UNIVERSIDAD TECNOLÓGICA ECOTEC

Facultad de ingenierías

TÍTULO DEL TRABAJO:

Videojuego educativo para ser usado como herramienta para enseñar la lógica de programación en niños de octavo grado

CARRERA:

Ingeniería en Sistemas

AUTOR:

Lucas Isaac Bonnard Silva

TUTOR:

Marcos Espinoza

Guayaquil-Ecuador

2020

DEDICATORIA

Dedico esta tesis a mis padres, que han estado para mí en todo momento y fueron un gran apoyo emocional durante el tiempo de redacción de este proyecto.

A mi Novia Abigail, quien me apoyo incondicional y me alentó a continuar inclusive en las horas más altas de la noche.

A mis amigos, que a pesar de no haber pasado tiempo con ellos, me apoyaron desde el inicio hasta el final de la tesis.

A mis profesores, por la paciencia que me tuvieron durante todo el proceso de tesis y toda mi carrera universitaria.

A mis mascotas, por brindarme su compañía incondicional durante todas las noches y madrugadas de esfuerzo y desvelo.

Para todos ellos es esta dedicatoria, debido a que si no fuera por ellos esta tesis no se hubiera podido culminar.

AGRADECIMIENTOS

En el presente trabajo de tesis, Me gustaría agradecerte a ti Dios, por haberme colmado de tus bendiciones y haber estado a mi lado en todo momento para cumplir esta meta.

Agradezco a cada uno de mis profesores, porque cada uno aporto con su grano de arena en mi formación universitaria. De forma especial, quisiera agradecer a y la Miss Erika Ascencio, por apoyarme desde mi primer día hasta el último, por su tiempo y su paciencia, por ayudarme a pesar de tener una agenda ocupada, y siempre la recordare como una excelente profesora.

Son muchas personas las cuales han formado parte de mi vida profesional, y quisiera agradecerle a uno y cada uno por sus consejos, tiempo, paciencia, cariño y consideración, pero si lo hiciera, este agradecimiento nunca terminaría, no hay palabras que expresen lo agradecido que estoy.

ANEXO N°16

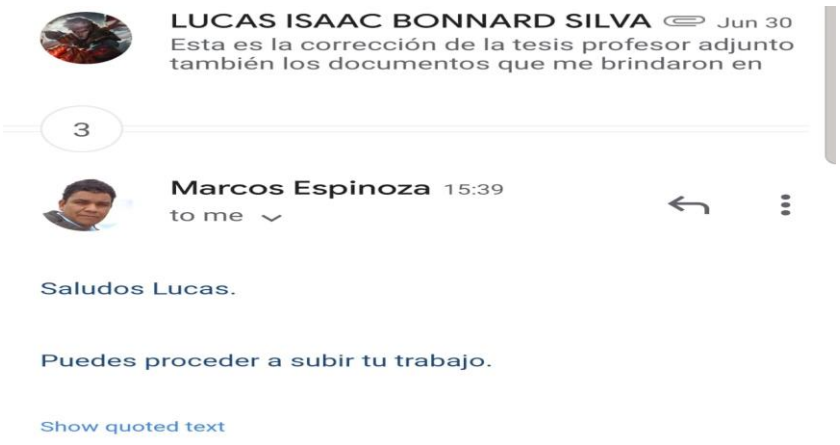
CERTIFICACION DE REVISION FINAL

QUE EL PRESENTE PROYECTO DE INVESTIGACIÓN TITULADO: **VIDEOJUEGO EDUCATIVO PARA SER USADO COMO HERRAMIENTA PARA ENSEÑAR LA LÓGICA DE PROGRAMACIÓN EN NIÑOS DE OCTAVO GRADO**

ACOGIÓ E INCORPORÓ TODAS LAS OBSERVACIONES REALIZADAS POR LOS MIEMBROS DEL TRIBUNAL ASIGNADO Y CUMPLE CON LA CALIDAD EXIGIDA PARA UN TRABAJO DE TITULACIÓN, POR LO QUE SE AUTORIZA A: **(Lucas Isaac Bonnard Silva,** QUE PROCEDA A SU PRESENTACION.

Samborondón, 02-07-2021

Nombres y Apellidos del Tutor: Marcos Antonio Espinoza Mina



ANEXO N°15

CERTIFICADO DEL PORCENTAJE DE COINCIDENCIAS

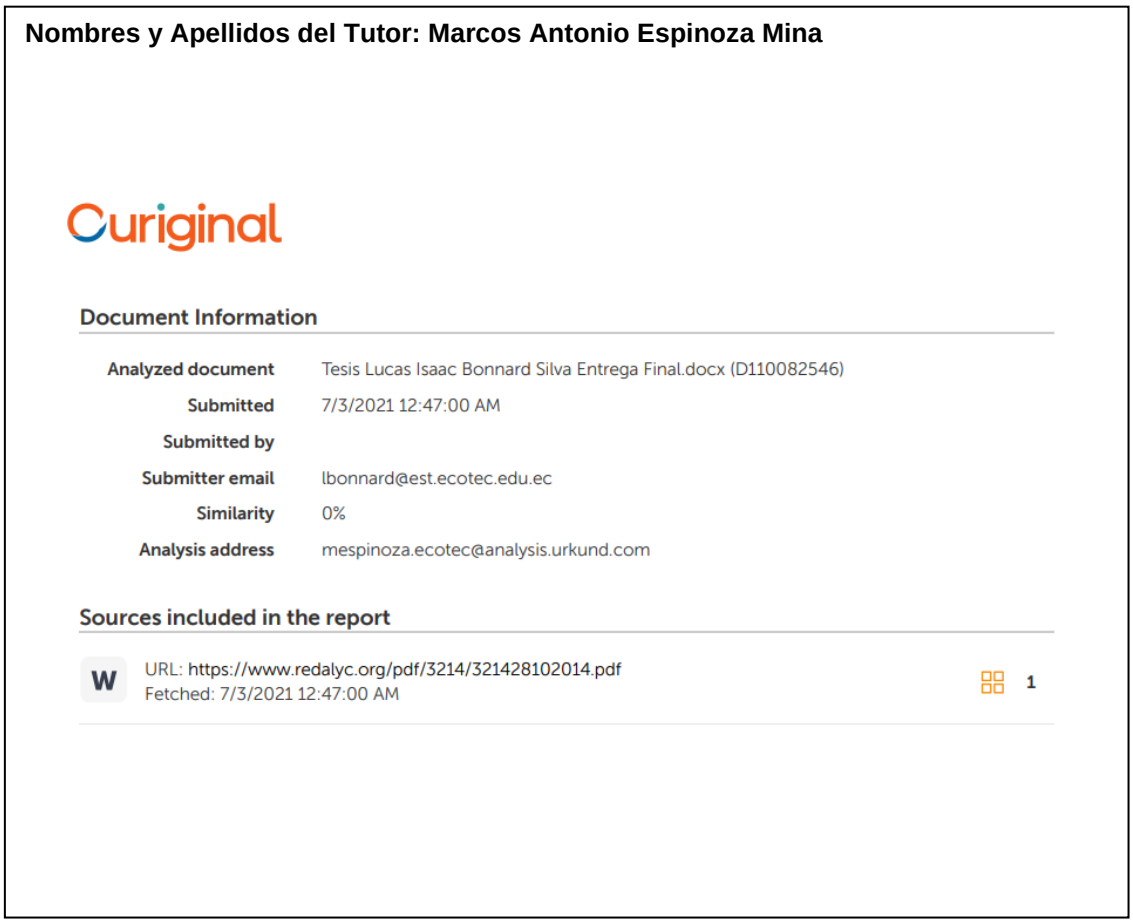
Habiendo sido nombrado Marcos Antonio Espinoza Mina tutor del trabajo de titulación" Videojuego educativo para ser usado como herramienta para enseñar la lógica de programación en niños de octavo grado" elaborado por Lucas Isaac Bonnard Silva **,** con mi respectiva supervisión como requerimiento parcial para la obtención del título Ingeniero en Sistemas con énfasis en sistemas.

Se informa que el mismo ha resultado tener un porcentaje de coincidencias 0% mismo que se puede verificar en el siguiente link:

https://secure.urkund.com/view/104129091-315924-105242

Adicional se adjunta print de pantalla de dicho resultado.

Nombres y Apellidos del Tutor: Marcos Antonio Espinoza Mina



RESUMEN

Las TIC son de suma importancia en fines educativos pues permite la representación y transmisión de información a través de múltiples formas expresivas provocando la motivación del usuario; ayudando a superar las limitaciones temporales y/o distancias geográficas entre docentes y educandos. Así mismo, facilita la extensión de la formación más allá de las estrategias tradicionales de la enseñanza presencial.

Se desarrolló un videojuego didáctico para facilitar a los docentes de la Unidad Educativa Thomas More la enseñanza del tema de Bucles en la sección de lógica de programación. Para esto se tuvo varias reuniones con los docentes donde se tomaron apuntes sobre la didáctica de enseñanza, llegando así a desarrollar un nivel en el juego obteniendo la aprobación de los maestros como un sistema eficiente para el aprendizaje de los estudiantes.

ABSTRACT

ICTs are of utmost importance for educational purposes as it allows the representation and transmission of information through multiple expressive forms, causing the user's motivation; helping to overcome temporary limitations and / or geographical distances between teachers and students, likewise, facilitating the extension of training beyond the traditional forms of face-to-face teaching.

A didactic video game was developed to facilitate the teachers of the Thomas More Educational Unit to teach the topic of Loops in the programming logic section. For this, several meetings were held with the teachers where notes were taken on the teaching didactics, thus developing a level in the game, obtaining the approval of the teachers as an efficient system for student learning.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

INTRODUCCIÓN

El uso de las Tecnologías de Información y Comunicación (TIC) con fines y transmisión de información a través de múltiples medios educativos permite la representación formas expresivas provocando la motivación del usuario; ayudando a superar las limitaciones temporales y/o distancias geográficas entre docentes y educandos. Así mismo, facilitan que se extienda la formación más allá de las formas tradicionales de la enseñanza presencial (Martinez, Ceceñas, & Martinez, 2014).

En ese contexto, se elaboran los videojuegos para satisfacer las necesidades de los niños que es manejar un aprendizaje en la lógica de programación, los cuales ayudan a fomentar la comunicación, inventiva, adaptabilidad, pensamiento crítico y persistencia. Los videojuegos en cierto modo son parte del aprendizaje, ya que algunos enseñan valores, formas de actuar e incluso crean nuevos intereses por temas como la historia, la geografía o la música (Vazquez, 2019). Por último, los videojuegos permiten al estudiante en la medida que va aprendiendo lo perciba como un desafío (Tamayo, 2018). Por lo cual, se debe aprovechar esta herramienta para desarrollar habilidades en los pequeños y así despertar su curiosidad en diferentes áreas de la educación.

Con lo anteriormente expuesto se busca que los niños adquieran el conocimiento sin la presión de estar estudiando, es decir, con mayor naturalidad. En ese sentido, la presente propuesta de tesis se enfoca en realizar un videojuego con una estructura de desafíos que incluya la resolución de problemas, para así brindarles a los docentes una nueva herramienta que puedan usar para despertar el interés de los niños y enseñarles lo que corresponde a la lógica de la programación de una manera poco usual y llamativa, ya que el enfoque que se tiene hoy en día únicamente consigue que los niños no presenten interés en esta área.

Cuando se enseña programación desde pequeños se fomenta la creatividad, el emprendimiento e independencia de pensamiento; aumenta la motivación, mejora la autonomía, se trabajan estrategias de resolución de problemas y se conocen diferentes formas de comunicación de ideas, además que esta forma de

trabajo es de prueba y error y se basa en la idea de aprender haciendo. (EDACOM, 2019).

La lógica de la programación es una técnica, la cual tiene como función el desarrollo de instrucciones en secuencia para definir un objetivo (Pelhon, 2019). La lógica de la programación brinda al estudiante la capacidad de un entendimiento más rápido cuando inicie sus estudios en algún lenguaje de programación debido a que la lógica le brinda las pautas e ideas básicas de cómo entender cualquier tipo de lenguaje, lo cual estimula su capacidad de resolución de problemas.

Planteamiento del Problema

Al realizar una revisión previa de los currículos tanto para el nivel de Educación General Básica como para el de Bachillerato General Unificado en Ecuador, se observa que los estudiantes, para avanzar hacia el perfil de salida del nivel correspondiente, deben desarrollar aprendizajes de las siguientes áreas de conocimiento: Lengua y Literatura, Matemática, Ciencias Naturales, Educación Física y Educación Cultural, Ciencias Sociales, Lengua Extranjera, y Artística (Ministerio de Educación, 2019). Siendo importante resaltar que la enseñanza de materias como informática, o de un nivel más avanzado como es la programación, no es obligatoria, la cual se debería considerar importante en la actualidad pues el mundo se encuentra en continuo avance tecnológico.

Se hace necesario entonces, a nivel de escuela, que la educación recibida por los estudiantes aporte al desarrollo de la capacidad de pensar y de aprender situando en primer plano la exigencia de una enseñanza continua o el aprender a aprender cómo es conocida (Quintero, 2016). El instruir a los niños mediante una programación lógica ayuda a comprender mejor a la sociedad actual, sus cambios tecnológicos constantes y las peculiaridades de las TIC. Aprender a programar es adaptarse y prepararse para triunfar en el mundo digital (UNIR, 2020).

De igual manera, existe la necesidad de que la educación básica ecuatoriana se apoye en herramientas que contenga una estructura llamativa y didáctica; que se use en las escuelas primarias para que los niños puedan explorar el mundo de la programación desde una edad temprana, permitiendo que las nuevas generaciones adquieran conocimiento a través de la tecnología.

Delimitación

Este proyecto tiene un carácter exploratorio porque indaga sobre una nueva forma en la que los videojuegos intervienen en el proceso de enseñanza en niños de octavo año de educación básica; a través de esta iniciativa se quiere emplear un desarrollo de aprendizaje por medio de estrategias comunicacionales.

Aprender programación en la niñez permite enfrentar errores desde un punto de vista positivo, de autocorrección y búsqueda de estos, los enfrenta a retos de resolución de problemas complejos, y a encontrar más de una solución empleando la creatividad (EDACOM, 2019)

La investigación sigue una lógica descriptiva pues explica cómo utilizar los videojuegos como herramienta para ayudar a la enseñanza en niños de conceptos complejos como programación, también describe las habilidades que se desarrollan con el uso de videojuegos educativos como habilidad mental, comprensión, estrategia, trabajo en equipo, además de preparar a los niños para la actualidad. Los videojuegos son una herramienta que los niños usan habitualmente, que ofrece múltiples posibilidades de interactuar, les permite adquirir nuevas experiencias de aprendizaje, desarrollo de estrategias, y resolver problemas (Santana, 2020).

Justificación

Considerando el contexto educativo actual, en el cual las aulas se han quedado atrás por la modalidad de educación virtual y la socialización e integración social se han limitado a comunicaciones por redes sociales; de tal manera se requiere que de esta forma los niños puedan aprender.

De ahí radica la importancia de integrar los videojuegos como herramientas que soporten el aprendizaje de una forma creativa, de manera que estos estimulen el desarrollo de las habilidades cognitivas, sociales y emocionales de los estudiantes.

Ecuador, es un país que se encuentra con frecuentes avances tecnológicos en el ámbito educativo, por lo tanto, quiere incluir el alfabetismo digital, para que no sea una desventaja a comparación de otros países, por ende, es de suma relevancia el conocimiento y uso potencial de las tecnologías. De manera que, la

ausencia de innovación e integración de recursos digitales en la educación imposibilita el avance de la sociedad. Sabiendo que, incluso aunque se innove en otros ámbitos, la educación es fundamental para el progreso social.

OBJETIVO GENERAL

Desarrollar un videojuego que sirva como herramienta a los docentes para la enseñanza de la lógica de programación en niños de octavo de básica.

OBJETIVOS ESPECÍFICOS

- Revisar a nivel teórico la didáctica de enseñanza de la lógica de programación, los diferentes usos educativos de los videojuegos y las estrategias empleadas.

- Desarrollar un videojuego que permita a los docentes la enseñanza a los niños de octavo año de educación básica, en la lógica de programación.

- Probar el videojuego con docentes que impartan la enseñanza a niños de octavo año de educación básica.

CAPÍTULO I

MARCO TEÓRICO

1. DIDÁCTICA DE LA ENSEÑANZA EN LÓGICA DE PROGRAMACIÓN

1.1. Didáctica en la enseñanza

La didáctica explica los fundamentos teóricos de la educación y la formación, e interviene en el proceso enseñanza-aprendizaje, donde es necesaria la combinación de la práctica, que resulta importante, pues el ser humano aprende mediante la experiencia, sin olvidar que debe ir acompañada de la teoría, la misma que debe ser aplicable a la realidad. También se puede mencionar que, la didáctica es la interacción que se genera entre todo lo que se encuentre alrededor, que pueda facilitar el proceso de enseñanza-aprendizaje como: la materia y su objetivo, el docente, los alumnos, el medio gráfico, directivos, infraestructura. Es una disciplina pedagógica pragmática y normativa, cuyo objetivo específico es incentivar y orientar eficazmente a los alumnos en sus aprendizajes (Blanco, 2017).

1.2 Tipos de didáctica

La didáctica se puede clasificar de la siguiente manera:

1.2.1 Didáctica Diferenciada

Surge a partir de la particularidad de elementos como los sujetos, contenido, ambiente, etc., promoviendo particularidades, anulándolas y modificándolas. Es considerado que es la didáctica que se adapta a la diversidad de los individuos y de los procesos (Raffino, 2020).

La didáctica diferenciada se aplica a situaciones en las cuales se está enseñando algo en específico, para esto se toman en consideración desde la edad hasta las competencias intelectuales de los estudiantes

1.2.2 Didáctica General

Conjunto de normas que para dirigir el proceso de enseñanza-aprendizaje, se fundamentan de manera global con enfoque educativo, utilizando métodos de enseñanza válidos sin importar el ámbito o materia en específico. Analiza los elementos que pueden repetirse en diversas ocasiones, y presenta prototipos para explicar lo analizado y poder aplicarlo (Raffino, 2020).

