INSTITUTO TECNOLÓGICO

SUPERIOR DE MISANTLA

VIDEOJUEGO PARA AYUDAR A NIÑOS A APRENDER A PROGRAMAR

TESIS

QUE PARA OBTENER EL TITULO DE

INGENIERO EN SISTEMAS COMPUTACIONALES

P R E S E N T A

CARLOS MIGUEL HERRERA ORDUÑA

ANA CAREN TICANTE HERNÁNDEZ

DIRECTOR:

DR. SIMÓN PEDRO ARGUIJO HERNÁNDEZ

CODIRECTORES:

MIA. ROBERTO ÁNGEL MELENDEZ ARMENTA

MSC. JOSÉ ANTONIO HIRAM VÁZQUEZ LÓPEZ

MISANTLA, VERACRUZ ENERO 2019.

INSTITUTO TECNOLÓGICO SUPERIOR DE MISANTLA
DIVISIÓN DE ESTUDIOS PROFESIONALES
AUTORIZACIÓN DE IMPRESIÓN DE TRABAJO DE TITULACIÓN

FECHA: 23 de Enero de 2019.

ASUNTO: **AUTORIZACIÓN DE IMPRESIÓN DE TESIS PROFESIONAL.**

A QUIEN CORRESPONDA:

Por medio de la presente hago constar que el (la) C:

CARLOS MIGUEL HERRERA ORDUÑA

pasante de la carrera de INGENIERÍA EN SISTEMAS COMPUTACIONALES con No. de Control 142T0190 ha cumplido satisfactoriamente con lo estipulado por el **Manual de Procedimientos para la Obtención del Título Profesional de Licenciatura** bajo la **opción Titulación Integral (Tesis Profesional)**

Por tal motivo se **Autoriza** la impresión del **Tema** titulado:

"VIDEOJUEGO PARA AYUDAR A NIÑOS A APRENDER A PROGRAMAR"

Dándose un plazo no mayor de un mes de la expedición de la presente a la solicitud del Acto de Recepción para la obtención del Titulo Profesional.

ATENTAMENTE

ING. GERBACIO TLAXALO ESPINOZA
DIVISIÓN DE ESTUDIOS PROFESIONALES

DIVISIÓN DE ESTUDIOS PROFESIONALES

Archivo.

VER. 01/03/09 F-SA--39

INSTITUTO TECNOLÓGICO SUPERIOR DE MISANTLA
DIVISIÓN DE ESTUDIOS PROFESIONALES
AUTORIZACIÓN DE IMPRESIÓN DE TRABAJO DE TITULACIÓN

FECHA: 23 de Enero de 2019.

ASUNTO: **AUTORIZACIÓN DE IMPRESIÓN DE TESIS PROFESIONAL.**

A QUIEN CORRESPONDA:

Por medio de la presente hago constar que el (la) C:

ANA CAREN TICANTE HERNÁNDEZ

pasante de la carrera de INGENIERÍA EN SISTEMAS COMPUTACIONALES con No. de Control 142T0225 ha cumplido satisfactoriamente con lo estipulado por el **Manual de Procedimientos para la Obtención del Título Profesional de Licenciatura** bajo la **opción Titulación Integral (Tesis Profesional)**

Por tal motivo se **Autoriza** la impresión del **Tema** titulado:

"VIDEOJUEGO PARA AYUDAR A NIÑOS A APRENDER A PROGRAMAR"

Dándose un plazo no mayor de un mes de la expedición de la presente a la solicitud del Acto de Recepción para la obtención del Titulo Profesional.

ATENTAMENTE

ING. GERBACIO TLAXALO ESPINOZA
DIVISIÓN DE ESTUDIOS PROFESIONALES

DIVISIÓN DE ESTUDIOS PROFESIONALES

Archivo.

VER. 01/03/09 F-SA--39

Dedicatoria

Este trabajo se lo dedico a mi madre, quien es el pilar más grande de mi vida, la mayoría de mis logros se los debo a ella ya que ella fue quien me crio desde que yo era pequeño, me inculco valores, me ha guiado para ser una buena persona, pero más que nada, por su amor.

También le dedico este trabajo a mi padre, a quien le agradezco profundamente su apoyo durante toda mi formación académica, su cariño, su paciencia y sus consejos. Gracias a él es que he llegado hasta donde estoy.

Su hijo, Carlos Miguel.

Dedicatoria

Este gran logro se lo dedico a la fortaleza de mi vida, mi familia, a mis padres por todo su trabajo y sacrificio, a mi madre por ser mi motivación, a mi padre por ser mi ejemplo a seguir, a Xanath por su apoyo y palabras de aliento, a Pepe por ser mi inspiración, a Xavi por ser mi cómplice y siempre contar con su apoyo, a Jacob por llenar nuestras vidas de alegría, a mi abuelo Cande por creer en mí.

Por ustedes he llegado hasta aquí, infinitas gracias.

Los amo, Ana Caren.

Agradecimientos

A mis padres por apoyarme incondicionalmente, motivarme e inculcarme valores que me han hecho una persona de bien, y que gracias a su apoyo ha sido posible haber estudiado una carrera, son las personas más importantes en mi formación y en mi vida. Agradezco a mi padre por su paciencia, confianza y cariño que han traspasado fronteras y espero que el concluir este trabajo y meta lo llene de orgullo.

A mi compañera de tesis Ana Caren, una persona especial con la cual he vivido grandes momentos en este vago suspiro que se llama vida, ella es una pieza importante en la realización de este trabajo y en mi vida. También le agradezco profundamente por guiarme a ser una mejor persona.

A mi asesor Dr. Pedro Arguijo, que es una gran persona y docente, quien me ha guiado, instruido, ayudado y aconsejado en el desarrollo de este trabajo, debo destacar, por encima de todo, su disponibilidad, paciencia y compromiso depositados en este trabajo.

A mis revisores de tesis que han mostrado compromiso, amabilidad, disponibilidad y han ayudado a refinar muchos aspectos de este trabajo.

A mi amiga Leyra y a su madre quienes facilitaron el permiso para acudir a la escuela primaria urbana "Niños Héroes" donde se mostró el videojuego a niños y se hicieron pruebas.

A mis maestros que me han formado a lo largo de mi carrera y quienes no solo me han facilitado el conocimiento para desarrollar este trabajo si no también me han aportado valores y aptitudes, algunos me han mostrado la importancia de ser una persona profesional, respetuosa, responsable, puntual, creativa y comprometida.

A mi institución y a todo su personal, porque cada uno de los involucrados de este sistema sirve como engranaje para hacer funcionar esta máquina de superación, crecimiento personal y académico. Además de que la experiencia a lo largo de mi carrera profesional y estadía en esta casa de estudio ha sido agradable, confortable y provechosa.

A mis dos grandes amigos Orlando y Oswaldo quienes estimo tanto, a Oswaldo le agradezco enormemente que nunca ha dudado a enseñarme algo que a mí se me dificulta y a él no, A Orlando porque con su gran carisma y forma de ser siempre me inspira en ser una buena persona, los dos son grandes amigos y personas que sin importar nada y sin esperar nada a cambio ellos estarán ahí.

Infinitas gracias, Carlos Miguel.

Agradecimientos

Agradezco infinitamente a mi madre por su amor y sacrificio, por convertirme en lo que soy, por sus palabras de aliento y motivación, por su comprensión, confianza y apoyo incondicional. Gracias mamá por estar en todo momento y jamás dejarme caer.

A mi padre por sus valores y virtudes inculcados, por sus consejos y por siempre creer en mí, gracias papá por apoyarme siempre, por acompañarme en todos mis sueños y metas, es un orgullo y un privilegio ser su hija.

Agradezco con mucho cariño a mis hermanos, ustedes son mi motivación, inspiración y mi fuerza, gracias por siempre poder confiar en ustedes y es por ustedes que siempre seguiré adelante.

A mi compañero de tesis, Carlos, gracias por hacer posible este logro, por tu paciencia, tu esfuerzo, tu dedicación, por depositar tu confianza en mí, muchas gracias por iluminar mi vida.

A mis compañeros, amigos de vida que siempre estuvieron ahí, gracias por toda su ayuda, por compartir momentos significativos conmigo, porque sobre todas las cosas siempre nos mantuvimos unidos.

A mi asesor de tesis Dr. Pedro Arguijo, por ser mi guía en este proceso, por su confianza, tiempo, enseñanza y dedicación, porque sus consejos y correcciones son parte fundamental de este logro, infinitas gracias.

Al personal académico del ITSM por ser pilares esenciales en mi desarrollo, gracias por compartir sus valiosos conocimientos, por ayudarme a crecer personal y profesionalmente.

A mi familia y a todas las personas importantes en mi vida que están conmigo siempre e incondicionalmente, gracias por aportar cosas buenas a mi vida y apoyarme en todo momento.

Con cariño, Ana Caren.

Resumen

El presente trabajo describe el diseño y desarrollo de un videojuego educativo que tiene como objetivo ayudar a niños a iniciarse en el mundo de la programación, el videojuego busca desarrollar la lógica de la programación en infantes, la forma en que se realiza es dando al jugador el control de un personaje el cual se controla mediante líneas de pseudocódigo que deben ser escritas en un campo de texto de manera correcta.

Se revisaron las propuestas de videojuegos educativos existentes y se investigaron los conceptos principales acerca del desarrollo de videojuegos, sus características, el diseño, desarrollo e implementación de estos.

El videojuego se creó a base de la metodología ágil Scrum ya que se adapta dinámicamente a sus necesidades y requerimientos, permitiendo flexibilidad, mitigación de riesgos y adaptación a cambios. Se utilizaron herramientas de software, como el motor gráfico Unity 3D, Asesprite, Tiled y Adobe Photoshop para el diseño y desarrollo del videojuego.

En el primer capítulo se explica la problemática, se establecen objetivos y se realiza una propuesta de solución al problema planteado. En el segundo capítulo se describe la terminología empleada del presente trabajo y se enlistan algunos trabajos relacionados. En el tercer capítulo se detalla la planificación, diseño y desarrollo del videojuego y en el cuarto capítulo se exponen los resultados obtenidos, así como las conclusiones del trabajo propuesto y posibles trabajos futuros.

Índice de contenido

Índice de figuras

Índice de tablas

Capítulo I. Generalidades

1.1 Introducción

Actualmente, la tecnología se ha convertido en la mano derecha de las personas, ésta ayuda a eliminar obstáculos de comunicación, ha permitido avances importantes en medicina, contribuye en la efectividad de los procesos empresariales y permite implementar nuevas técnicas de enseñanza, entre muchas otras aplicaciones que mejoran la calidad de vida de la sociedad. Gran parte de estas tecnologías operan a base de código escrito en un lenguaje de programación que permite automatizar tareas tediosas y reducir el tiempo en el cual se ejecutan los procesos; según el Dr. Carlos Enrique Ramírez, profesor Investigador de la Universidad Politécnica de Tulancingo, la programación no es para eruditos o genios, se encuentra en la vida cotidiana de las personas y cree que en un futuro cercano será una habilidad que las personas deben poseer sin importar su oficio o profesión. Esta habilidad permite desarrollar el pensamiento creativo y lógico, la claridad en la comunicación, aumenta la productividad y la eficiencia en la resolución de problemas [1].

La enseñanza de la programación es vital para el desarrollo tecnológico de la sociedad y así como con los idiomas, entre menos edad tenga una persona le será más fácil entender un lenguaje. Según la psicología educativa, la edad apropiada para aprender a programar depende de la complejidad del código de programación, si el código es sencillo, podría ser a principio de la educación primaria, si tiene un mayor grado de complejidad es recomendable esperar al segundo o tercer ciclo de primaria, si es aún más complejo, mejor en la etapa de secundaria. El formato de lenguaje debe ajustarse a la edad del infante y se recomienda que tenga un formato gráfico, lingüístico y matemático. Se debe de convertir en una actividad de aprendizaje, además, debe resultar divertido y motivacional, por lo que el componente juego resulta esencial en las primeras edades [2].

Desde su aparición, los videojuegos y juegos en general han sido parte del desarrollo social y cultural de las personas y llegan a influir en su desarrollo físico, sensorial, mental, creativo, etc.

Dada su importancia, en la vida cotidiana, se han creado videojuegos educativos con el objetivo de generar estrategias y métodos de enseñanza alternativos a los tradicionales, animando a los infantes y jóvenes a obtener conocimiento con una herramienta atractiva y motivadora a través de retos y actividades que van presentando dichos videojuegos.

1.2 Planteamiento del problema.

Aprender a programar es el objetivo que se plantea mucha gente, pero esto no es una tarea sencilla, ya que se conjuntan diferentes factores, por mencionar algunos: tiempo, dedicación, método de enseñanza, pensamiento abstracto y, además, se necesita tener habilidad para resolver problemas.

Jorge Iván Fuentes en su artículo "Dificultades de aprender a programar" sostiene que aprender a programar no es como adquirir cualquier otro conocimiento, no es un proceso definido por un algoritmo, no es memorización, no basta con aprender las palabras reservadas de un lenguaje de programación, aprender a programar consiste en plasmar, mediante un lenguaje de programación, la forma de solucionar un problema. Cada problema se soluciona de manera distinta y cada programador lo resuelve de una forma diferente. Es ahí donde radica la dificultad de aprender a programar; de tener un problema y crear una solución. [3]

Los videojuegos son una parte imprescindible en el entretenimiento, según un estudio a nivel mundial existen alrededor de 1,200 millones de video jugadores en el mundo [4]. Otros estudios realizados mencionan los beneficios de jugar videojuegos, siempre que se practique de forma moderada, algunos de estos estudios recogen el hecho de que los videojuegos, utilizados según criterios pedagógicos, pueden aportar interesantes beneficios al mundo educativo [5].

Los videojuegos han encontrado su carácter lúdico dentro de los salones de clase y la educación informal. Existen videojuegos educativos para aprender muchas

cosas, los más básicos enseñan a leer, sumar, restar o hasta tocar instrumentos; pero también existen videojuegos de carácter más serio y profesional, dedicados al aprendizaje de profesiones, la mayoría de estos últimos abordan el tema utilizando simulaciones de la vida diaria como, por ejemplo, ayudar a la capacitación de empleados en una empresa.

En un estudio sobre el uso de los videojuegos aplicados a educación básica como herramienta didáctica se encontró que el 80% de los profesores consideran los videojuegos como herramienta eficaz y el 87% afirma que favorece las habilidades cognitivas [6].

Las habilidades cognitivas son aquellas que analizan y coordinan información recibida para transformarla en conocimiento, recogen la información, la analizan, la procesan y la guardan en la memoria, para poder utilizarla cuando se requiera [7].

Las principales habilidades cognitivas que se desarrollan con el uso de los videojuegos son la concentración y focalización, capacidad analítica, estratégica, toma de decisiones, capacidad de comprensión, planificación de acciones, entre otros.

Por otra parte, el gran desafío de las personas que quieren aprender a programar es comprender la lógica de la programación, la secuencia de instrucciones para la ejecución de una tarea y la resolución de problemas en general, para esto se requieren de la mayoría de las habilidades cognitivas que se desarrollan al jugar un videojuego.

Por lo que una aplicación adicional que se le puede dar a los videojuegos consiste en utilizarlos como una herramienta que ayude a aprender a programar, ya que podría facilitar esta tarea, haciendo el proceso de aprendizaje más atractivo y efectivo.

Debido a la dificultad de aprender a programar se plantea el desarrollo de un videojuego capaz de enseñar las bases de la lógica de programación a través de una serie de niveles, en las cuales el jugador codifica las acciones que debe realizar un personaje. Así como la implementación de un minijuego en el que el usuario pueda

desarrollar habilidades cognitivas que le permitan mejorar la habilidad de resolución de problemas.

1.3 Antecedentes

Para videojuegos en educación, se opta por hacer uso de juegos de estrategia, aventuras y juegos basados en simulaciones o modelos de la vida real. Belli y López [8], señalan que, juegos como Civilization o Rise of Nations hacen hincapié en la experiencia que tiene el jugador con respecto al protagonista de la historia. También mencionan que, en otros casos, el aprendizaje se obtiene por un modelo de práctica, como por ejemplo Tony Hawk's Pro Skater, juego que sumerge al jugador dentro de la cultura del skate, donde este aprende cosas como, diseñar sus propias patinetas, saber cuáles son las ropas indicadas o hasta realizar maniobras. En la mayoría de los videojuegos se proporciona un entorno de experimentación en primera persona donde el jugador interactúa con el contexto creado, toma decisiones y percibe consecuencias.

Los juegos epistémicos, son aquellos que tienen como objetivo formar profesionales, haciéndolos pensar como profesionales. Están diseñados para ser herramientas pedagógicas, donde los sujetos aprenden a trabajar como médicos, abogados, arquitectos, ingenieros, periodistas y otras profesiones.

Industry Player es un simulador de empresa basada en datos del mundo real de Tycoon Systems, empresa especializada en simulaciones del ramo empresarial, Industry Player permite diseñar una empresa con 10 millones de dólares como capital, la cual debe competir en el ramo empresarial que le corresponde hasta ser la mejor [9]. En Holanda, VSTEP (Virtual Safety Training and Education Platform), ha desarrollado muchas simulaciones en 3D para entrenar trabajadores operarios de campos petrolíferos, servicios de emergencia, autoridades portuarias, al personal de hospitales y al ejército [10].

NWN ScriptEase es un proyecto de la Universidad de Alberta Canadá acerca de una herramienta que facilite el uso de NWNScript, para docentes sin conocimientos de programación, de modo que puedan crear adaptaciones

educativas, como por ejemplo, el proyecto de la Escuela de Periodismo y Comunicación de Masas de New Media Institute (EEUU), donde se pone a los alumnos en el papel de periodistas, y mediante simulaciones de situaciones reales, tienen que elaborar artículos e investigaciones periodísticas [11].

Existen sitios web dedicados a dar cursos de programación utilizando diversos lenguajes, pero muchos aspirantes todavía tienen problemas para iniciarse ya que para aprender a programar se necesita pensar de forma abstracta, saber programar se trata de tomar un problema concreto y crear una solución que pueda ser generalizada, sin embargo, esta no es la forma natural en la que los seres humanos piensan, y de hecho ésta es una de las razones por las que muchas personas abandonan el interés por aprender a programar [12].

Una alternativa que ayuda a iniciar personas en la programación es el uso de los videojuegos, para esto existen una gran variedad de videojuegos que se encuentran en la web, ya sea que se pueden jugar online desde un navegador o ser instalados en un dispositivo móvil.

1.3.1 Code combat.

Code combat es una es una plataforma interactiva en línea, un juego con estilo de rol, este juego está enfocado a niños de más de 9 años o adultos y está pensado para que los jugadores puedan aprender a programar lenguajes de programación específicos, por ejemplo, JavaScript acompañado del lenguaje de etiquetas HTML, a diferencia del videojuego propuesto en este trabajo, que utiliza el pseudocódigo en lugar de lenguajes específicos.

En Code combat el jugador toma el papel del personaje que el elija, y mediante hechizos y acciones a través de comandos de programación el juego avanza con diferentes niveles, aumentando el nivel complejidad [13].

1.3.2 CodinGame

CodinGame es otro juego que corre sobre el navegador, el cual le permite al jugador practicar sus habilidades de programación combinándolas con un videojuego de disparos y aventura. En este juego se derrotan enemigos,

programando comandos en la pantalla que se asemeja a una consola y que se encuentra en la parte derecha de la interfaz. Entre los más de 25 lenguajes que maneja el juego se encuentran: C#, C++, JavaScript, PHP, Haskell, Java, Python y Swift aumentando la dificultad según los retos que ofrezca cada lenguaje [14].

El videojuego propuesto en este trabajo es instalable y no requiere de conexión a internet para interactuar con él, a diferencia de este, CodinGame corre sobre una plataforma web por lo que es necesario contar con una conexión a internet, además está enfocado a usuarios adolescentes y adultos que les permite obtener conocimientos de lenguajes de programación específicos.

1.3.3 Code Monkey.

Code Monkey es un juego educativo para niños, en donde un mono recoge bananas para "salvar el mundo", este juego que tiene como objetivo enseñar fundamentos de programación haciendo uso de CoffeeScript para tal tarea, el juego trata de ir superando las pantallas recogiendo un plátano que será puesto en un lugar específico del escenario, por lo que el jugador deberá dar instrucciones usando CoffeeScript para así superar los obstáculos que se interpongan entre el plátano y el mono.

