ basado en c# y Wiimote, para su aplicación en la Empresa Virtual Learni.* PhD thesis, Escuela Politecnica del Ejercito, 2013.

[12] C. T. Miller, *Games: purpose and potential in education.* Springer Science & Business Media, 2008.

[13] D. V. Fernandez, C. M. Angelina, and EspaCursos, *Desarrollo de Videojuegos: Arquitectura del Motor de Videojuegos.* Cursos en Español, Oct. 2011.

[14] Pixelsmil.com, "Qué significa "Indie"," 2012. [Online] Available: `http://www.pixelsmil.com/2012/05/que-significa-indie.html`.

[15] J. P. Martín, "Penetración de los videojuegos educativos e infantiles en España desde el 2005 al 2007," *Comunicación y Pedagogía: Nuevas tecnologías y recursos didácticos*, no. 229, pp. 23–28, 2008.

[16] L. N. de Calidad del Software (España), "Metodologías y Ciclos de Vida," 2009. [Online] Available: `https://goo.gl/bkO7yO`.

[17] P. Kruchten, *The rational unified process: an introduction.* Addison-Wesley Professional, 2011.

[18] Msdn.microsoft, "Microsoft Solutions Framework (MSF)," 2016. [Online] Available: `https://msdn.microsoft.com/es-es/library/jj161047(v=vs.120).aspx`.

[19] O. Pastrana, "5 beneficios de aplicar metodologías ágiles en el desarrollo de software," 2014. [Online] Available: `https://goo.gl/wCMAvt`.

[20] N. Acerenza, A. Coppes, G. Mesa, A. Viera, E. Fernández, T. Laurenzo, and D. Vallespir, "Una Metodología para Desarrollo de Videojuegos," 2009.

[21] O. Trejos, *La esencia de la Lógica de Programación.* 2010.

[22] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software.* Addison Wesley, 2010.

[23] UnrealEngine.com, "Unreal Engine," 2016. [Online] Available: https://www.unrealengine.com/blog.

[24] Unity3D.com, "Unity - Game engine, tools and multiplatform," 2016. [Online] Available: https://unity3d.com/es/unity.

[25] GodotEngine.org, "Welcome to Godot Engine's documentation," 2016. [Online] Available: http://docs.godotengine.org/en/latest/.

[26] T. Demachy, "Extreme Game Development: Right on Time, Every Time," 2013. [Online] Available: https://goo.gl/XPS9O4.

[27] A. Barreno, G. Alexander, S. Guaraca, and M. Gonzalo, *Adaptación de las metodologías ágiles Scrum y Extreme Game Development en una metodología para el desarrollo de videojuegos en Android. Caso práctico: Desarrollo de un videojuego.* PhD thesis, Sept. 2013.

[28] Eclipse.org, "SUM para Desarrollo de Videojuegos," 2008. [Online] Available: http://www.gemserk.com/sum.

[29] B. Foundation, "Blender | free and open 3d creation software," 2016. [Online] Available: https://www.blender.org/about/.

[30] Gimp.org, "Gimp - gnu image manipulation program," 2016. [Online] Available: https://www.gimp.org/about/.

[31] Inkscape.org, "Inkscape | tutorials and help," 2016. [Online] Available: http://www.inkscape.org/about/.

[32] Audacityteam.org, "Audacity | manuals and documentation," 2016. [Online] Available: http://www.audacityteam.org/about/.

[33] Microsoft, "Visual studio | developer tools and services," 2016. [Online] Available: https://www.visualstudio.com/es/vs/community/.

[34] Develover.Android.com, "Android studio y sdk tools," 2016. [Online] Available: https://developer.android.com/studio/index.html.

[35] Support.google.com, "Google play | transaction fees," 2016. [Online] Available: https://goo.gl/MFDnE4.

[36] Developer.amazon.com, "Amazon app distribution agreement," 2016. [Online] Available: https://developer.amazon.com/public/support/legal/da.

[37] Gamejolt.com, "Terms of use on game jolt," 2016. [Online] Available: http://gamejolt.com/terms.