1.2.3 Didáctica Especial

Más conocida como específica, estudia los métodos y prácticas realizados en cada campo, disciplina o materia de estudio, dando pie a diferenciaciones entre los métodos y las prácticas, además de valorar y determinar los beneficios de los alumnos según la materia. Retoma las normas creadas por la didáctica general y las aplica específicamente en una materia en particular, por lo que resulta más específica que la didáctica diferencial (Raffino, 2020).

1.2.4 Lógica de programación

Esta gira en torno al concepto de relacionar los elementos, por medio de una serie coherente de ideas y razonamientos lógicos, en donde la representación intelectual de un objeto puede ser la solución para un problema. La Lógica es ciencia de relaciones porque estudia el pensamiento y, pensar es establecer relaciones (EcuRed, 2020).

Cuando se enseña la lógica de programación a niños, les permite mejorar el razonamiento, facilita el aprendizaje de áreas como matemática, estimula la creatividad, les permite desarrollar la capacidad de atención y fomenta el trabajo en equipo. Aprender programación tiene muchos beneficios para los niños como la estimulación del pensamiento crítico, reconocer patrones y secuencias, crear algoritmos, diseñar pruebas para encontrar y corregir errores (Castro, 2020). Esto le permite al niño poder desarrollarse en diferentes situaciones, sin mayor problema.

La lógica de la programación es el paso que se requiere para poder ser un buen programador, ya que esta le brinda al estudiante la capacidad de entender o comprender de una manera más fluida la programación, una vez que la lógica se comprende, el alumno puede aprender de forma clara y sencilla, obteniendo conocimientos adquiridos de acorde a la programación.

Más allá del poder facilitar el aprendizaje de programación, la lógica de la programación puede brindar al alumno más tipos de beneficios, se puede desarrollar la capacidad de análisis y de síntesis además mejora la capacidad de

resolver problemas y la agilidad en toma de decisiones (Palacios, 2018). (Castellanos, Castellanos, Salazar, & Casas, 2016)

1.2.5 Teoría de videojuegos aplicados a la enseñanza

La teoría de videojuegos surge a principios del siglo XXI. Esta teoría, introducida por Mark Wolf y Bernard Perron, marca una diferencia con la teoría de juegos tradicional, la cual proporciona pautas para la comprensión y construcción de juegos. La diferencia radica en que, en los videojuegos, el computador revela y dirige las reglas del juego. Pero más allá de eso, el diseñador del juego cumple un rol fundamental en la creación del universo del juego y, por tanto, de sus reglas (Guardiola & Natkin, 2005). En ese sentido, estas últimas se vuelven poco susceptibles de modificación por parte de los usuarios. A su vez, este contexto no es considerado en las actividades lúdicas tradicionales.

Dadas las implicaciones cognitivas de los videojuegos, estos han trascendido al ámbito educativo, por lo que es cada vez más común su utilización en la enseñanza de diversas asignaturas (Núñez-Barriopedro, Sanz-Gómez, & Ravina-Ripoll, 2020). Su aplicación en las aulas ha demostrado que proporcionan condiciones idóneas para estimular la exploración y experimentación, y el desarrollo de habilidades comunicativas, intelectuales, sociales y afectivas en los estudiantes (Martínez, Egea, & Arias, 2018).

En específico, la teoría de juegos aplicada a la enseñanza abarca todas las necesidades del estudiante, es decir, le proporciona un papel protagónico y autogestionado en un ambiente en donde este tendrá la libertad de cometer errores y aprender a partir de ellos (Martínez, Egea, & Arias, 2018). Así mismo, podrá desarrollar su capacidad de resolución de problemas a través de la lógica que, per se, poseen los videojuegos.

1.2.6. Diferentes usos educativos de los videojuegos

Los profesores hoy en día tienen acceso a muchas herramientas educativas, las cuales permiten al docente mejorar el aprendizaje del alumno y desarrollar mejor sus habilidades. En la actualidad se cuenta con una herramienta con alto poder a la hora de diseñar procesos de aprendizajes que están basados en la motivación,

diversión, inmersión e interactividad enfocados en un propósito pedagógico (Castellanos, Castellanos, Salazar, & Casas, 2016).

Muchos videojuegos que se usan actualmente como entretenimiento pueden ser utilizados como una herramienta pedagógica dependiendo como el docente decida manejarlo, es un amplio margen de tipos de conocimientos que un niño puede adquirir luego de una experiencia con los videojuegos empleados de una manera pedagógica, unas cuantas a mencionar serian: Lectura, pensamiento lógico, observación, espacialidad, geografía, vocabulario, ortografía, resolución de problemas, planificación de estrategias, e idiomas, como sabemos los niños tienen la capacidad de aprender un idioma mucho más rápido que los adultos, por ende al presentarle un video juego a un niño en donde le enseñen a base de retos la parte de la gramática y el vocabulario, este podrá asimilar de forma divertida la información y mejorará su léxico.

Existen diferentes tipos de videojuegos y cada uno puede ser usado para dar enseñanzas totalmente distintas, tal como lo presenta (Ferrer, 2001). Respecto a ello, en la tabla 1 se resumen los tipos de videojuegos más conocidos que se relacionan con la estimulación del desarrollo de habilidades cognitivas, emocionales y sociales.

Tabla 1: Videojuegoseduca - Clasificación de los videojuegos

Arcade	(Juegos tipo plataforma, luchas) Ejemplos: Pacman, Mario, Sonic, Doom, Quake, Street Fighter, Arcanoid. Mejoran la orientación espacial del estudiante, ayuda a la visualización de problemas y a resolverlos.
Deportes	Ejemplos: FIFA, PC Fútbol, NBA, Formula I Grand Prix, Need For Speed. Permiten el desarrollo de habilidades de coordinación psicomotora y el aprendizaje las reglas y estrategias de los deportes.
Juegos de aventura y rol	Ejemplos: King Quest, Indiana Jones, Monkey Island, Final Fantasy, Tomb Raider and Pokémon. Se desarrollan Motivaciones nuevas enfocadas a una temática respectiva, la cual será estudiada en clases de una manera más sistemática.
Simuladores y construcción	(Aviones, maquinarias, ciudades). Ejemplos: MINECRAFT, Simulador de vuelo Microsoft, Sim City, Tamagotchi, The Incredible Machine, Theme Park. Permite desarrollar interés en la experimentación y en la investigación de funcionamiento de máquinas, situaciones y fenómenos.
Juegos de Estrategia	Ejemplos: Strategy, Warcraft, Age of Empires, Civilization, Lemmings, Black & White and Centurion. Desarrolla el ingenio que permite estrategias para lograr un objetivo y prevenir las acciones del enemigo y actuar ante estos.
Puzzle y juegos de lógica	Ejemplos: 7th.Guest, Tetris. Desarrollan la imaginación, creatividad, la percepción espacial y la lógica.
Juegos de preguntas	Ejemplos: Trivial, Carmen Sandiego. Estos juegos pueden servir para el rendimiento de examen o repaso de temas específicos.

1.2.7 Aplicaciones de los videojuegos en trabajos relacionados

Un ejemplo de la implementación de los videojuegos en unas aulas de clase ubicada en Ourense España, con un total de 93 alumnos, La intervención se realizó utilizando el videojuego “WII MUSIC” pertenecientes a la consola NINTENDO WII, y Earmaster, un videojuego disponible para computadora. Cabe recalcar que la clase en la cual se estaba implementando es dinámica y se produce por medio de la música. En la tabla 2 se describen las características de los videojuegos aplicados en el caso de estudio previamente mencionado.

Tabla 2: Videojuegos usados en unidad educativa.

Videojuego	Wii Music	Earmaster versión 4.0
Plataforma	Nintendo Wii	Windows / Mac
Género	Otros, Música	Entrenamiento auditivo
Desarrollador	Nintendo	EarMasterAp -Dinamarca
Jugadores	De uno a cuatro	Individual
Características generales	El wii music es un programa didáctico que presenta varias alternativas, como minijuegos. Trabaja de manera entretenida o recreativa ayudando en el aspecto rítmico y en la percepción auditiva, adquiriendo conocimiento musical e identificación de los instrumentos	Ear Master es un programa estimulante auditivo que ayuda en la clasificación auditiva. Está cuenta con un amplía variedad de prácticas como son la identificación y comparación de intervalos, identificación de acordes y de escalas, dictado rítmico entre otros, cuyos ejercicios creativos aumenten el nivel de dificultad conforme avancen.

Los 2 juegos fueron implementados de manera que se complementen. El Wii Music se lo presento utilizando la consola, este juego es totalmente lúdico e infantil con el que se trabajó el aspecto rítmico y auditivo. Por otro lado, el Earmaster para su utilización y desarrollo se implementó la didáctica por turnos utilizando la pizarra digital, ya que permite una mejor interacción y llama más la atención del estudiante, cabe recalcar que este último no es un juego enfocado a un público infantil, ya que la sobriedad del juego recalca que es para un público más maduro, a pesar de dicho detalle el juego se presentó de una manera que sea atractiva. Además, se hizo necesaria una explicación teórica días antes y posteriores del uso del juego para confirmar la retentiva.

El Wii Music se utilizó para la enseñanza rítmica, entrenamiento auditivo y el conocimiento de instrumentos musicales y repertorio, por otra parte, el Earmaster

se utilizó para poder reafirmar lo aprendido en el Wii Music a base de pruebas y preguntas.

Otro claro ejemplo de la capacidad que tienen los videojuegos en la rama de la educación fue “Games for Learning Summit” siendo la primera cumbre de videojuegos para el aprendizaje, la cual tubo desarrollo en abril del 2015, en esta cumbre participaron expertos en la educación, profesores, estudiantes y desarrolladores de videojuegos, con el objetivo de encontrar nuevas estrategias pedagógicas para generar un mayor interés de aprendizaje en el alumno

En Escocia, los videojuegos llegan a ser utilizados como una herramienta de aprendizaje hasta con los infantes (Bourne & Salgado, 2016). “The Consolarium”, es un proyecto en Escocia financiado por dicho gobierno, el cual incentiva al uso de los videojuegos en las aulas de clase, para que de esta manera los niños puedan desarrollar valores cívicos, respeto y educación.

Ubicada en Nueva york, se puede encontrar la primera escuela pública basada en videojuegos, fue creada en el año 2009, su metodología está enfocada en la enseñanza por medio de la gamificación. Los estudiantes de esta escuela aprenden superando retos y niveles basados en los juegos, el alumno es motivado a investigar más acerca del tema en el cual está enfocado el juego que está jugando y de esta manera poder superar los niveles, cada juego es creado de acuerdo con el tema de estudio.

El objetivo es crear en el estudiante un entusiasmo por aprender, de tal manera el alumno no se sienta forzado a tener que investigar para obtener una buena calificación.

1.2.8 Metodologías de desarrollo de videojuegos

En los últimos años, los estudios sobre la implementación de videojuegos en el ámbito educativo han aumentado, dado el interés de la comunidad educativa en conocer sus efectos sobre los elementos del entorno de aprendizaje: alumnado, metodología, contenidos, relación docente-aprendiz, entre otros (Dorado & Gewerc, 2017).

Sin embargo, al momento de considerar como referencia el diseño y metodología llevada a cabo por los estudios realizados, es necesario tomar en cuenta la variable cultural como delimitante para acoger a aquellos estudios que se asocien con el contexto en el cual se desarrolla el presente trabajo. En ese sentido, en los siguientes párrafos se destaca el método más relevante de la literatura en lo que respecta a los videojuegos como herramienta educativa.

De forma general, se sugiere que el ciclo de vida iterativo de videojuegos consta de cuatro etapas: pre-producción, producción, pruebas y post-producción. En la etapa de pre-producción se define la idea del juego, así como los requisitos técnicos y la planificación de producción. En la etapa de producción, como su nombre lo indica, se ejecuta lo planificado en la etapa anterior. Luego, para asegurar la calidad de funcionamiento del juego, se realizan pruebas. Por último, la fase de post-producción constituye una serie de pruebas para evaluar el rendimiento del videojuego (Prieto, 2018).

Jiménez et. al (2016) coinciden con el ciclo anterior; no obstante, en el caso de los videojuegos aplicados a la educación, proponen reducir el proceso a tres fases: pre-producción, producción y post-producción. Siendo la planificación de producción, desarrollo de software e implementación pedagógica, respectivamente.

En contraste, Paderewski et. al (2017) proponen cinco módulos para el diseño o arquitectura del juego y estos son: diseño, personalización, de juego, de monitorización y gestión de grupos. Según los autores, estos módulos posibilitarán el diseño de un videojuego educativo que cumpla las condiciones adecuadas para su efectividad.

Dado el énfasis e implicaciones prácticas que poseen las metodologías propuestas por Jiménez et. al (2016) y Prieto (2018), considerando también los propuesto por Paderewski et. al (2017); para este proyecto se consideran las siguientes fases: concepción, diseño, planificación, producción y pruebas. En la ilustración 1 se muestra el flujo de las fases mencionadas.

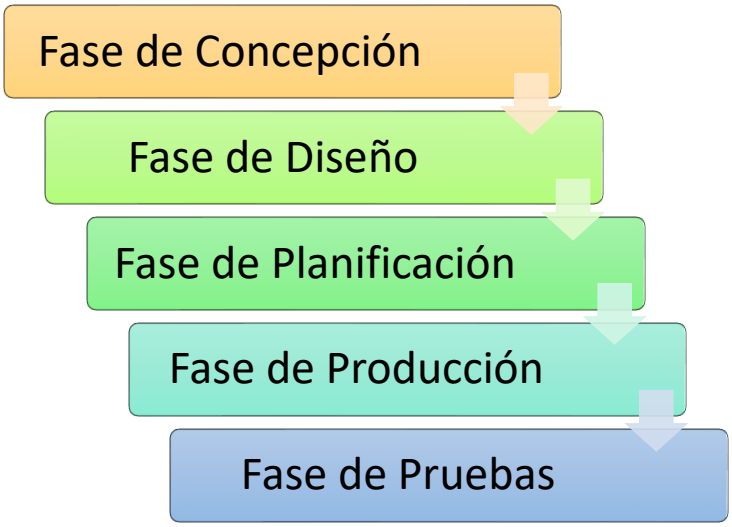


Ilustración 1 Fases del desarrollo de un videojuego

1.2.9 Metodologías de desarrollo del software

Aunque las metodologías de desarrollo de software no varían en lo esencial, sí que se puede hablar de modelos o marcos de trabajo distintos. Estos corresponden a métodos de trabajo que han sido creados para satisfacer necesidades específicas en los proyectos y entre los más utilizados se encuentran: Modelo en cascada, Modelo de desarrollo evolutivo y modelo de componentes (Perez, 2016)

Modelo Cascada: este modelo se basa en la condición de que mientras mayor sea el número de tareas ejecutadas, las actividades estarán cada vez más relacionadas. Además, considera que la especificación, validación y evolución son las acciones principales para el desarrollo de un software. Resultando también determinantes el diseño, la implementación y las pruebas en la elaboración de un software.

Modelo de desarrollo evolutivo: En este modelo las actividades de especificación, desarrollo y validación están entrelazadas; y estas son las más importantes. Específicamente, el punto de partida del modelo se desarrolla de forma rápida y va evolucionando de acuerdo con la dinámica del proyecto y a las

peticiones del consumidor. Todo el proceso es una evolución continua que se detiene cuando los objetivos iniciales han sido alcanzados.

Modelo de componentes: Se trata de un modelo que se desarrolla del trabajo que otros han llevado a cabo, reutilizando las partes que ya no aportan beneficios a otros proyectos e integrándolas en una nueva metodología de desarrollo. En resumen, el modelo se basa en la adecuación y adaptación de partes de modelos no desarrollados, que al final de este acaban cobrando un nuevo valor y asumiendo otras funciones.

Ante todo, lo mencionado, se seleccionó el modelo tipo cascada por ser el más adecuado a la línea de lo que sería la creación de un videojuego aplicado a la enseñanza. Además, hace énfasis en la implementación de pruebas para confirmar que el software funciona de manera correcta. A su vez, resalta la importancia de cada paso; ya que al intentar corregir algo, pone en riesgo todo el proyecto y los resultados de cada fase pasan a ser parte de la fase siguiente.

CAPÍTULO II

METODOLOGÍA DEL PROCESO DE DESARROLLO DE LA PROPUESTA TECNOLÓGICA

2.1. Enfoque de la investigación

Los enfoques de investigación cualitativa se enfocan en un análisis profundo (Mata, 2019). Este proyecto tiene un enfoque cualitativo, ya que para el desarrollo del proyecto se tomó en cuenta una realidad social. Con base a la información obtenida en entrevistas y reuniones, se pudo delimitar los parámetros para el desarrollo de este, el cual es un videojuego que puede ser usado como herramienta en la enseñanza de la lógica de la programación.

2.2. Tipo de investigación

La investigación sigue una lógica descriptiva, pues explica cómo utilizar los videojuegos como herramienta para ayudar a la enseñanza en niños de conceptos complejos como programación. También describe las habilidades que se desarrollan con el uso de videojuegos educativos, tales como: habilidad mental, comprensión, estrategia, trabajo en equipo y preparación para la actualidad. Por todo aquello, los videojuegos constituyen una herramienta que ofrece múltiples posibilidades de interactuar, adquirir nuevas experiencias de aprendizaje, desarrollo de estrategias y resolución de problemas (Santana, 2020).

2.3. Periodo y lugar donde se desarrolla la propuesta tecnológica

El sitio donde se dio lugar para el desarrollo de la propuesta tecnológica fue la Unidad Educativa Thomas More y el periodo para dicho proyecto fue entre el 25 de abril del 2021 hasta el 11 de junio del 2021.

2.4. Universo y muestra

El universo es un conjunto de elementos definidos en un atributo, y se valoran todos los elementos que lo componen. En ese sentido, el universo de esta investigación se limita a un solo profesor, puesto que el proyecto está enfocado a niños que se encuentran cursando 8vo año de educación, y para este nivel solo hay un profesor encargado de la materia. Además de que se conoce que, cuando la población es pequeña, se toma el muestreo completo.

Por otro lado, para reforzar esta investigación, se escogió a dos docentes especialistas en la enseñanza, quienes también corresponden a la muestra para tomar en cuenta. Una muestra indica que un grupo reúne las características, las cuales son importantes para una investigación (Espinoza, 2016).