A diferencia del trabajo propuesto que como se mencionó anteriormente, utiliza el pseudocódigo para enseñar las bases de la programación, Code Monkey utiliza CoffeeScript para enseñar a los infantes, es una plataforma web y para tener acceso a todas sus herramientas se debe de pagar una subscripción [15].

1.4 Justificación

Debido a la dificultad de aprender a programar, este trabajo busca facilitar la introducción a la programación a niños mayores a 7 años ya que a partir de esa edad son capaces de tener objetivos y razonamientos concretos, pueden usar símbolos y dar soluciones intuitivas a los problemas. Se desarrolla un videojuego por los beneficios que pueden explotarse con su implementación en el ámbito educativo, además, éste permite ayudar a niños a obtener nuevos conocimientos y habilidades, facilitando el proceso de aprendizaje de programación [16]. Al ser un videojuego,

los niños no deben asistir a una institución o plaza para obtener el conocimiento, ya que esto es posible con el contenido del videojuego, además de que esta herramienta puede estar al alcance, en la seguridad de su casa, por lo que el niño no corre peligro.

Los videojuegos educativos son una forma con la que al cerebro le gusta aprender, ya que tienen la capacidad de facilitar los conocimientos. Los videojuegos motivan vía diversión, además animan a probar diferentes maneras de aprender y pensar, también proporcionan un ambiente educativo-virtual e interactivo [17]. Desde un punto de vista educativo, los videojuegos pueden aumentar la motivación para el aprendizaje y reducir la carga de un instructor, además, ofrecen la oportunidad de practicar; que es una forma de aprender, mejorando la comprensión y adquisición de nuevos conocimientos. Otro efecto evidente del uso de videojuegos en la educación es la adquisición de competencias digitales, por ejemplo, el uso de la computadora personal y su sistema operativo, procesamiento y gestión de la información, expresión creativa y actitud equilibrada frente a la tecnología. Gran parte de los niños que por medio de videojuegos comienzan en el mundo tecnológico, obtienen capacidades propias de alfabetización digital [18].

Los videojuegos proporcionan un entorno de aprendizaje rico y complejo, donde tiene como aspectos positivos: la motivación, el aprendizaje de contenidos y tareas, los procedimientos y destrezas manuales, mejora la retención de conocimientos básicos, aumenta la memoria a largo plazo e incrementa el interés hacia determinadas materias.

1.5 Objetivo general y específicos.

Desarrollar un videojuego capaz de ayudar a niños de 7 a comprender los principios básicos de la programación.

En la búsqueda de llevar acabo la meta planteada es necesario establecer una lista de objetivos específicos:

- Revisar las diferentes propuestas de videojuegos existentes que se han utilizado para enseñar, así como, las soluciones implementadas.

- Identificar los elementos involucrados en el desarrollo de videojuegos educativos.
- Investigar y estudiar cuáles son los elementos necesarios para que el videojuego sea capaz de ayudar a los usuarios a desarrollar la lógica básica de programación.
- Implementar las soluciones que más se adecuan para enseñar los conocimientos básicos necesarios para aprender a programar.
- Analizar y escoger la plataforma apropiada para la implementación de este videojuego.
- Diseñar, desarrollar e implementar un videojuego que sea agradable al usuario.

1.6 Propuesta de solución

La problemática que se planea resolver es la de ayudar a niños a iniciarse en el mundo de la programación, este trabajo propone como solución el uso de un videojuego educativo debido a sus ventajas que ofrece frente a la problemática. Se tiene como premisa ayudar a un personaje, Scott Pilgrim, el cual se describe en la sección 3.3.10, a llegar al final del mapa para encontrar una guitarra siguiendo una secuencia de instrucciones, para así asistir al usuario en la comprensión del concepto de algoritmo que es uno de los principios de la programación.

La manera en que se introducen las acciones del personaje se realiza mediante la forma más cercana a los lenguajes de programación que es el pseudocódigo, éste es un lenguaje falso con palabras clave que el jugador debe codificar con cierta sintaxis predefinida similar a la de los lenguajes de programación más conocidos, se optó que las instrucciones del videojuego sean en el idioma español ya que el público al que va dirigido el videojuego son menores hispanohablantes.

En el videojuego, al usuario se le presentan situaciones en las que debe pensar en una estrategia de solución en donde logre organizar instrucciones para formar la estructura del programa que será ejecutado dentro del juego desarrollando así su lógica de programación.

1.7 Alcances y limitaciones.

Alcances

- Diseñar e implementar un videojuego educativo que sirva como herramienta para iniciar a niños en la programación.
- Se pretende abarcar conocimientos básicos, así como la lógica necesaria para aprender a programar.
- El producto estará implementado para un jugador.
- El producto será desarrollado para PC, porque en una computadora es donde usualmente se programa y las personas que desean programar deberían de estar familiarizados con su uso.
- El juego será gratuito.

Limitaciones.

- El proyecto de investigación no cuenta con un experto que evalúe la retroalimentación entre el juego y el jugador.
- Falta de un experto que evalúe cuales técnicas de aprendizaje son mejores para aplicar en un videojuego educativo, por lo que se limitará a escoger las técnicas de los juegos para programación más conocidos y jugados actualmente.
- El desarrollo del producto se realizará en software gratuito, privándose así de algunas herramientas de paga que el entorno de desarrollo ofrece.
- Falta de usuarios suficientes para interactuar con el software.

Capítulo II. Marco Teórico

2.1 Definición de juego

Un juego es un ejercicio recreativo sometido a reglas en el que normalmente se gana o se pierde y en la que se pone en práctica alguna capacidad o destreza ya sea mental, física o ambas.

2.2 Definición de videojuego

Un videojuego es un juego digital interactivo, con independencia de su soporte, este software está creado con fines de entretenimiento en general y se basa en la interacción, principalmente, con personas o personajes ficticios. Para que un videojuego pueda ser usado se necesita utilizar un aparato electrónico, por ejemplo, una computadora, una consola de videojuegos, una tableta electrónica, un teléfono móvil entre otros.

2.3 Tipos de videojuegos

Los videojuegos se clasifican de diversas maneras, por ejemplo, por género en los que pueden destacarse:

- Juegos de acción: Son juego de luchas y de peleas, el objetivo de estos juegos consiste básicamente en eliminar enemigos.
- Árcade: Son juegos de plataformas, laberintos o aventuras. El objetivo principal de este tipo de juegos es que el usuario supere escenarios y pantalla para seguir jugando, por lo general tienen un ritmo rápido y requieren tiempos de reacción mínimos.
- Deportivos: Son juegos que simulan la mecánica de un deporte, también suele llamarse así a los juegos que promueven la competitividad y valores propios de los deportes físicos.

- Estrategia: Este tipo de juegos hacen uso de la planificación de recursos o acciones para lograr un objetivo.
- Simulación: Estos juegos se caracterizan por permitirle al jugador experimentar e investigar el funcionamiento de máquinas, fenómenos, situaciones y asumir el mando.

2.4 Juegos serios

Los juegos serios, son juegos que están diseñados con un propósito más formativo que de entretenimiento, los juegos serios tienen un propósito educativo explícito y cuidadosamente planeado. Este tipo de juegos ayudan a la comprensión y aprendizaje de nuevos conceptos y habilidades. Una característica principal de este tipo de juegos es que no utilizan mundos de fantasía, y se basan en mundos o entornos reales, por esto los jugadores experimentan problemas reales, donde aprenden de sus errores y adquieren experiencia.

La taxonomía de los juegos serios se rige por siete modalidades según Salvat [19]:

- Juegos para la salud.
- Juegos publicitarios.
- Juegos para la formación
- Juegos para la educación.
- Juegos para la ciencia y la investigación.
- Producción y juegos como empleo, estas a su vez están asociadas a 7 sectores:
    - Gobiernos y ONGs (Organizaciones no gubernamentales).
    - Defensa.
    - Sistema de Salud.
    - Marketing y comunicaciones.
    - Educación.
    - Empresas.
    - Industria.

Los juegos serios se caracterizan por hacer uso de simulaciones, un ejemplo claro de esto es Virtual Safety and Education Platform [10], que es un juego que tiene como finalidad capacitar operarios de servicios de emergencia, operarios de maquinaria, personal de hospitales, ejercito, etc. Industry Player es otro juego, el cual se usa como herramienta para la formación de trabajadores en la empresa Tycoon Systems [9].

2.5 Juegos epistémicos

Son juegos que buscan la formación del jugador como profesional, ya que, este tipo de juegos introduce al jugador en entornos reales donde deberá poner en práctica conocimientos y competencias profesionales para resolver problemas cotidianos de distintas profesiones.

Los juegos epistémicos se adaptan a las exigencias del aprendizaje de la actualidad, porque los estudiantes toman roles de profesionistas, mientras se les inculca conocimiento de la profesión misma, conocimiento que pueden aplicar en el juego [20]. Un ejemplo de estos es el juego desarrollado por la Escuela de Administración y Dirección de Empresas Sloan del MIT que sumerge el jugador en el papel de un alto directivo de una compañía de videojuegos, y en el cual deberá desarrollar estrategias competitivas para el crecimiento de la compañía en la industria.

Los juegos epistémicos se basan en marcos epistémicos, y están estructurados de acuerdo con la cultura de la profesión a la que pertenecen, es decir por habilidades, valores, conocimiento y una epistemología de cómo actúan los profesionales creativos.

2.6 Trabajos relacionados

En un trabajo titulado “Videojuego Educativo para el Aprendizaje de SQL” de la Universidad Complutense de Madrid desarrollaron un proyecto cuyo objetivo principal es la realización de un videojuego que sirva como herramienta de aprendizaje y que permita adquirir una serie de conocimientos sobre el lenguaje de base de datos SQL, se desarrolló un software con herramientas gratuitas, capaz de

facilitar al usuario adquirir una serie de conocimientos del lenguaje SQL de forma estructurada, y de tal manera que la dificultad aumente mientras avanza el juego, además, está diseñado de tal forma que puedan hacerse ampliaciones sin necesidad de realizar cambios estructurados [21].

"Diseñando Videojuegos para Aprender de Forma Divertida: En Busca del Equilibrio Perdido" expone el equilibrio de diversión y aprendizaje que deben existir en un videojuego educativo, también remarca la importancia de que un juego educativo debe ser divertido (equilibradamente) para el jugador, así aprovechando el interés que pueda generar en él [22].

En el trabajo de tesis "Creación de un videojuego 3D educativo con metodología basada en la modificación de código" el autor describe el análisis, diseño e implementación de un videojuego 3D orientado a la enseñanza de conocimientos de programación. El videojuego se basa en modificar partes de un código de programación mediante una interfaz gráfica para que el personaje del juego complete niveles y avance en el juego. La metodología del juego permite que al modificar el código de muestren sus efectos en tiempo real [23].

"Implementación del prototipo de un videojuego interactivo para reforzar el aprendizaje del inglés en niños de 3 a 5 años" es un trabajo de tesis, donde los autores pretenden reforzar el aprendizaje del idioma inglés en niños de preescolar de 3 a 5 años planteando la aplicación de un videojuego como herramienta para motivar la enseñanza de este idioma. Además, los autores recopilan los métodos de enseñanza que se utilizan en las instituciones educativas en la actualidad con el objetivo de adaptar estas técnicas e introducirlas en el videojuego [24].

En el trabajo de tesis "Diseño e implementación de un videojuego de acción-aventura 2D para dispositivos android utilizando el motor unity" el autor aborda el tema del desarrollo de videojuegos utilizando tecnologías y herramientas especializadas, que permiten reducir tanto tiempo y complejidad que supone la producción de un videojuego. El autor, como objetivo, muestra la accesibilidad y gran potencial que poseen herramientas como Unity. El videojuego es de tipo

plataformas 2D el cual relata una historia ficticia relacionada con el campus Fernando May, sus docentes, estudiantes y edificios [25].

En el artículo "El proceso productivo del videojuego: fases de producción" el autor describe las distintas fases por las que pasa la producción de un videojuego, y como cierre también plantea la vinculación entre lo audiovisual y la informática, y también indica como a través de la optimización de los recursos disponibles en cada una de estas áreas se puede llegar a una estructura industrial y productiva ideal para una industria cultural como el videojuego [26].

En el trabajo de tesis "La lúdica de juegos en el aprendizaje de la programación orientada a objetos: un prototipo en c#" el autor implementa y desarrolla un videojuego cuyo objetivo consiste en aportar los cimientos teóricos de la programación orientada a objetos, mediante el estudio paulatino de los conceptos de clase, herencia, polimorfismo, sobrecarga de métodos y sentencias de control; el videojuego motiva el aprendizaje por medio de una serie de mini juegos donde los participantes interactúan con los conceptos de una manera lúdica [27].

2.7 Terminología empleada

2.7.1 Sprite

Se trata de un tipo de mapa de bits dibujados en la pantalla de ordenador por hardware gráfico especializado sin cálculos adicionales de la CPU.

También se denomina sprite a una serie de imágenes unidas en un mismo archivo una al lado de otra y que representan al mismo personaje (u objeto) en distintas posiciones. A diferencia de un archivo GIF animado, todas las imágenes están visibles en el mismo momento y puede estar codificado en un formato de imagen más ligero (en general PNG). Esto se utiliza principalmente en videojuegos para teléfonos móviles en formato J2ME, ya que el tiempo de carga de una imagen es muy superior al que ocupa seleccionar y mostrar un área de una imagen ya cargada en la memoria. También se utiliza en el diseño de skins para un software con la finalidad de reducir el número de archivos en el paquete que los contiene.

2.7.2 Asesprite

Aseprite es una herramienta de arte en píxeles que permite crear animaciones 2D para videojuegos.

Características:

- Creación de sprites usando capas y marcos como conceptos separados.
- Crear imágenes en RGB o modos de color indexados.
- Abrir y guardar animaciones GIF, secuencia de archivos PNG, etc.
- Una línea de tiempo donde puede copiar / mover capas, marcos o imágenes específicas
- Herramientas de animación: vista previa de animación en tiempo real.
- Herramientas de Pixel art: sombreado, trazos perfectos de píxeles, rotación, modo mosaico.
- Exportar e importar hojas de sprites.

2.7.3 Tiled

Tiled es un editor de nivel de software libre. Admite la edición de mapas de teselas en varias proyecciones (ortogonal, isométrica, hexagonal) y también admite niveles de construcción con imágenes libremente posicionadas, giradas o escaladas o anotándolas con objetos de diversas formas.

Características:

- Mapas ortogonales e isométricos.
- Formato de archivo XML.
- Soporte para múltiples capas tanto de tiles como de objetos.
- Soporte para múltiples tilesets.
- Soporte de color clave y transparencias.
- Tamaño de los tiles y mapas totalmente personalizable.

2.7.4 Unity

Unity es un motor de videojuego multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft

Windows, OS X, Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas. El motor gráfico utiliza OpenGL (en Windows, Mac y Linux), Direct3D (solo en Windows), OpenGL ES (en Android y iOS), e interfaces propietarias (Wii). Tiene soporte para mapeado de relieve, mapeado de reflejos, mapeado por paralaje, oclusión ambiental en espacio de pantalla, sombras dinámicas utilizando mapas de sombras, render a textura y efectos de post-procesamiento de pantalla completa.

El script se basa en Mono, la implementación de código abierto de .NET Framework. Los programadores pueden utilizar UnityScript (un lenguaje personalizado inspirado en la sintaxis ECMAScript), C# o Boo (que tiene una sintaxis inspirada en Python). A partir de la versión 3.0 añade una versión personalizada de MonoDevelop para la depuración de scripts.

2.7.5 Adobe Photoshop

Adobe Photoshop es un editor de gráficos desarrollado por Adobe Systems Incorporated y utilizado principalmente para el retoque de fotografías y gráficos. Traducido al español significa “taller de fotos” y es el líder mundial dentro del mercado de las aplicaciones de edición de imágenes en general.

Capítulo III. Desarrollo

3.1 Metodología de desarrollo

Una definición de metodología es "el conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentos y aspectos de formación para los desarrolladores de sistemas de información" [28].

Una metodología de desarrollo de software es una serie de pasos que se realizan de tal forma que permite a los desarrolladores elegir las técnicas para cada fase del proyecto y definir principalmente los participantes, sus actividades a desarrollar y una estructura de tiempo en el que deben de realizarse. Seguir una metodología de desarrollo facilita la planificación, control, gestión de requisitos, operación, mantenimiento y validación de proyectos [29].

Para elegir una metodología de desarrollo se necesita considerar que existen 2 principales clasificaciones, las tradicionales donde todo tiene que ser controlado y genera extensa documentación desde la planificación hasta la implementación del software, y la cual se utiliza en proyectos grandes donde su rendimiento y nivel de calidad son indispensables. La segunda, son las metodologías ágiles, donde parte de la información es obviada y se enfoca más en el desarrollo en menor tiempo, se utiliza en proyectos cuyos requisitos cambian constantemente antes de su finalización.

El conjunto de etapas y subetapas que sigue un proyecto, se denominan ciclo de vida del software, éste es fundamental para elegir la metodología de desarrollo a seguir ya que define las necesidades desde su concepción, los cambios o ajustes que puedan presentarse, el mantenimiento necesario, la realización de pruebas, hasta llegar a su finalización.

Tomando en cuenta estos puntos, se eligió utilizar la metodología de desarrollo ágil Scrum, porque demuestra un desarrollo que se adapta dinámicamente a las necesidades y requerimientos de la creación de un videojuego, además, se

pueden realizar tareas simultáneamente, permite flexibilidad, mitigación de riesgos y adaptación a los cambios a los que se pueda someter en todo su ciclo de vida.

3.1.1 Fases de la metodología Scrum

Scrum permite trabajar en una serie de iteraciones en equipo. El proceso de Scrum para este desarrollo se define en 3 fases secuenciales: pre-game, game y post-game.

Como puede verse en la figura 1, la fase de pre-game se realiza en una única iteración que comprende la definición del concepto del juego y la planificación de este, la fase game comprende a la implementación del videojuego y se trabaja de forma iterativa e incremental, la última fase tiene como objetivo evaluar y ajustar distintos aspectos del juego, eliminar errores detectados y, por último, entregar la versión final.

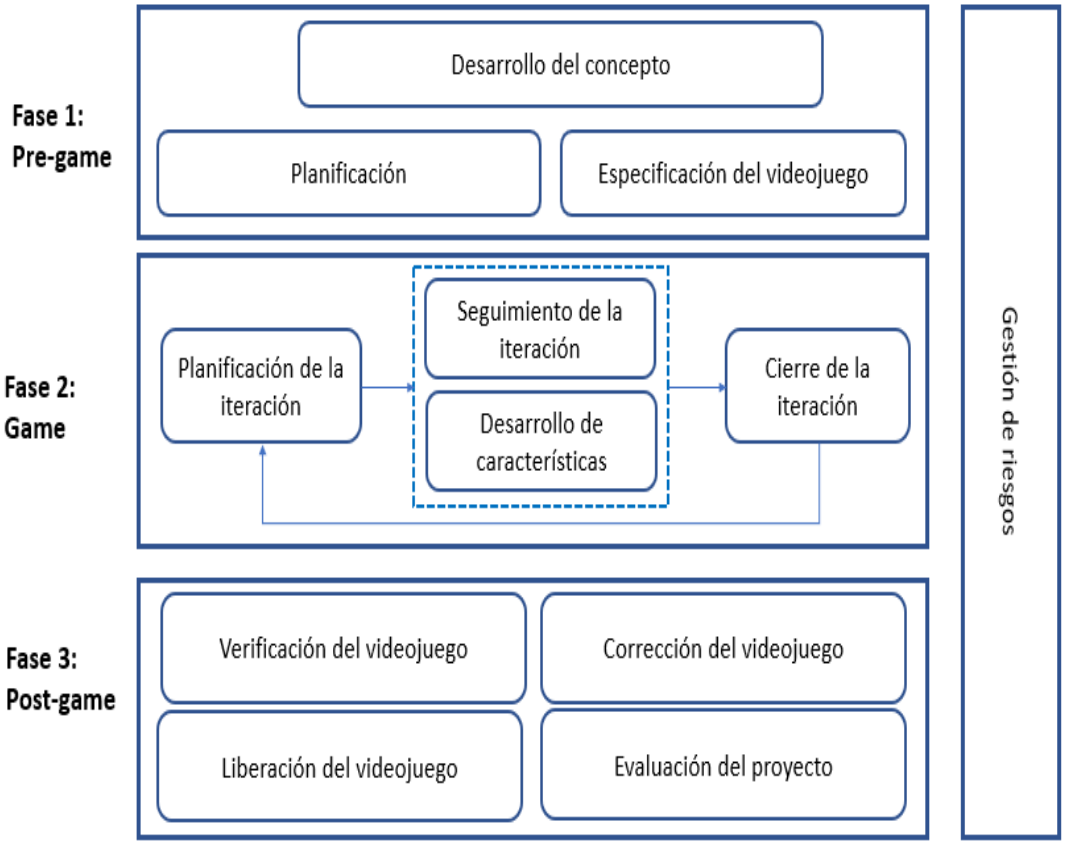


Figura 1. Diagrama de fases del desarrollo.