[38] H. Rahman, "Hello remi by hussein rahman," 2016. [Online] Available: http://gamejolt.com/games/hello-remi/209204.

Anexos y Apéndices

Anexo A

Tarjetas de tareas

Tarjeta de tarea	
Número de Tarea: T01	**Historia de Usuario (Nro. y Nombre):** HU09 - Programador
Nombre Tarea: Generación de las casillas del mapa	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 04 de julio del 2016	**Fin:** 05 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Se debe generar las casillas del mapa de forma automática con un id de coordenada única para cada una.	

Tabla A.1: Tarjeta de Tarea "T01"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T02	**Historia de Usuario (Nro. y Nombre):** HU10 - Programador
Nombre Tarea: Acciones para el personaje	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 06 de julio del 2016	**Fin:** 07 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Crear acciones que permitan el movimiento y la interacción del personaje con el mundo.	

Tabla A.2: Tarjeta de Tarea "T02"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T03	**Historia de Usuario (Nro. y Nombre):** HU14 - Programador
Nombre Tarea: Comandos básicos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 08 de julio del 2016	**Fin:** 09 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Crear una nomenclatura de comandos básica para realizar las acciones del personaje.	

Tabla A.3: Tarjeta de Tarea "T03"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T04	**Historia de Usuario (Nro. y Nombre):** HU15 - Diseñador
Nombre Tarea: Diseño y texturización del personaje principal	
Tipo de Tarea: Diseño	**Puntos Estimados:** 16 horas
Inicio: 10 de julio del 2016	**Fin:** 11 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Diseñar al personaje principal en base a los bocetos creados durante la fase de pre-juego	

Tabla A.4: Tarjeta de Tarea "T04"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T05	**Historia de Usuario (Nro. y Nombre):** HU20 - Diseñador
Nombre Tarea: Animaciones del personaje	
Tipo de Tarea: Diseño	**Puntos Estimados:** 16 horas
Inicio: 12 de julio del 2016	**Fin:** 13 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Crear animaciones al personaje para que responda según las acciones que realice.	

Tabla A.5: Tarjeta de Tarea "T05"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T06	**Historia de Usuario (Nro. y Nombre):** HU24 - Programador
Nombre Tarea: Interfaz de comandos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 12 horas
Inicio: 14 de julio del 2016	**Fin:** 15 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Crear una interfaz para ingresar los comandos utilizando botones fáciles de entender.	

Tabla A.6: Tarjeta de Tarea "T06"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T07	**Historia de Usuario (Nro. y Nombre):** HU26 - Programador
Nombre Tarea: Meta del juego	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 12 horas
Inicio: 15 de julio del 2016	**Fin:** 16 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Crear un objeto que simbolice la meta de cada nivel.	

Tabla A.7: Tarjeta de Tarea "T07"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T08	**Historia de Usuario (Nro. y Nombre):** HU11 - Diseñador
Nombre Tarea: Diseño del enemigo	
Tipo de Tarea: Diseño	**Puntos Estimados:** 16 horas
Inicio: 18 de julio del 2016	**Fin:** 19 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Diseñar al enemigo en base a los bocetos realizados en la fase pre-juego.	

Tabla A.8: Tarjeta de Tarea "T08"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T09	**Historia de Usuario (Nro. y Nombre):** HU12 - Programador
Nombre Tarea: Generación de niveles	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 20 de julio del 2016	**Fin:** 21 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Generar los niveles de forma aleatoria utilizando el proceso de generación de casillas.	

Tabla A.9: Tarjeta de Tarea "T09"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T10	**Historia de Usuario (Nro. y Nombre):** HU13 - Programador
Nombre Tarea: Movimiento de cámaras	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 22 de julio del 2016	**Fin:** 23 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Mover las cámaras de forma manual y automática utilizando el touch de las pantallas.	