2.5. Definición y comportamiento de las principales variables incluidas en el estudio

En este proyecto se tiene una variable independiente y una variable dependiente. La variable independiente corresponde a la aplicación del videojuego que sería utilizado como herramienta de enseñanza, y la variable dependiente sería el proceso de enseñanza utilizando dicho videojuego. En la tabla 3 se detalla los conceptos, indicadores e instrumentos o métodos de las variables.

Tabla 3: Definición y comportamiento de las principales variables incluidas en el estudio

VARIABLE	CONCEPTUALIZACIÓN	INDICADORES	INSTRUMENTOS Y/O MÉTODOS
Aplicación de Videojuego	El videojuego tiene aspectos positivos con respecto al aprendizaje de nuevas habilidades (Fuentes & Perez, 2015).	Para la creación del videojuego se dedicó más de más de 30 horas y este se basó en las especificaciones obtenidas de la entrevistas y reuniones realizadas.	Entrevistas y reuniones
Proceso de aprendizaje y enseñanza	Un videojuego puede tener como finalidad enseñar una materia en concreto y llegar a tal punto de poder llegar a entrenar a	Los profesores encargados de calificar o aprobar el proyecto dieron un visto bueno	Entrevistas y reuniones

	profesionales (Iberdrola, 2021).	con respecto al objetivo de este proyecto que es un videojuego que sea usado como herramienta de enseñanza de la lógica de la programación.	

2.6. Métodos empleados e instrumentos de la investigación

El método empírico es utilizado todos los días para encontrar soluciones o respuestas a los problemas que nos rodean (Ceron, Perea, & Figueroa, 2020). En la línea de obtención de datos e investigación de este proyecto se emplea un método empírico, ya que para obtener dicha información se valió de entrevistas y reuniones de grupo; las cuales fueron instrumentos que brindaron información para la creación del videojuego y la presentación del proyecto.

Por una parte, las reuniones de grupo, también conocidas como “grupos focales” o en inglés “focus group”, representan una de las herramientas más utilizados en el ámbito de investigación cualitativa (Robinson, 2020). El éxito de esta herramienta proviene de su capacidad para captar las experiencias y percepciones de los participantes, permitiendo una comprensión profunda de estas. Por ello, se recomienda el uso de esta herramienta en investigaciones asociadas a la educación, para mejorar la experiencia de los estudiantes y el personal educativo en general (Rosenthal, 2016).

Por otro lado, las entrevistas, al igual que los grupos focales, también corresponden a una de las herramientas más empleadas para la ejecución de investigaciones de carácter cualitativo. Pero más allá de eso, estas proporcionan más información sobre los pensamientos, sentimientos y percepciones del entrevistado; diferenciándola de los grupos focales (Álvarez-Álvarez & Vejo-Sainz, 2017). Por ello, es recomendable el uso combinado de las entrevistas y grupos focales, ya que las entrevistas permiten consolidar la información “superficial”

obtenida a partir de las reuniones de grupo (Guest, Namey, Taylor, Eley, & McKenna, 2017).

Respecto a lo detallado previamente, para el desarrollo de este proyecto se ejecutó un total de 2 reuniones y 2 entrevistas antes de la presentación y posterior aprobación del proyecto. En la primera reunión se presentó la idea del proyecto, es decir, la utilización de los videojuegos en el ámbito educativo. A su vez, este encuentro posibilitó identificar las necesidad del alumnado y definir de forma más específica la propuesta de solución. Luego se desarrolló la primera entrevista para decidir los temas adecuados que el videojuego debería abarcar. Posteriormente, en la segunda reunión se presentó un storyboard del videojuego considerando los posibles niveles que debería poseer. Finalmente, en la última entrevista se acordó de forma descriptiva el nivel de videojuego a desarrollar, es decir, el bucle; basado en los interés del docente.

Cabe mencionar que la primera y segunda reunión de grupo se llevó a cabo el 25 de abril y 1 de mayo, respectivamente. Mientras que la primera y segunda entrevista se desarrolló el 26 de abril y 4 de mayo, respectivamente. Estas fueron ejecutadas en la Unidad Educativa Thomas More en conjunto con el docente responsable de octavo grado. A partir de estas reuniones y entrevistas fue posible continuar con las actividades propuestas para la solución. Del anexo 1 al 7 se muestran los formatos empleados en la entrevista y grupo focal.

CAPÍTULO III

ANÁLISIS E INTERPRETACIÓN DE LOS RESULTADOS

3. PRUEBAS FUNCIONALES

Para la verificación del correcto funcionamiento del videojuego se realizaron diferentes pruebas, con el fin de poder observar y analizar las inconsistencias que se pueda tener tanto en los personajes como en la interfaz del programa, además de revisar la información obtenida para responder a los diferentes cuestionamientos que se plantearon para esta investigación.

3.1. Pruebas unitarias

Este tipo de prueba se centra en una pieza o parte específica del software, estas pueden ser módulos, procedimientos o funciones, cualquier parte del software puede someterse a dicha prueba (Ramos, 2017). Para esta prueba se revisó de manera individual los siguientes aspectos: funciones básicas del personaje y movimiento del enemigo. Se seleccionaron dichas mecánicas, debido a que no requieren una interacción con otros objetos para desempeñar su función.

3.1.1. Funciones básicas del personaje

Para esta prueba se revisó de manera individual las siguientes funcionalidades del personaje:

- Movimiento
- Mecánica de salto
- Acción de golpe
- Reaparición post desaparición

3.1.1.1. Movimiento

Para esta prueba se le envió la orden al personaje para que el corriera en el mapa. Dicha orden se realizó al presionar los botones correspondientes a las flechas de izquierda y derecha. El objetivo principal de estas pruebas era poder confirmar si el personaje dejaba de realizar la acción al soltar dichos botones; es decir, saber si los botones funcionaban como un pulsador o como un interruptor que, al presionarlo una vez, para desactivarlo, debería volverse a presionar.

Para ello se le ordenó al personaje que corriera de forma continua en ambas direcciones. Al soltar el botón de manera repentina, el resultado que se esperaba obtener con esta acción era que el personaje se detenga en el momento; en otras palabras, se necesitaba que los botones trabajaran como pulsadores y, efectivamente, el personaje se detuvo.

En una ocasión que se realizó una prueba, se pudo observar que el personaje siguió corriendo, a pesar de soltar el botón. Por ende, se realizó una revisión dentro del algoritmo y se comprobó que no había ningún error en su programación. No obstante, se procedió a examinar el hardware y se confirmó que el botón se había quedado atascado; por lo cual el personaje seguía desempeñando la función que era enviada por el botón. Así que se descartó un error de codificación.

Posteriormente, al limpiar el teclado y repetir la prueba, se confirmó que el personaje podía realizar de manera correcta la acción de correr y parar. Por lo tanto, la programación realizada para el movimiento del personaje era correcta y cumplía en totalidad su objetivo.

3.1.1.2. Mecánicas de salto

Correspondiente a esta prueba, se procedió a ubicar al personaje en el inicio del juego y enviarle la orden de salto con la tecla de dirección hacia arriba, el personaje salto con normalidad, al confirmar que el mismo reaccionaba a la orden de salto, se continuó enviando dicha orden con un lapso de 2 segundos entre cada una para asegurar que todo funcione de manera correcta. Luego de corroborar que la mecánica se esté dando de forma correcta, se procedió a confirmar el salto doble, el cual representa a un segundo salto al haber presionado por segunda vez la tecla de dirección hacia arriba. A continuación, se realizaron algunas pruebas mandando la orden del doble salto, en diferentes tiempos, obteniendo resultados favorables, es decir, que las mecánicas funcionan y cumplen su objetivo.

Existe una mecánica de salto diferente a la ya antes mencionada, la cual es el salto de emergencia o precaución, ocurre cuando el personaje se encuentra cayendo debido de cualquier factor a excepción de un salto, para esta prueba se

ubicó al enemigo de tal manera que se dirija al vacío y se envió la orden de salto, cabe recalcar que el salto normal está condicionado que funcione solo si está tocando una base, por lo tanto, este salto se diferencia ya que el personaje saltaría sin tener algún asentamiento. Al estar cayendo se le ordeno al personaje saltar, y efectivamente el personaje realizo con éxito dicho salto de emergencia o precaución, se realizó esta prueba un total 5 veces, las cuales en todas se cumplió de manera correcta la acción.

3.1.1.3. Acción de golpe

En esta prueba se procedió a revisar si la acción de ataque o golpe se ejecutaba de la manera correcta, dicha acción fue programada para realizarse al presionar la barra espaciadora, la animación de golpe o ataque se desarrolló con perfección. Para confirmar que la programación fue hecha de manera adecuada, se optó por presionar la barra espaciadora un total de 5 veces de manera consecutivas, el resultado fue que el personaje desarrollo la animación de golpe a la perfección.

3.1.1.4. Reaparición

Una de las principales funciones del personaje, es no poder ser eliminado, se programó que cada vez que el personaje salga o caiga fuera de las áreas delimitadas por la cámara ,este reaparecería en la posición en la que se encontraba al iniciar el nivel , para esta prueba se tomó al personaje y se lo ubico de manera que cayera al vació, para esto se podía obtener 2 posibles resultados, primero que el personaje no se redirija a la zona inicial, como esta previsto, o que reaparezca en la misma zona donde cayo, y siga cayendo de manera infinita cada vez a una velocidad más alta, esto provocaría un colapso dentro del juego, el segundo resultado sería que el personaje reapareciera en la zona delimitada. Después que el personaje paso los limites designados dentro del área de cámara, reapareció exitosamente en la zona de inicio, se procedió a realizar la misma prueba en varias zonas del mapa, y en cada una se tuvo éxito.

Para confirmar que la codificación relacionada a la reaparición había sido realizada de la manera adecuada, se cambió las coordenadas en las cuales el

personaje reaparecería después de salir del área de cámara designada, de tal manera se optó por cambiar las coordenadas un total de 3 veces. La primera coordenada que se colocó en una de las plataformas móviles, la segunda coordenada fue sobre una plataforma estática ubicada en el extremo derecho del mapa, y la tercera coordenada fue en una zona por encima del mapa, de tal manera que cuando el personaje desaparezca, aparecería arriba del mapa y caería hasta tocar el suelo. Después de estos 3 cambios de coordenadas, se confirmó que la codificación con respecto a la reaparición del personaje principal se encontraba funcionando de manera adecuada.

3.1.2. Movimiento del enemigo

Para esta prueba se requirió netamente la observación, ya que basado en su programación, el enemigo tiene movimientos independientes, además de tener la función de no detenerse y cada vez que su velocidad es reducida a 0, cambiaria su dirección de manera automática, dicha prueba fue un éxito ya que basado en su programación y en la observación de sus acciones, el enemigo cumple con éxito su función de movimiento.

3.2. Pruebas de integración

El objetivo de esta prueba es validar la integración de 2 componentes para realizar o completar una acción. Se revisaron los 2 tipos de interacciones que se encuentran dentro del juego, las cuales son: Funcionalidad de las plataformas e Interacción entre enemigo y protagonista (Lee, 2020).

3.2.1. Funcionalidad de las plataformas

Dentro del juego se puede encontrar 2 tipos de plataformas: Plataformas móviles y plataformas que se despliegan al vacío al ser tocadas. Ambas plataformas en un punto del juego tienen que interactuar de manera directa con el protagonista.

3.2.1.1. Plataformas Móviles

La plataforma móvil posee su propia velocidad. Dentro de la codificación se programó que, el protagonista, al tocar dicha plataforma, viaje a la velocidad de esta. Cabe la posibilidad que dicho código falle cuando el personaje esté saltando de una plataforma a otra. Debido a esto, se realizó una prueba que consistió en ordenar al protagonista que salte entre las plataformas móviles. De tal manera se podría confirmar si en alguna de las colisiones entre personaje y plataforma, sus velocidades dejaran de coincidir. Después de ordenar al personaje que salte continuamente de plataforma en plataforma un total de 5 veces, se confirmó que la interacción entre esta era correcta.

3.2.1.2. Plataformas que se despliegan al vacío al ser tocadas

Basado en la programación de esta plataforma, esta solo se desplegaría al vacío si el personaje principal la toca, en cuestión dicha plataforma solo cumpliría su función si es tocada por el personaje, dichas plataformas también posen 2 características extras, las cuales son: reaparecer al ser tocadas después de 2 segundos y brindarle al personaje tiempo de saltar, en contexto la plataforma no se desplegaría al vació de inmediato, si no que le brindaría al personaje un tiempo de 0.5 segundos.

La prueba consistió en ordenar al personaje que atraviese la línea de plataformas que se encuentran dentro del juego, esto lo haría de ida y vuelta un total de 5 veces, con esta prueba se podría confirmar: el tiempo de respuesta de la plataforma al ser tocada, la función de despliegue al vacío y la capacidad de reaparecer luego de transcurrir 2 segundos. Al completar dicha actividad en las plataformas, se pudo confirmar que las características antes mencionadas se cumplen de manera correcta.

3.2.2. Interacción enemigo - protagonista

Para esta prueba se confirmó 3 aspectos en específicos, para que estos ocurran se requiere la interacción de los objetos “Player” y “Enemy” correspondientes al protagonista y enemigo, dichos aspectos son los siguientes:

- Efectividad de ataque del protagonista
- Animación de desaparición
- Reaparición de enemigo al ser eliminado

Se inició la prueba ordenando al protagonista atacar al objeto enemigo, de esta manera se confirmó la efectividad del golpe. A su vez, se optó por revisar de manera interna los límites que tenía dicho golpe, ya que esta mecánica se había configurado de tal manera que el golpe del protagonista alcanzara una distancia en específico, para que luego se pudiera atacar a los enemigos sin que estén muy cerca del personaje principal. Con la prueba mencionada se logró confirmar que las limitaciones del puño se encontraban bien.

Luego de efectuar el golpe, el enemigo automáticamente era eliminado; con esto también se logró confirmar la animación de desaparición de este. Después de dicha eliminación, el enemigo reapareció en el punto especificado en la programación, cumpliendo así con lo planificado. Dadas las configuraciones, se confirmaron los 3 aspectos mencionados previamente.

Adicionalmente, luego del primer golpe, se ordenó al protagonista eliminar enemigos constantemente para así confirmar si existía alguna falla o límite en la reaparición de los enemigos y, a su vez, terminar de confirmar la funcionalidad del ataque. Esto se realizó durante 10 segundos, al finalizar ese tiempo se confirmó que la interacción protagonista-enemigo funcionaba y se desempeñaba conforme a lo adecuado.

3.2.3. Prueba de Menú

En específico, para la prueba del menú se procedió a comprobar la conexión que poseía con la escena del nivel ya realizado; para esto se inició el juego un total de 5 veces y se accedió al nivel de bucle usando su respectivo botón. A su vez, se comprobaron que los otros botones estén deshabilitados, ya que solo tenía que estar habilitado el botón correspondiente al nivel de bucle. Con esta prueba se pudo confirmar que la conexión entre las escenas de menú y nivel funcionan adecuadamente.

3.3. Prueba de aceptación

Se enfocan en la aceptación de los criterios designados previamente por el usuario; es decir, la funcionalidad del juego, el diseño amigable para el estudiante y la sencillez en la accesibilidad. En la presentación del Software a los docentes de la Unidad Educativa Bilingüe Thomas More, cada uno de ellos probó de manera individual el nivel presentado.

Durante las pruebas se revisaron las mecánicas básicas de cada objeto en el software y que dicho programa cumpla su función, la cual era ser usada como herramienta en la enseñanza de la lógica de la programación. Cada docente revisó el programa durante un estimado de 3 a 5 minutos. Después de un par de intentos por parte de cada uno de los docentes, todos acordaron que el videojuego propuesto cumplía su función y podía ser usado como ejemplo para representar lo que es un Bucle (Mera, 2015).

CAPÍTULO IV

IMPLEMENTACIÓN DE LA SOLUCIÓN TECNOLÓGICA

4. Metodología de implementación

La metodología que se optó por elegir en este proyecto fue el tipo cascada, pues sus partes son: análisis, diseño, implementación, verificación y mantenimiento, lo que se adapta perfectamente a la creación de un videojuego, independientemente del propósito que tenga, debe pasar por varias etapas, desde el diseño, bocetos, programación y mantenimiento del Beta.

Se considera la metodología de cascada como la más utilizada por su enfoque en la documentación, donde toma un papel primordial el equipo de análisis y diseño, debido a que es ahí donde se documentarán las mecánicas del juego. (Echeverri, 2014)

Para el desarrollo del proyecto y basándose en la metodología tipo cascada, el cual se dividió en 6 Fases:

1. Fase de Concepción
2. Fase de Diseño
3. Fase de planificación
4. Fase de Desarrollo
5. Fase de pruebas
6. Fase de presentación

Fase de Concepción: Se determina el género del videojuego y el proceso, y se constituye un guion grafico (StoryBoards). Este último se diseñó y se implementó como se muestra en la siguiente ilustración.

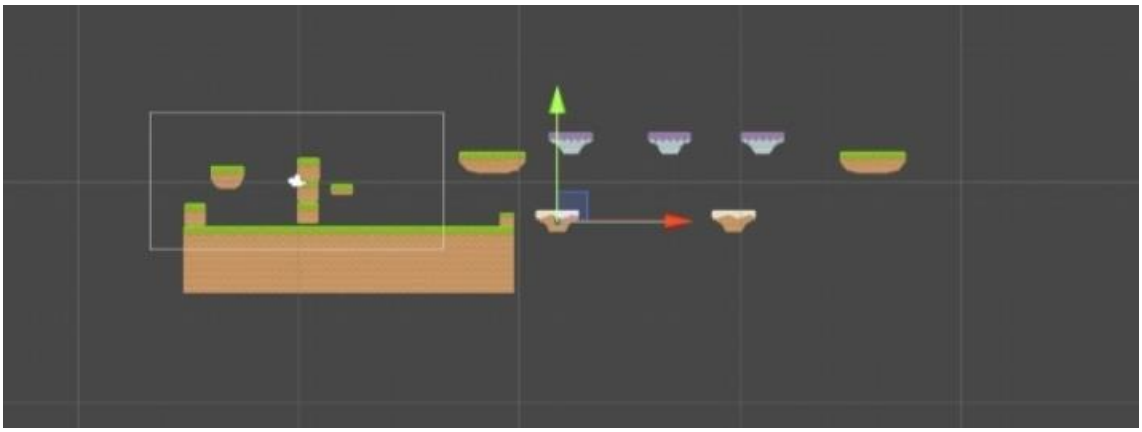


Ilustración 2 Guion gráfico

Fase de diseño: se define los elementos y objetivos que componen el juego, se crean bocetos, se desarrolla la historia, el personaje principal etc.