3.2 Planificación

3.2.1 Título

El título del videojuego es “Code Adventure”.

Figura 2. Título del videojuego.

3.2.2 Género

El género es una forma de clasificar a los videojuegos, principalmente se toma en cuenta en la mecánica del juego, aunque también se considera la temática, la estética, los controles y objetivos, entre otros. La clasificación por géneros es un sistema que se basa en las coincidencias entre distintos videojuegos por su jugabilidad básica y se ha formado a medida que han surgido numerosos juegos con características en común que permiten agruparlos, de esta manera se han podido dividir en subgéneros cada vez más específicos, por ejemplo los juegos de estrategia se caracterizan por manipular a varios personajes u objetos a la vez haciendo uso de la inteligencia y la planeación, para este caso existen 2 grandes subgéneros, estrategia en tiempo real y estrategia por turnos, también pueden clasificarse en juegos de guerra, construcción de imperios o juegos de gestión de tiempos.

Un videojuego puede pertenecer a más de un género, es común que muchos de ellos combinen más de 2 géneros distintos, normalmente en estos casos, se considera del género que predomine en él.

El desarrollo de "Code Adventure" entra dentro de este caso ya que es una combinación de distintos géneros, se considera un juego de estrategia ya que se caracteriza por manipular personajes para lograr objetivos haciendo uso de la inteligencia y la planificación. También contiene, aunque a menor medida, características de los juegos de rol, ya que se interactúa con el personaje y se complementa con una historia. El minijuego que contiene "Code Adventure" es del género puzzle ya que prueba la inteligencia del jugador para resolver problemas, en este caso, del tipo lógico.

Por otro lado, el género puede considerarse como educativo, ya que está dirigido a aportar conocimientos y habilidades al jugador durante el transcurso del juego.

3.2.3 Arquitectura del videojuego

El videojuego puede ser jugado desde cualquier sistema operativo sobre plataforma PC.

3.2.4 Versiones

Alpha: La versión Alpha es la primera versión funcional del videojuego. Esta primera versión cuenta con 10 niveles y no tiene sonidos de fondo, permitió identificar errores y áreas de oportunidad en ciertos aspectos del juego.

Beta: A la versión Beta del videojuego se añadió un apartado con 6 niveles extra en una modalidad de minijuego. Se añadieron sonidos de fondo. Se identificaron líneas de código que pueden ser optimizadas y validaciones no implementadas.

Versión 1.0: En la versión final del videojuego en el cual se aplicaron todos los cambios y modificaciones resultantes de la versión beta, se añadieron 10 niveles más, teniendo un total de 20, el código está optimizado, se diseñaron mejores interfaces para algunas pantallas y se verificaron las validaciones.

3.2.5 Mecánica

3.2.5.1 Mecánica del juego principal

En esta sección se describe la mecánica del juego principal, en la figura 3 se muestra una vista general de la pantalla de juego del nivel 1, misma que es similar para los demás niveles, únicamente cambiando el mapa, la posición y existencia de enemigos y obstáculos.

Figura 3. Pantalla del nivel 1.

Para completar el nivel, Scott, personaje principal detallado en la sección 3.3.10, debe pasar por cada uno de los obstáculos que se le presenten en el mapa, éstos pueden ser enemigos o rocas que deberá de destruir, también debe conseguir la guitarra de cada nivel, estas sirven como orientación mostrando el final del nivel, el

jugador debe de introducir una serie de comandos que se traducirán en acciones que Scott realizará.

En la pantalla del juego principal existe un campo de texto donde el usuario debe escribir el código. Las líneas introducidas en el campo de texto permitidas son las siguientes:

Movimiento:

Existen cuatro tipos de acciones relacionadas con movimiento, al ejecutarse éstas permiten el desplazamiento del personaje hacia la izquierda, derecha, arriba o abajo, dependiendo de la acción que se trate, el jugador debe introducir el número de pasos que dará el personaje hacia la dirección indicada, el juego permite un número de pasos entre 1 y 20.

En la tabla número 1 se observa la sintaxis de las acciones correspondientes a movimiento, donde # es el número de pasos:

Acción	Sintaxis
Moverse a la derecha.	Scott.Derecha(#);
Moverse a la izquierda.	Scott.Izquierda(#);
Moverse para arriba.	Scott.Arriba(#);
Moverse para abajo	Scott.Abajo(#);

Tabla 1. Acciones de movimiento del personaje.

Ataque:

El personaje también puede realizar una acción de ataque, éste se realiza hacia donde se encuentre orientado el personaje, por ejemplo, en la figura 4 se muestra al personaje realizando un ataque cuando su orientación es hacia la derecha.

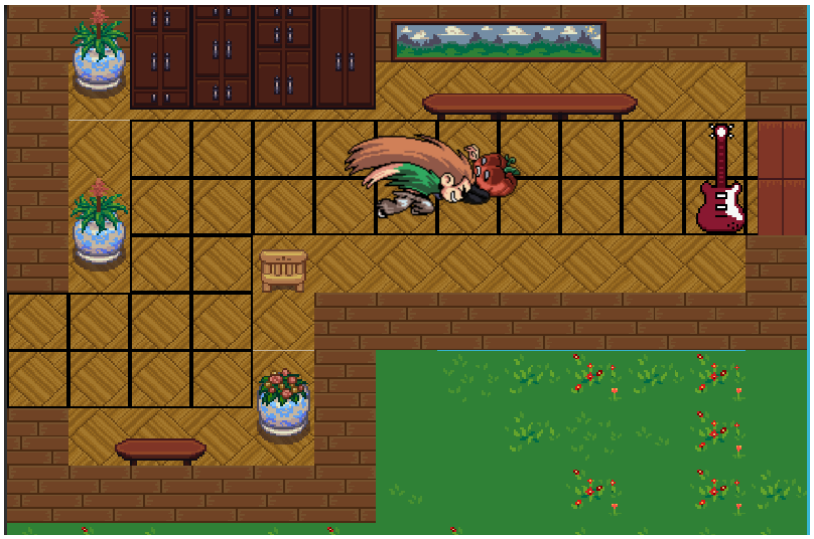


Figura 4. Ataque del personaje.

El ataque acaba con los obstáculos que el personaje se pueda encontrar en su camino, éstos pueden ser enemigos o piedras. Su sintaxis se encuentra en la tabla número 2.

Acción	Sintaxis
Atacar con un cabezazo.	Scott.Cabezazo();

Tabla 2. Acciones de ataque del personaje.

Después de que el jugador introduce el código que quiere ejecutar, debe pulsar el botón "Correr", se observa en la figura 5.

Figura 5. Botón de correr.

A continuación, el código introducido se analiza por el script "Script_Juego", esto para evaluar cada una de las líneas introducidas y encontrar posibles errores de sintaxis, si las líneas de código introducidas son correctas estas proceden a interpretarse, traduciéndose en acciones que realiza el personaje del

juego, en la sección 3.4.3 se describe a más detalle el funcionamiento del script "Script_Juego".

3.2.5.2 Mecánica del minijuego

Se creó una sección de minijuegos con el objetivo de enseñar a los niños lo que es la sucesión de pasos para llegar a un resultado, en donde el jugador tiene que acomodar ciertas imágenes para que la historia siga un orden congruente, en total, este minijuego consta de 6 niveles. En la figura 6 se muestra el nivel 2 del minijuego con las imágenes desordenadas.

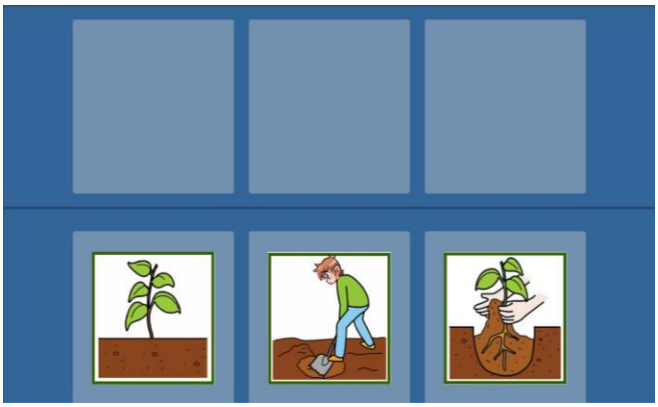


Figura 6. Nivel 2 desordenado.

En la figura número 7, las imágenes se ordenaron de forma cronológica.

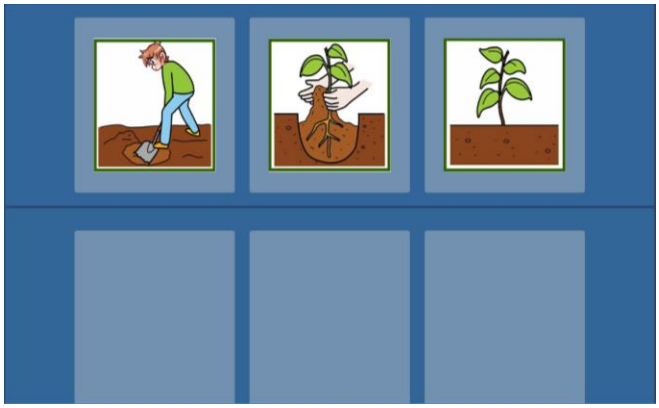


Figura 7. Nivel 2 ordenado correctamente.

3.2.6 Cámara

Para el caso de "Code Adventure" la plataforma de desarrollo es en 2D, lo recomendado para este tipo de juegos es una cámara fija, su posición, distancia focal y campo de visión no son cambiantes, por lo que, además, permite consumir menos recursos.

3.2.7 Periféricos

Ya que el videojuego es para PC, los periféricos fundamentales para jugar son teclado y ratón. El teclado permite capturar las instrucciones dadas por el jugador y el ratón permite desplazarse por las diferentes pantallas y niveles, así como poner en marcha las líneas de código escritas, haciendo clic en los botones que se encuentran en el videojuego. Para una mejor experiencia se recomienda utilizar bocinas que permiten escuchar el sonido de fondo del videojuego.

3.2.8 Público

El jugador, es el usuario final de los videojuegos, es a quien se debe de analizar para ofrecer una buena experiencia del juego, saber que le gusta, como se divierte para poder ofrecerle un producto adecuado a sus necesidades. En el mundo de los videojuegos hay distintos grupos de jugadores, Rodrigo Villanueva, periodista de la revista Level Up, hace una clasificación por la taxonomía del jugador donde se toma en cuenta su forma de ser, su personalidad y sus preferencias de carácter psicológico [30]. El mercado de los videojuegos también se divide en hombres y mujeres, en general, los hombres prefieren juegos que suponen desafíos, competitividad y destructividad, y las preferencias femeninas incluyen resolución de puzzles, exploración de mundos y cuidados de otros.

El público de un videojuego también puede definirse de acuerdo con su experiencia, están los jugadores profesionales cuyas habilidades están muy desarrolladas, pasan muchas horas practicando para torneos y hasta pueden llegar a vivir de esto. Los jugadores duros son aquellos que dedican muchas horas al día jugando, su objetivo es tener puntuaciones máximas, no busca solo un medio de entretenimiento sino un reto o aventura. Los jugadores casuales no están interesados

en mejorar sus habilidades o tener puntuaciones altas, para ellos es un pasatiempo y no le invierten muchas horas de su día a los videojuegos.

Con la diversidad de aparatos electrónicos existentes, muchos jugadores no pueden permitirse contar con todas las plataformas del mercado, así que otra forma de dividir a los usuarios finales es según la plataforma a la que jueguen, las más comunes son los jugadores de PC, que se dedican a jugar juegos para computadoras, muchos otros utilizan consolas para jugar, sus smartphones y tabletas.

Una clasificación fundamental del público de los videojuegos es por edades, se debe de regular el contenido presentado de acuerdo con el rango de edad al que va enfocado, hay contenido especializado en infantes desde los 0 años, de preescolar, adolescencia, adultos, etc.

El público al que va dirigido "Code Adventure" son niños desde los 7 años, que es cuando un infante lee y razona.

3.2.9 Dificultad

En un videojuego la dificultad se regula normalmente mediante las misiones, con unos escenarios más cortos y sencillos al principio y a medida que el personaje avanza se incrementa la duración y complejidad, esto con el objetivo de atraer la atención de los jugadores y mantener su interés.

Para determinar la dificultad de un videojuego se requiere considerar aspectos como el diseño de los escenarios, la historia, el modo y el ritmo de juego. Ya que, por ejemplo, si el ritmo del juego es lento y los niveles tienen una dificultad alta en donde se requiera de mucho tiempo para pasarlos, probablemente el usuario pierda interés en el juego.

Se determinan tres tipos de dificultad en los videojuegos, la física, donde se ponen a prueba las habilidades cognitivas y motoras en conjunto, la mental, que es una dificultad deductiva que muestra la agilidad del usuario ante la resolución de problemas y la emocional, donde interactúan factores como el miedo, la ansiedad, entre otros [31].

Para el caso de “Code Adventure”, la dificultad mental es la que predomina ya que no intervienen habilidades motoras ni emocionales, se determinó que la dificultad sea incremental, que los primeros niveles se resuelvan con pocas líneas de código y al avanzar se requieran de más líneas de código para resolver el nivel correspondiente. Por lo que se determinó que la dificultad de este juego se componga por fácil, medio y difícil.

3.2.10 Captura de requisitos funcionales y no funcionales

Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. El proceso de descubrir, analizar, documentar y verificar estos requerimientos se denomina ingeniería de requerimientos. La ingeniería de requisitos es importante en el desarrollo de un software ya que permite especificar los requerimientos de un software y esto sirve como base para desarrollar dicho sistema. En la ingeniería de requisitos se suele usar el termino stakeholder, que se acuña a la parte interesada del software en desarrollo.

El descubrimiento de requerimientos es el proceso de recoger información sobre el sistema propuesto y extraer los requerimientos del usuario y del sistema de esta información. A continuación, se resumen algunas de las técnicas del descubrimiento de requisitos descritas por Ian Sommerville [32]:

- **Puntos de vista:** Un punto clave del análisis orientado a puntos de vista es, reconocer varias perspectivas y proporcionar un marco de trabajo para descubrir conflictos en los requerimientos propuestos por diferentes stakeholders.

  Los puntos de vista se pueden utilizar como una forma de clasificar los stakeholders y otras fuentes de requerimientos. Existen 3 tipos genéricos de puntos de vista:

  - Puntos de vista de los interactuadores: Estos proporcionan requerimientos detallados del sistema que cubren las características e interfaces de este.

- Puntos de vista indirectos: Son aquellos que proporcionan requerimientos y restricciones organizacionales de alto nivel, representan los stakeholders que no utilizan el sistema pero que influyen en los requerimientos de algún modo.
- Puntos de vista del dominio: Estos representan las características y restricciones del dominio que influyen en los requerimientos del sistema.

- **Entrevistas:** Las entrevistas formales o informales con stakeholders del sistema son parte de la mayoría de los procesos de la ingeniería de requerimientos. En estas entrevistas, el equipo de ingeniería de requerimientos hace preguntas a los stakeholders sobre el sistema que utilizan y sobre el sistema a desarrollar. Las entrevistas sirven para obtener una comprensión general de lo que hacen los stakeholders, cómo podrían interactuar con el sistema y las dificultades a las que se enfrenta con los sistemas actuales. Los requerimientos provienen de las respuestas a estas preguntas. Las entrevistas pueden ser de dos tipos:
  - Entrevistas cerradas: donde los stakeholders responden a un conjunto predefinido de preguntas.
  - Entrevistas abiertas donde no hay un programa predefinido. El equipo de la ingeniería de requerimientos examina una serie de cuestiones con los stakeholders del sistema y, por lo tanto, desarrolla una mejor comprensión de sus necesidades.

  En la práctica, las entrevistas con los stakeholders normalmente son una mezcla de estos.

- **Escenarios:** Las personas encuentran más fácil dar ejemplos de la vida real que descripciones abstractas. Pueden comprender y criticar un escenario de cómo podrían interactuar con un sistema de software. Los ingenieros de requerimientos pueden utilizar la información obtenida de esta discusión para formular los requerimientos reales del sistema. Los escenarios son

descripciones de ejemplos de las sesiones de interacción. Cada escenario abarca una o más posibles interacciones.

- **Casos de uso:** Los casos de uso son una técnica que se basa en escenarios para la obtención de requerimientos. Actualmente se han convertido en una característica fundamental de la notación de UML, que se utiliza para describir modelos de sistemas orientados a objetos. En su forma más simple un caso de uso identifica el tipo de interacción y los actores involucrados. Los casos de uso identifican las interacciones particulares con el sistema.

3.2.10.1 Casos de uso

Para obtener los requerimientos del videojuego se utilizaron casos de uso, ya que estos ayudan a describir la interacción que tiene el usuario con el software en distintos escenarios. A continuación, se muestran y se describen los casos de uso diseñados.

Caso de uso: Continuar juego

Figura 8. Caso de uso "Continuar juego".

Identificador	Continuar juego	
Descripción	Este caso de uso se genera cuando el jugador quiere continuar su partida en el último nivel jugado.	
Secuencia Normal	**Paso**	**Acción**
	1	El jugador continúa partida en el último nivel videojuego.
	2	El videojuego carga el último nivel completado.
Excepciones	**Paso**	**Acción**
	2	El jugador no ha completado ningún nivel, por lo tanto, se carga el nivel 1.
Rendimiento	El tiempo estimado para que se reproduzca este caso de uso es de 5-15 segundos.	
Frecuencia	Alta	
Importancia	Vital	
Comentarios	Ninguno	

Tabla 3. Descripción de caso de uso "Continuar juego".

Caso de uso: Ir al menú de selección de nivel del juego principal

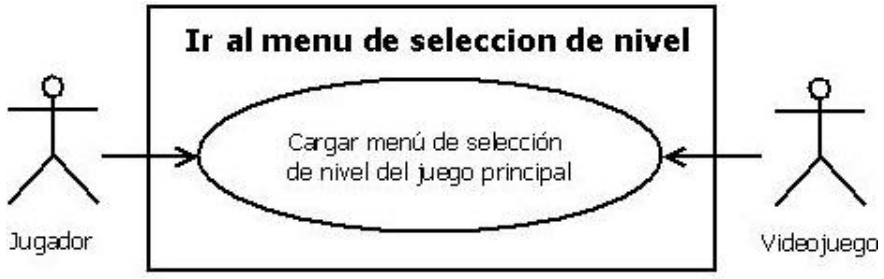


Figura 9. Caso de uso "Ir al menú de selección de nivel".

Identificador	Ir al menú de selección de nivel	
Descripción	Este caso de uso se desarrolla cuando el jugador decide escoger un nivel determinado para jugar.	
Secuencia Normal	**Paso**	**Acción**
	1	El jugador escoge la opción "Escoger nivel", en automático el videojuego cargara esa pantalla.
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	El tiempo estimado para que se reproduzca este caso de uso es de 1-5 segundos.	
Frecuencia	Media	
Importancia	Vital	
Comentarios	Ninguno	

Tabla 4. Descripción de caso de uso "Ir al menú de selección de nivel".

Caso de uso: Ir al menú de selección de nivel del minijuego

Figura 10. Caso de uso "Selección del minijuego".