Tabla A.10: Tarjeta de Tarea "T10"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T11	**Historia de Usuario (Nro. y Nombre):** HU21 - Programador
Nombre Tarea: Control de colisiones del personaje	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 24 de julio del 2016	**Fin:** 25 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Utilizar sensores que permitan detectar la colisión del personaje con los objetos del mundo	

Tabla A.11: Tarjeta de Tarea "T11"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T12	**Historia de Usuario (Nro. y Nombre):** HU22 - Programador
Nombre Tarea: Panel flotante de pausa	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 26 de julio del 2016	**Fin:** 27 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Crear un panel flotante de pausa que permita reiniciar y salir del juego en cualquier momento.	

Tabla A.12: Tarjeta de Tarea "T12"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T13	**Historia de Usuario (Nro. y Nombre):** HU23 - Programador
Nombre Tarea: Notificación de errores	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 24 horas
Inicio: 28 de julio del 2016	**Fin:** 30 de julio del 2016
Miembro responsable: Hussein Rahman	
Descripción: Crear paneles flotantes para notificar diferentes tipos de errores durante la partida.	

Tabla A.13: Tarjeta de Tarea "T13"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T14	**Historia de Usuario (Nro. y Nombre):** HU17 - Programador
Nombre Tarea: Acciones avanzadas del personaje	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 1 de agosto del 2016	**Fin:** 2 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Crear acciones avanzadas para la interacción del personaje con el mundo.	

Tabla A.14: Tarjeta de Tarea "T14"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T15	**Historia de Usuario (Nro. y Nombre):** HU18 - Programador
Nombre Tarea: Comandos avanzados	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 3 de agosto del 2016	**Fin:** 4 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Crear una nomenclatura de comandos avanzados para realizar las acciones avanzadas del personaje.	

Tabla A.15: Tarjeta de Tarea "T15"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T16	**Historia de Usuario (Nro. y Nombre):** HU19 - Diseñador
Nombre Tarea: Decoración del mundo	
Tipo de Tarea: Diseño	**Puntos Estimados:** 24 horas
Inicio: 5 de agosto del 2016	**Fin:** 7 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Diseñar la decoración del mapa para mejorar la estética del juego.	

Tabla A.16: Tarjeta de Tarea "T16"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T17	**Historia de Usuario (Nro. y Nombre):** HU25 - Programador
Nombre Tarea: Gestión de powerUps	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 8 de agosto del 2016	**Fin:** 9 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Permitir la obtención de powerUps para el desbloqueo de armas.	

Tabla A.17: Tarjeta de Tarea "T17"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T18	**Historia de Usuario (Nro. y Nombre):** HU27 - Programador
Nombre Tarea: Gestión de vidas	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 10 de agosto del 2016	**Fin:** 11 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Crear un proceso que determine la cantidad de vidas al iniciar la partida y las condiciones para perder vidas.	

Tabla A.18: Tarjeta de Tarea "T18"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T19	**Historia de Usuario (Nro. y Nombre):** HU28 - Sonidista
Nombre Tarea: Música de fondo	
Tipo de Tarea: Sonido	**Puntos Estimados:** 16 horas
Inicio: 12 de agosto del 2016	**Fin:** 13 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Buscar y editar música de fondo acorde con las características del juego.	

Tabla A.19: Tarjeta de Tarea "T19"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T20	**Historia de Usuario (Nro. y Nombre):** HU04 - Programador
Nombre Tarea: Interfaz de ayuda	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 15 de agosto del 2016	**Fin:** 16 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Diseñar y programar la interfaz encargada de mostrar ayuda referente a los controles y comandos del juego.	

Tabla A.20: Tarjeta de Tarea "T20"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T21	**Historia de Usuario (Nro. y Nombre):** HU21 - Programador
Nombre Tarea: Interfaz gráfica de selección de niveles	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 12 horas
Inicio: 17 de agosto del 2016	**Fin:** 18 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Diseñar y programar la interfaz encargada de mostrar los niveles de dificultad del juego.	

Tabla A.21: Tarjeta de Tarea "T21"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T22	**Historia de Usuario (Nro. y Nombre):** HU08 - Programador
Nombre Tarea: Interfaz gráfica de puntuaciones	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 12 horas
Inicio: 18 de agosto del 2016	**Fin:** 19 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Diseñar y programar la interfaz encargada de mostrar las puntuaciones obtenidas al finalizar el nivel.	