Fase de planificación: se crea un cronograma, para así organizar las actividades que se llevaran a cabo hasta el día final de la presentación del proyecto.

Fase de desarrollo: se empieza el desarrollo, es decir, los bocetos pasan a ser diseños y se empieza a crear el software.

Fase de prueba: al finalizar el software se revisa detalladamente y se realizan pruebas para constatar que todo fluya y funcione de acuerdo con lo esperado.

Fase de presentación: es cuando el producto ya ha sido corregido, funciona de acuerdo con lo estipulado con el cliente y cumple con todos los objetivos esperados, en este caso se presenta el juego al profesor encargado de darle visto bueno.

4.1. Fase de concepción

Se realizó una recopilación de datos de videojuegos que permiten a los niños aprender a programar, por ejemplo, human resources machine, Flappy bird, code combat, entre otros, siendo que estos videojuegos enseñan un aspecto fundamental de la programación y la lógica de esta, existen muchos otros que permiten de manera inconsciente el aprendizaje de las bases del lenguaje de programación.

Basado en la investigación de los tipos de videojuegos y didáctica de la enseñanza, se llegó a la conclusión de crear un videojuego que sea utilizado como una herramienta para el profesor en las aulas de clase, y así facilitar la enseñanza y desarrollo de la lógica de la programación.

Se optó que el videojuego sea con niveles seleccionables y que cada nivel sea o muy corto o largo, basado en lo que se va a enseñar, sea este un tema complejo como los vectores, son una estructura de datos que almacena un grupo de datos del mismo tipo y se lo define con un nombre único, es así como este nivel se convierte en el más largo, a diferencia de lo que es un bucle, secuencia de códigos que se ejecuta repetitivamente en un ciclo sin fin que, por su fácil

comprensión, es el más corto, pero sirve como herramienta al maestro para dejar en claro la idea (Cantillo, Nieves, & Pacocha, 2018).

Luego de analizar el desarrollo del videojuego, se realizó una lluvia de ideas sobre cómo sería el personaje. Al inicio el personaje se lo idealizó con características de los juegos arcade de los 80, pero esta idea se eliminó, ya que el videojuego pretende llegar a un público de no mayor a 14 años, teniendo en cuenta esto, se optó por modificar el aspecto del personaje pasando de ser un conejo humanoide a ser un robot en una estación espacial, porque sería más acorde a los objetivos que tiene el videojuego y a la temática del mismo, se eligió un diseño caricaturesco y de baja estatura por ser más llamativo y agradable para el público al que va dirigido, es decir, para llegar al resultado final se realizaron muchos bocetos.

Luego de decidir como sería el personaje se pensó en que una excelente idea sería otorgar al personaje una característica física que sea representativa, tal como en Mario Bros su signo representativo es su gorra roja y en Legend of Zelda es su espada, luego de realizar muchas pruebas del objeto que combinaría con un robot, se decidió por colocarle una bufanda que se mueva a la par de los movimientos del personaje, para darle fluidez y armonía al videojuego. Se realizó varios bocetos y muchos videos cortos de la bufanda y sus movimientos, para consolidar la idea de lo que se buscaba antes de entrar a la siguiente fase.

4.2. Fase de diseño

Con todos los bocetos obtenidos se diseñó al personaje en el cual se centra el videojuego, mejorando detalles como lo son los ojos, además de agregar orejas y de diseñar la bufanda. Al respecto, los bocetos del personaje se ilustran en los anexos 8 y 9.

Para llevar el diseño del personaje a programación se creó un diseño por cada fragmento de movimiento, de esta manera los diseños puedan ser utilizados en el motor UNITY el mismo que fue seleccionado por su excelente repertorio de juegos en 2d, un gran ejemplo es CUPHEAD. Este videojuego llamó la atención ya que su desarrollo tomo 7 años, y todo su diseño fue hecho a mano, según los

creadores del juego se utilizaron métodos de dibujo como acuarelas, oleo, lápices de colores, entre otros, y la digitalización de diseño tomó ese largo tiempo puesto que se tenía que programar junto a la digitalización de los diseños realizados a mano. Para los movimientos del personaje como: saltar, caminar, atacar o caer, entre todos se hicieron más de 30 diseños debido a que por cada frame se realiza un diseño.

Con respecto al entorno, se realizaron algunos diseños variados tanto de plataforma como de obstáculos, estos diseños fueron creados de manera aleatoria para poder tener más opciones extras por si hay que hacer cambios de último minuto. Esta característica se muestra en la ilustración 3.

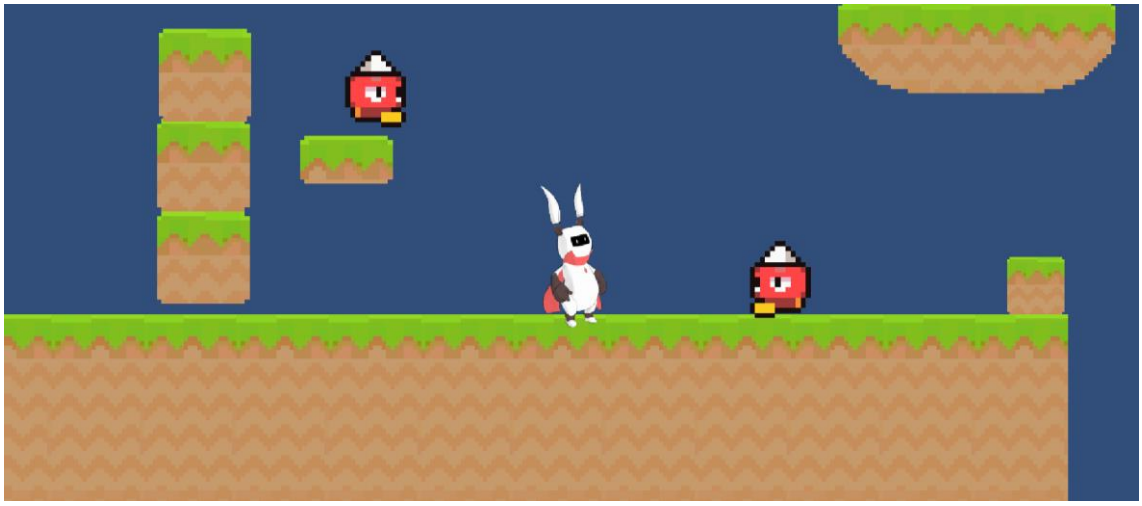


Ilustración 3 Diseño de obstáculos en el entorno del personaje

El lenguaje de programación que se utilizó es C#, debido que es el lenguaje compatible con UNITY y se desarrolló la codificación dentro del programa Visual Studio.

Basado en la información obtenida en las entrevistas con el profesor, se decidió que el nivel que se programaría para la enseñanza sería el LOOP/bucle, que es un círculo infinito de acciones, al iniciar el nivel aparecerá un sin número de enemigos de manera constante y en el momento que elimines uno aparecerá otro, estos no podrán eliminar al personaje principal, pero si causaran una molestia al intentar avanzar. Una vez que el usuario logra llega al final del nivel, será teletransportado al inicio de este, el jugador no podrá eliminar al personaje principal

intentando hacer que se caiga de alguna plataforma, ya que al momento de morir será reubicado/teletransportado al inicio del nivel. De esta manera se representa el loop/bucle ya que el nivel nunca acabaría y así podrá ser una herramienta la cual el profesor podrá utilizar para explicar lo que es el loop.

Los enemigos no son más altos ni tienen un diseño tan complejo como el personaje principal, inicialmente se pensó que los enemigos atacaran con espadas, pero esta idea se descartó y se optó por que los enemigos sean solo un obstáculo en el camino mientras el personaje principal intenta avanzar, y solo podrán ser eliminados temporalmente con un golpe del protagonista. Para el diseño de su ataque se tomaron algunos bocetos ya que este ataque tiene que cubrir un área el cual sea compatible con la programación en el motor de UNITY.

En un principio se pensaba realizar una historia escondida en el juego que se comprendería al tener todos los niveles terminados, y cada nivel tendría un mensaje y así dar una trama que resulte interesante para el niño, ya que un videojuego puede tener una trama sin que esta se mencione como tal. Pero dicha idea se omitió debido a que el software está enfocado en ser una herramienta para los docentes, y no para ser usada como videojuego de libre acceso para los niños.

4.3. Fase de planificación

En esta etapa se determinó el tiempo que se le emplearía tanto al videojuego como al desarrollo de este documento ya que ambos se tenían que realizar en conjunto para poder cumplir los requerimientos de la Universidad. Se opto por desarrollar en la primera semana las mecánicas básicas del juego, es decir, la posibilidad de caminar y saltar. Durante la segunda semana se procedió a crear más interacción del personaje con su entorno.

En la tercera semana se comenzó a realizar la parte textual de este proyecto, como: recopilar todos los datos y acciones que se han realizado durante las semanas posteriores, explicarlos y reflejarlos en el informe de la fase de producción.

En la cuarta semana se integró el diseño final del personaje principal, se implementó las acciones de los enemigos junto a la programación de este, y se

desarrolló en conjunto la interacción de ambos, para que exista dicha interacción se tiene que codificar. Estos diseños se muestran en los anexos del 10 al 13.

En la quita semana se realizaron las pruebas en el software final, se realizaron pruebas funcionales específicamente. Durante la sexta semana se presentaría el demo a la persona encargada de probar y determinar si este proyecto es apto para ser utilizado como una herramienta para que los docentes puedan enseñar a los niños la lógica de programación para posteriormente entregar la documentación del proyecto al tutor designado para son respectiva revisión.

Tabla 4: Avance del programa

Semana 1	Basando en la investigación hecha, se inició desarrollando las mecánicas básicas del personaje protagonista del video principal y el entorno que rodearía a este
Semana 2	Durante la segunda semana se codifico y configuro las interacciones que posee el personaje principal con el entorno.
Semana 3	Se registro los avances del proyecto de manera escrita y para la correcta redacción de este se procedió a realizar reuniones en las cuales se realizaron todas las consultas respectivas al Tutor designado.
Semana 4	Se integro al programa el diseño final del personaje principal y se inicio con el desarrollo del enemigo junto a sus mecánicas básicas
Semana 5	Se inicio el proceso de pruebas para confirmar la eficiencia del software.

Semana 6 y días restantes	Se presento el demo del proyecto a los docentes de la unidad educativa bilingüe Thomas More, posteriormente se envió la documentación del proyecto al Docente encargado para su respectiva revisión.

4.4. Fase de producción

4.4.1. Creación del proyecto

Lo primero que se realizo fue crear el archivo de UNITY, se seleccionó el formato 2d, luego de manera automática el programa preparara la escena y la organización de herramientas para poder crear un proyecto en 2d. Posteriormente, se importaron los diseños hechos al nuevo proyecto de UNITY, los mismos que pertenecen a el movimiento del personaje, reacciones de este, objetos, entornos, plataformas y enemigos. Dichos diseños estarían divididos en plantillas, como parte del diseño y desarrollo se crea una plantilla dedicada solo para el protagonista y todas sus acciones a realizar, otra platilla tendría todos los objetos a utilizarse en el proyecto, desde plataformas o blocs para uso de creación de objetos nuevos, junto a esta plantilla también estarán ubicados los enemigos y sus acciones.

Después de importar los diseños a Assets el cual es un espacio para Directorios, funciones y diseños, se crearía un directorio en el cual se introducirán dichas plantillas, este directorio se lo denominó Graphics. Una vez que los diseños estén dentro de este directorio, en la platilla de las plataformas se cambiaria el SPRITE MODE a múltiple para poder separar cada diseño de las platillas, y tenerlos de una manera individual para manejarlos mejor.

Posteriormente se accedería al director de Sprite para poder separar las imágenes y corregir la división que realiza UNITY a cada una de las plantillas y modificarla para que dicha separación encaje de manera exacta con los diseños hechos. Una vez separado los diseños en la zona de inspección, se modificó el

FILTER MODE (modo de filtro) de Bilinear a Point, luego de esto se permitirá desplegar los Sprites en la zona de assets y así confirmar que los diseños se separaron de la manera adecuada y estarían listos para ser implementados.

4.4.2. Primeras plataformas

Se seleccionó el diseño del cual estaría hecho la primera plataforma para pruebas con el personaje el cual solamente sería un cuadrado con el cual se crearía varias plataformas al duplicarlo y unirlo. Al introducir el cuadro con el que se iniciara a crear la plataforma base, se corregiría el tamaño de pixel por unidad a 21 para que de esta forma sea un tamaño adecuado para un videojuego ya que si fuera de otra manera seria muy pequeño y no se pudiera apreciar en una pantalla de una manera adecuada. Además, se disminuyó el tamaño de la cámara para que de esta manera no sea un inconveniente visual al momento de reproducir lo creado.

El diseño seleccionado posteriormente se le modificó el nombre a PLATAFORMA 1x1 y se crearía un nuevo directorio en el cual todos los diseños renombrados o recreados se ubicarían, este directorio tendría el nombre de PREFAPS. Una vez que se creó esta plataforma se la borraría, ya que se la creó para un uso de emergencia. Se creó una plataforma usando los sprites 123, 125, y 155, y se las unió con el nombre de PLATAFORMA 3x1 y se la agregó al directorio de PREFABS para que pueda ser una herramienta que se pueda usar en cada momento que se necesite una plataforma con una dimensión 3x1.

Para la creación del suelo se usó el SPRITE 152 y se lo duplicó constantemente y se lo agrupó hasta poder crear una nueva figura, dicho conjunto se lo agrupó y se le dió el nombre de PLATAFORMA 6x3, y se la agregó al directorio, de tal manera que si se requiere extender el suelo solo se tomaría la plataforma ya creada una y otra vez, ya que esta se encuentra guardada en el directorio como una nueva herramienta de trabajo.

Para este punto ya se tendría 3 plataformas de diferentes características físicas respectivamente, después del desarrollo del entorno, comenzó el proceso de creación del personaje y sus mecánicas básicas.

4.4.3. Personaje principal (para pruebas iniciales)

Para las acciones del personaje protagonista del videojuego, se requirió dividir la plantilla de la misma manera que se dividió la plantilla de las plataformas. Luego de realizar la división y configuración de la plantilla del personaje se procedió a configurar las acciones básicas que son:

- Animación al estar quieto
- Animación de caída
- Animación y programación de caminata
- Animación de salto
- Animación de loop/Bucle
- Ataque

Antes de cualquier tipo de configuración la prioridad era la introducción del personaje en el juego, para esto se seleccionó el primer Sprite de la plantilla, se le corrigió el tamaño variando la cantidad de pixeles por unidad y se le denomino a este Sprite con el nombre de PLAYER.

Dentro de la herramienta ANIMATION se seleccionó la opción “crear” para general un directorio dentro de “assets”, en el cual se encuentran las animaciones del personaje, este directorio tendría el nombre de ANIMATION, en el cual se designaría un nombre para cada animación.

4.4.3.1. Animación al estar quieto

El nombre que se le dio a esta animación en el directorio de animaciones fue Player_idle.

Una vez creado el nombre de la animación, se tomaron los sprites pertenecientes a la misma, los cuales se encontraban en el Directorio “Graphics” para luego ser ubicados en el orden pertenecientes en la herramienta “Animator”, se configuró la velocidad de la animación y se duplicó el ultimo Sprite para que la animación se pueda apreciar de una manera más fluida. Esta animación se muestra en la ilustración 4.

Ilustración 4 Animación del personaje en estado "quieto"

4.4.3.2. Animación de caída

El nombre que se le dió a esta animación en el directorio de animaciones fue Player_fall.

A diferencia de las demás animaciones, esta no pose varios sprites y solo se movió un Sprite a la herramienta Animator. Esta animación se muestra en la ilustración 5.

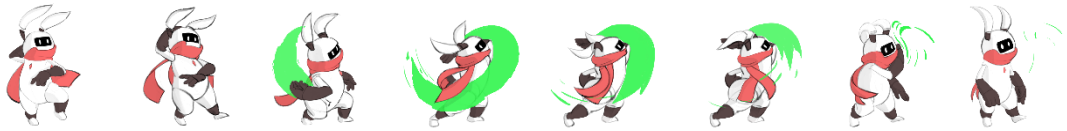


Ilustración 5 Animación del personaje en estado de "caída"

4.4.3.3. Animación de caminata

El nombre que se le dió a esta animación en el directorio de animaciones fue Player_walk.

Al igual que la animación anterior se tomaron los Sprites correspondientes a la animación caminata y se las lleva a la herramienta Animator y se configuraría la velocidad en que esta se reproduce

Desde la herramienta ANIMATOR las 3 animaciones ya creadas aparecerán, en ese momento se desarrolló la lógica, en la cual se ubicó las 3 de manera de triángulo y se las interconectó basadas en cada animación, de esta manera cada acción tiene una consecuencia lógica. Esta animación se ilustra en la ilustración 6.

Ilustración 6 Animación del personaje en estado de "caminata"

4.4.3.4 Colisiones y gravedad en el personaje

Cuando se creó las animaciones básicas, se le brindó al objeto PLAYER un Boxcollider de formato 2D para gestionar las colisiones, y un RegidBody para brindar al personaje gravedad y fuerza, siendo este de formato 2D y para que el personaje tenga un tipo de cuerpo dinámico se configuro el BoxCollider permitiendo que al personaje principal le afectara tanto la gravedad como el choque con otros objetos en su entorno, por ejemplo: las plataformas, suelos, objetos o enemigos. Para que la caída fuera más lineal y sin rotaciones se congeló la rotación en x en el rigidbody en la opción constrastraint.

4.4.3.5. Caminata, caída y estado base

Para iniciar la programación se añade al objeto PLAYER un scrip denominado playercontroler, y para usar este scrip se usó la herramienta visualStudio.

Se comenzó creando una variable publica llamada Speed con una fuera de 2f, se realizó un nuevo Void llamado FixedUpdate para evitar fallos cuando se trabaja con físicas.

Posteriormente, con un Getaxis se detecta cuando se presiona el eje horizontal, estas funciones Unity las comprende por defecto, posterior a esto se creó una variable privada de tipo Ridisbody2d a la cual se le denominó “RB1D” para que esta brinde una fuerza específica en la misma dirección que se enfoque el personaje.