Identificador	Ir al menú de selección de nivel de minijuego	
Descripción	Este caso de uso se desarrolla cuando el jugador decide escoger un nivel del minijuego para jugar.	
Secuencia Normal	**Paso**	**Acción**
	1	El jugador escoge la opción "Escoger nivel Minijuego", en automático el videojuego cargara esa pantalla.
Excepciones	**Paso**	**Acción**
	-	-
Rendimiento	El tiempo estimado para que se reproduzca este caso de uso es de 1-5 segundos.	
Frecuencia	Media	
Importancia	Vital	
Comentarios	Ninguno	

Tabla 5. Descripción de caso de uso "Selección de nivel de minijuego".

Caso de uso: Ir a pantalla de Acerca_de

Figura 11. Caso de uso "Ir a pantalla de Acerca de".

Identificador	Ir a pantalla de Acerca_de	
Descripción	Este caso de uso se presenta cuando el usuario se dirige a la pantalla de acerca de, esta pantalla muestra información de los desarrolladores del videojuego.	
Secuencia Normal	**Paso**	**Acción**
	1	El jugador escoge la opción "Acerca_De", en automático el videojuego cargará esa pantalla.
Excepciones	**Paso**	**Acción**
	-	**-**
Rendimiento	El tiempo para que se reproduzca este caso de uso es de 1-5 segundos.	
Frecuencia	Baja	
Importancia	Baja	
Comentarios	Ninguno	

Tabla 6. Descripción de caso de uso "Ir a pantalla de Acerca_de".

Caso de uso: Salir del juego

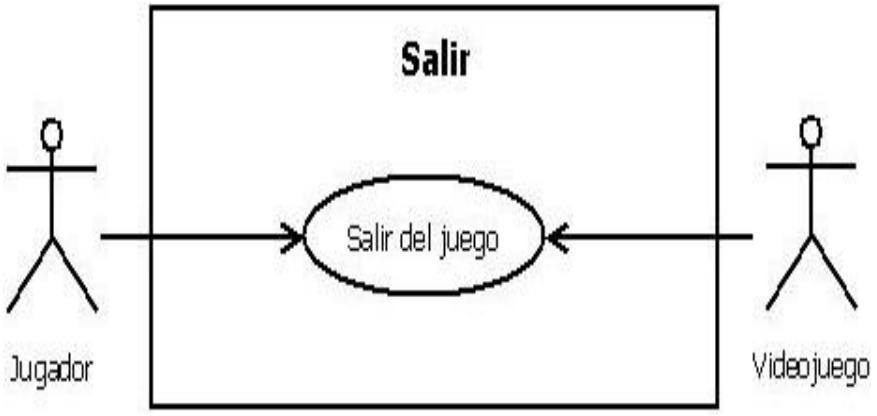


Figura 12. Caso de uso "Salir del videojuego".

Identificador	Salir	
Descripción	Caso de uso que sucede cuando el usuario quiere salir del videojuego.	
Secuencia Normal	**Paso**	**Acción**
	1	El jugador escoge la opción "Salir", en automático el videojuego se cierra.
Excepciones	**Paso**	**Acción**
	-	**-**
Rendimiento	El tiempo que se estima para que se reproduzca este caso de uso es de 1-5 segundos.	
Frecuencia	Alta	
Importancia	Importante	
Comentarios	Ninguno	

Tabla 7. Descripción de caso de uso "Salir".

Caso de uso: Selección de nivel del juego principal

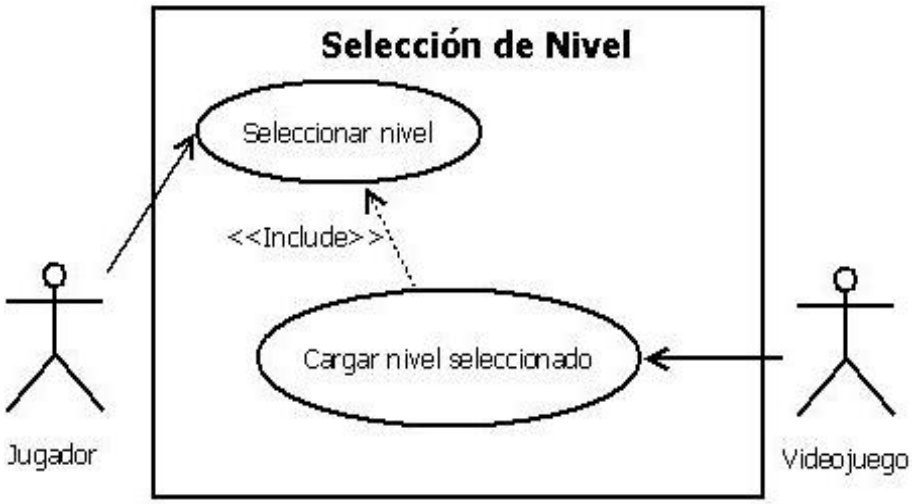


Figura 13. Caso de uso "Selección de nivel".

Identificador	Selección de nivel	
Descripción	Escenario que se presenta cuando el jugado selecciona un nivel del videojuego principal.	
Secuencia Normal	**Paso**	**Acción**
	1	El jugador selecciona el nivel que quiere jugar.
	2	El videojuego deberá carga el nivel seleccionado por el jugador.
Excepciones	**Paso**	**Acción**
	1	En el caso del que un nivel no esté desbloqueado aun, el jugador no podrá seleccionarlo y por ende el videojuego no podrá cargar el nivel.
	2	Si el jugador no selecciona algún nivel el videojuego no cargará ningún nivel.
Rendimiento	El tiempo estimado en que se debe completar esta serie de acciones depende del jugador.	
Frecuencia	Media	
Importancia	Importante	
Comentarios	Si el jugador no ha completado ningún nivel, solo podrá seleccionar el nivel 1.	

Tabla 8. Descripción de caso de uso "Selección de nivel".

Caso de uso: Selección de nivel del minijuego

Figura 14. Caso de uso "Selección de nivel Minijuego".

Identificador	Selección de nivel minijuego		
Descripción	Escenario que se presenta cuando el jugado selecciona un nivel del minijuego.		
Secuencia Normal	**Paso**	**Acción**	
	1	El jugador selecciona el nivel que quiere jugar.	
	2	El videojuego deberá carga el nivel seleccionado por el jugador.	
Excepciones	**Paso**	**Acción**	
	1	Si el jugador no selecciona algún nivel el videojuego no cargara ningún nivel.	
Rendimiento	El tiempo estimado en que se debe completar esta serie de acciones depende del jugador.		
Frecuencia	Media		
Importancia	Importante		
Comentarios	El jugador podrá acceder a cualquier nivel del minijuego.		

Tabla 9. Descripción de caso de uso "Selección de nivel de minijuego".

Caso de uso: Consultar información Acerca_de:

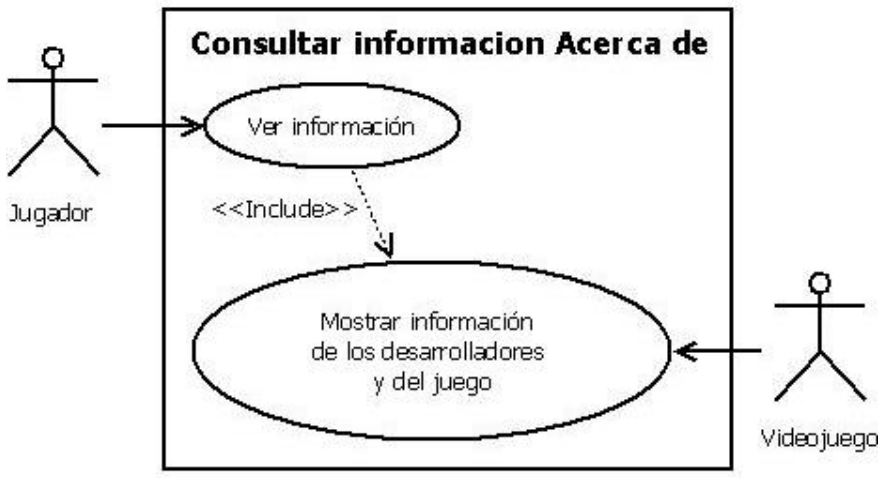


Figura 15. Caso de uso "Consultar información Acerca de".

Identificador	Ir a pantalla de Acerca_de		
Descripción	Este caso de uso se presenta cuando el usuario se dirige a la pantalla de acerca de, esta pantalla muestra información de los desarrolladores del videojuego.		
Secuencia Normal	**Paso**	**Acción**	
	1	El jugador escoge la opción "Acerca_De", en automático el videojuego cargara esa pantalla.	
Excepciones	**Paso**	**Acción**	
	-	-	
Rendimiento	El tiempo que se estima para que se reproduzca este caso de uso es de 1-5 segundos.		
Frecuencia	Baja		
Importancia	Baja		
Comentarios	Ninguno		

Tabla 10. Descripción de caso de uso "Ir a pantalla de Acerca de".

Caso de uso: Consultar ayuda

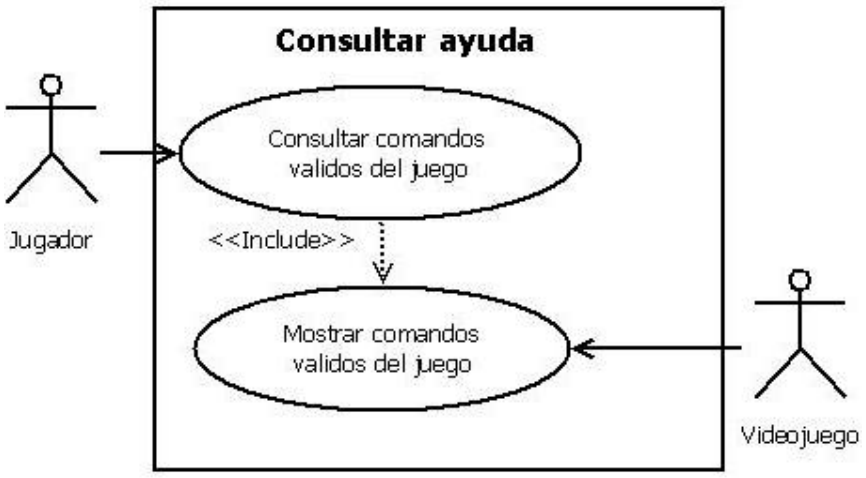


Figura 16. Caso de uso "Consultar ayuda".

Identificador	Mostrar Ayuda	
Descripción	Este caso de uso sucede cuando el usuario tiene dudas acerca de cómo se juega el videojuego.	
Secuencia Normal	**Paso**	**Acción**
	1	El jugador consulta información sobre como jugar
	2	El juego muestra la información acerca de cómo jugar
Excepciones	**Paso**	**Acción**
	-	**-**
Rendimiento	El tiempo estimado para que se reproduzca este caso de uso es de 1-5 segundos	
Frecuencia	Media	
Importancia	Importante	
Comentarios	Ninguno	

Tabla 11. Descripción de caso de uso "Mostrar ayuda".

Caso de uso: Minijuego

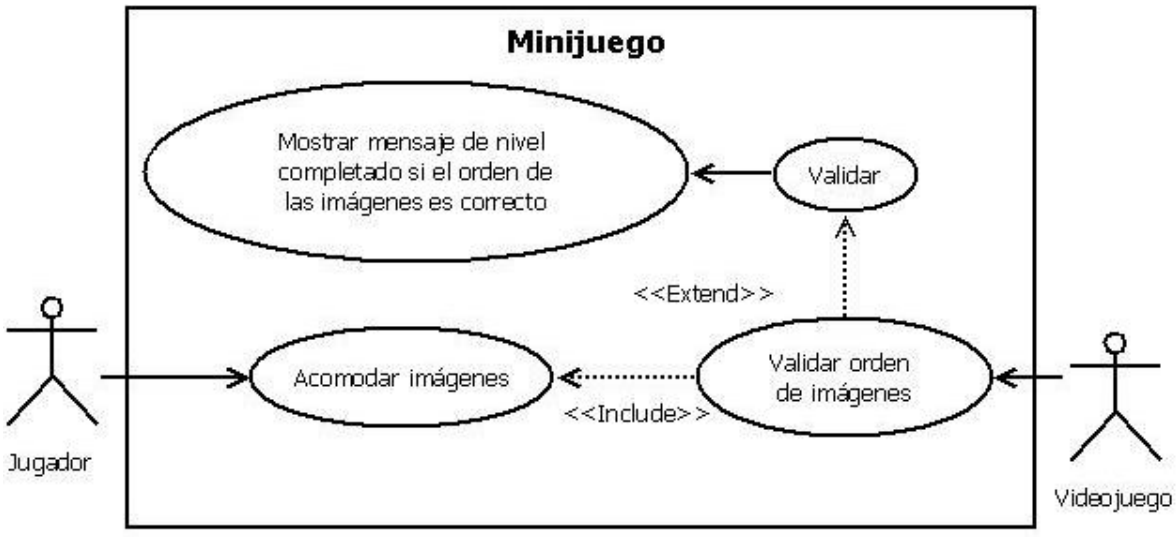


Figura 17. Caso de uso "Minijuego".

Identificador	Minijuego		
Descripción	Caso de uso presentado al jugar un minijuego.		
Secuencia Normal	**Paso**	**Acción**	
	1	El jugador ordena imágenes y el juego valida el orden.	
	2	Mostrar mensaje de nivel completado	
Excepciones	**Paso**	**Acción**	
	3	El jugador puede no completar el minijuego, vuelve al menú principal o sale del videojuego.	
Rendimiento	El tiempo estimado en que se debe completar este caso de uso depende del tiempo que se tome el jugador para resolver el nivel.		
Frecuencia	Alta	Importancia	Vital
Comentarios	La validación se ejecuta constantemente, solo se puede completar el nivel hasta que el orden de las imágenes sea correcto.		

Tabla 12. Descripción de caso de uso "Minijuego".

Caso de uso: Juego principal

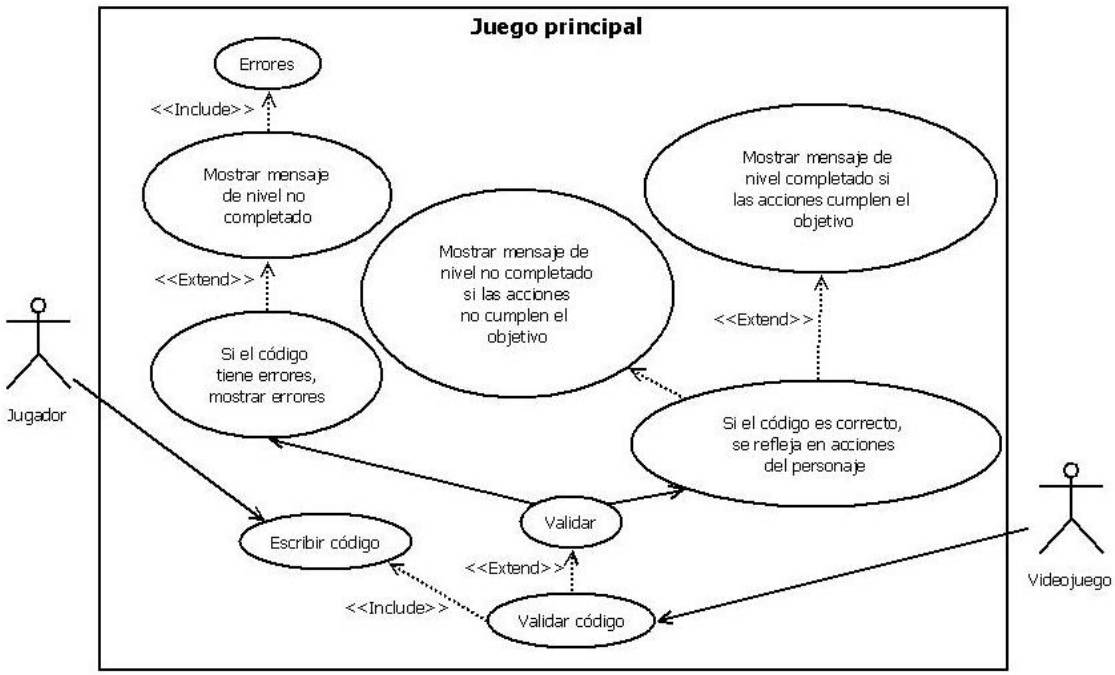


Figura 18. Caso de uso "juego principal".

Identificador	Juego principal	
Descripción	Este caso de uso representa cuando el usuario está jugando cualquier nivel del juego principal.	
Secuencia Normal	**Paso**	**Acción**
	1	El jugador escribe código.
	2	El videojuego valida el código.
	3	El videojuego traduce en acciones del personaje el código introducido por el jugador.
	4	El videojuego ejecuta acciones del personaje.
	5	El videojuego muestra mensaje de nivel completado.
Excepciones	**Paso**	**Acción**
	2	Si el código está vacío se muestra un mensaje de error
	2	Si el código introducido por el usuario tiene errores, se muestra un mensaje de error.
		Las acciones ejecutadas por el personaje no cumplieron con el objetivo del nivel, se muestra mensaje de nivel fallado.
Rendimiento		El tiempo estimado en que se debe completar este caso de uso depende del tiempo que se tome el jugador para resolver el nivel.
Frecuencia		Alta
Importancia		Vital
Comentarios		Ninguno

Tabla 13. Descripción de caso de uso "Juego principal".

3.2.10.2 Requerimientos funcionales

Los requerimientos funcionales expresan cómo interacciona un sistema con su entorno, sus detalles técnicos y en general, las funcionalidades específicas que debe cumplir. De la tabla 14 a la tabla 24, corresponden con los requisitos funcionales del videojuego:

Identificación del requerimiento:	**RF01**
Nombre del Requerimiento:	Iniciar el juego.
Descripción del requerimiento:	El usuario debe dar clic en el ejecutable del juego. El sistema lanza el juego sin errores y cargando los recursos gráficos necesarios.
Requerimiento NO funcional:	• RNF01 • RNF02 • RNF03
Prioridad del requerimiento: **Alta**	

Tabla 14. RF01 "Iniciar el juego".

Identificación del requerimiento:	**RF02**
Nombre del Requerimiento:	Mostrar el menú principal.
Descripción del requerimiento:	El sistema carga el menú principal. El usuario puede seleccionar la opción que desee.
Requerimiento NO funcional:	• RNF01 • RNF02 • RNF03
Prioridad del requerimiento: **Alta**	

Tabla 15. RF02 "Mostrar el menú principal".

Identificación del requerimiento:	**RF03**
Nombre del Requerimiento:	Mostrar el selector de niveles del menú principal.
Descripción del requerimiento:	El usuario selecciona el selector de niveles en el menú principal. El sistema muestra la interfaz del selector de niveles. El usuario puede seleccionar un nivel o regresar al menú principal.
Requerimiento NO funcional:	• RNF01 • RNF02 • RNF03
Prioridad del requerimiento:	**Alta**

Tabla 16. RF03 "Selector de niveles".

Identificación del requerimiento:	**RF04**
Nombre del Requerimiento:	Mostrar el selector de niveles del minijuego.
Descripción del requerimiento:	El usuario selecciona el selector de niveles del minijuego en el menú principal. El sistema muestra la interfaz del selector de niveles del minijuego. El usuario puede seleccionar un nivel o regresar al menú principal.
Requerimiento NO funcional:	• RNF01 • RNF02 • RNF03
Prioridad del requerimiento: **Alta**	

Tabla 17. RF04 "Selector de niveles de minijuego".

Identificación del requerimiento:	**RF05**
Nombre del Requerimiento:	Mostrar la pantalla de “Acerca de”.
Descripción del requerimiento:	El usuario selecciona “Acerca de” en el menú principal. El sistema muestra la interfaz de “Acerca de”. El usuario puede regresar al menú principal.
Requerimiento NO funcional:	• RNF01 • RNF02 • RNF03
Prioridad del requerimiento: **Baja**	

Tabla 18. RF05 "Mostrar pantalla de "Acerca de".

Identificación del requerimiento:	**RF06**
Nombre del Requerimiento:	Salir del videojuego.
Descripción del requerimiento:	El usuario selecciona el botón de “Salir”. El sistema cierra el juego.
Requerimiento NO funcional:	• RNF01 RNF02 RNF03
Prioridad del requerimiento:	**Alta**

Tabla 19. RF06 "Salir del videojuego".