Tabla A.22: Tarjeta de Tarea "T22"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T23	**Historia de Usuario (Nro. y Nombre):** HU16 - Programador
Nombre Tarea: Interfaz gráfica de selección de idioma	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 12 horas
Inicio: 19 de agosto del 2016	**Fin:** 20 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Diseñar y programar la interfaz de selección de idioma (ingles y español)	

Tabla A.23: Tarjeta de Tarea "T23"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T24	**Historia de Usuario (Nro. y Nombre):** HU01 - Programador
Nombre Tarea: Interfaz gráfica del menú principal	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 12 horas
Inicio: 20 de agosto del 2016	**Fin:** 21 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Diseñar y programar la interfaz encargada de mostrar el menú principal del juego.	

Tabla A.24: Tarjeta de Tarea "T24"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T25	**Historia de Usuario (Nro. y Nombre):** HU02 - Programador
Nombre Tarea: Interfaz gráfica del storyboard	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 12 horas
Inicio: 21 de agosto del 2016	**Fin:** 22 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Diseñar y programar la interfaz encargada de mostrar el storyboard del juego	

Tabla A.25: Tarjeta de Tarea "T25"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T26	**Historia de Usuario (Nro. y Nombre):** HU05 - Programador
Nombre Tarea: Interfaz gráfica de créditos e información del juego	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 12 horas
Inicio: 23 de agosto del 2016	**Fin:** 24 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Diseñar y programar la interfaz encargada de mostrar los créditos del juego (datos del autor y recursos usados)	

Tabla A.26: Tarjeta de Tarea "T26"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T27	**Historia de Usuario (Nro. y Nombre):** HU07 - Programador
Nombre Tarea: Selección del atuendo del personaje	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 12 horas
Inicio: 24 de agosto del 2016	**Fin:** 25 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Programar una interfaz que permita seleccionar el atuendo del personaje principal antes de iniciar la partida.	

Tabla A.27: Tarjeta de Tarea "T27"

Elaborado por: Hussein Rahman

Tarjeta de tarea	
Número de Tarea: T28	**Historia de Usuario (Nro. y Nombre):** HU03 - Programador
Nombre Tarea: Interfaz gráfica de configuración del juego	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 16 horas
Inicio: 26 de agosto del 2016	**Fin:** 27 de agosto del 2016
Miembro responsable: Hussein Rahman	
Descripción: Diseñar y programar la interfaz encargada de configurar aspectos gráficos y de audio del juego.	

Tabla A.28: Tarjeta de Tarea "T28"

Elaborado por: Hussein Rahman

Anexo B

Manual de usuario

Contenido

1. REQUISITOS MINIMOS

a. Windows

- OS: Windows XP, 7, 8, 10.
- GPU: DX9 (shader model 3.0) o DX11 (256 VRAM)
- CPU: Dual Core 2.2 GHz
- RAM: 1Gb ram

b. Linux

- OS: Ubuntu, Fedora, Arch, etc.
- GPU: OpenGL ES 2.0 o superior con (256 VRAM)
- CPU: Dual Core 2.2 GHz
- RAM: 1Gb ram

c. Android

- OS: 2.3.1 o posterior
- CPU: ARMv7 (Cortex) con OpenGL ES 2.0 o posterior.
- RAM: 1Gb

Nota: En dispositivos Android con menos de 1gb de ram, cambiar a la calidad de gráficos mínimos en las opciones de configuración del juego.

2. INSTALACIÓN

a. Windows

1. Descargar el archivo setup_remi.exe
2. Ejecutar el instalador y seguir los pasos del asistente.
3. Ejecutar el juego desde el acceso directo del escritorio.

b. Linux

1. Descargar el archivo setup_remi.zip y descomprimirlo.
2. Otorgar permisos de ejecución "chmod +x" al archivo remi.x86.
3. Ejecutar el archivo remi.x86 con el comando "./".

c. Android

1. Descargar el archivo setup_remi.apk
2. Habilitar la instalación de orígenes desconocidos en la configuración de seguridad.