Luego se la buscaría automáticamente como un componente interno dentro del start, este Ridisbody2d hace referencia al ridigbody2d que se encuentra dentro del programa que es el que se configuró con anterioridad, luego en el fixedUpdate se tomaría el rb2d y se le añadiría una fuerza dándole o utilizando un vector de 2

dimensiones apuntando hacia la derecha y este se multiplicaría por Speed, que es la velocidad más la dirección que se almacena en H.

Una vez que de determinar todo lo anterior se puso un límite de velocidad, lo cual se hizo utilizando una variable publica a la cual se la nombró como MAXSPEED y se le brindo una fuerza de prueba de 5 que será la fuerza máxima de velocidad. En el FixedUpdate se declaró una variable flotante llamada Limited speed la cual llama a MathF que es una función matemática junto con Clamp otorgando el límite mínimo y máximo de velocidad. A continuación, se procedió a actualizar los parámetros de la animación declarando una variable privada de tipo Animator con nombre “anim” y en el void start se le brindaría a “admin” las funciones del Animator de Unity de la misma manera que se hizo con el “RB2D” y delimitar la velocidad en el anim las configuramos dentro del void update.

Tocar el suelo es algo que también se tiene que determinar en la codificación, ya que con esto el personaje tendrá la posibilidad de asentarse en plataformas después de un salto o de una caída, para esto se creó una variable pública de tipo booleano llamada grounded y en el void update permite que admin tenga el valor de grounded para que pueda ser utilizado dentro del personaje.

Para configurar lo mencionado previamente, se debe solucionar un problema, cuando el personaje requiere movimiento continuo inesperadamente se queda atascado, esto sucede debido a que su colider es de forma cuadrada ,y debido a esto se impactaba con los coliders de las otras plataformas, así que para solucionar dicho inconveniente dentro de PLAYER se creó otro objeto nuevo al cual se le otorga un “CIRCLECOLIDER2D”, se lo ubicó por los pies del personaje demo y se le determino un tamaño en consideración al personaje, y el colider cuadrado se colocó a la altura de la cintura del personaje.

Una vez solucionado esto se le agregó el componente llamado “CheckGround”. Se creo este scrip que realiza lo ya hecho en el playercontroller usando un getcomponent inparent y así buscar todos los componentes que tiene Playercontroller. Se crea otro script siendo ubicado en los pies, los mismos que tendrán contacto con el piso, de tal manera este objeto valida el contacto a tierra y el padre confirma las funciones a realizar. En el mismo objeto se sobrescribió 2

métodos internos para detectar las colisiones: “ONCOLISIONSTAY2D” y “VOID ONLOCISIONEXIT2D” y dentro de cada uno se determinará cuando grounded es true y cuando es false, lo que permite determinar cuando hay una colisión o no, es decir, que cuando el personaje se encuentre cayendo ya no continúe caminando si no que desempeñe la función de caída.

Después de esto se podrá confirmar que el personaje demo puede caminar y desempeñar la animación de caminar y al detenerse desempeñara la animación de estar quieto que consiste en parpadear y moverse como si respirara o inclinará sus rodillas, depende de la percepción del que este viendo la animación.

Después de este punto a las plataformas se les otorgó un tag de nombre “GROUNDED”, para identificar que lo que está tocando el personaje con sus pies es suelo, para eso en los últimos 2 voids creados se les añadió un IF a cada uno llamando al tag GROUNDED, de esta manera se delimitó que las plataformas son suelo.

Luego de varias pruebas se pudo observar que la movilidad del personaje demo no era fluida, para solucionar ese inconveniente dentro de animator, se procedió a desactivar el EXIT TIME entre los conectores de cada animacion, el EXIT TIME daba un pequeño tiempo extra antes de iniciar la caminata, dando la ilusión de que el personaje no reaccionara al momento de darle la orden de moverse.

Al desactivar la función mencionada, el personaje caminó y desempeñó la acción que se le manda, sin tener que poseer un tiempo de espera antes de estas, dentro de las mismas opciones se realizaron algunas modificaciones en las configuraciones para que el movimiento sea más fluido y corregir pequeños errores como: que el personaje demo se resbale más de lo que debería al frenar, por lo que cada que se hacía una prueba de caminata el personaje se caía de la plataforma.

Hasta este momento el personaje Demo se puede mover de izquierda a derecha pero no gira al mandarle que vaya hacia la dirección opuesta, independientemente a que dirección vaya siempre estará mirando a la derecha, para cambiar esto el personaje tiene que cambiar el eje X de 1 a -1 cuando se le

pide que vaya a la dirección opuesta de la cual está mirando, que sería izquierda, teniendo en cuenta que H posee la dirección del movimiento se realizaron 2 if en el script con la función “transform.localscale” y se determinó la posición a la cual estaría observando el personaje demo cuando si cumpla el if. Y de esta manera el personaje en ese momento ya fue capaz de girar cuando se le da la orden que vaya a la dirección opuesta de la que se le solicito originalmente.

4.4.3.6. Salto

Para que el personaje demo pueda saltar cuando se dé la orden y a su vez realice la animación correspondiente se realizó lo siguiente: Dentro del Script de Player se agregó 2 nuevas variables, “JUMPPOWER” correspondiente a la fuerza de salto, la cual es una variable pública y la segunda variable sería de carácter privada booleana que se llamó “JUMP”.

La acción de presionar el botón para que el personaje salte se realizó en el Update y la implementación de la fuerza se realizó en el fixedupdate, la razón por la cual se separaron las acciones es debido a que el fixedupdate no funciona de la manera correcta cuando se trata de acciones dirigidas por botones, debido a esto es recomendable que este tipo de acciones como lo es el salto se realice su codificación dentro del update.

En el Void Update utilizando un IF se creó la codificación para cuando el usuario presione la flecha con dirección hacia arriba se despliegue la orden de salto, para representar la flecha hacia arriba, Unity usa el comando “UpArrow”.

En el void fixedupdate para implementar la fuerza del salto se usó un IF en el cual se añadió la fuerza de impulso, y después de terminar este IF se añadió un JUMP = false, esa falsedad se añadió para que no se repita la orden hasta que el usuario vuelva a presionar el botón designado para saltar.

Hasta ese momento el personaje relativamente ya puede saltar, más su salto se asemeja a un salto con poca gravedad, para poder solucionar esto se modificó la gravedad en los ajustes del programa, se modificó también el impulso ya que al aumentar la gravedad el impulso del personaje también fue afectado. Una vez configurada la gravedad y el impulso, el personaje demo logro ejecutar el salto de

una manera más fluida y natural. Luego se pulieron unos cuantos detalles extra como lo es el impacto contra el suelo y demás detalles pequeños relacionados con el salto del personaje demo.

4.4.4. Integración de protagonista

Con las animaciones básicas del personajes implementadas y probadas, dicha programación paso a integrarse al diseño del personaje final, el cual es el personaje que estaría en el resultado final del juego, dicho personaje final fue introducido tanto en la programación como en las animaciones básicas.

4.4.4.1. Reaparición post eliminación del personaje principal

Ahora que el personaje ya poseía la capacidad de realizar las acciones básicas, se procedió a crear un loop o bucle para su muerte, el cual consiste en que el personaje, aunque se caiga de la altura más alta o salte del mapa, este reaparezca en la sección inicial, de esa manera haciendo imposible su muerte y creando un bucle, además cuando el usuario intente terminar el nivel o avance hasta el final, el juego se reinicie en un ciclo sin fin.

Para lo mencionado, se creó un código dentro del Script del personaje. Esta línea de código se llamó “OnBecameInvisible”, en el cual especificando la posición con un vector de 3 posiciones se determinó la posición de reaparición, la reaparición ocurriría siempre que el personaje salga del área de cámara, en el momento que el personaje saliera de dicha área, automáticamente reaparecería en las coordenadas determinadas en el Vector de 3 posiciones.

4.4.5. PLATAFORMAS

Para continuar con el objetivo del nivel, se procedió a insertar y programar las plataformas que se desplegarían al vacío tras ser pisadas. El objetivo fue hacer una línea de plataformas que sean el camino para llegar a una plataforma estática, la cual sería el supuesto final del nivel, ya que el juego terminaría al pasar esta plataforma.

Las plataformas que comprenderían dicho camino se derrumbarían al momento de ser pisadas por el personaje, debido a esto, para poder brindarle una

ayuda al usuario, se decidió crear plataformas móviles en la zona inferior del mapa, de tal manera el jugador tendrá la oportunidad de evitar la hipotética muerte, estas plataformas se moverán hacia adelante y hacia atrás. Lo primero que se programa son las plataformas inferiores móviles para luego trabajar en las que se despliegan al vacío luego de ser pisadas.

4.5.1. Plataformas Móviles

Para las plataformas móviles se creó una plataforma a partir de 2 sprites, los cuales fueron ubicados en posiciones opuestas, debido a la forma de la plataforma se decidió por implementarle un “pligon colider”, el cual permitiría a dicha plataforma poder tener colisión contra otros objetos, se editó la forma del polígono con respecto a la plataforma y se creó un material al cual se le denomino “deslizante’’. Este impide que el personaje quede atorado en las esquinas, se les aplicó el material a todas las plataformas del juego para evitar futuros errores.

Debido a la ausencia de tag ’’grounded” el personaje principal no podría interactuar con esta plataforma, para este caso en especial se decidió crear un nuevo tag con el nombre de ‘’plataform’’ y se creó un código dentro del checkground para que el personaje no solo detecte suelo con el tag “grounded”, sino que también pueda asentarse cuando este toque plataformas con el tag ‘’plataform’’ , con respecto a la codificación, se reutilizo la línea de código usada para especificar el tag “grounded”.

Con la plataforma ya creada ahora se procedió a crear el movimiento que esta haría. Para esto se creó un objeto dentro del objeto de la plataforma misma, el cual fue denominado como “Target”, y dentro se creó un script que se denominó “Drawsceneline” el cual delimitaría la línea por la cual la plataforma se movería del punto A al punto B. Estos 2 puntos fueron declarados como 2 variables públicas de tipo tranform con los nombres de “from” y “to”.

Para lo mencionado se declaró un método llamado “OnDrawGizmosSelected” y usando un if se creó la condición de si los valores “start" y “from” no son nulos se ordenó hacer una línea entre estos 2 puntos, ya dentro del programa se declaró que el “from” sería la plataforma móvil y el TO sería

el objeto target, y dentro del objeto target se repitió el script ya usado para que se reflejara la línea de manera apropiada. Dentro del mismo código se definió que se dibujará un círculo en las puntas de los extremos de cada línea para así estas sean más fáciles de monitorear. Dentro de Unity se realizaron configuraciones para que las esferas previamente hechas se transformaran en iconos que resalte que punto pertenece a “from” y a “to”, en este caso cual sería la plataforma y el target.

Después de haber definido todo el punto se procedió a crear el movimiento de la plataforma. Dentro del objeto “plataforma móvil” se creó otro script con el mismo nombre del objeto. En esta se declaró 2 variables públicas, una con el nombre de “Speed” el cual representaría la velocidad de movimiento que tendría la plataforma, y otra con el nombre de “Target” al cual se le asignara el objeto “target”, para esto se creó el método “fixed update” y con un IF se desarrolló una condicional para comprobar si el target es diferente de null, y se utilizaría la función de unity “Movetoward” la cual nos permite movernos si le brindamos la información que nos solicita. que son el punto de inicio y el punto final más la velocidad.

Hasta este punto la plataforma se movería, pero esta no retornaría a su punto inicial, para que esta retorne se integró a la codificación un par de variables privadas tipo vector 3 para almacenar las posiciones, las cuales serían “Start” y “End”. Dentro del primer IF que se encuentra en el Método Start, se agregaron las variables antes mencionadas y se delimitaron sus posiciones las cuales pertenecerían a: “Start” la posición de la plataforma y “end” a el destino que se determinó con “Target” y para confirmar que la plataforma retorne cuando llega a su objetivo se agregó un nuevo IF dentro del método FixedUpdate en la cual condiciona que si la posición de la plataforma es la misma que la del Target entonces el “target” pasaría a convertirse en “Start”, de tal manera el “Start” original seria ahora “target”, y las acciones ya codificadas se repetirían constantemente , lo cual haría que la plataforma se mueva constantemente de ida y vuelta.

4.5.2. Velocidad compartida entre protagonista y plataforma

Con la plataforma móvil ya creada, se desarrolló un inconveniente, el cual era que el personaje no se movía de la misma forma que la plataforma, presentando problemas. Esto se podía visualizar a lo que el personaje saltaba a la plataforma móvil, y este no se movía con la plataforma, es decir si la plataforma esta de manera vertical el personaje pensaba que está cayendo, pero no es así, el simplemente debía tomar la misma orientación que la plataforma, y seguir con el juego, lo mismo pasaba si la plataforma se movía de manera horizontal, ya que el personaje tendría que caminar para mantenerse sobre esta, por ende, al no viajar a su misma velocidad, este se queda estático hasta caer al vacío, en donde se procede a que el personaje vuelva a iniciar, hasta que el pueda completar la prueba.

Para corregir estos errores se agregaron líneas de código dentro del script “check ground”, donde se encuentra la programación del personaje, dentro de “check ground”, en los If que condicionan la colisión con el Tag “Platform”, se agregó que si ocurre una colisión el personaje pasaría a ser hijo de objeto plataforma, con esto al jugador convertirse en hijo de la plataforma este viajaría a la misma velocidad y de la misma manera, que esta, también se agregó que cuando ese salte automáticamente deje de pertenecerle a la plataforma y vuelva a ser el personaje un objeto independiente, para qué pueda continuar con su trabajo.

Al diagnosticar algunos errores se pudo descubrir un nuevo inconveniente, donde el personaje brinca de una altura un poco más elevada que la plataforma que se mueve de manera vertical , el personaje a pesar de ya haber aterrizado en la plataforma, sigue desarrollando la animación de caída como si fuera un bucle, para poder corregir esto se declaró en el scrip “check ground” una variable privada rigidboddy2d con el nombre de “rb2d” y se utilizó un método llamado “OnCollisionEnter2d” y en este se copió el if con el cual se tenía la condición de colisión con la plataforma y se declaró que la velocidad sería igual a un vector de 3 dimensiones vacío, de tal manera se ordenó que cuando el personaje toque una plataforma su velocidad siempre será 0, con respecto a la plataforma en la que va el personaje, de esta manera el personaje no tenderá a querer moverse o salirse de la plataforma, sino que se encontrará estático en ella.

Con estos cambios, el personaje al estar en plataformas móviles, tanto verticales como horizontales, viajaría a la velocidad de la plataforma, y se podrá mover sobre estas sin problema alguno.

4.5.3. Plataforma que se despliega al vacío

Culminado el proceso de la plataforma móvil, se prosiguió a crear la plataforma que se despliega al vacío luego de ser tocada por el personaje principal, para la creación de esta plataforma se copió la plataforma móvil ya hecha y se ubicó dicha copia arriba de la plataforma móvil, en la copia se eliminó el objeto “target” que se encontraba dentro del objeto “plataforma móvil” copiado y también se eliminó los scripts que tenía, se los retiro dándoles clic derecho en la esquina superior derecha de cada uno y seleccionando la opción remove component, y se le cambio de nombre a plataforma “Falling”.

Para poder diferenciarla se le cambio los sprites a unos que poseen la misma forma y los mismos detalles, pero con un color distinto al original, después de esto se arrastró el objeto a la carpeta prefabs para de esa manera solo tener que tomarlo de esa carpeta cada vez que queramos una nueva plataforma de este tipo. A esta plataforma se le dio un RigidBody 2d y se le designó que fuera de tipo kinetico, para que se mantenga estático en el aire, esto sería solo de manera temporal, ya que con la programación que se realizaría continuación esto cambiaria, para que de esta manera la plataforma caiga al ser pisada o tocada.

Posteriormente se le otorgó un nuevo script al que se denominó “PlataformaFalling”, en el cual se declaró una variable privada RigidBody2d con nombre “rb2d” y en el inicio se declaró que “rbd2” pertenece al rigidboddy2d tal como lo hemos hecho en todos los scrips hasta ahora. Se declaró un método OncollisionEnter2d llamado “COL” el cual hace referencia al objeto contra el cual se está colisionando. Dentro de este se creó una condición usando un if y una función de Unity llamada “Comparetag”, esta condición tendría la función que cuando el objeto con tag “player” toque dicha plataforma esta cambie su RigidBoddy y deje de ser tipo kinematic y en consecuencia a esto, la plataforma se caería debido a que fue tocada por el jugador.

4.5.4. Colisión y rotación de la plataforma al caer

Cuando esta plataforma empieza a caer, esta colisionaba con los demás elementos del juego, también le rotación era afectada, lo que significa que rota basado en el sector que se le aplique fuerza. Para corregir la rotación se procedió a crear una variable privada PolygonCollider2d con el nombre de “pc2d” y en el último if que se creó se agregó y se añadió la condición que si el personaje toca la plataforma el poligoncooiden2d de la plataforma se convertiría en triguer, lo cual permitiría que la plataforma caiga sin interactuar con ningún otro elemento y cancele la rotación previamente mencionada.

Además, se le cambió el tag a la plataforma por el tag “grounded” , debido a que si esto no se hacía, la condición que tiene el personaje de convertirse en hijo cuando toca la plataforma nunca desaparecería, ya que la plataforma desaparece al caer junto con el personaje y al reaparecer el personaje, aparecería y reaparecería dando la ilusión que está cayendo.

Luego de lograr cumplir con el objetivo de dicha plataforma, se le aplicó a esta la capacidad de brindarle al personaje un poco de tiempo para salta antes de que esta se despliegue al vacío, para poder realizar esta nueva característica se implementó una variable publica tipo flotante llamada “FallDelay” y se le brindo una fuerza de 1f , se tomaron las condiciones del if con respecto a “rb2d” y “pc2s” y se ubicaron en un nuevo método que se le denomino “Fall”.

Dentro del if previamente mencionado, se realizó un “Invoke” el cual llamó al método “Fall” , el invoke permite determinar cuándo queremos que algo ocurra, de esta manera cuando el personaje salte a este tipo de plataformas , exista un lapso de tiempo para poder saltar a otra antes que dicha plataforma se despliegue al vacío, esto se realizó con el fin de que el usuario, desarrolle la habilidad para realizar un novimiento o tomar una decisión de forma rápida, en donde se busca que esta sea la mejor, para continuar en el juego, y que este no vuelva al inicio.