Identificación del requerimiento:	**RF07**
Nombre del Requerimiento:	Iniciar nivel.
Descripción del requerimiento:	El usuario selecciona un nivel del videojuego. El sistema muestra la interfaz del nivel seleccionado. El usuario puede comenzar a jugar, ver la ayuda o regresar al menú principal.
Requerimiento NO funcional:	• RNF01 RNF02 RNF03
Prioridad del requerimiento:	**Alta**

Tabla 20. RF07 "Iniciar nivel".

Identificación del requerimiento:	**RF08**
Nombre del Requerimiento:	Escribir el código y ejecutarlo.
Descripción del requerimiento:	El usuario escribe el código para resolver el nivel. El usuario da clic en el botón “Correr”. El sistema mueve al personaje de acuerdo con las acciones dadas por el usuario. El sistema verifica el código escrito por el usuario. Si el código es correcto, el sistema muestra la interfaz de “Nivel completado”, si el código es incorrecto, el sistema muestra la interfaz de “Reintentar”.
Requerimiento NO funcional:	• RNF01 RNF02 RNF03
Prioridad del requerimiento:	**Alta**

Tabla 21. RF08 "Escribir el código y ejecutarlo".

Identificación del requerimiento:	**RF09**
Nombre del Requerimiento:	Reiniciar nivel.
Descripción del requerimiento:	El usuario da clic en el botón de "Reintentar". El sistema carga la interfaz del nivel actual. El usuario puede jugar nuevamente o regresar al menú principal.
Requerimiento NO funcional:	• RNF01 • RNF02 • RNF03
Prioridad del requerimiento: **Alta**	

Tabla 22. RF09 "Reiniciar nivel".

Identificación del requerimiento:	**RF010**
Nombre del Requerimiento:	Mostrar pantalla de ayuda.
Descripción del requerimiento:	El usuario da clic en el botón de ayuda. El sistema carga la interfaz de ayuda.
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento: **Alta**	

Tabla 23. RF010 "Mostrar pantalla de ayuda".

Identificación del requerimiento:	**RF011**
Nombre del Requerimiento:	Regresar al menú principal.
Descripción del requerimiento:	El usuario da clic en el botón de "Volver". El sistema carga la interfaz del menú principal.
Requerimiento NO funcional:	• RNF01 • RNF02 • RNF03
Prioridad del requerimiento: **Alta**	

Tabla 24. RF011 "Regresar al menú principal".

3.2.10.3 Requerimientos no funcionales

Los requerimientos no funcionales representan características generales y/o restricciones de un sistema, las tablas 25, 26 y 27 corresponden a los requisitos no funcionales del videojuego:

Identificación del requerimiento:	**RNF01**
Nombre del Requerimiento:	Usabilidad del videojuego.
Descripción del requerimiento:	El videojuego ha de ser intuitivo al uso, debe ser fácil de utilizar a simple vista.
Prioridad del requerimiento: **Alta**	

Tabla 25. RNF01 "Usabilidad del videojuego".

Identificación del requerimiento:	**RNF02**
Nombre del Requerimiento:	Rendimiento del videojuego.
Descripción del requerimiento:	El videojuego debe de ejecutarse sin errores en computadoras con requisitos mínimos de software.
Prioridad del requerimiento: **Alta**	

Tabla 26. RNF02 "Rendimiento del videojuego".

Identificación del requerimiento:	**RNF03**
Nombre del Requerimiento:	Compatibilidad con resoluciones.
Descripción del requerimiento:	La aplicación será compatible con diferentes resoluciones de pantalla.
Prioridad del requerimiento: **Alta**	

Tabla 27. RNF03 "Compatibilidad con resoluciones".

3.3 Diseño

Para el desarrollo de un videojuego es fundamental comenzar imaginando la historia, cómo serán los personajes y el escenario en el cual se desarrolla el juego, con que objetos va a interactuar y los objetivos a alcanzar.

Todos estos factores deben estar muy bien definidos antes de comenzar con el diseño o la programación ya que son las bases para un buen desarrollo.

3.3.1 Diseño de botones, iconos y otros elementos de la interfaz

Para contar con una interfaz que vaya de acuerdo con el estilo píxel art del juego se necesitó diseñar ciertos elementos indispensables para el juego, por ejemplo: botones y paneles que forman parte de la interfaz gráfica del usuario, estos elementos se diseñaron con Asesprite y Photoshop, para posteriormente utilizarse en Unity.

También se utilizó el tipo de fuente "04b03", ya que esta va de acuerdo con el estilo píxel art, esta fuente se encontró en la página web (https://www.dafont.com/es/04b-03.font). Los botones diseñados se muestran en la figura 19.

Figura 19. Botones diseñados.

En la figura 20, se pueden ver algunos recursos diseñados que sirven para mostrar información al usuario para que logre identificar los diferentes niveles del videojuego por su número.

Figura 20. Recursos para mostrar información.

Los elementos de la figura 21 corresponden a paneles que sirven como contenedores conformados por botones, texto y otros elementos de la interfaz gráfica.

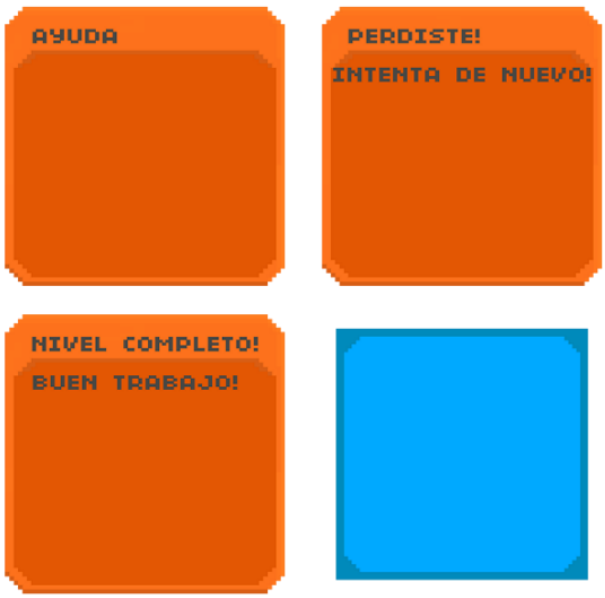


Figura 21. Paneles diseñados.

En la etapa de desarrollo se dejaron atrás algunos elementos diseñados que no se utilizaron, ya que mientras se construían las escenas se descartaron aquellos que no armonizaban con las escenas o que fueron evolucionando y se les hicieron cambios de color o forma, algunos de estos elementos descartados se muestran en la figura 22.

Figura 22. Elementos descartados.

3.3.2 Interfaces

En la figura 23, se muestra la interfaz diseñada para la pantalla principal del videojuego, donde se pueden seleccionar las opciones de continuar, seleccionar un nivel en específico o un nivel de los minijuegos, información sobre los desarrolladores y la opción de salir. Para hacer esta interfaz más agradable a la vista, se colocó en la parte inferior izquierda una animación de Scott Pilgrim y Ramona Flowers, personajes descritos en la sección 3.3.10.

Figura 23. Diseño de interfaz del menú principal.

3.3.3 Menú de selección de niveles del juego principal

Para que el usuario pueda seleccionar un nivel para jugar, se creó una interfaz que muestra todos los niveles existentes, tanto los niveles desbloqueados y los bloqueados.

Dado que el número de niveles es 20 y esto toma mucho espacio, para distribuir todos los botones de los niveles en un solo contenedor, se colocó un Scroll Rect Vertical, este es un componente visual de Unity que permite desplazarse sobre un contenido amplio que es mostrado en un área pequeña. El resultado del diseño de la interfaz de selección de nivel se muestra en la figura 24.

Figura 24. Diseño de la interfaz de "Escoger nivel".

3.3.4 Menú de selección de niveles del minijuego

También se realizó una interfaz para que el usuario seleccione cualquier nivel del minijuego, detallados en la sección 3.3.7, dando clic en alguno de los recuadros que corresponden a los seis niveles existentes, mostrados en la figura 25.

Figura 25. Diseño de la interfaz de “Escoger nivel” del minijuego.

3.3.5 Pantalla Acerca_De

En la figura 26, se observa el diseño de la interfaz que muestra la información de los desarrolladores y un botón para regresar al menú principal. Para armonizar más esta interfaz se colocó una animación de Scott Pilgrim y otra animación de Ramona Flowers, dichas animaciones se reproducen cuando el usuario pulsa los botones con los iconos de los personajes.

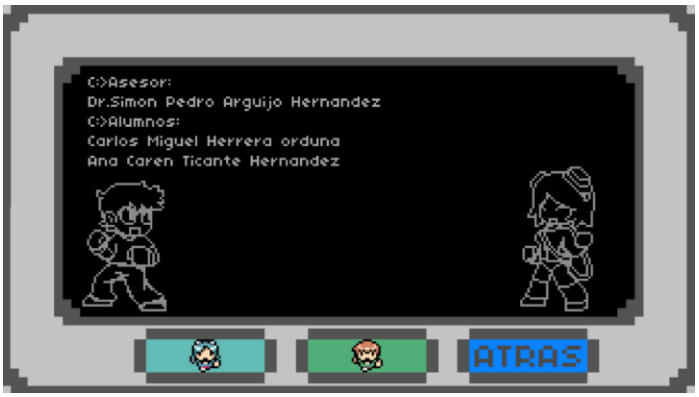


Figura 26. Pantalla de "Acerca de".

3.3.6 Juego principal

Para la pantalla de juego principal se distribuyó el contenido como se puede observar en la figura 27, tomando como base una resolución de 1280x720, cada nivel cuenta con un panel principal, mientras que los mapas son diferentes para cada nivel, los niveles se describen en la sección 3.3.11.1.

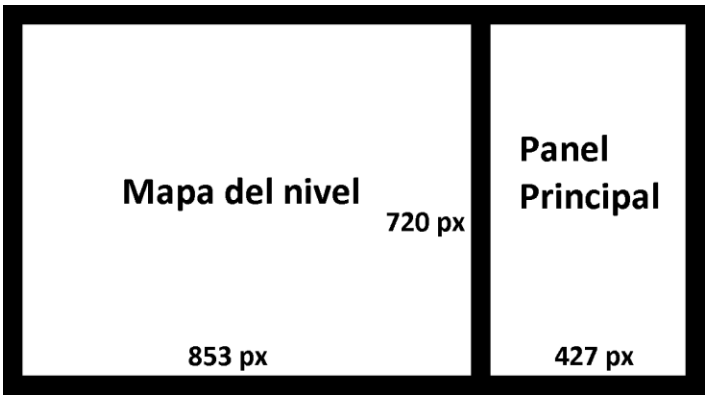


Figura 27. Distribución de contenido.

El panel del juego principal se muestra en la figura 28, este está constituido de un título que identifica el número de nivel, un elemento de entrada de texto de Unity, un botón que ejecuta el código ingresado por el usuario, un botón de ayuda y otro botón que direcciona al menú principal.

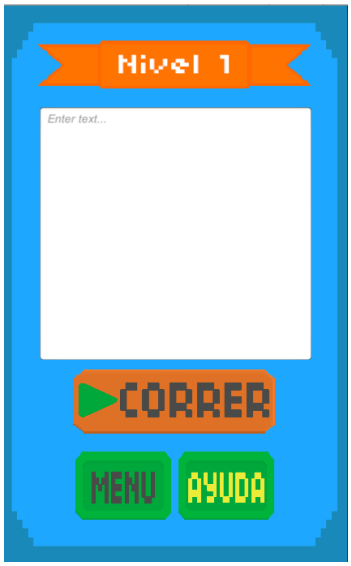


Figura 28. Panel principal de cada nivel.

En la figura 29, se puede ver un panel que se muestra al usuario cuando complete un nivel del juego principal.

Figura 29. Pantalla de nivel completado.

Por el contrario, si el usuario falla el nivel y este no es completado se exhibe el panel de la figura 30.

Figura 30. Pantalla de nivel perdido.

En la figura 31, se muestra el panel de ayuda, en este panel se visualizan todos los comandos válidos del juego.

Figura 31. Pantalla de ayuda.

3.3.7 Minijuego

En la pantalla correspondiente a un nivel del minijuego, se colocaron 2 paneles horizontales paralelos, el panel de arriba contiene los slots donde el jugador debe acomodar las imágenes ordenadamente, por otro lado, el panel de abajo es el que contiene las imágenes a acomodar; el número de slots depende del número de imágenes a ordenar. El número total de niveles del minijuego es de 6, y cada nivel contiene una sucesión de imágenes distintas a organizar. En la figura 32 se muestran los paneles mencionados anteriormente con sus respectivos slots, esta vista pertenece al nivel 2 del minijuego donde se deben de ordenar 3 imágenes, los niveles del minijuego se describen en el apartado 3.3.11.2.

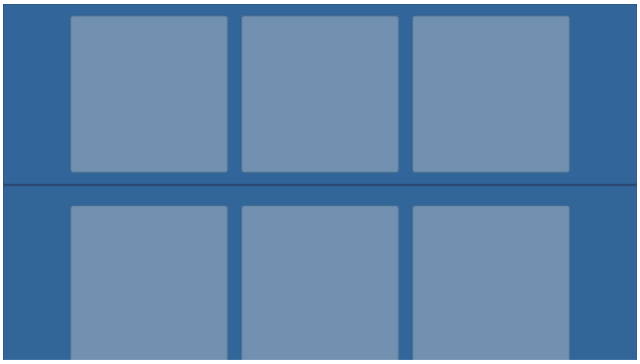


Figura 32. Distribución del minijuego.

También se diseñó un panel que se muestra cuando el jugador completa un nivel del minijuego, ver figura 33.

Figura 33. Pantalla del minijuego completado.

3.3.8 Imágenes y recursos

La página web de donde se tomaron los sprites de los personajes es (http://www.spriters-resource.com), la cual forma parte de un grupo de sitios web encargadas de distribuir y recolectar materiales de videojuegos. En esta página se establece que todos los recursos alojados en estas plataformas pueden ser utilizados por cualquier persona sin infringir políticas de derecho de autor siempre y cuando sea para desarrollos personales, estudiantiles, y cualquier proyecto que no sea comercializable.

Dentro de la página web (https://opengameart.org) se encuentran recursos para la ambientación de las escenas, los objetos mostrados en la figura 34 son algunos de los utilizados en el diseño de los mapas del videojuego, estos son libres y pueden ser utilizados bajo los mismos términos que los sprites del personaje principal.

Figura 34. Recursos utilizados.

Se tomaron algunos de estos recursos para ser colocados en las escenas, sin embargo, se necesitó de otros con los que no se contaba por lo que se diseñaron objetos con la herramienta Asesprite que está enfocada en la creación de imágenes del tipo pixel art.

3.3.9 Desbloqueables

Un objeto desbloqueable es aquel que se encuentra disponible después de completar ciertos requisitos, por ejemplo, pasar todos los niveles o completar un modo de juego. En "Code Adventure" se incluyeron una serie de guitarras desbloqueables con distintos diseños, en la tabla 29 se pueden visualizar cada una de ellas y el nivel en el que se encuentran.

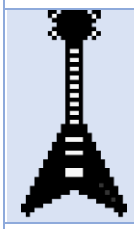
Guitarras				
				
Nivel 1	Nivel 2	Nivel 3	Nivel 4	Nivel 5
				
Nivel 6	Nivel 7	Nivel 8	Nivel 9	Nivel 10
				
Nivel 11	Nivel 12	Nivel 13	Nivel 14	Nivel 15
				
Nivel 16	Nivel 17	Nivel 18	Nivel 19	Nivel 20

Tabla 28. Logros desbloqueables.

3.3.10 Personajes

Para que el videojuego cumpla con algunos de sus objetivos es importante que su personaje principal sea agradable con los usuarios finales, con esto en mente se determinó que debería ser una persona o animal en forma caricaturesca con colores vibrantes, de preferencia con un estilo pixel art para combinar con los escenarios que se tenían en mente.

Después de investigar se encontró un personaje que encajaba con los requisitos mencionados, es un personaje relativamente popular dentro del mundo de los videojuegos lo que da la oportunidad de que los usuarios lo conozcan y sea agradable para ellos.

Scott Pilgrim es un personaje diseñado para una serie de novelas gráficas publicadas desde el año 2004 hasta el 2010, el cómic sigue una historia de comedia y romance, el guión original cuenta con referencias a la cultura popular, los videojuegos y la música norteamericana ya que Scott es un bajista de una banda de rock.

Sin embargo, en este videojuego se le da un giro diferente, teniendo como objetivo avanzar hasta el final del camino para conseguir una guitarra y así completar el nivel que desbloqueará el siguiente, consiguiendo así una secuencia entre todos los escenarios. Al conseguir todas sus guitarras, Scott se reúne con todos sus amigos y celebran con una gran fiesta en el castillo de un rey.

Existen otros personajes dentro del universo de Scott Pilgrim que se tomaron para complementar la historia del videojuego. En la tabla número 28 se describen cada uno de los personajes utilizados en el juego.

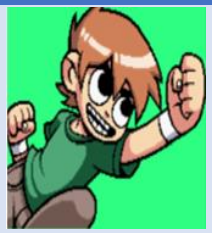
Nombre	Imagen	Descripción	Personaje jugable	Rol en el juego
Scott Pilgrim		Personaje principal que tiene como objetivo llegar al final del mapa para obtener guitarras y avanzar de nivel.	Si	Personaje principal del juego, este puede ser controlado por acciones en forma de código que el jugador introduce.
Linus		Enemigo de Scott, se encarga de impedirle el paso en diversos niveles.	No	Personaje que sirve como obstáculo, Scott tiene que derrotarlo para llegar al final del mapa.
Ramona Flowers		Personaje secundario, encargada de presentar los niveles del minijuego.	No	Este personaje solamente se encuentra presente como acompañante gráfico en el minijuego.

Tabla 29. Personajes.

3.3.11 Niveles

3.3.11.1 Juego principal

El juego cuenta con 20 niveles, los primeros doce se desarrollan dentro de la casa de Scott y los ocho restantes se ambientan en jardines y áreas verdes. Para el diseño de cada uno de estos niveles se usó la herramienta Tiled que permite construir mapas utilizando los tiles descargados previamente. Posteriormente, dichos mapas se exportaron a la plataforma de desarrollo del videojuego.

Se procede a describir cada uno de los niveles a través de las escenas diseñadas, en las cuales el jugador debe introducir líneas de código para programar los movimientos del personaje y así llegar hasta el final del mapa, descrito con mayor detalle en la sección 3.2.5 correspondiente a la mecánica:

Nivel 1		
	**Dificultad:** Fácil	
	Obstáculos	
	Rocas: 0	**Enemigos:** **0**
	Solución: Scott.Derecha(5);	

Tabla 28. Nivel 1.

Nivel 2		
	**Dificultad:** Fácil	
	Obstáculos	
	Rocas: 0	**Enemigos:** **0**
	Solución: Scott.Abajo(5);	

Tabla 29. Nivel 2.

Nivel 3		
	**Dificultad:** Fácil	
	Obstáculos	
	Rocas: 0	**Enemigos:0**
	Solución: Scott.Izquierda(6);	

Tabla 30. Nivel 3.

Nivel 4	
	**Dificultad:** Fácil **Obstáculos** **Rocas**: 0 **Enemigos:** **0** **Solución:** Scott.Izquierda(4); Scott.Abajo(6);

Tabla 31. Nivel 4.

Nivel 5	
	**Dificultad:** Fácil **Obstáculos:** **Rocas**: 0 **Enemigos:** **0** **Solución:** Scott.Abajo(6); Scott.Derecha(4);

Tabla 32. Nivel 5.

Nivel 6		
	**Dificultad:** Fácil	
	Obstáculos:	
	Rocas: 0	**Enemigos:** **0**
	Solución: Scott.Derecha(4); Scott.Arriba(4);	

Tabla 33. Nivel 6.

Nivel 7		
	**Dificultad:** Fácil	
	Obstáculos	
	Rocas: 0	**Enemigos:** **0**
	Solución: Scott.Arriba(2); Scott.Derecha(7);	

Tabla 34. Nivel 7.