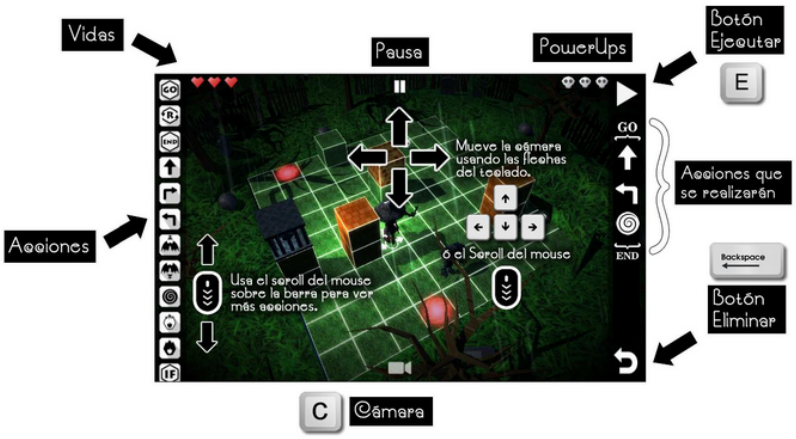
3. Instalar el juego usando el asistente de instalación.
4. Ejecutar el juego.

Nota: En algunos dispositivos se cierra la aplicación durante la primera ejecución debido a problemas de compatibilidad con diferentes modelos de GPU, se soluciona durante la segunda ejecución.

3. CONTROLES

a. Windows y Linux

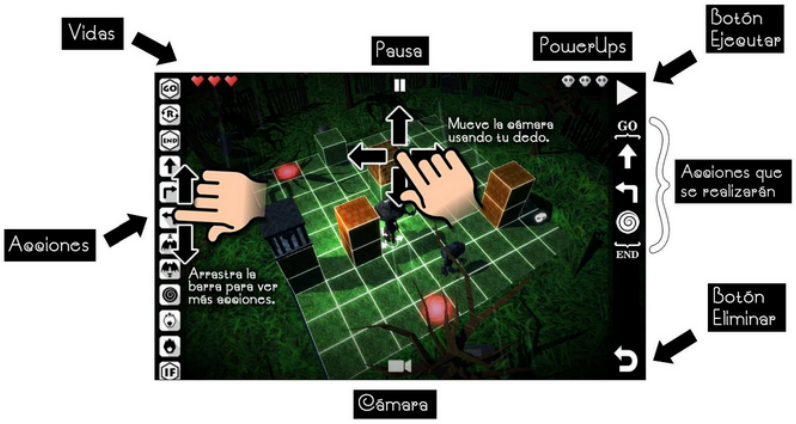
Los controles en la versión de Windows son el teclado y el mouse, tal y como se puede observar en la siguiente imagen.



1. **Vidas.** - El juego inicia con 3 vidas, si colisiona con un enemigo pierde una vida.
2. **Acciones.** - Son las interaccione del personaje con el mundo, como: moverse, girar, volar, etc.
3. **Cámara.** - Con la tecla C es posible cambiar entre perspectiva aérea y tercera persona
4. **PowerUps.** - Son items que permiten al personaje utilizar magia cuando obtiene 3 powerUps.
5. **Botón ejecutar.** – Permite ejecutar las acciones del ingresadas.
6. **Botón eliminar.** - Permite eliminar las acciones ingresadas.

b. Android

La interfaz de la versión Android es similar a la versión de PC con la diferencia que para esta versión se utilizan controles táctiles en lugar de un teclado físico.