4.4.6. Cámara

Ya con todas estas características se procedió a insertar una de las funciones más importantes de un videojuego, la cual sería la cámara en la cual se realizará todo el juego, ya que hasta ahora solamente se había definido los límites de esta y los movimientos que se pueden realizar en este entorno, más si el personaje se mueve o intenta avanzar más allá del tamaño dispuesto para la cámara, este saldría de los límites del juego y seria teletransportado al inicio. En esta parte del proceso de desarrollo se integró la capacidad de que la cámara siga al personaje por todo el transcurso del juego, hasta donde se determine que serían los límites del videojuego.

Para logar conseguir que la cámara siga al personaje se creó un script dentro del objeto Main Camera el cual fue denominado “CameraFollow”, en este se declaró un GameObject con el nombre de “Follow”, dentro del programa Unity se seleccionó a “player” dentro del campo “Follow “en el objeto “MainCamara”, este campo se activó al crear la Variable publica “Follow”.

Dentro del script se alteró el update para convertirlo en un FixedUpdate, dentro se creó una variable flotante con el nombre “posX” en el cual se le asignó la posición del personaje, más solo en el eje de las x, debajo de este se repitió el mismo proceso, pero en este caso se declaró una variable flotante, pero para la posición en el eje Y con el nombre “posY”, posteriormente se cambió la posición de la cámara.

Con todo definido, se cambió la posición de la cámara haciendo referencia a un nuevo vector de 3 dimensiones al que se le brindo las posiciones de “posX” y “posY” y se mantuvo la posición z igual. A partir de este punto la cámara ya cumplió su función de seguir al personaje mientras este ejecuta sus movimientos, para alcanzar a superar todos los obstáculos que se le imponen, por ende, se optó con perfeccionar un poco más la cámara de movimientos, de tal manera que el juego sea más llamativo para el estudiante, y sienta la necesidad de querer volver a jugar.

Para mejorar el movimiento de la cámara se le dio limitaciones a esta, tales como: hasta donde o desde donde esta puede moverse con el personaje, con el fin

de que tenga las limitaciones que se le han realizado al persona, y pueda interactuar de forma correcta con él, es decir al momento que el personaje decida moverse hacia arriba o hacia abajo, la cámara no se desplace innecesariamente o muy hacia arriba o muy hacia abajo, ya que esto no permitiría apreciar de forma correcta el juego, o simplemente le cause una molestia al usuario, ya que podría causar la perdida de la perspectiva al jugador de lo que se encuentra al lado opuesto de su movimiento, y lo haría de cierta forma perder el avance que ya ha ganado.

Para realizar las mejoras mencionadas se procedió a realizar ciertas modificaciones del script, donde se crearon 2 vectores públicos con los nombres “minCamPos “y “maxCamPos”, haciendo referencia a la posición mínima y máxima que tendría la cámara al moverse. Dentro del tranform.position con función de movimiento de cámara se retabuló la información y de las variables “posX” y “posY” se utilizó la función clam para delimitar así las capacidades de movimiento de la cámara, tanto movimiento hacía arriba como hacia abajo.

Dentro de Unity en las configuraciones de cámara se activó la opción de determinar las posiciones máximas y mínimas de la Cámara, el personaje al saltar y moverse tendría un límite, y debido a esto no se pierde la perspectiva del juego y se aprecia de mejor manera el entorno, ya que ahora al saltar la cámara ya no sube de manera innecesaria haciendo perder al usuario la posibilidad de ver el suelo.

A partir de este momento la Cámara ya posee límites y restricciones, pero se le brindó un efecto de suavizado de tal manera que el movimiento de la misma no sea tan bruscos para el espectador, para ello se creó una variable pública de tipo flotante con el nombre de “smothtime”, haciendo referencia a los segundos de retardo o suavizados que hay antes que la Cámara se mueva con el personaje, para que la misma pueda moverse hacia una dirección, y gestionar la velocidad, se creó una variable interna gestionada como un vector de 2 dimensiones privada llamada “velocity” y dentro de “posX” y “posY” con un “mathf” se llamó a la función “Smothdamp” en la cual sus campos fueron llenados con la posición , la velocidad y el “smothtime” respectivamente.

Por ello, en unity ya se podría determinar el “smothtime” directamente llenando el campo dentro de la Main cam, se le designo un retardo de 0.15

segundos, lo cual permito que la Cámara se mueva con más fluidez, de una manera más natural y menos forzada.

4.4.6.1. Movimiento de Cámara

En este punto de la etapa de desarrollo, se decidió añadirle una mecánica extra al personaje , ya que debido a las contantes plataformas y a sus movimientos, puede existir la posibilidad de que el jugador no alcance a llegar a una plataforma en específico, o pueda tener ciertos problemas y caer, debido a que no presionó la barra espaciadora en el momento exacto, así que se optó por brindarle al personaje la capacidad de dar un salto doble, el cual se había mencionado con anterioridad, que sería el poder dar un salto en el aire luego de haber dado ya dado uno, para esto se declaró una variable booleana en el script del personaje, esta variable tendría el nombre de “dobleJump”, dentro del if que confirma el salto normal, debajo de la confirmación del “Jump”.

Luego se validó el segundo salto con un doblejump= true, estas 2 afirmaciones irían dentro de un nuevo if que condicionaría si “grounded” es verdadero, y posteriormente se creó un else if que condiciona el “doblejump” que da como resultado un “Jump” positivo y un “doublejump” negativo, con el fin de que el personaje no pueda realizar un tercer salto adicional a los que ya se dio, hasta este punto el personaje ya logro lo que es salto doble, debido a que al saltar este puede llegar a usar la misma fuerza del primer salto, y tendería a saltar una altura demasiada extensa, se optó por limitar dicha fuerza, para esto solamente se cambió la potencia de salto, mas no se duplique la fuerza que ya se había usado para el primer salto , que corresponde al “Jump Power”.

4.4.7. Salto de precaución

Para esta acción se procedió a realizar algunas pruebas de forma minuciosas, para comprobar que el personaje podía realizar saltos de precaución en diferentes situaciones, en donde sean necesarias, uno de estos saltos podría ser cual el personaje se este cayendo, a causa de no haberse detenido a tiempo, usualmente esto pasa en muchos juegos, en donde la persona aplasta mucho tiempo la tecla para correr, o simplemente se queda atascada por un momento

dicha tecla, por consecuente el personaje se puede caer de una plataforma o del suelo como tal.

Para ello se realizó el salto de precaución, en donde solo podrá ser usado cuando el personaje realice las caídas antes mencionadas, en donde se le permitirá regresar a la plataforma sin perder el avance del juego, caso contrario no se podrá realizar dicha acción, y tendrá que regresar a su posición inicial.

Para resolver esto se procedió a realizar a crear un if que condicione que, si “grouden” es verdadero, este personaje podrá realizar un doble salto, mas no un salto normal, de esta manera tendría un salto extra almacenado dentro de la memoria, el cual solo puede ser usado en casos necesarios y óptimos para la continuación del juego. El personaje al momento de estar en una caída que no fue causada por un salto podrá acudir a dicho salto de precaución, de esa manea poder llegar a su objetivo o retornar su avance en el juego.

4.4.8. Enemigos

Después de haber definido todas las actividades o acciones básicas del personaje, se procedió a la creación de los enemigos, los mismos que no serían capaces de lastimar al personaje principal debido al objetivo del nivel, pero si le ocasionarán una molestia durante su movimiento, con el objetivo de que el niño puede realizar una estrategia o buscar una solución ante esa molestia que es proporcionada por el juego como un obstáculo adicional.

Para la creación de dichos enemigos se tuvo que importar los sprites, para luego crear un nuevo objeto con el nombre de “Enemy”, este objeto tiene como imagen un Sprite del enemigo, el enemigo en estado base para ser exacto, a dicho objeto nuevo se le dió un “circle collider” debido a su forma y este collider se editó de tal manera que tome la forma del enemigo, cabe recalcar que el enemigo como tal es circular, así que el collider se adaptó de una forma más natural sin necesidad de numerosas modificaciones.

Adicional a lo dicho anteriormente se tiene que para que el enemigo pueda interactuar con su entorno se le dió un rigidbody 2d y se congelo la rotación en el eje z ,de esta manera si es golpeado o empujado, este no se girará en su mismo

eje, al igual que al personaje principal, al enemigo se le creo un script con el nombre de “EnemyController”, debido a ciertas similitudes se optó por copiar las variables de “speed” y “maxspeed” del “PlayerController” y pegarlas en el “EnemyCOntroller” y a ambas se le brindo una fuerza de 1f.

Luego se declaró un RigidBody2d el cual se recupera dentro del start, a diferencia del personaje principal, al enemigo no se le daría órdenes para que se mueva, sino que se movería de manera independiente, para esto se copió el “adforce” del “Playercontroller” y se lo pego en el “enemycontroller” pero se omitió la H, ya que en este caso no sería necesaria, y después de esto se copió la línea de código que representa la función de “limitedSpeed” , todo fue ubicado en el fixedupdate.

Cuando se culminó de desarrollar las características y funcionas básicas , se debía que integrar la capacidad que el enemigo se volteara cada vez que impactara algo, este impacto ocasionaría que la velocidad el personaje disminuya a 0 , por lo tanto la condición seria que siempre que el personaje baje su velocidad a 0, cambie su dirección y se encamine a la dirección opuesta, para esto se realizó un If dentro del fixed update en el cual condicionaba que si la velocidad era mayor a -0.01f y menor a 0.01f , por consecuencia “speed” sería igual a “-speed”.

Posteriormente se le indico al “rb2d” cuál es la nueva velocidad copiando el código de rb2d.velocity más se remplazó el limitedvelocity por Speed, se optó por la velocidad 0.01 y -0.01 ya que esto permite que el enemigo retorne de una manera más fluida, en cambio si la condición hubiese sido directamente con 0, tendría que llegar a una estática total para poder retomar su dirección, y esto no es tan llamativo a la vista, ya que el enemigo tendría que quedarse inmóvil por unos segundos antes de redireccionarse.

4.4.8.1. Caminata del enemigo

Con la realización del movimiento, también se procedió a crear la nueva animación del Enemigo, la cual sería la animación de caminata, esta se llamó Enemy_Walk. Se procedió a escoger los sprites correspondientes a la animación de caminata y ubicarlos en la línea de animación, como esta es la única animación

que se reproduce de manera continua no hay que delimitarle condiciones o permisos ya que el enemigo siempre estará caminando, en este punto el enemigo camina, colisiona, disminuye su velocidad casi a 0, retorna en dirección opuesta.

Sin embargo, presentaba un pequeño problema, el cual era que al girar, este no lo hacía por completo, por ende, daba la ilusión de que caminaba de espaldas, lo cual era algo que no deberá realizar, para corregir este detalle, se copió las líneas de código de PLayerControler que hacen referencia la transformación del LocalScale los cuales corresponden a 2 if respectivamente y se remplazó la H por el speed y se alteraron los signos < y> por > y< y se designó que el valor que hagan referencia cada uno sea a 0 , de esta manera el personaje ya puede caminar y retornar como lo hacía inicialmente, si no que procederá a realizar la acción correcta.

4.4.8.2. Eliminación del enemigo

Para la siguiente etapa de desarrollo del enemigo, como en todo juego se requiere eliminar de alguna u otra forma los obstáculos que se tiene, y en este caso es el enemigo, por lo cual se decidió realizar una se configuración en la condición con respecto a su eliminación, para esto, no solo había que programar la animación de eliminación del enemigo, sino también el ataque del protagonista, y que dicho ataque haga efecto sobre el enemigo.

Antes de proseguir con la programación del enemigo, se optó por configurar y programar el ataque del protagonista primero, ya que sin esta la animación de muerte del enemigo seria innecesaria, para eso se creó una nueva a animación llamada player_attack, en la cual se insertó y configuró los sprites correspondientes a un ataque del personaje y se creó dentro de animator la condición de Attack y esta se determinó que fuese de tipo trigger, posteriormente se creó una transición desde el Anystate al player_attack con la condición Attack en dicha transición, hasta este momento se creó la entrada de la animación al personaje, mas no la salida.

Para que haya una salida por así decirlo se hicieron 2 transiciones, una a player_idle y otra a player_walk, en la cual la condición para player_idle es que el speed sea menor a 0.1 y para player_walk sea mayor a 0.1, con esto la animación

de ataque no se repetirá en un bucle y el personaje podrá atacar independientemente el estado en el que se encuentre, podrá estar caminando o estar sin movimiento igual podrá atacar sin que esta acción se repita infinitamente sin control alguno.

Con todo determinado y configurado, dentro del scrip playerController dentro del Update se hizo la condición que al presionar la barra espaciadora el personaje tendría que realizar el ataque, además se configuraron los exit time para que la animación se vea más fluida y detallada.

Dentro del script se definió un objeto llamado animatorstateinfo para almacenar el estado del animador, el stateinfo lo que hace es comprobar si se está realizando una animación en específico y la designamos a una variable booleana, entonces se comprueba que si se está atacando se activaría dicha variable boooleana con el nombre attacking, luego dentro del if se añadió un && más !attaking, para que de esta manera solo se pueda atacar cuando no se encuentre realizando dicha acción, esto más que nada fue para que haya un mejor orden dentro de las funcionalidades del videojuego.

Dentro del objeto player se creó un nuevo objeto llamado Attack al cual se le dió un polligoncolider2d y este collider se modificó para que cubra el área la cual el ataque se encargaría de cubrir, dentro del player en su scrip se recuper al polligoncollider y se le brindo el nombre de nombre de attackcollider , para este punto lo que falta es que la animación solamente cumpla su función cuando esta se encuentre desarrollando , y se resalta que la función de es eliminar a los enemigos, así que se tendría que delimitar que la animación de ataque solamente pueda eliminar al enemigo cuando se encuentre a la mitad de su ejecución, debido a que es cuando el puño del personaje alcanza su punto más largo, dentro del script desactivamos el attackingcollider en el Start para que se encuentre apagado siempre, y para que funcione y resalte la función que se mencionó, se tuvo que añadir un IF en la parte más baja de Update en la cual se comprueba si se está atacando con la variable booleana attacking.

Con respecto a lo anterior, dentro se determinó una variable de tipo flotante con el nombre de playbacktime, la cual se encargaría de almacenar el tiempo que

lleva la animación realizándose y se almacenaría la variable del stateinfo nomalizedtime, la cual va guardando el tiempo de duración de la animación y dentro del mismo if se creó otro if en el cual se utilizó el playbacktime para condicionar el momento en el cual la animación se encuentra en el punto que se necesita, sería igual a TRUE caso contrario con un else sería false.

Ahora se procedió a integrar la muerte de los enemigos, con esto se insertó los sprites correspondientes y por consiguiente se hicieron todas las configuraciones para que dicha animación corra de manera fluida y atractiva a la vista , las animación se la denomino como Enemy_Destroy y se le desactivaría el loop, al personaje se le creo un nuevo scrip llamado destroyable, en el cual se determinaron 2 variables, una fue destroystate y la segunda fue timefordisable , una se encargaría de guardar y bajo condición If reproducir la animacion de destrucción cuando haya la colisión correspondiente, y la segunda se encarga de eliminar la posibilidad de colisión, cuando los enemigos son derrotados estos se convierten una nube de humo pequeña, dicha variable determinaría el momento en el cual el personaje principal podría atravesar la nube de humo dejada por el enemigo destruido.

Hasta este punto del juego ya se tenía todo lo necesario para darlo por finalizado, más faltaba un pequeño detalle, el cual era la reaparición de los enemigos, se intentó crear un método pero se optó por crear algo conocido coloquialmente en el mundo de los videojuegos como Respawner, el cual es un objeto que se encargaría de crear un enemigo siempre que el enemigo creado originalmente fuese destruido, habría 3 Respawners en el inicio del mapa en posiciones diferentes, para que de esta manera no haya una acumulación de enemigos en un mismo lugar reapareciendo.

En congruencia con lo anterior, se tomó al enemigo ya creado y se lo almacenó en prefabs y se lo eliminó del juego, debido a esto, tendríamos al enemigo en una carpeta ya configurado, solo se tendría que arrastrarlo, se hizo esto para poder tener un área más limpia con la cual trabajar para la creación del Respawner, ya que con el enemigo dentro del nivel se podría llegar a crear un poco de confusión cuando se configura.

Respecto a lo anterior, la creación de este fue más sencilla de lo que alguien podría tener idea, ya que se basó únicamente en 2 gameobjects con el nombre de prefabenemigo y _enemigo, los cuales uno haría referencia al prefabs del enemigo que se almaceno y el otro se encargaría a hacer referencia al enemigo en cuestión, dentro del Update se creó un IF el cual designaría que si _enemigo es igual a Null dicho _enemigo sería igual a una instancia de prefabenemigo como un game object y luego aparecería en la posición en la que se encuentra el objeto en el mapa.

A partir de este punto se dió el paso final que fue extraer el juego del programa, ya que a partir de este momento el juego ya estaba culminado, de tal manera que el jugador al iniciar este nivel entraría a un bucle infinito en el cual no podría morir ni terminar el nivel, ya que regresaría al punto de partida y los enemigos al ser eliminados estos siempre reaparecerán de manera infinita.

4.4.9. Creación de menú

En cuanto al desarrollo del menú, para esto se creó una carpeta, esto significaría que el juego tendría 2 escenas; una sería el nivel y la otra el menú. Dentro de las configuraciones de Unity se designó que el menú sea el que se presente primero. Con las herramientas brindadas por Unity se crearon botones y en líneas de código se configuró una función, la cual permitía bajar el brillo. Esto último se logró gracias a que se ubicó una Canvas de color negro encima de todo y este se desvanecería al gusto del usuario. Por último, para el menú se designaron 3 botones, de los cuales el primero era el único funcional, ya que a ese botón se le codificó la función de acceder a la escena del nivel de bucle.

4.5. Fase de pruebas

4.5.1. Pruebas funcionales

Uno de los tipos de pruebas que se le realizo al Software son las funcionales, estas pruebas se realizan para comprobar la funcionalidad y la usabilidad de un software. (Lee, 2020)

Las pruebas funcionales que se desarrollaron para este proyecto fueron pruebas unitarias y de integración, permitiendo mejorar el programa de forma coherente.