Nivel 8		
	**Dificultad:** Media	
	Obstáculos	
	Rocas: 0	**Enemigos: 1**
	Solución: Scott.Derecha(3); Scott.Arriba(3); Scott.Derecha(1); Scott.Cabezazo(); Scott.Derecha(3);	

Tabla 35. Nivel 8.

Nivel 9		
	**Dificultad:** Difícil	
	Obstáculos	
	Rocas: 0	**Enemigos: 1**
	Solución: Scott.Derecha(2); Scott.Cabezazo(); Scott.Derecha(2); Scott.Abajo(3); Scott.Izquierda(3); Scott.Abajo(1);	

Tabla 36. Nivel 9.

Nivel 10	
	**Dificultad: Media** **Obstáculos:** **Rocas**: 0 **Enemigos: 1** **Solución:** Scott.Abajo(3); Scott.Derecha(1); Scott.Cabezazo(); Scott.Derecha(2); Scott.Abajo(4);

Tabla 37. Nivel 10.

Nivel 11	
	**Dificultad: Media** **Obstáculos:** **Rocas**: 0 **Enemigos: 1** **Solución:** Scott.Abajo(3); Scott.Izquierda(1); Scott.Cabezazo(); Scott.Izquierda(3); Scott.Abajo(4);

Tabla 38. Nivel 11.

Nivel 12		
	**Dificultad: Dificil**	
	Obstáculos:	
	Rocas: 0	**Enemigos:** **1**
	Solución: Scott.Abajo(1); Scott.Derecha(3); Scott.Abajo(3); Scott.Izquierda(3); Scott.Cabezazo(); Scott.Izquierda(3);	

Tabla 39. Nivel 12.

Nivel 13		
	**Dificultad: Media**	
	Obstáculos:	
	Rocas: 1	**Enemigos:** **0**
	Solución: Scott.Izquierda(3); Scott.Cabezazo(); Scott.Izquierda(6);	

Tabla 40. Nivel13.

Nivel 14		
	**Dificultad:** **Media**	
	Obstáculos:	
	Rocas: 1	**Enemigos:** **0**
	Solución: Scott.Arriba(3); Scott.Izquierda(2); Scott.Cabezazo(); Scott.Izquierda(5); Scott.Abajo(4); Scott.Izquierda(4);	

Tabla 41. Nivel 14.

Nivel 15		
	**Dificultad:** **Media**	
	Obstáculos:	
	Rocas: 0	**Enemigos:** **1**
	Solución: Scott.Izquierda(4); Scott.Cabezazo(); Scott.Izquierda(3); Scott.Abajo(4);	

Tabla 42. Nivel 15.

Nivel 16		
	**Dificultad: Media**	
	Obstáculos:	
	Rocas: 1	**Enemigos:** **0**
	Solución: Scott.Abajo(3); Scott.Izquierda(1); Scott.Cabezazo(); Scott.Izquierda(3);	

Tabla 43. Nivel 16.

Nivel 17		
	**Dificultad: Difícil**	
	Obstáculos	
	Rocas: 1	**Enemigos:** **1**
	Solución: Scott.Izquierda(2); Scott.Abajo(2); Scott.Izquierda(5); Scott.Arriba(2); Scott.Derecha(3); Scott.Arriba(3); Scott.Izquierda(7);	

Tabla 44. Nivel 17.

Nivel 18		
	**Dificultad: Difícil**	
	Obstáculos	
	Rocas: 1	**Enemigos: 1**
	Solución: Scott.Abajo(5); Scott.Izquierda(2); Scott.Cabezazo(); Scott.Izquierda(2); Scott.Arriba(5); Scott.Izquierda(3); Scott.Cabezazo(); Scott.Izquierda(4);	

Tabla 45. Nivel 18.

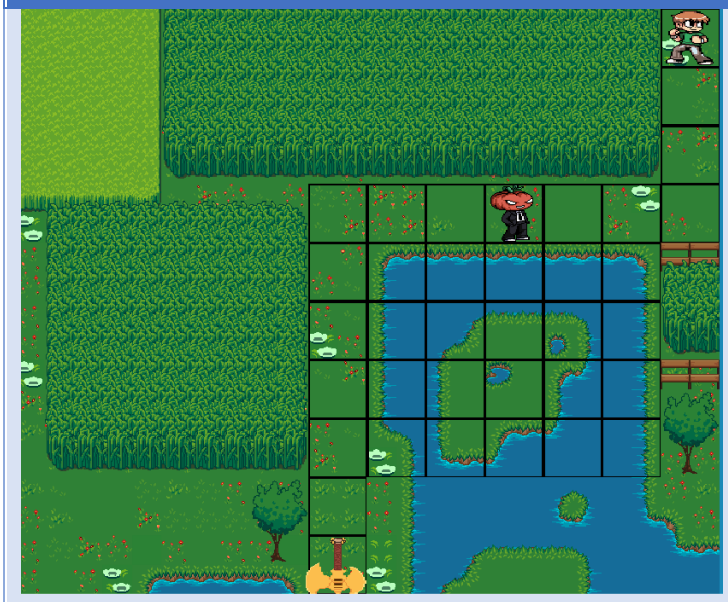
Nivel 19		
	**Dificultad: Difícil**	
	Obstáculos	
	Rocas: 0	**Enemigos: 1**
	Solución: Scott.Abajo(3); Scott.Izquierda(2); Scott.Cabezazo(); Scott.Izquierda(4); Scott.Abajo(6);	

Tabla 46. Nivel 19.

Nivel 20	
	**Dificultad:** **Difícil** **Obstáculos** **Rocas**: 1 **Enemigos:** **1** **Solución:** Scott.Abajo(3); Scott.Derecha(1); Scott.Cabezazo(); Scott.Derecha(2); Scott.Abajo(2); Scott.Cabezazo(); Scott.Abajo(4);

Tabla 47. Nivel 20.

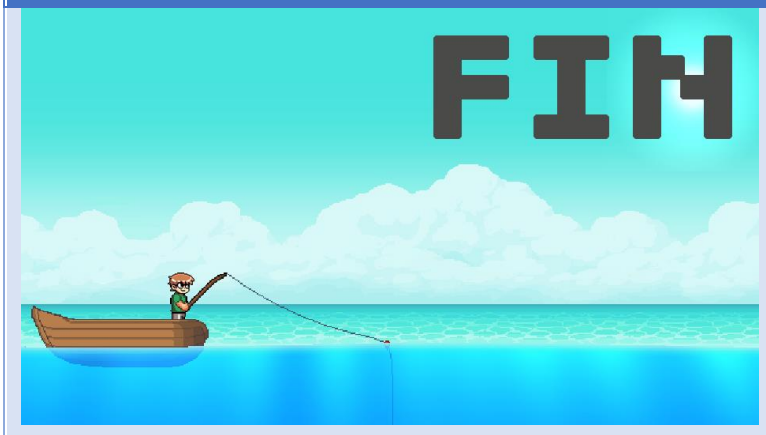
Nivel 1	
	**Dificultad:** **N/A** **Obstáculos** **Rocas**: 0 **Enemigos:** **0** **Descripción:** Esta es la pantalla que aparece al completar todos los niveles.

Tabla 48. Escena final.

3.3.11.2 Minijuego

Se creó una sección de minijuegos con el objetivo de enseñar a los niños lo que es la sucesión de pasos para llegar a un resultado, en donde tienen que acomodar ciertas imágenes para que la historia siga un orden congruente, en total, este minijuego consta de 6 niveles.

En la figura 35, se muestra la primera escena que está diseñada para que sea la de menor dificultad, consta de 3 imágenes en desorden que deben ordenarse de tal forma que se forme una secuencia congruente. En este caso las imágenes corresponden a un joven con dolor de muela, el orden correcto sería poner en el primer recuadro al joven sintiendo el dolor de muela, después la imagen que señala que debe ir al dentista y por último la imagen en la que se muestra feliz y sin ninguna dolencia.

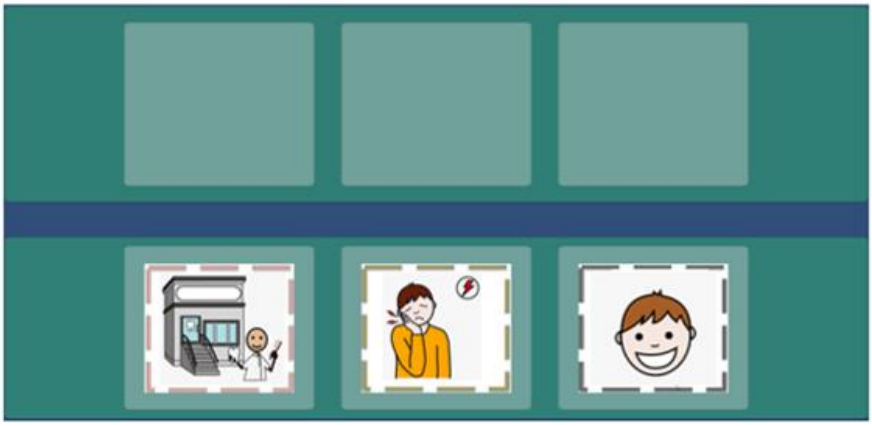


Figura 35. Escena 1 del minijuego.

En la segunda escena se deben de acomodar las imágenes en el orden en el que crece una planta, al igual que en el anterior consta de 3 elementos a ordenar, como se puede ver en la figura 36, en la parte superior de la interfaz se encuentran 3 recuadros vacíos, el usuario debe arrastrar cada una de las imágenes a estos recuadros en el orden correcto, por lo que en el primer recuadro se debe de arrastrar a Scott haciendo un hueco en la tierra, posteriormente se debe de acomodar la imagen en la que está plantando y por último, deslizar al último recuadro la imagen en la que el árbol ya está plantado.

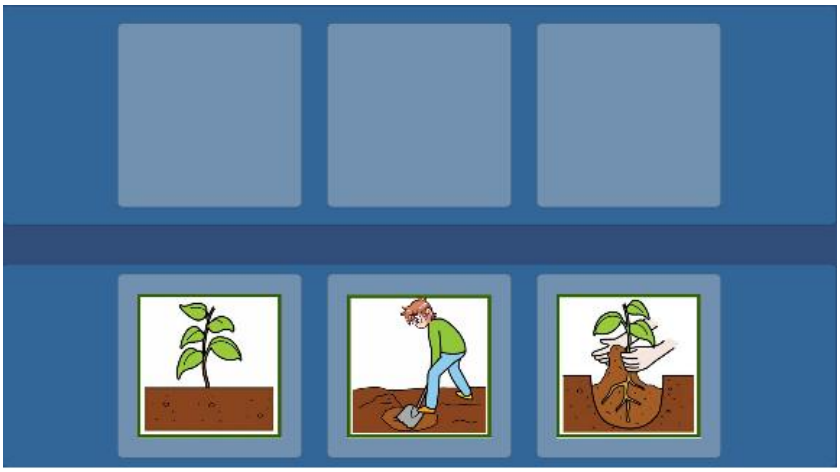


Figura 36. Escena 2 del minijuego.

La tercera escena diseñada representa el proceso en el que se ingiere una manzana, para este nivel se aumenta el grado de dificultada ya que son cuatro las imágenes a ordenar, como se puede ver en la figura 37. La imagen que debe ser arrastrada al primer recuadro es en la que se puede ver a la manzana completa, la que sigue es en la que se muestra la manzana con una mordida, después la imagen que tiene una manzana con 2 mordidas y, por último, la imagen que tiene a la manzana comida.

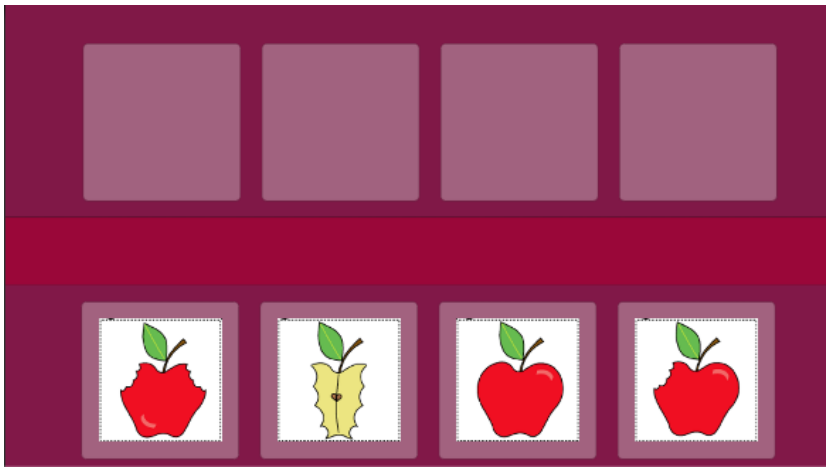


Figura 37. Escena 3 del minijuego.

La escena 4 del minijuego es de 4 imágenes a acodar como la anterior y representa las acciones que sigue un niño al momento de jugar con unas resbaladillas, tal como se muestra en la figura 38. Para este nivel debe de arrastrarse al primer recuadro, la imagen del niño que apenas subirá a las resbaladillas, al segundo recuadro, la imagen en la que el niño ya se encuentra arriba, la siguiente es la imagen en la que el niño ya se está desplazando y por último debe ser colocada la imagen en la que el niño ya se encuentra abajo.

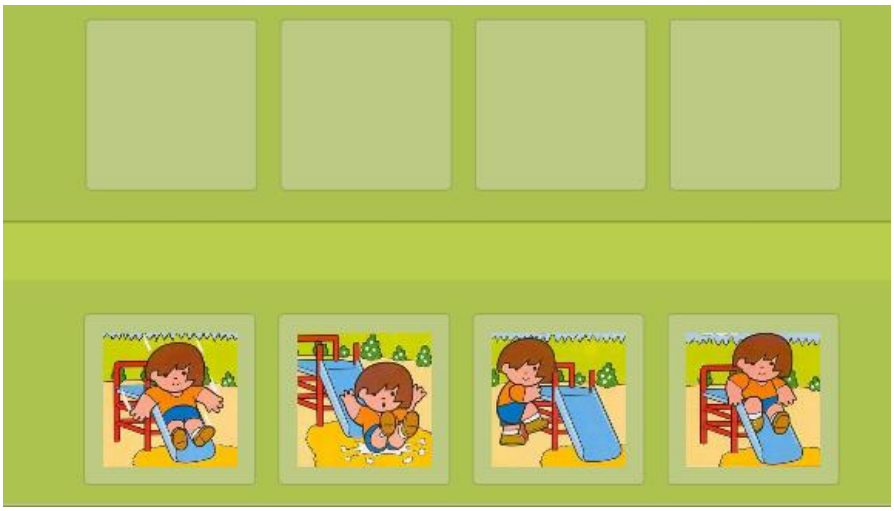


Figura 38. Escena 4 del minijuego.

La escena 5 muestra la historia de una niña que está haciendo un muñeco de nieve, como en los niveles anteriores, las imágenes están en desorden y el objetivo del jugador ordenarlas en los recuadros superiores vacíos que pueden visualizarse en la figura 39. En el primer recuadro se debe de colocar la imagen de la niña que apenas comienza a crear el muñeco de nieve, el segundo recuadro de la parte superior corresponde a la imagen de la niña que ya formó el cuerpo del hombre de nieve, la tercera posición corresponde a la imagen de la niña que coloca los ojos y la nariz del muñeco de nieve y por último la imagen en la que se encuentra terminado.

Figura 39. Escena 5 del minijuego.

En la figura 34 se muestra el sexto y último nivel de los minijuegos, éste se considera el del nivel de dificultad más avanzado ya que cuenta con 5 imágenes que deben de ser ordenadas en los recuadros superiores de la interfaz. En el primer recuadro debe ir la imagen de la niña en un clima soleado, después la imagen en la que se muestra el cielo nublado, posteriormente la imagen en la que se puede apreciar que comienza a llover, la imagen que sigue en la que la niña abrirá el paraguas y el último recuadro corresponde a la imagen donde la niña ya está utilizando el paraguas y puede seguir su camino.

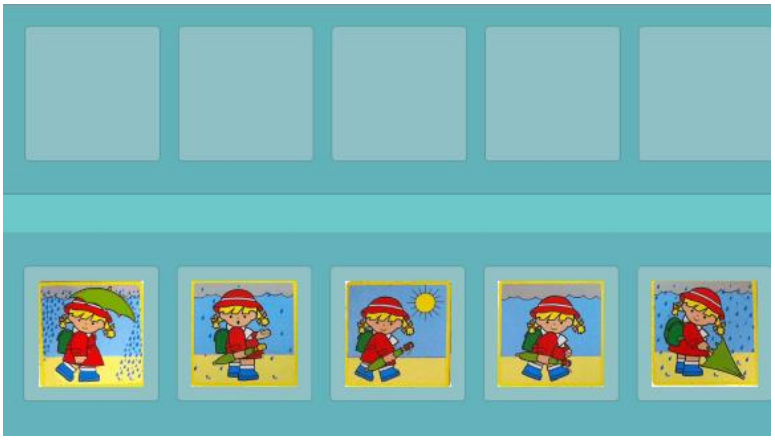


Figura 40. Escena 6 del minijuego.

3.4 Desarrollo

3.4.1 Desarrollo del menú principal.

Para el funcionamiento del menú principal, ver imagen 41, se programó un script llamado "MenuPrincipal", este script además de darle función a cada una de las opciones de la lista permite posicionar la flecha que se asemeja al símbolo ">" en la opción a seleccionar, ya que este menú se controla a través de las teclas "up" y "down" del teclado. El código que permite este movimiento de la flecha se muestra en la imagen 42.

Figura 41. Menú principal.

```
public GameObject flecha, lista;
int indice = 0;

void Start () {
  Dibujar();
}

void Update () {
  bool up = Input.GetKeyDown("up");
  bool down = Input.GetKeyDown("down");

  if (up) indice--;
  if (down) indice++;
  if (indice > lista.transform.childCount-1) indice = 0;
  else if (indice < 0) indice = lista.transform.childCount-1;
  if (up || down) Dibujar();
  if (Input.GetKeyDown("return")) Accion();
}

void Dibujar(){
  Transform opcion = lista.transform.GetChild(indice);
  flecha.transform.position = opcion.position; // Pivote a la izquierda
```

Figura 42. Movimiento de la flecha.

La mayoría de las opciones del menú direccionan a otras pantallas, exceptuando la opción salir, la gestión de cada opción del menú se realiza mediante el método mostrado en la figura 43:

```
void Accion(){
  Transform opcion = lista.transform.GetChild(indice);
  if(opcion.gameObject.name == "Continuar"){
    Application.LoadLevel("mapa1");
  }
  else if(opcion.gameObject.name == "EscojeNivel"){
    Application.LoadLevel("Niveles_Menu");
  }
  else if(opcion.gameObject.name == "Minijuego"){
    Application.LoadLevel("Minijuego_Menu");
  }
  else if(opcion.gameObject.name == "AcercaDe"){
    Application.LoadLevel("AcercaDe");
  }
  else if (opcion.gameObject.name == "Salir"){ // Conseguir nombre del objeto
    print("Cerrando juego... En playmode y HTML no funciona");
    Application.Quit();
```

Figura 43. Direccionamiento de pantallas.

3.4.2 Acerca de

Se desarrollo el script "BotonesAcercaDe", para dar función a los tres botones con los que cuenta esta pantalla, ver figura 44. El primer botón que se encuentra de color azul reproduce una animación de Ramona Flowers, mientras que el segundo botón de color verde reproduce una animación de Scott Pilgrim. El tercer botón sirve para regresar al menú principal del juego.

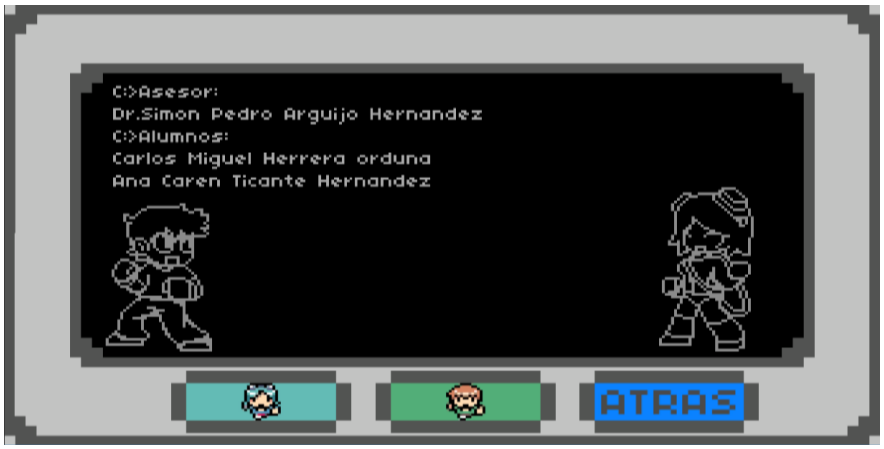


Figura 44. Pantalla "Acerca de".