4. INTERFAZ DE USUARIO

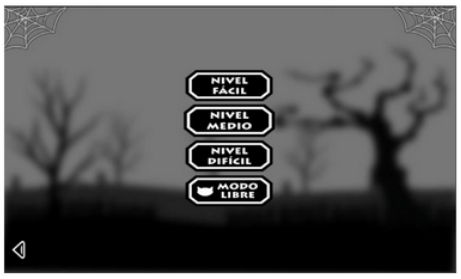
El juego está dividido en diferentes pantallas como se muestran a continuación:

a. Selección de idioma. - Mediante esta pantalla se puede iniciar el juego en inglés o español.

b. **Menú principal.** - Es la pantalla inicial, permite iniciar, salir, ver los créditos y entrar a la configuración.

c. **Storyboard.** - Muestra la historia del juego.

d. **Menú niveles.** - Permite seleccionar el nivel de dificultad o entrar al modo libre, cada nivel es creado aleatoriamente.

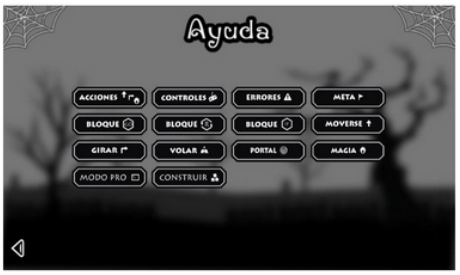


e. **Créditos.** - Información del autor y recursos externos utilizados.

f. **Configuración.** - Permite cambiar la calidad gráfica para mejorar el rendimiento o establecer el volumen de la música del juego.

g. **Puntuación.** - Se muestra cuando el nivel es finalizado con éxito, indicando 3, 2 o 1 calabaza según el tiempo utilizado.

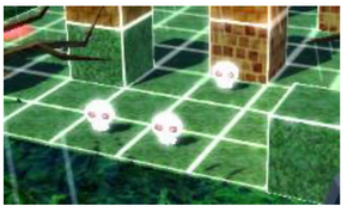
h. **Ayuda.** - Se activa desde el panel pausa y permite entender el funcionamiento de los diferentes elementos del juego.



i. **Nivel perdido.** - Esta pantalla se muestra cuando el jugador pierde todas sus vidas.

5. META DEL JUEGO

a. **Obtener powerUps.** - Tres powerUps son necesarios para activar la magia, la cual ayuda a destruir a los enemigos.



b. **Destruir a todos los enemigos.** - Después de eliminar a todos los enemigos del mapa, la prisión que encierra al gatito se abrirá.

Nota: La magia blanca destruye a los enemigos negros, mientras que la magia negra vence a los enemigos blancos.

c. **Rescatar al gatito. -** Para finalizar cada nivel es necesario llegar a la posición de la prisión abierta.

6. MENSAJES DE ERROR

Los errores que ocurran durante la partida, serán mostrados mediante una notificación visible en la parte superior, existen algunos tipos de errores y sus causas, tal y como se muestra a continuación:

a. **Error de movimiento. -** Este error sucede cuando se intenta mover en casillas que no existen en el mapa.

b. **Error de colisión. -** Cuando el personaje colisiona con un objeto del mapa (columna, enemigo, prisión bloqueada).

c. **Error de acciones.** - Cuando las acciones están ingresadas incorrectamente se mostrará este error.

Nota: El error es causado por falta del comando de apertura del bloque, se podría iniciar con (GO ó R).

d. **Error de coordenadas.** – Son errores causados por ingresar coordenadas incorrectas en los comandos de construcción, para más información véase el (**numeral 10-b**).

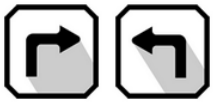
e. **Error de construcción.** - Son errores causados por la sintaxis incorrecta de los comandos de construcción, para más información (véase el **numeral 10-b**).

7. ACCIONES

Las acciones permiten interactuar al personaje con su entorno y de esta manera desplazarse por el mapa para cumplir con los objetivos del juego. Las acciones por si solas no funcionan ya que deben estar agrupadas dentro de un bloque de acciones (véase el **numeral 6**).

a. **Mover.** - Permite moverse una o varias casillas hacia adelante.

b. **Girar.** - Permite girar ¼ hacia la derecha o izquierda.



c. **Volar.** - Permite moverse horizontalmente una o varias casillas.

d. **Aterrizar.** - Permite regresar a la tierra (Solo si el personaje está en el aire).

e. **Portal.** - Permite utilizar portales para transportarse entre una casilla a otra.

f. **Magia.** – Permite utilizar magia blanca (destruye enemigos negros) y magia blanca (destruye enemigos blancos).

g. **Colisión.** - Acción especial para ser usada junto con el bloque IF **(véase numeral 6-c)** y determinar con qué tipo de enemigo colisionó el personaje.