4.5.2. Pruebas unitarias

Este tipo de prueba se centra en una pieza o parte específica del software, esta puede ser módulos, procedimientos o funciones, cualquier parte del software puede someterse a dicha prueba. (Lee, 2020)

Para esta prueba se revisó de manera individual los siguientes aspectos: funciones básicas del personaje, funcionalidad de las plataformas, funcionalidad de enemigos.

4.5.3. Funciones básicas del personaje

Para esta prueba se revisó de manera individual las siguientes funcionalidades del personaje:

- Movimiento
- Mecánica de salto
- Acción de golpe
- Reaparición post desaparición

4.5.4. Movilidad de enemigos

A base de observación se confirmó el movimiento independiente y correcto que posee el enemigo.

4.5.5. Pruebas de integración

El objetivo de este tipo de prueba es validar la integración de 2 componentes para realizar o completar una acción. En esta etapa se realizaron pruebas a los elementos que para cumplir una acción determinada necesita una interacción de otro elemento y de esta manera trabajar en conjunto. (Lee, 2020)

4.5.6. Funcionalidad de las Plataformas

Dentro del nivel hay 2 tipos de plataforma, las móviles, y las que se despliegan al vacío al ser tocadas. Para esta prueba confirmo la interacción que tiene el personaje con estas.

4.5.7. Interacción enemigo y protagonista

Para esta prueba se confirmó 3 aspectos en específicos que para que estos ocurran se requiere la interacción de los objetos “Player” y “Enemy” correspondientes al protagonista y enemigo, dichos aspectos son los siguientes:

- Efectividad de ataque del protagonista
- Animación de desaparición
- Reaparición de enemigo luego ser eliminado

4.5.8. Pruebas de menú

En específico, para esta prueba se procedió a confirmar el enlace entre la escena del menú con la del nivel de bucle

4.6. Fase de presentación

La presentación final del proyecto se realizó el día 8 de junio del 2021, en dicha presentación el proyecto fue revisado por 3 integrantes de la Unidad educativa Thomas More: Msc. Erica Lainez Rectora de la Institución, Msc. Willian Wadamud Docente encargada de impartir la asignatura de programación/computación y Msc. María Auxiliadora Campos Vicerrectora. En donde cada docente procedió a realizar una evaluación de manera individual al juego, de tal manera de que puedan analizar a detalle cada nivel del juego, y sus obstáculos.

Durante la presentación del proyecto se expuso las ideas de mejora para los demás niveles que podría llegar tener el juego y como sería cada uno de ellos. Para esto se procedió a enseñar los posibles diseños que podría llegar a tener el juego.

Adicional a lo mencionado se explicó que lo temas que representaban cada uno de los diseños fueron propuestos a base de condiciones y variables respectivamente, como se diseñó el juego principal.

La presentación del proyecto tuvo una duración de 25 a 30 minutos, debido a que cada uno de los docentes revisó y analizó el programa de manera individual.

Luego de evaluar la presentación final del proyecto, los docentes llegaron a la conclusión que el proyecto es una herramienta viable que si puede ser usar para impartir diferentes conocimientos de lógica programable a los alumnos de 8vo y dieron su aprobación firmada, las cuales pueden ser encontradas en Anexos.

CONCLUSIONES

El objetivo general de esta tesis es cumplir los requerimientos para que puedan servir como herramienta para el docente en la enseñanza de la lógica de la programación a niños de 8vo año. Gracias a toda la información recaudada por medio de grupos focales, entrevistas e investigaciones, se logró desarrollar el videojuego con un solo nivel; cuya utilidad consiste en la demostración y enseñanza de lo que es el Bucle, el cual fue escogido por el docente.

Además, en respuesta a los criterios definidos a partir de las entrevistas y grupos focales, se creó un entorno amigable para el usuario, de fácil acceso e intuitivo. A su vez, se creó un personaje capaz de moverse alrededor del entorno creado, permitiendo al usuario visualizar todas las cosas de su ambiente y, de esta manera, generar una experiencia de aprendizaje creativa.

Por otro lado, en congruencia con los objetivos específicos de este trabajo, el videojuego fue presentado en la Unidad Educativa a varios docentes expertos en el área y se llegó a la conclusión de que correspondía a una herramienta interactiva, la cual podía llegar a ser beneficiosa para los estudiantes. Esto debido a que, desde la perspectiva del personal docente, facilita la comprensión por parte de los estudiantes en lo que respecta a temas complicados de la programación. De esta manera, se evidenció el cumplimiento de las expectativas de las partes interesadas.

Por último, en el presente trabajo también se logró ejecutar una revisión teórica sobre la didáctica de la enseñanza de la lógica de programación, los usos educativos de los videojuegos y sus aplicaciones en trabajos relacionados; de la cual se concluyó que los videojuegos corresponden a una novedosa herramienta de aprendizaje que cada vez más se va integrando a la educación como soporte al desarrollo de habilidades cognitivas, emocionales y sociales. Finalmente, dicha revisión guio la ejecución de las fases de creación del videojuego.

RECOMENDACIONES

Antes de culminar, estas son algunas recomendaciones basadas en los resultados que se obtuvieron del presente estudio:

Como primer punto se recomienda extender la investigación con respecto a las didácticas de la enseñanza de la lógica de programación y las ciencias exactas en general, ya que de esta manera se podrá obtener una visión completa sobre cómo hacer que el videojuego personalice sus características, a fin de que el estudiante pueda comprender satisfactoriamente los temas asociados a esas áreas.

De forma análoga, sería adecuado ampliar la muestra de participantes en los grupos focales y las entrevistas, así como la frecuencia de estas. Esto con el objetivo de mejorar la experiencia del usuario a través de las percepciones, sentimientos y opiniones profundas captadas por estos instrumentos de investigación. Es decir, una muestra mayor podría brindar descripciones más específicas de las necesidades de los estudiantes, las cuales aportarían a la mejora del videojuego.

Adicionalmente, para futuros trabajos y para quienes deseen replicar o mejorar el desarrollo, de tal manera se emplea las siguientes recomendaciones:

- Mantener separados los sprites de los objetos que se vayan a incluir en el videojuego, ya que esto permitirá trabajar de una manera más organizada.
- No juntar la programación de objetos móviles con el entorno principal del juego, ya que esto puede causar problemas en la codificación de ambos.
- No exportar el trabajo a otro motor, ya que el proyecto está completamente con el lenguaje C# y el cambio a otro motor causaría un colapso debido al cambio de lenguaje.
- Evitar usar coliders cuadrados, ya que estos causaran problemas cuando el personaje interactúe con cualquier tipo de objeto

rectangular o con puntas, optar por usar coliders circulares o con forma de polígono, ya que esto disminuyen la posibilidad que el personaje tenga algún percance cuando interactúe con otros objetos.

- Revisar minuciosamente todas las funciones del personaje, con el fin de que al jugarlo los niños no presenten problemas por el estancamiento o por un bucle infinito por el que esté pasando el personaje.
- Reciclar los códigos de las plataformas en el caso que se dese diseñar nuevos modelos, de esta manera los nuevos modelos ya tendrían el código necesario para realizar las funciones de las antiguas plataformas.
- No unir la codificación que posee el personaje para la interacción con el suelo, con la que pose para el resto de las acciones. Unir estas 2 solo complicaría más la codificación y afectaría directamente la interacción con los objetos que dicho personaje toque o colisione debido a que el colíder que usa para caminar seria el mismo que tendría para las colisiones que pueden ocurrir contra su cabeza
- Tener muy en cuenta las medidas, ya que 1cm puede hacer la diferencia en el proceso de animación y puede afectar de manera directa como se vea la fluidez en el juego
- Reutilizar la programación de los enemigos, el código creado para que los enemigos reaparezcan tras ser eliminados es algo complicado, la mejor opción es reciclar dicho código para cualquier otro enemigo que se dese poner, en código se puede delimitar el tiempo que toma un enemigo en reaparecer, y esa característica puede llegar a ser muy útil en niveles futuros.
- Evitar crear relaciones innecesarias en las animaciones tanto del personaje o de cualquier objeto, ya que las animaciones van de la mano con codificación, y cuando un numero alto de relaciones entre ellas, existe una alta probabilidad de confusión y dicha confusión puede ser el detonante que errores de codificación.

- Definir los objetos con nombres totalmente diferentes es una recomendación para tener en cuenta a la hora de programar, ya que estos pueden crear confusión y colocar una línea de código con el nombre de un objeto equivocado debido a que dichos objetos poseen nombres similares, desencadenara un error con una dificultad de ser hallado alta.

BIBLIOGRAFÍA

Álvarez-Álvarez, C., & Vejo-Sainz, R. (2017). ¿Cómo se sitúan las escuelas españolas del medio rural ante la innovación? Un estudio. *Aula Abierta, 45*(1), 25-32. doi:10.17811/rifie.45.1.2017.25-32

Blanco, H. (2017). La didáctica en la práctica docente. *Boletín Científico de la Escuela Superior Atotonilco de Tula.* Obtenido de https://repository.uaeh.edu.mx/revistas/index.php/atotonilco/article/view/2197

Bourne, C., & Salgado, V. (22 de Diciembre de 2016). *AIKA.* Obtenido de AIKA: http://www.aikaeducacion.com/tendencias/los-videojuegos-transforman-aula/

Cantillo, L., Nieves, A., & Pacocha, L. (25 de Agosto de 2018). *docsity*. Obtenido de docsity: https://www.docsity.com/es/vectores-en-programacion/4112434/

Castellanos, Y., Castellanos, C., Salazar, J., & Casas, W. (2016). EL VIDEOJUEGO COMO RECURSO EDUCATIVO. *Escolme*. Obtenido de http://www.escolme.edu.co/revista/index.php/cies/article/view/71/68

Castro, M. (2020). El mundo programado por los niños. *GK Revista electrónica.* Obtenido de https://gk.city/2020/10/25/beneficios-aprender-programacion-ninos-ninas/

Ceron, A., Perea, M., & Figueroa, J. (2020). *Métodos empíricos de la investigacion.* Obtenido de https://www.uaeh.edu.mx/docencia/P_Presentaciones/icea/asignatura/mercadotecnia/2020/metodos-empiricos.pdf

Dorado, S., & Gewerc, A. (2017). El profesorado español en la creación de de materiales didácticos: Los videojuegos educativos. *Digital Education Review*(31), 176-195. Recuperado el junio de 2021, de https://dialnet.unirioja.es/servlet/articulo?codigo=6052467

Echeverri, A. (19 de Octubre de 2014). *Ingenioteka*. Obtenido de Ingenioteka: http://www.ingenioteka.com/metodologia-de-cascada-en-los-videojuegos/

EcuRed. (2020). Lógica de programación. Obtenido de https://www.ecured.cu/L%C3%B3gica_de_programaci%C3%B3n

EDACOM. (2019). Importancia de enseñar programación a niños. *EDACOM*.

Espinoza, E. (2016). *Universo, muestra y muestreo.*

Ferrer, S. (2001). *videojuegoseduca.* Obtenido de videojuegoseduca: http://ardilladigital.com/DOCUMENTOS/TECNOLOGIA%20EDUCATIVA/TICs/T8%20VIDEOJUEGOS/08%20LOS%20VIDEOJUEGOS.pdf

Fuentes, L., & Perez, L. (2015). Los videojuegos y sus efectos en escolares de Sincelejo, Sucre. *Red de Revistas Científicas de América Latina, el Caribe, España y Portugal*, 318-328.

Guardiola, E., & Natkin, S. (2005). Game Theory and video game, a new approach of game theory to analyze and conceive game systems. *International Conference on Computer Games*, 166-170. Recuperado el junio de 2021, de https://www.researchgate.net/publication/228573564_Game_Theory_and_video_game_a_new_approach_of_game_theory_to_analyze_and_conceive_game_systems

Guest, G., Namey, E., Taylor, J., Eley, N., & McKenna, K. (2017). Comparing focus groups and individual interviews: findings from a randomized study. *International Journal of Social Research Methodology, 20*(6), 693-708. doi:10.1080/13645579.2017.1281601

Iberdrola. (2021). *Iberdrola*. Obtenido de Iberdrola: https://www.iberdrola.com/talento/beneficios-videojuegos-aprendizaje

Jiménez-Hernández, E., Piattini, M., & Revillagigedo-Tulais, A. (2016). Methodology to construct educational video games in software engineering. *4th International Conference in Software Engineering Research and Innovation*. doi: 10.1109/CONISOFT.2016.25

Lee, G. (16 de Octuber de 2020). *Loadview*. Obtenido de Loadview: https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/

Martínez, J., Egea, A., & Arias, L. (2018). Evaluación de un videojuego educativo de contenido histórico. La opinión de los estudiantes. *Revista Latinoamericana de Tecnología Educativa, 17*(1), 62. Obtenido de http://dehesa.unex.es/flexpaper/template.html?path=/bitstream/10662/7997/1/1695-288X_17_1_61.pdf#page=1

Martinez, L., Ceceñas, P., & Martinez, D. (2014). Obtenido de http://www.upd.edu.mx/PDF/Libros/Tics.pdf

Mata, L. (28 de Mayo de 2019). *Investigalia.* Obtenido de Investigalia: https://investigaliacr.com/investigacion/el-enfoque-cualitativo-de-investigacion/

Mera, J. (2015). Análisis del proceso de pruebas de calidad de software. *Ingeniería Solidaria*.

Núñez-Barriopedro, E., Sanz-Gómez, Y., & Ravina-Ripoll, R. (2020). Los videojuegos en la educación: Beneficios y perjuicios. *Educare, 24*(2). doi:10.15359/ree.24-2.12

Paderewski, P., Fuentes-García, M., Gutiérrez, F., Padilla-Zea, N., & López-Arcos, J. (2017). Diseño de videojuegos orientado a la educación emocional. *V Congreso Internacional de Videojuegos y Educación (CIVE'17)*. Recuperado el junio de 2021, de https://riull.ull.es/xmlui/bitstream/handle/915/6684/CIVE17_paper_21.pdf?sequence=1&isAllowed=y

Palacios, J. (2018). *JIMDO*. Obtenido de JIMDO: https://sofiapalacio.jimdofree.com/cuarto-periodo/contenido/

Pelhon, L. (2019). Lógica de programación: el primer paso para aprender a programar. *HostGator*. Obtenido de https://www.hostgator.mx/blog/logica-de-programacion-primer-paso/

Perez, A. (26 de Junio de 2016). *OBS: Business School*. Obtenido de Business School: https://www.obsbusiness.school/blog/que-son-las-metodologias-de-desarrollo-de-software

Prieto, Á. (2018). *Metodología de desarrollo para videojuegos educativos, basado en ntaciones gráficas.* Granada: Universidad de Granada. Recuperado el junio de 2021

Raffino, M. (2020). ¿Qué es la Didáctica? Obtenido de https://concepto.de/didactica/

Ramos, J. (2017). Los diferentes tipos de testing en el desarrollo de software. Trujillo, Perú. Recuperado el junio de 2021, de https://programacionymas.com/blog/tipos-de-testing-en-desarrollo-de-software

Robinson, J. (2020). Using focus groups. En S. Delamont, M. Ward, & S. Delamont (Edits.), *Handbook of Qualitative Research in Education* (págs. 338–348). doi:10.4337/9781788977159.00040

Rosenthal, M. (2016). Qualitative research methods: Why, when, and how to conduct interviews and focus groups in pharmacy research. *Currents in Pharmacy Teaching and Learning, 8*(4), 509-516. doi:10.1016/j.cptl.2016.03.021

Santana, E. (2020). *aulaPlaneta.* Obtenido de aulaPlaneta: https://www.aulaplaneta.com/2020/11/16/firmas-invitadas/el-papel-de-los-videojuegos-en-el-aula-del-futuro/

Tamayo, G. (2018). Videojuego educativo para evaluación de operaciones básicas: suma, resta, multiplicación y división.

Vazquez. (2019). La transformación de los videojuegos como nueva industria del entretenimiento.

ANEXOS

.

THOMAS MORE
SAMBORONDÓN
Educación inicial - Básica - Bachillerato

Samborondón, 8 de junio de 2021

A QUIÉN LE INTERESE

Por medio de la presente debo indicar que yo, **William Wadamud , Docente** de la materia de programación/computación **de la Unidad Educativa Thomas More**, he revisado el proyecto del alumno: **Lucas Isaac Bonnard Silva** el cual corresponde a un videojuego que pueda servir como herramienta para el docente para poder enseñar la lógica de programación en los más pequeños.

Por lo anteriormente expuesto manifiesto que es viable el proyecto, dado que dicho nivel si puede servir como una herramienta para representar lo que es un bucle de una manera que los más pequeños puedan entenderlo.

Sin otro particular, me suscribo

Atentamente,

Msc. William Wadamud
Docente
Unidad Educativa Thomas More

ANEXO 1: FIRMA DE APROBACIÓN 1

THOMAS MORE
SAMBORONDÓN
Educación inicial - Básica - Bachillerato

Samborondón, 8 de junio de 2021

A QUIÉN LE INTERESE

Por medio de la presente debo indicar que **yo, María Auxiliadora Campos; Vicerrectora de la Unidad Educativa Thomas More** , he revisado el proyecto del alumno **Lucas Isaac Bonnard Silva** el cual corresponde a un videojuego que pueda servir como herramienta para el docente para poder enseñar la lógica de programación en los más pequeños.

Por lo anteriormente expuesto manifiesto que es viable el proyecto, dado que dicho nivel si puede servir como una herramienta para representar lo que es un bucle de una manera que los más pequeños puedan entenderlo.

Sin otro particular, me suscribo

Atentamente,

María Campos S.

Msc. María Auxiliadora Campos
Vicerrectora
Unidad Educativa Thomas More

Dirección: Km. 13 Vía a Samborondón (detrás de la Universidad Ecotec)
Tel.: (04) 372 5100 0980766666
Email: admisiones@thomasmore.edu.ec / www.thomasmore.edu.ec

ANEXO 2: FIRMA DE APROBACIÓN 2

THOMAS MORE
SAMBORONDÓN
Educación inicial - Básica - Bachillerato

Samborondón, 8 de junio de 2021

A QUIÉN LE INTERESE

Por medio de la presente debo indicar que yo, **Erika Laínez Román, Rectora de la Unidad Educativa Thomas More**, he revisado el proyecto del alumno: **Lucas Isaac Bonnard Silva** el cual corresponde a un videojuego que pueda servir como herramienta para el docente para poder enseñar la lógica de programación en los más pequeños.