En el script “BotonesAcercaDe” primero se inicializan los “GameObject” que contienen los sprites, sus componentes “Animator” y unas variables de estado. Las variables de estado sirven para detectar si se ha pulsado uno de los dos botones que reproducen animación. La figura muestra lo anterior mencionado.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BotonesAcercaDe : MonoBehaviour {

  public GameObject Scott;
  public GameObject Ramona;
  Animator animScott;
  Animator animRamona;

  bool ScottTaunt = false;
  bool RamonaTaunt = false;
  // Use this for initialization
  void Start () {
    animScott = Scott.GetComponent<Animator>();
    animRamona = Ramona.GetComponent<Animator>();
  }
```

Figura 45 Script "BotonesAcercaDe".

Los métodos que cambian el valor de las variables auxiliares se muestran en la figura #, estos métodos se ejecutan cuando se oprimen los botones azul y verde.

```
public void ScottButton(){
  ScottTaunt=true;
}
public void RamonaButton(){
  RamonaTaunt=true;
}
```

Figura 46. Métodos de los botones de animación.

Para reproducir cualquiera de las dos animaciones primero se obtiene el estado del “animator” de cada “GameObject” para conocer si ya se está reproduciendo la animación en cuestión, si lo anterior es falso y la variable auxiliar

es verdadera entonces se procede a reproducir la animación, todo esto se realiza dentro del método update y se muestra en la figura 47.

```
void Update () {
  AnimatorStateInfo stateInfoanimScott = animScott.GetCurrentAnimatorStateInfo(0);
  bool TauntingScott = stateInfoanimScott.IsName("ScottAnimacion");
  if(ScottTaunt==true&&TauntingScott==false){
    animScott.SetTrigger("Taunt");
    ScottTaunt = false;
  }
  AnimatorStateInfo stateInfoanimRamona = animRamona.GetCurrentAnimatorStateInfo(0);
  bool TauntingRamona  = stateInfoanimRamona.IsName("RamonaAnimacion");
  if(RamonaTaunt==true&&TauntingRamona==false){
    animRamona.SetTrigger("Taunt");
    RamonaTaunt = false;
  }

}
```

Figura 47. Método Update.

3.4.3 Juego Principal

La pantalla del juego principal contiene cuatro scripts, llamados "ScriptInterfaz", "Analizador", "Movimiento" y "Guitarra" en la figura 48 se observa una vista general del nivel 1.

Figura 48. Pantalla del nivel 1.

“ScriptInterfaz” contiene todos los métodos relacionados a los botones con los que cuentan los paneles de esta pantalla, estos métodos direccionan a otras pantallas y también se encargan de activar o desactivar ciertos elementos de la interfaz, haciendo estos elementos visibles solo cuando se necesite.

En “ScriptInterfaz” primero se inicializan los “GameObject”, un elemento de tipo “Text” y otro de tipo “Input Field”; estos son elementos de interfaz de Unity. Los “GameObject” hacen referencia a paneles de la interfaz, el “Input Field” a una caja de entrada de texto y el “Text” es un componente que muestra texto. Para que este script se pueda comunicar con el script de movimiento del personaje Scott Pilgrim se crea una variable de nombre “pj_mov” y de tipo “Movimiento” (así se llama el script de movimiento del personaje). También se inicializan variables string que contienen nombres de las escenas correspondientes al nivel actual y al nivel siguiente. La figura 49 muestra lo anterior descrito.

```
public GameObject Panel_Help;
public GameObject Panel_LevelCompleted;
public GameObject Panel_LevelFailed;
public Text Texto_Error;
public InputField Campo_de_Texto;

public Movimiento pj_mov;

public string NivelActual;
public string SiguienteNivel;
```

Figura 49. Declaración de variables.

El botón de la izquierda mostrado en la figura 50, se encuentra en la interfaz principal de la pantalla de juego y muestra el panel de ayuda, mientras que el botón de la derecha oculta el panel de ayuda, los métodos correspondientes de estos botones se muestran en la figura 51.

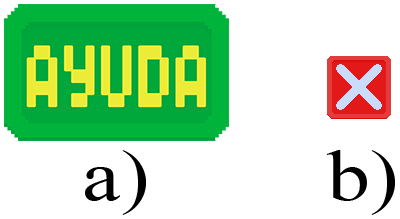


Figura 50. Botones de ayuda.

```
public void AbrirVentanaAyuda(){
  Panel_Help.SetActive(true);
}
public void CerrarrVentanaAyuda(){
  Panel_Help.SetActive(false);
}
```

Figura 51. Métodos de los botones.

Cuando el usuario completa un nivel, se ejecuta el método "AbrirVentanaNivelCompletado", este método activa el panel de "nivel completado". Si el usuario falla un nivel se ejecuta el método "AbrirVentanaNivelFallado" y se muestra el panel de "nivel fallado". El método "IrAMenuPrincipal" direcciona a la pantalla del menú principal, mientras que "VolverAIntentar" es un método que se le atribuye a un botón del panel de "nivel fallado" y este vuelve a cargar el nivel actual, por último, el método "CargaSiguienteNivel" puede ser ejecutado cuando el usuario termina un nivel. Esta parte del script se puede observar en la figura 52.

```
public void AbrirVentanaNivelCompletado(){
  Panel_LevelCompleted.SetActive(true);
}
public void AbrirVentanaNivelFallado(){
  Panel_LevelFailed.SetActive(true);
}
public void IrAMenuPrincipal(){
  Application.LoadLevel("MenuPrincipal");
}
public void VolverAIntentar(){
  Application.LoadLevel(NivelActual);
}
public void CargaSiguienteNivel(){
  Application.LoadLevel(SiguienteNivel);
}
```

Figura 52. Métodos para abrir las ventanas.

Por último, el método "AnalizarCodigo" se ejecuta cuando el jugador pulsa el botón correr, el cual se muestra en la figura 53, este método crea un objeto de la clase "Analizador" y le manda al constructor lo introducido por el usuario, después, si el script analizador encontró errores se mostrará el panel de nivel fallado con los errores encontrados, por el contrario, si no hubo ningún error se obtiene un arreglo con la información de las acciones que realizará el personaje y se procede a ejecutar dichas acciones mediante el script "Movimiento" el método "AnalizarCodigo" del botón correr se muestra en la figura 54.

Figura 53. Botón correr.

```
public void AnalizarCodigo(){
  string codigo=Campo_de_Texto.text;
  Analizador analizar= new Analizador(codigo);
  if(string.IsNullOrEmpty (analizar.error) != true){
    print(analizar.error);
    AbrirVentanaNivelFallado(analizar.error);
  }
  else{
    pj_mov=GameObject.Find("Scott_Sprite").GetComponent<Movimiento>();
    pj_mov.Empezar(analizar.Acciones);
  }

}
```

Figura 54. Método "Analizar código".

El script “Analizador” se encarga de analizar la cadena de texto introducida por el usuario, encontrar errores de sintaxis, y almacenar un nuevo vector con los valores de las acciones que se ejecutaran en el juego, a continuación, se describe a detalle el funcionamiento de este script.

Primero se importan las siguientes librerías de Unity, ver figura 55.

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
```

Figura 55. Librerías.

Después se inician siguientes variables, donde “error” es una variable de tipo string que almacena posibles errores que se encuentren en el texto introducido por el usuario, puede verse en la figura 56.

```
  char[] codigoaChars;
  List<string> listaComandos;
  public string [,] Acciones;
  public string error="";
```

Figura 56. Variables.

El constructor recibe un parámetro de tipo string y este corresponde al texto introducido por el usuario, el constructor se muestra en la figura 57.

En el constructor se realizan los siguientes pasos:

1.- El arreglo "codigoChars" almacena el texto introducido por el usuario en un arreglo de tipo char.

2.- Se ejecuta el método "obtenerComandos" para obtener el valor de la lista "listaComandos", este método se describe más adelante.

3.- Después se compara el tamaño de la lista "listaComandos", con el afán de conocer si tiene un tamaño diferente a 0, si lo anterior es verdadero se ejecuta el método "analizarComandos" donde se analiza cada elemento de "listaComandos", por el contrario, si el tamaño de la lista es de 0 entonces se puede decir que el usuario no escribió ningún texto.

```
public Analizador(string codigo){
  codigoaChars = codigo.ToCharArray ();
  listaComandos=obtenerComandos(codigoaChars);
  if(listaComandos.Count!=0){
    Acciones = new string[listaComandos.Count, 3];
    analizarComandos(listaComandos);
  }
  else{
    error=error+"\n"+"Error texto vacio.";
  }
```

Figura 57. Analizador de código.

El método "obtenerComandos" convierte las líneas de texto introducidas por el usuario en elementos de una lista, como se muestra en la figura 58.

```
public List<string>  obtenerComandos(char[] chars){
    List<string> lista = new List<string> ();
    string linea="";
    for (int i = 0; i < chars.Length; i++) {
      if(chars[i]!='\n'){
        linea=linea+chars[i];
        if(i==chars.Length-1){
          lista.Add(linea);
          linea="";
        }
      }
      else{
        lista.Add(linea);
        linea="";
      }
    }
    return lista;
  }
```

Figura 58. Método "obtenerComandos".

El método "analizarComandos" recibe como parámetro el valor de "listaComandos" y recorre la lista para evaluar cada elemento de esta por medio métodos auxiliares que analizan su sintaxis, véase figura 59.

```
public void analizarComandos(List<string> lista){
  for(int i =0;i<lista.Count;i++){
    char[] linea=lista[i].ToCharArray();
    if(linea.Length>=15&&linea.Length<=20){
      analizarNombre(linea,i);
    }
    else{
      error=error+"\n"+"Error Linea #"+(i+1)+ ": Texto invalido, revisa los comandos
    }
  }
}
```

Figura 59. Método "analizarComandos".

El método analizar nombre, examina que los primeros 5 caracteres de la línea introducida formen la cadena de texto "Scott"como se muestra en la figura 60.

```
public void analizarNombre(char[] linea, int i){
  if((linea[0]+""+linea[1]+""+linea[2]+""+linea[3]+""+linea[4]).Equals("Scott")){
    analizarpunto(linea,i);
  }
  else{
    error=error+"\n"+"Error Linea #"+(i+1)+ ": Escribe correctamente el nombre del
  }
}
```

Figura 60. Método "analizarNombre".

En la figura 61 se muestra el método analizar nombre, éste examina que el sexto carácter sea un punto.

```
  public void analizarpunto(char[] linea, int i){
  if(linea[5]=='.'){
    analizarAccion(linea,i);
  }
  else{
    error=error+"\n"+"Error Linea #"+(i+1)+ ": Falto un '.'  (punto ) despues del nombre
  }
}
```

Figura 61. Método "analizarpunto".

El método analizarAccion, mostrado en la figura 62, examina lo que se encuentra después del punto, y esto determina si se encuentra una acción valida en la cadena de caracteres.

```
public void analizarAccion(char[] linea, int i){
  if((linea[6]+""+linea[7]+""+linea[8]+""+linea[9]+""+linea[10]).Equals("Abajo")&linea.Length>=158
  }
  else if((linea[6]+""+linea[7]+""+linea[8]+""+linea[9]+""+linea[10]+""+linea[11]).Equals("Arriba"
  }
  else if((linea[6]+""+linea[7]+""+linea[8]+""+linea[9]+""+linea[10]+""+linea[11]+""+linea[12]).Ec
  }
  else if((linea[6]+""+linea[7]+""+linea[8]+""+linea[9]+""+linea[10]+""+linea[11]+""+linea[12]+""+
  }
  else if((linea[6]+""+linea[7]+""+linea[8]+""+linea[9]+""+linea[10]+""+linea[11]+""+linea[12]+""+
  }
  else{
    error=error+"\n"+"Error Linea #"+(i+1)+ ": Accion invalida, las acciones posibles son: 'Arriba
  }
}
```

Figura 62. Método "analizarAccion".

Los métodos analizarParentecisI y analizarParentecisF mostrados en la figura 63, examinan si la cadena de texto contiene el paréntesis que abre y el paréntesis que cierra.

```
public void analizarParentesisI(char[] linea,int n, int i){
  if(linea[n]=='('){
    if(n!=14){
      //otro
      analizarNumero(linea,n+1,i);
    }
    else{
      //cabezazo
      analizarParentecisF(linea,n+1,i);
    }
  }
  else{
    error=error+"\n"+"Error Linea #"+(i+1)+ ": Falto un '(' despues de la accion.";
  }
}
public void analizarParentecisF(char[] linea,int n, int i){
  if (linea[n]==')'){
    analizarPuntoComa(linea,n+1, i);
  }
  else{
    error=error+"\n"+"Error Linea #"+(i+1)+ ": Despues de un numero, cierra el paren
  }
}
```

Figura 63.Métodos para los paréntesis.

El método “analizarNumero” examina los números dentro de los paréntesis, se puede ver en la figura 64.

```
public void analizarNumero(char[] linea,int n, int i){
  if(linea [n] >= '0' && linea [n] <= '9'){
    Acciones[i,1]=""+linea[n];
    if(linea [n+1] >= '0' && linea [n+1] <= '9'){
      Acciones[i,1]=Acciones[i,1]+""+linea[n];
      analizarParentecisF(linea,n+2,i);
    }
    else{
      analizarParentecisF(linea,n+1,i);
    }
  }
  else{
    error=error+"\n"+"Error Linea #"+(i+1)+ ": Despues
  }
}
```

Figura 64. Método "analizarNumero".

El método “analizarPuntoComa” inspecciona que la línea de texto termine con punto y coma, véase figura 65.

```
public void analizarPuntoComa(char[] linea,int n, int i){
  if(linea[n]==';'){
  }
  else{
    error=error+"\n"+"Error Linea #"+(i+1)+ ": Termina con ;";
  }
}
```

Figura 65. Método "analizarPuntoComa".

3.4.3.1 Movimiento

EL script movimiento es el que se encarga de darle movimiento al personaje, en este script primero se inicializa un GameObject referente al personaje que se moverá y otros componentes de este, como su Rigidbody2d, Animator, CircleCollider, también se inicializan algunas variables de estado y otras utilizadas para movimiento, como se puede ver en la figura 66.

```
GameObject pj;
Rigidbody2D rb2d;
Animator anim;
CircleCollider2D attackCollider;

public float speed = 3f;
public int  Direccion_Inicial;
int Direccion;
public Vector3 posicioninicial;
public string [,] Acciones;
Vector2 mov;
public bool empezar=false;
double longitud_Paso=0.9996;
//double longitud_Paso=0.916675;
//double longitud_Paso=0.900025;
int i=0;
int j=0;
double [,] Posiciones=new double[3, 2];
public Guitarra Objetivo;
public ScriptInterfaz Gui;
AnimatorStateInfo stateInfo;
bool attacking;
bool attacking2;
bool atacando;
void Start () {
  pj = GameObject.FindWithTag ("Player");
  anim = pj.GetComponent<Animator>();
  rb2d = pj.GetComponent<Rigidbody2D>();
  attackCollider = transform.GetChild(0).GetComponent<CircleCollider2D>();
  attackCollider.enabled = false;
  pj.transform.position = posicioninicial;
  anim.SetInteger ("Direccion", Direccion_Inicial);
  Direccion=Direccion_Inicial;
  //Acciones=new string[,]{{"derecho","2",""}};
  mov = new Vector2(0,0);
}
```

Figura 66. Movimiento.

Para que el personaje se mueva primero se debe obtener el arreglo que contiene la información de las acciones a realizar, el script "ScriptInterfaz" obtiene este arreglo de la clase "Analizador" y además lo envía al script" Movimiento". Después de que se obtenga el arreglo con la información, se recorre dicho arreglo. Si la acción que contiene el elemento del arreglo es de movimiento, entonces se reproduce la animación correspondiente y se modifica el vector movimiento, (1,0) para la dirección derecha, (-1,0) para izquierda, (0,1) para arriba y (0,-1) para abajo; por otro lado, si la acción es de ataque se reproduce la animación de "cabezazo" y el vector de movimiento es (0,0), véase figura 67.

```
47      if(Acciones[i,0].Equals("derecha")){
48        mov= new Vector2(1,0);
49        if(rb2d.position[0]<=double.Parse(Acciones[i,2])){
50          Moviendo();
51        }
52        else{
53          j=0;
54          Siguiente();
55        }
56      }
57      else if(Acciones[i,0].Equals("izquierda")){
58        mov= new Vector2(-1,0);
59        if(rb2d.position[0]>=double.Parse(Acciones[i,2])){
60          Moviendo();
61        }
62        else{
63          j=0;
64          Siguiente();
65        }
66      }
67      else if(Acciones[i,0].Equals("arriba")){
68        mov= new Vector2(0,1);
69        if(rb2d.position[1]<=double.Parse(Acciones[i,2])){
70          Moviendo();
71        }
72        else{
73          j=0;
74          Siguiente();
75        }
76      }
77      else if(Acciones[i,0].Equals("abajo")){
78        mov= new Vector2(0,-1);
79        if(rb2d.position[1]>=double.Parse(Acciones[i,2])){
80          Moviendo();
81        }
82        else{
83          j=0;
84          Siguiente();
85        }
86      }
87      else if(Acciones[i,0].Equals("cabezazo")){
88        stateInfo = anim.GetCurrentAnimatorStateInfo(0);
89          attacking = stateInfo.IsName("Ataque_Derecha");
90        attacking2 = stateInfo.IsName("Ataque_Izquierda");
91        if(atacando==false&&!attacking&& !attacking2){
92          anim.SetTrigger("Atacking");
93          atacando=true;
94        }
95        else if(atacando==true&&!attacking&& !attacking2){
96          atacando=false;
97          Siguiente();
98        }
99
100     }
```

Figura 67. Script "Interfaz".

Para que se realice el movimiento utilizando las físicas de Unity se utiliza el método mostrado en la figura 68.

```
130     void FixedUpdate () {
131     // Nos movemos en el fixed por las físicas
132     rb2d.MovePosition(rb2d.position + mov * speed * T
133     }
```

Figura 68. Método "FixedUpdate".

El método ValidarVictoria, mostrado en la figura 69, se utiliza para validar si el jugador cumplió con el objetivo del juego, se obtiene una variable de estado del script "ScriptGuitarra" que guarda si hubo una colisión entre el personaje del juego y la guitarra del final del nivel.

```
public void validarVictoria (){
  Objetivo = GameObject.Find ("Guitarra_Sprite").GetComponent<Guitarra> ();
  Gui = GameObject.Find ("InterfazGraficaUsuario").GetComponent<ScriptInterfaz> ();
    if (Objetivo.ObjetivoCompleted==true) {
      Gui.AbrirVentanaNivelCompletado();
    }
    else {
      Gui.AbrirVentanaNivelFallado("Codigo inneficiente");
    }
}
```

Figura 69. Método"ValidarVictoria".

Si el jugador cumplió con el objetivo, se muestra el panel de nivel completado, ver figura 70.

Figura 70. Nivel completado.

Por el contrario, si el código introducido por el usuario no cumple con el objetivo, o si hubo errores al analizar las líneas introducidas, se sale del programa y muestra el panel de nivel fallado, donde se muestra un mensaje acerca de la causa de por qué no se completó el nivel, así como de posibles errores de sintaxis, ver figura 71.

Figura 71. Nivel fallido.