8. BLOQUES

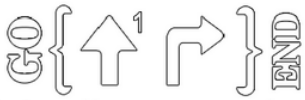
Los bloques son comandos especiales que permiten agrupar acciones para su posterior ejecución, existen 3 tipos principales de bloques, los cuales son:

a. **Bloque GO. -** Este bloque ejecutará una sola vez las acciones que se encuentren agrupadas. Para utilizar las acciones dentro del bloque GO se lo hace de la siguiente manera:

Entrada

Nota: El comando END es utilizado para cerrar los bloques de acciones.

Salida

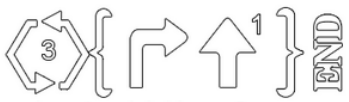


Nota: En superíndice de la acción mover es un parámetro numérico que indica cuantos pasos avanzará el personaje, este valor puede ser cambiado desde el panel parámetros (véase **numeral 7-a**).

b. **Bloque R.-** El bloque "Repeat" permite ejecutar n veces las acciones que se encuentren agrupadas. Para utilizar las acciones dentro del bloque R se lo hace de la siguiente manera:

Entrada

Salida



Nota: En número 3 en el centro del bloque R es un parámetro numérico que indica cuantas veces se repetirán las acciones dentro del bloque (véase **numeral 7-a**).

c. **Bloque IF.** - Es un bloque especial que permite elegir entre dos posibles resultados en base a una condición planteada, es quizás el bloque más complejo de usar. Para utilizar las acciones dentro del bloque IF se lo hace de la siguiente manera:

Entrada

Nota: La acción colisión abrirá el panel colisión (véase **numeral 7-c**) para seleccionar entre dos posibles condiciones:

Si el enemigo frente al jugador es de color blanco  , o si el enemigo es de color negro  .

Salida

Nota: El bloque IF utiliza los comandos TRUE y FALSE para agrupar acciones en caso de cumplirse o no la condición inicial, además se usa el comando ENDIF para finalizar el bloque IF.

9. PANELES FLOTANTES

Los paneles flotantes son pequeñas ventanas que se muestran con determinados comandos que requieren parámetros (mover, volar, colisión y bloque R).

a. **Panel parámetros.** – Se mostrará solo con los comandos (mover, volar y bloque R), permitiendo cambiar el valor numérico entre 0 y 99. Contiene 4 botones como se muestra a continuación:

1. **Botón más.** – Incrementa el valor numérico.
2. **Botón menos.** – Disminuye el valor numérico
3. **Botón ok.** – Agrega el valor numérico a la acción.
4. **Botón x.**– Cierra la ventana y cancela la acción.

b. **Panel colisión.** – Se mostrará solo con la acción colisión (véase los **numerales 5-g y 6-c**), permitiendo seleccionar entre dos posibles condiciones. La funcionalidad de los botones es idéntica al panel parámetros.

c. **Panel pausa.** – Este panel se mostrará si se presiona el botón pausa ubicado en la parte superior. Contiene cuatro botones como se muestra a continuación:

1. **Botón reiniciar. -** Reinicia los valores por defecto del nivel actual para empezar de nuevo.
2. **Botón ayuda. –** Muestra una pantalla con tópicos de ayuda sobre diferentes elementos del juego (véase **numeral 3-h**).
3. **Botón salir. -** Sirve para salir del nivel actual, regresando al menú de selección de niveles (véase **numeral 3-c**).
4. **Botón x.-** Cierra el panel pausa.

10. MODO PRO

Es un modo avanzado del juego que permite utilizar pequeños scripts para realizar la ejecución de acciones.