Por lo anteriormente expuesto manifiesto que es viable el proyecto, dado que dicho nivel si puede servir como una herramienta para representar lo que es un bucle de una manera que los más pequeños puedan entenderlo.

Sin otro particular, me suscribo

Atentamente,

Mgs. Erika Laínez Román
Directora General/Rectora
Unidad Educativa Thomas More

Dirección: Km. 13 Vía a Samborondón (detrás de la Universidad Ecotec)
Tel.: (04) 372 5100 0980766666
Email: admisiones@thomasmore.edu.ec / www.thomasmore.edu.ec

ANEXO 3: FIRMA DE APROBACIÓN 3

FECHA	Abril 25 2020	HORA	15h:35	LUGAR	Thomas More

Integrates de la Reunion	Cargos/Puesto
Lucas Bonnard	Estudiante a desarrollar la tesis
Msc William Wadamud	Docente de computación/programación

TEMA A TRATAR
Reunion 1
OBSERVACIONES
Se hizo la presentación del tema con respeto a la utilización de videojuegos como herramienta para enseñar la lógica de la programación

Firma del Profesor

Firma del Alumno Responsable del proyecto

ANEXO 4: FORMATO DE REUNIÓN 1

FECHA	Abril 25 2020	HORA	15h:35	LUGAR	Thomas More

Integrates de la Reunion	Cargos/Puesto
Lucas Bonnard	Estudiante a desarrollar la tesis
Msc. William Wadamud	Docente de computación/programación

TEMA A TRATAR
Entrevista 1
OBSERVACIONES
Se recibieron los temas adecuados que el videojuego debería tener.

Firma del Profesor

Firma del Alumno Responsable del proyecto

ANEXO 5: FORMATO DE ENTREVISTA 1

FECHA	Abril 25 2020	HORA	15h:35	LUGAR	Thomas More

Integrates de la Reunion	Cargos/Puesto
Lucas Bonnard	Estudiante para desarrollar la tesis
Msc. William Guadamud	Docente de computación/programación

TEMA A TRATAR
Reunion2
OBSERVACIONES
Basándose en la entrevista efectuada, se presentan ideas de como serian cada nivel del videojuego basados en los temas brindados .

Firma del Profesor

Firma del Alumno Responsable del proyecto

ANEXO 6: FORMATO DE REUNIÓN 2

FECHA	Abril 25 2020	HORA	15h:35	LUGAR	Thomas More

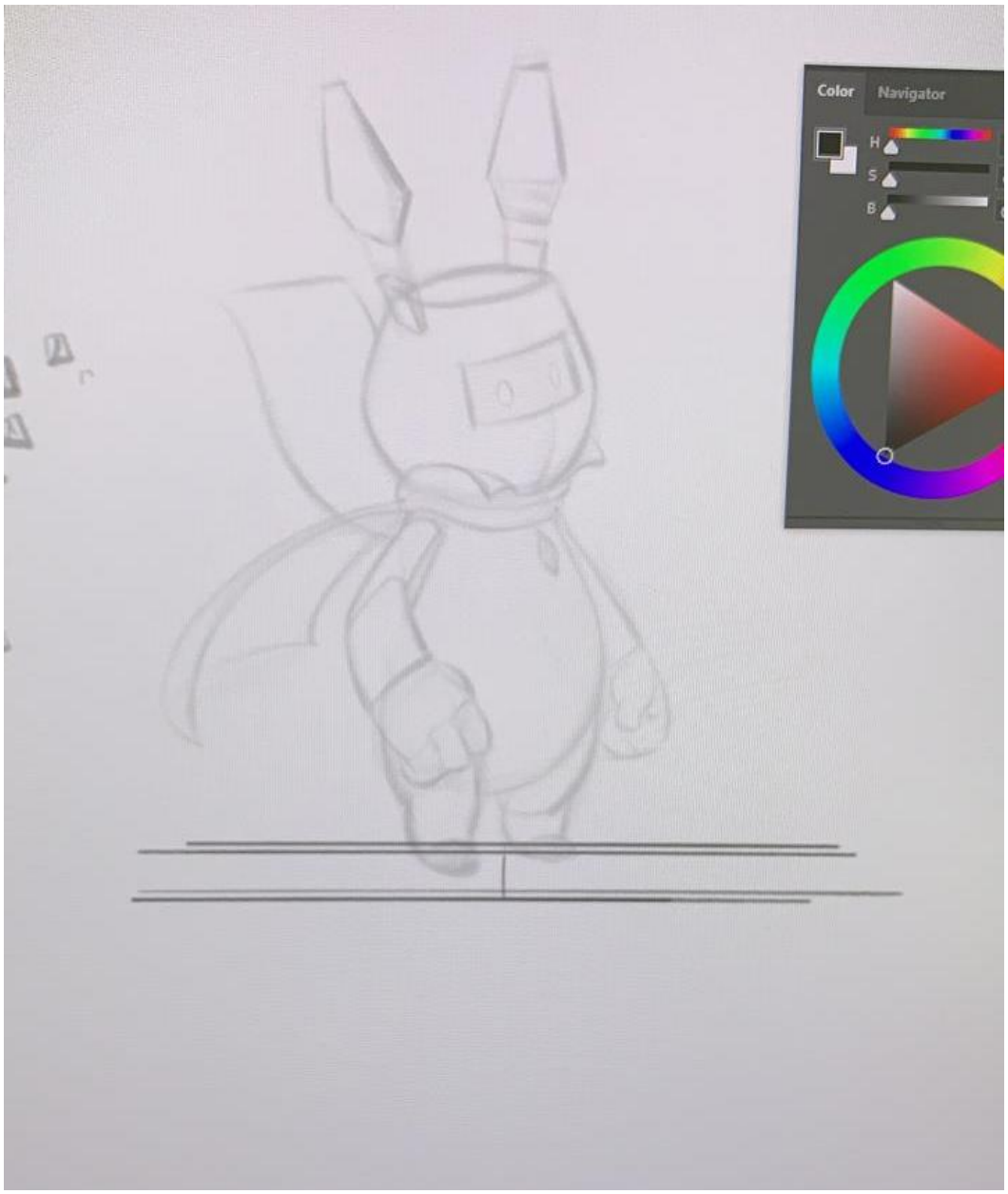
Integrates de la Reunion	Cargos/Puesto
Lucas Bonnard	Estudiante a desarrollar la tesis
Msc. William Guadamud	Docente de computación/programación

TEMA A TRATAR
Entrevista 2
OBSERVACIONES
Acuerdan finalmente de manera descriptiva cuales serían los niveles de videojuegos y se selecciona Bucle como tema para el demo del Videojuego

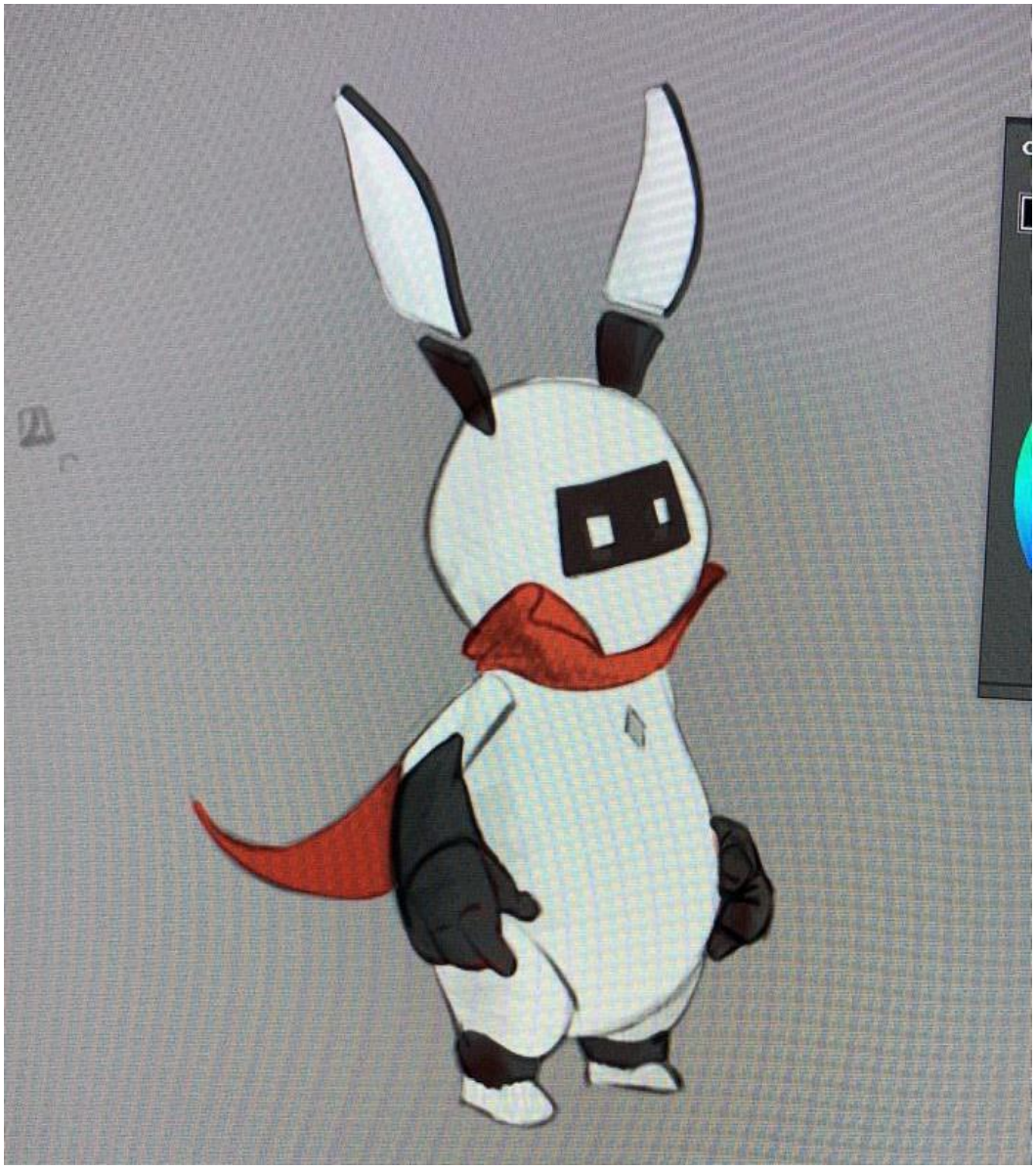
Firma del Profesor

Firma del Alumno Responsable del proyecto

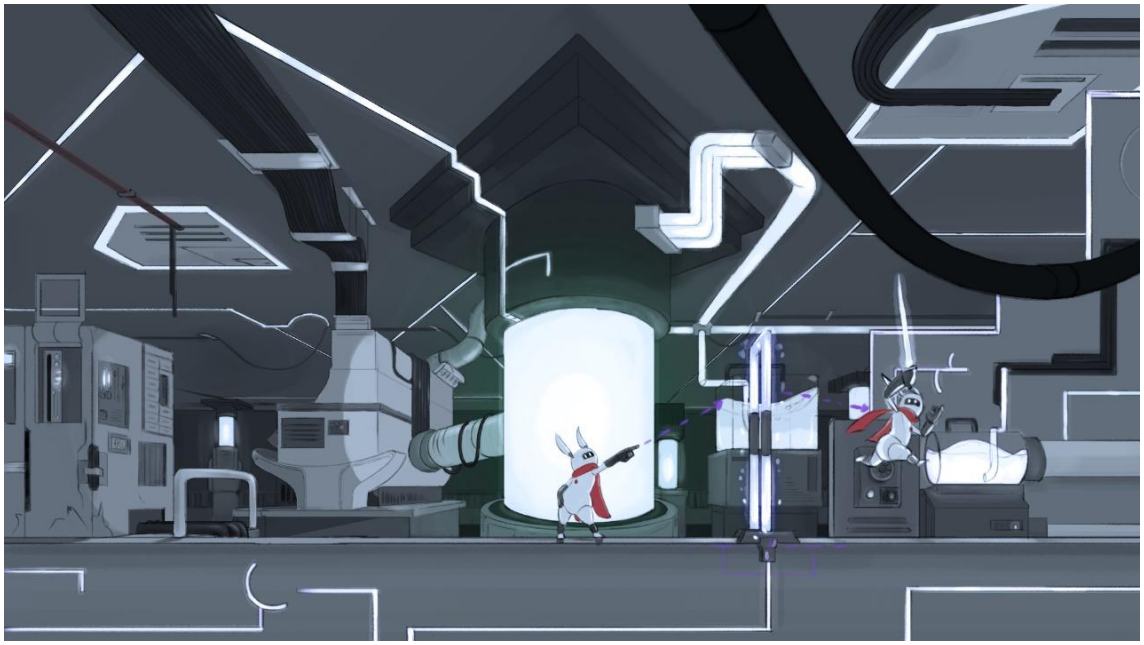
ANEXO 7: FORMATO DE ENTREVISTA 2



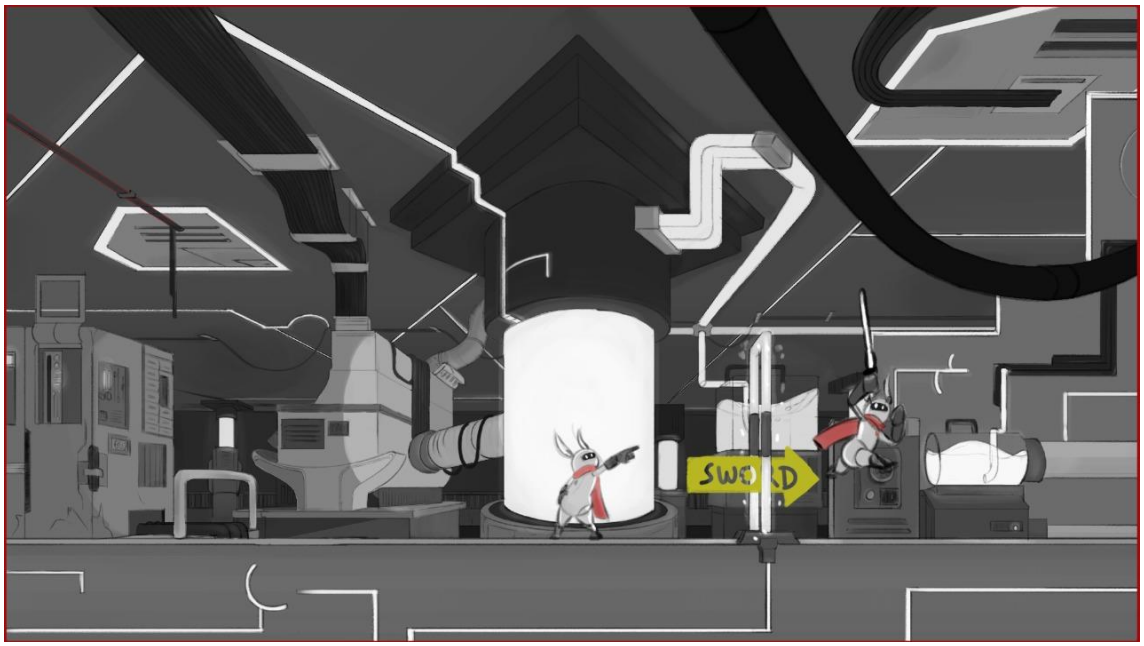
ANEXO 8: PRIMER BOCETO DEL PERSONAJE



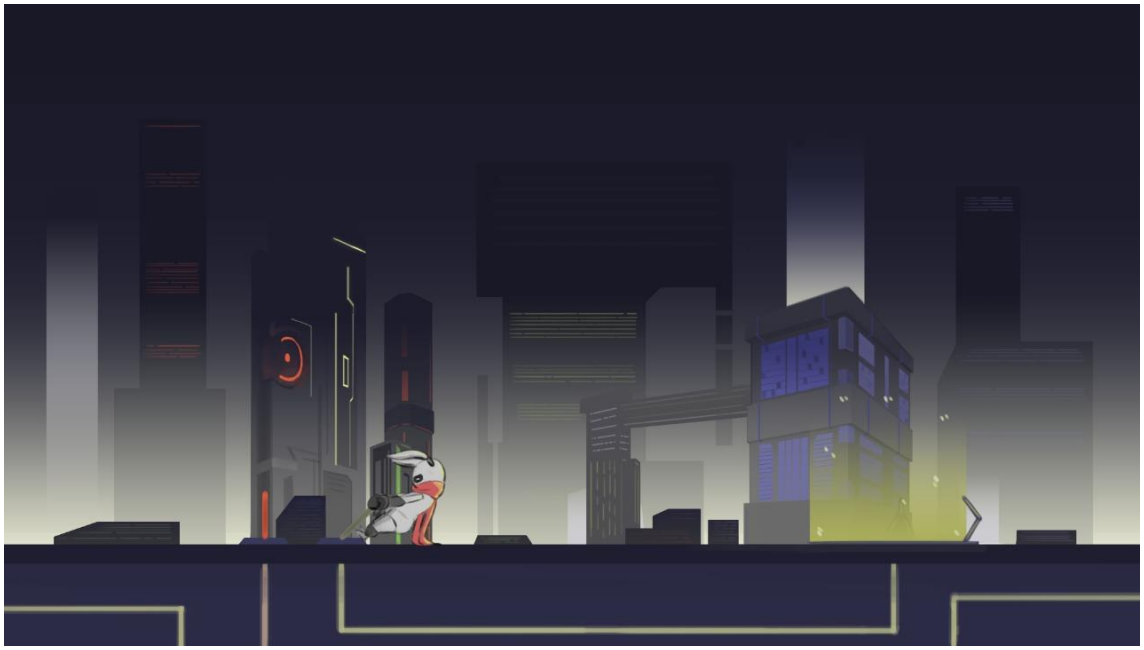
ANEXO 9: SEGUNDO BOCETO DEL PERSONAJE



ANEXO 10: DISEÑO DE VARIABLES 1



ANEXO 11: DISEÑO DE VARIABLES 2



ANEXO 12: DISEÑO DE CONDICIÓN 1

ANEXO 13: DISEÑO DE CONDICIÓN 2