3.4.3.2 Guitarra

El "ScriptGuitarra" se encuentra en la guitarra de cada uno de los niveles y detecta si el al terminar el número de pasos el personaje del juego existió una colisión entre el personaje y la guitarra lo que significa que el usuario llego al final del mapa. El script "ScriptGuitarra" se muestra en la figura 72.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class bo : MonoBehaviour {
7     public bool ObjetivoCompleted=false;
8     public GameObject ui;
9
10 void OnTriggerEnter2D(Collider2D other){
11     if(other.tag=="Player"){
12         print("ha chocado");
13         ObjetivoCompleted=true;
14         nivelcompletado();
15
16     }
17 }
18 public void nivelcompletado(){
19     ui.SetActive(true);
```

Figura 72. Script "ScriptGuitarra".

3.4.3.3 Objetos destruibles

Se creó un script llamado “name”, este script se utilizó en el objeto roca y el enemigo del juego, el script detecta cuando existe una colisión del ataque del personaje principal, si existe dicha colisión se reproduce una animación del objeto correspondiente siendo destruido, véase figura 73.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Destroyable : MonoBehaviour {
    public string destroyState;
    public float timeForDisable;
    Animator anim;
    void Start () {
        anim = GetComponent<Animator>();
    }
    IEnumerator OnTriggerEnter2D (Collider2D col) {
        if (col.tag == "Attack") {
            anim.Play(destroyState);
            yield return new WaitForSeconds(timeForDisable);
            foreach(Collider2D c in GetComponents<Collider2D>()){
                c.enabled = false;
            }
        }
    }
    void Update () {
        AnimatorStateInfo stateInfo = anim.GetCurrentAnimatorStateInfo(0);
        if (stateInfo.IsName(destroyState) && stateInfo.normalizedTime >= 1) {
            Destroy(gameObject);
        }

    }

}
```

Figura 73. Destruir objetos.

3.4.3.4 Seleccionar nivel

Las pantallas de seleccionar nivel del juego principal y seleccionar nivel del minijuego usan dos scripts llamados "LoadLevel" e "IrMenuPrincipal". El script "IrMenuPrincipal" se encarga de regresar al jugador al menú principal y su código se muestra en la figura 74.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class IrMenuPrincipal : MonoBehaviour {
  public void IrAMenuPrincipal(){
    SceneManager.LoadScene("MenuPrincipal");
  }
}
```

Figura 74. Seleccionar nivel.

El script "LoadLevel" se encarga de abrir el nivel seleccionado por el jugador, este script se encuentra en cada uno de los botones referentes a los niveles en el menú de selección de nivel, el código de este script se muestra en la figura 75.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LoadLevel : MonoBehaviour {
  public void Cargar_Nivel_Minijuego(string NameNextLevel){
    Application.LoadLevel(NameNextLevel);
  }
}
```

Figura 75. Script "LoadLevel".

3.4.4 Minijuego

El código mostrado en la figura 76, corresponde al funcionamiento del nivel 1 de los minijuegos, en donde se van a comparar que las imágenes estén en el orden correcto.

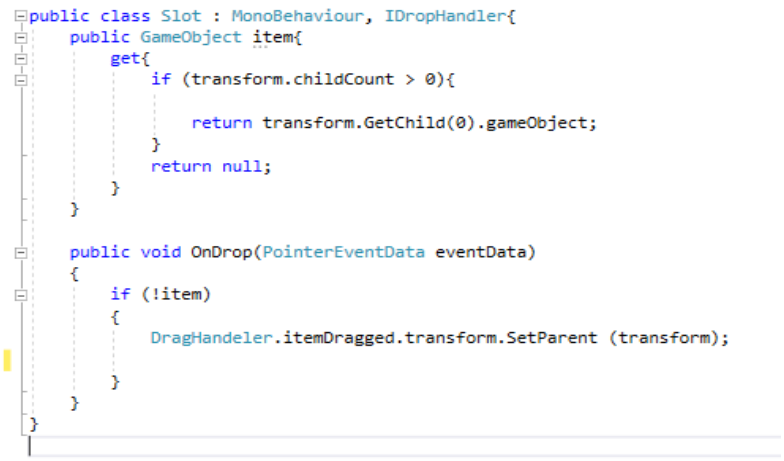
```
public GameObject pollo, huevo, gallo, slot1, slot2, slot3;
public GameObject ui;
void Start () ...

void Update () {

    if (slot1.transform.childCount >0  && slot2.transform.childCount >0 && slot3.transform.childCount >0)
    {
        string sl1 = slot1.transform.GetChild(0) + "";
        string sl2 = slot2.transform.GetChild(0) + "";
        string sl3 = slot3.transform.GetChild(0) + "";
        print(sl1 + " " + sl2 + " " + sl3);

        if (sl1 == "huevo (UnityEngine.RectTransform)" && sl2 == "pollo (UnityEngine.RectTransform)"
            && sl3 == "gallo (UnityEngine.RectTransform)")
        {

            print("coinciden");
                LevelCompleted ();

        }
        else
        {
            print("no coinciden");
        }
    }
    else print("no existe");

}
public void LevelCompleted (){
    ui.SetActive(true);
}
```

Figura 76. Script del nivel 1 del minijuego.

Si el usuario ordena las imágenes correctamente, se ejecuta el método LevelCompleted mostrado en la anterior figura, este método activa el panel de nivel completado, el cual se puede ver en la figura 77.

Figura 77. Nivel completado.

El script con nombre "Slot" tiene como objetivo acomodar las imágenes dentro de los slots deseados que son los cuadros que se muestran en la interfaz y parte del código se muestra en la figura 78.



```
public class Slot : MonoBehaviour, IDropHandler{
    public GameObject item{
        get{
            if (transform.childCount > 0){

                return transform.GetChild(0).gameObject;
            }
            return null;
        }
    }

    public void OnDrop(PointerEventData eventData)
    {
        if (!item)
        {
            DragHandeler.itemDragged.transform.SetParent (transform);

        }
    }
}
```

Figura 78. Código del script "Slot".

Capítulo IV. Resultados

En el presente trabajo se propuso el desarrollo de un videojuego para enseñar conceptos básicos de programación a niños, considerando los procesos de desarrollo y diseño de personajes y escenas, partiendo de conocimientos previos complementados con investigaciones y basándose principalmente en conceptos detallados en trabajos relacionados. Todos estos conocimientos obtenidos, se aplicaron con éxito en un videojuego interactivo que cumple con sus objetivos planteados.

A continuación, se describirán los resultados obtenidos, las conclusiones del presente trabajo y los posibles trabajos futuros que pueden continuar desarrollándose como resultado de la investigación.

4.1 Resultados de la investigación

Las pruebas de usabilidad son uno de los métodos más comunes para medir un videojuego. La usabilidad es un atributo de calidad que evalúa la calidad de experiencia del usuario e identifica la facilidad de uso de un sistema, en una prueba de usabilidad el videojuego se presenta a un grupo de usuarios objetivo, es decir, al público al que va dirigido principalmente el juego.

Lo que se busca es identificar posibles errores que los desarrolladores hayan pasado por alto, no es necesario presentar el producto final, basta con un prototipo o una versión avanzada del software, ya que se recomienda que dicha prueba se aplique al menos tres semanas antes de la entrega final [33].

Con el objetivo de realizar una prueba de usabilidad y probar "Code Adventure", además de observar el comportamiento de los niños frente al videojuego y verificar que este cumpla con el objetivo esperado que consiste en desarrollar la lógica de programación, se creó un demo de 7 niveles que incluyen los

distintos grados de dificultad y, además, los niveles del minijuego. Para aplicar dicha prueba, se solicitó un permiso a la directora de la escuela primaria urbana "Niños Héroes" ubicada en la colonia Espaldilla de Misantla, Veracruz, en respuesta a esta petición la directora accedió a que niños de quinto y sexto grado participaran en esta prueba.

El grupo de infantes constó de 12 niñas y 16 niños, de entre 9 y 11 años.

Figura 79. Participantes.

El día en que se llevó acabo la prueba, primero se dio una introducción a los niños acerca del objetivo del juego y de su participación, se respondieron todas las dudas existentes.

Posteriormente se iniciaron las pruebas, debido a que el grupo de alumnos era numeroso, se optó por que probaran el juego en parejas y así realizar pruebas con la mayor cantidad de niños posibles.

Por parte de los aplicadores de la prueba se observó los siguiente:

- Agrupar a los alumnos en parejas resulto provechoso, ya que se notó una retroalimentación debido al trabajo en equipo.

- A los niños que interactúan frecuentemente con una computadora se les facilitó el manejo del videojuego, en contraste con los que no.
- Los niños se interesaron más por el videojuego en comparación con las niñas.
- El videojuego despertó en los infantes un interés, sobre como con la programación se puede crear un videojuego.
- Se notó una dificultad en los niños para aprender las reglas sintácticas de los comandos que controlan el personaje del videojuego.

Figura 80. Niños haciendo las pruebas.

Con el propósito de recabar información sobre la opinión de los niños frente al software, se diseñó la encuesta mostrada en la tabla número 26, que consta de 10 preguntas aplicadas a cada niño después de interactuar con el demo del videojuego.

.

NUM	PREGUNTAS	POSIBLES RESPUESTAS				
1	¿Te gustó el juego?	1. No	2. Regular	3. Si		
2	Del 1 al 5, ¿Qué tan fácil es de jugar?	1	2	3	4	5
3	¿Es divertido?	1. Poco	2. Regular	3. Mucho		
4	¿Te gustó el minijuego?	1. Poco	2.Regular	3.Mucho		
5	Del 1 al 5, ¿Qué tanto te agradó el personaje?	1	2	3	4	5
6	¿Volverías a jugarlo?	1. No	2.Tal vez	3. Si		
7	¿Lo recomendarías con tus amigos?	1. No	2. Tal vez	3. Si		
8	¿Aprendiste algo nuevo?	1. No	2. Tal vez	3. Si		
9	¿Encuentras interesante el mundo de la programación después de jugar este juego?	1. Poco	2. Regular	3. Mucho		
10	¿Calificación final del videojuego?	1	2	3	4	5

Tabla 49. Encuesta.

Para el diseño de esta encuesta se utilizaron preguntas con un lenguaje sencillo para que los niños pudieran comprenderlas fácilmente, además, se utilizaron respuestas de opción múltiple por la misma razón y para que la información pudiera ser plasmada estadísticamente.

Para obtener datos de la encuesta, se utilizó una escala en la que a cada posible respuesta se le asigna un valor, por ejemplo: no=1, tal vez=2, si=3, dichos valores se encuentran asignados en la tabla 51. En algunas de las preguntas se pedía asignar una calificación del 1 al 5, porque no fue necesario realizar una conversión.

Una vez teniendo solamente valores ordinales, se obtuvieron los promedios de cada una de las preguntas, los cuales se muestran en la figura 81.

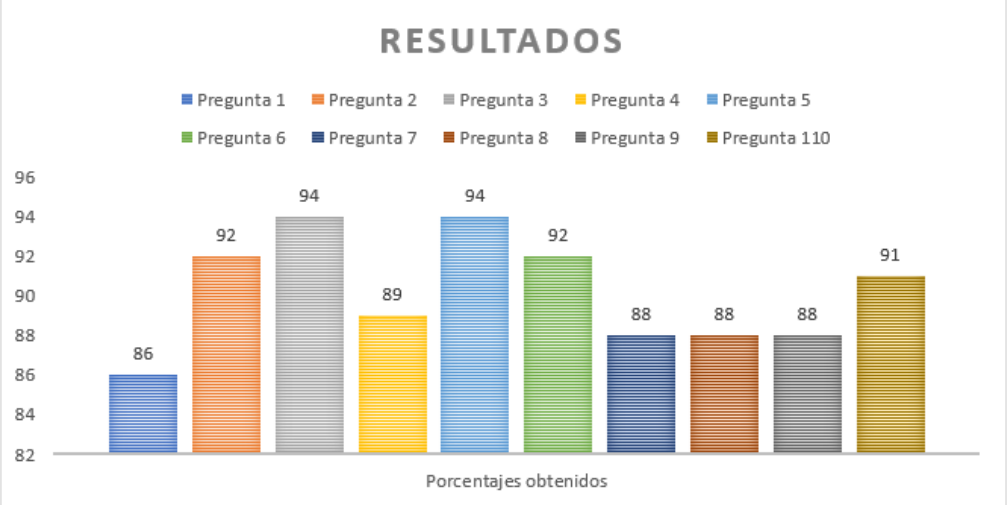


Figura 81. Porcentajes obtenidos.

El promedio general obtenido con los datos recolectados con la encuesta es de 90.2%.

Una vez realizadas las validaciones y cada una de las pruebas, se identifica que el juego satisface con las acciones verificadas anteriormente, brindando confiabilidad y funcionalidad en cada uno de los niveles del juego.

Se evidencia el correcto funcionamiento de los elementos del videojuego, dando un visto bueno en cada una de las pruebas realizadas y confirmando que el juego cumple con cada una de las especificaciones vistas en los objetivos.

Para comprobar el correcto funcionamiento del videojuego se subieron 3 videos a la plataforma de Youtube donde se puede ver cómo que se resuelven los niveles. El enlace donde pueden visualizarse es el siguiente:

(https://www.youtube.com/channel/UCR4zCZzbPqzABXadhEufFBw).

4.2 Conclusiones

Se revisaron las diferentes propuestas de videojuegos que tienen como objetivo ayudar personas a aprender a programar, se analizaron factores como: gameplay, diseños, temáticas, características de los personajes, dificultad, tópicos de programación que tratan y cómo los tratan. Todo lo anterior con la finalidad de tomar aspectos clave para que el videojuego a desarrollar sea capaz de transmitir conocimientos de programación.

De acuerdo con los resultados anteriores, se desarrollaron y diseñaron los mapas e interfaces del juego, y también ayudaron a delimitar el proyecto.

El objetivo fundamental de este trabajo de investigación era desarrollar un videojuego que fuera capaz de ayudar a niños a inicializarse en el mundo de la programación, haciendo más divertida y llevadera la inducción en la programación, la propuesta fue apegarse a las características principales con las que cuenta un videojuego.

4.3 Trabajos futuros

Como continuación de este trabajo y como en cualquier otro proyecto de investigación, existen diversas líneas de investigación que quedan abiertas y en las que es posible continuar trabajando. Durante el desarrollo surgieron algunas líneas futuras que se dejaron abiertas y que se esperan abordar en un futuro; algunas de ellas, están más directamente relacionadas con este trabajo y son el resultado de cuestiones que han ido surgiendo durante la realización de éste.

A continuación, se presentan algunos trabajos futuros que pueden desarrollarse como resultado de esta investigación o que, por exceder el alcance de esta tesis, no han podido ser tratados con la suficiente profundidad. Además, se sugieren algunos desarrollos específicos para apoyar y mejorar el modelo y metodología propuestos. Entre los posibles trabajos futuros destacan:

- Estudio extenso de las técnicas de aprendizaje.
- Investigación profunda sobre desarrollo de videojuegos educativos.
- Aumentar la cantidad de niveles al juego.
- Desarrollar funciones que se enfoquen en la enseñanza de conceptos de programación más complejos.
- Desarrollar minijuegos nuevos.

Bibliografía

1. C. Enríquez, "La importancia de Aprender a Programar," (miércoles, 20.06.2018). Recuperado de: http://www.milenio.com/opinion/varios-autores/universidad-politecnica-de-tulancingo/la-importancia-de-aprender-a-programar Último acceso: junio 2018.

2. A. Herranz, "A qué edad se debe aprender código?", Vida Tecnológica, (18 de septiembre de 2017). Recuperado de https://www.bloglenovo.es/a-que-edad-se-debe-aprender-codigo/ Último acceso: junio 2018.

3. J. I. Fuentes-Rosado, M. Moo-Medina, "Dificultades de aprender a programar, "Revista *Educación en Ingeniería*, *12*, 24, 76-82 (2017).

4. "McDonald The Global Games Market Will Reach $108.9 Billion in 2017 With Mobile Taking 42%," (20 de abril 2017). Recuperado de: https://newzoo.com/insights/articles/the-global-games-market-will-reach-108-9-billion-in-2017-with-mobile-taking-42/ Último acceso: junio 2018.

5. B. Marciano, "Juegos serios y entrenamiento en la sociedad digital," Teoría de la Educación. Educación y Cultura en la Sociedad de la Información, 9, 3 (2008).

6. S. Belli, C. López, "Breve historia de los videojuegos," Athenea Digital. Revista de pensamiento e investigación social 14, 159-179 (2008).

7. E. Delgado, E. C. C, "Desarrollo de habilidades cognitivas mediante videojuegos en niños de educación básica," Revista Iberoamericana para la Investigación y el Desarrollo Educativo ISSN: 2007-2619, 12, (2015).

8. A. I. R. Elizondo, J. A. H. Bernal, y M. S.R. Montoya, "Desarrollo de habilidades cognitivas con aprendizaje móvil: un estudio de casos," *Comunicar: Revista científica iberoamericana de comunicación y educación*, 34, 201-209 (2010).

9. R. L. de Guevara Cortés, "Los simuladores de negocios por computadora. Una herramienta de inmersión a la gestión financiera."

10. VStep, "RescueSim-Virtual Emergency Response Training", Product Information available online: www.rescuesim.com, (2011). Último acceso: junio 2018.

11. M. McNaughton, M. Cutumisu, D. Szafron, J. Schaeffer, J. Redford, D. Parker, "ScriptEase: Generative design patterns for computer role-playing games. In Proceedings of the 19th IEEE international conference on Automated software engineering," IEEE Computer Society," pp. 88-99, (2004, September).

12. J. I. Fuentes-Rosado, M. Moo-Medina, "Dificultades de aprender a programar", *Revista Educación en Ingeniería*, *12*, 24, 76-82. (2017).

13. Recuperado de: https://codecombat.com/about Ultimo acceso: junio 2018

14. Recuperado de: https://www.codingame.com/about/team. Último acceso: junio 2018.

15. Recuperado de: https://www.playcodemonkey.com/about/us Ultimo acceso: junio 2018

16. T. de Piaget, "Desarrollo Cognitivo: Las Teorías de Piaget y de Vygotsky," (2007).

17. F.E. Balerdi, "Videojuegos y educación. Comunicar", Revista científica iberoamericana de comunicación y educación, 10, 171-180 (1998).

18. A. García-Valcárcel Muñoz-Repiso, "Las competencias digitales en el ámbito educativo", (2016).

19. B. Gros Salvat, "Certezas e interrogantes acerca del uso de los videojuegos para el aprendizaje", Revista Internacional de Comunicación Audiovisual, Publicidad y Literatura, 1,7, 251-264 (2009).

20. M. Epistémicos, P. Reproductivas, "Juegos Epistérmicos1", Journal of online Education, 1, 6. (2005).

21. M. Moreno, E Remesal, F. Rivera Rodríguez, "Videojuego educativo para el aprendizaje de SQL", (2007).

22. N. Padilla-Zea, N. Medina-Medina, P. Paderewski, F. L.Gutiérrez, J. R. López-Arcos, "Diseñando Videojuegos para Aprender de Forma Divertida: En Busca del Equilibrio Perdido", (2011).

23. J. Orea Moya, "Creación de un videojuego 3D educativo con metodología basada en la modificación de código".

24. M. Álvarez, N. B. Cárdenas, J. Álvaro, "Implementación del prototipo de un videojuego interactivo para reforzar el aprendizaje del inglés en niños de 3 a 5 años".

25. V. Aravena, E. Cristian, "Diseño e implementación de un videojuego de acción-aventura 2d para dispositivos android utilizando el motor unity".

26. A. M. Manrubia Pereira, "El proceso productivo del videojuego: fases de producción". Historia y Comunicación Social, 19, 791-805, (2014).

27. R. Botero Tabares. "La lúdica de juegos en el aprendizaje de la programación orientada a objetos: un prototipo en c#", (2012).

28. L. Garcia. Informática, Temario Específico.

29. J. R. Nader, "Metodología de Desarrollo de Software: MBM (Metodologia Basada en Modelos)", *Ingeniare*, 16, 113-127, (2014).

30. R. Villanueva, "Los 4 tipos de Gamer según la psicología", (2014).. Recuperado de: https://www.levelup.com/articulos/283250/Los-4-tipos-de-gamer-segun-la-psicologia.

31. E. J. López Vallejo, "*Conceptualización y desarrollo de un videojuego RPG para PC* ", Bachelor's thesis, Universitat Politècnica de Catalunya, (2014).

32. I. Sommerville, Ingeniería del software. Pearson Educación, (2005).

33. D. E. de Videojuegos, "Libro Blanco del Desarrollo Español de Videojuegos 2016", *Desarrollo Español de Videojuegos*, (2016).