Para acceder al modo pro se lo hace por medio del botón Pro mode ubicado en la parte inferior derecha de la pantalla.

a. Scripts modo pro

La sintaxis de las acciones es la siguiente:

Acciones	Comandos
Mover	move.n
Girar derecha	turn.r
Girar izquierda	turn.l
Volar	fly.n
Aterrizar	land
Portal	use.portal
Magia blanca	magic.l
Magia negra	magic.d
Colisión enemigo blanco	enemy.l

Colisión enemigo blanco	enemy.d

Nota: n es el parámetro numérico, l=light, d=dark

La sintaxis de los bloques de acciones es la siguiente:

Bloques	Comandos
GO	go{}
R	repeat(n){}
IF	if(condición)then{}else{}

A continuación, se muestran una serie de ejemplos del modo de uso de los scripts.

Entrada	Salida
go{move.1,turn.r,use.portal}	avanzar un cuadro, girar a la derecho y usar el portal.
go{fly.2} repeat(5){turn.l}	volar 2 cuadros y girar a la izquierda 5 veces.
if(enemy.d)then{magic.l}else{magic.d, move.3,turn.l,use.portal}	Si el enemigo frente al jugador es negro, entonces usar magia blanca, de otro modo usar magia negra, avanzar 3 cuadros, girar a la izquierda y usar el portal.

Nota: usar un espacio para separar un bloque de otro, los comandos solo son permitidos en minúscula y sin espacios.

11. MODO LIBRE

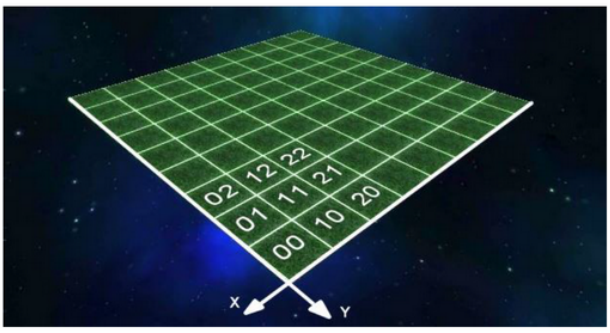
El modo libre es un modo especial en el cual se utiliza para practicar el uso de comandos y adicionalmente se puede utilizar comandos para construir los objetos del mundo tales como: columnas, powerups, portales, enemigos y la salida.

Para acceder al modo libre se lo hace desde el botón ubicado en la pantalla de selección de niveles, (véase **numeral 3-c**)

a. **Scripts construcción.** - Para crear objetos se utiliza un bloque especial llamado "create", en la siguiente tabla se puede observar el uso del comando.

Acciones	Comandos
Crear columnas	create(column){xyh;xyh..}
Crear powerups	create(powerup){xy;xy...}
Crear portales	create(portal){x1y1x2y2;...}
Crear enemigos	create(enemy){xy;xy...}
Crear salida	create(exit){xy}

x = coordenada x
y = coordenada y
h = altura
1 = posición portal inicial
2 = posición portal final



Ejemplos de creación

Columnas: `create(column){021;342;034}`
PowerUps: `create(powerup){01;45;80}`
Portales: `create(portal){2010;4580}`
Enemigos: `create(powerup){11;25}`
Salida: `create(powerup){80}`

Nota: los powerups se crean en pares y se pueden crear hasta 6 pares, solo se puede crear una salida

b. Mensajes de error. - Existen 3 errores que pueden ocurrir al momento de utilizar comandos de creación.

Error	Causa
Coordenadas incorrectas	Es causado con la entrada incorrecta de coordenadas, letras o símbolos en lugar de números por ejemplo.
Solo permitido en modo libre	Cuando se intenta crear objetos fuera del modo libre.
Primero debes crear un enemigo	Cuando se intenta crear la salida sin enemigos creados.

12. DESCARGA DEL JUEGO

El juego se encuentra publicado en la siguiente página, para ser descargado de forma gratuita:

http://gamejolt.com/games/hello-remi/209204