UNIVERSIDAD SERGIO ARBOLEDA

ESCUELA DE CIENCIAS EXACTAS E INGENIERÍA

Desarrollo de Videojuegos con Unity: Blink

Autor/es

Jair Duván Ayala Duarte

Tutor/es

Omar Ferney Álvarez

Documento Técnico

Bogotá D.C. Colombia Octubre 2021

Resumen

A continuación se presenta el proyecto realizado en el semillero Makers, dirigido por el docente Omar Álvarez, en el cual se desarrolló un juego mediante el entorno de Unity, esto anterior para afianzar el entorno de desarrollo y abrir una brecha estudiantil en la cual puedan surgir mayores desarrollos en el futuro, mediante esta plataforma, el presente documento permitirá conocer el proyecto realizado además que servirá como base para los que están iniciando en el desarrollo de videojuegos en Unity. Hay que tener en cuenta que el documento va transcurriendo conforme a las escenas creadas en el proyecto(el desarrollo del mismo), por lo cual algunas de estas pueden ser muy parecidas pero cambian en su funcionalidad. Al momento de realizar el proyecto surgieron ciertos problemas debido a que los documentos utilizados [8] dan las bases para iniciar en el desarrollo, pero hay cosas que son exclusivas de cada programa u juego por lo cual el desarrollo personal y la investigación hacen mucha presencia.

Palabras Clave: Videojuego, Unity, Escenas, Enemigos

Abstract

The following is the project conducted in the Makers seedbed, led by the teacher Omar Alvarez, in which a game was developed using the Unity environment, this previous to strengthen the development environment and open a student gap in which may arise greater developments in the future, through this platform, this document will allow to know the project also will serve as a basis for those who are starting in the development of video games in Unity. It must be taken into account that the document is going according to the scenes created in the project (the development of the same), so some of these may be very similar but change in its functionality. At the moment of realizing the project certain problems arose due to the fact that the documents used [8] give the bases to initiate in the development, but there are things that are exclusive of each program or game for which the personal development and the investigation make a lot of presence.

Keywords: Videogame, Unity, Scenes, Enemies

Motivación y justificación

El presente documento tiene por objeto principal explicar detalladamente el proceso en la creación de un video juego desde cero, teniendo en cuenta su estructura, mediante el código utilizado o en relación con el aspecto. Todo esto con el fin de mostrar su diseño y ejecución plasmado en un resultado final.

La razón del desarrollo de este documento y del proyecto en general: es dar a conocer el entorno de desarrollo de Unity y así poder explotar su potencial, además de lo anterior se tiene en cuenta que un video juego es una forma útil y práctica para fortalecer conocimientos de lógica y programación.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

El presente trabajo se refiere al desarrollo de videojuegos, más precisamente en el entorno de desarrollo de Unity, hoy en día las personas se encuentran rodeadas de múltiples formas de adquirir conocimiento ya sea mediante libros, cursos o todo tipo de información que viaja por la nube como documentos o videos, además de las múltiples formas de aprendizaje también hay muchos campos de aplicación que vienen con estas, mediante este proyecto se pretende explicar la creación de un videojuego, además de mostrar las formas de aplicación de los conocimientos adquiridos durante la carrera de ingeniería de Sistemas y Telecomunicaciones, haciendo énfasis en el ámbito de la programación.

Este proyecto fue ideado, diseñado y desarrollado durante la estadía en el semillero Makers y surgió a partir de la implementación de una nueva rama en este semillero, una rama que saliera un poco de lo que se venía manejando en ese momento en el semillero, de tal forma que se permita en el futuro incluir nuevos estudiantes, para que estos puedan acercarse a la programación, investigación y el desarrollo de videojuegos, explotando así sus habilidades y llegando a más personas de una forma didáctica como puede ser jugar.

Durante el desarrollo se realizaron una serie de investigaciones que llevaron a utilizar un método incremental el cual permite el planteamiento de las escenas, su respectivo modelamiento, la construcción o implementación (en este caso de la escena o el nivel a tratar) y así mismo su ejecución para generar una retroalimentación de la cual se va a ver afectado el nivel o escena siguiente. Lo anterior se puede ver visualizado durante el documento en el cual se plantea su diseño inicial y cómo va evolucionando este con respecto al paso del mismo, hay que tener en cuenta que durante esto se presentaron ciertos inconvenientes en temas de arte como viene siendo el diseño de los personajes y entorno, música como la de inicio, efectos del medio ambiente, efectos del personaje, entre otros.

1.1. Objetivos

1.1.1. Objetivos generales

Proporcionar un videojuego y a su vez un documento el cual pueda ser visto como una guía para la creación, adaptación y planteamiento de nuevos proyectos en el semillero Makers y afines. El juego estará basado en un personaje sencillo el cual tiene como objetivo moverse por los niveles de plataformas 2D, teniendo como meta abrir un cofre del tesoro que lo hará cambiar entre niveles.

1.1.2. Objetivos específicos

- Mediante software de diseño como lo es GIMP diseñar personajes, enemigos y parte del entorno del juego como lo son los árboles, arbustos, rocas y terreno.
- Utilizando el entorno de desarrollo, desarrollar animaciones de personajes que den sensación de movimiento.
- Utilizando el entorno de desarrollo, animar parte del entorno para generar niveles más llamativos.
- Mediante un lenguaje de programación multiparadigma dar control lógico a los objetos de las escenas.
- Mediante un lenguaje de programación multiparadigma y las animaciones generar eventos que se ejecuten dependiendo de si se cumplen ciertas condiciones.
- Mediante un lenguaje de programación multiparadigma y los eventos generados por los usuarios dar control al personaje que es manejado por el usuario.
- Utilizando el entorno de desarrollo, crear enemigos que cumplan con un patrón de movimiento.
- Utilizando el entorno de desarrollo y el lenguaje de programación multiparadigma, crear enemigos que interactúen con el personaje.
- Mediante el gestor Canvas crear un sistema de menú sobre el cual puedan interactuar los usuarios.

1.2. Marco Teórico

1.2.1. Desarrollo Histórico de los videojuegos

Existe un debate de cuál fue el primer videojuego o mejor dicho: el primer juego electrónico que se visualizó mediante una pantalla, pero sin duda alguna esto se remonta a el juego de tres en raya también llamado "OXO" desarrollado por Alexander S.Douglas en 1952, en el cual se enfrentaba el jugador contra la computadora. Más

adelante en 1958 William Higginbotham creó "Tennis for Two" el cual enfrentaba a dos jugadores y funcionaba mediante un algoritmo de trayectoria y un osciloscopio, permitiendo así que dos humanos se enfrentaran entre sí. Ya para 1972 se puede decir que inician de forma comercial los videojuegos cuando Ralph Baer considerado el padre de los videojuegos sacó al mercado una consola que se conectaba a los televisores y permitía jugar ping pong, la anterior sin guardar progresos o marcar puntos de forma automática. [4]

A partir de este momento salta el boom de los videojuegos y es cuando llega ATARI al mercado, mejorando así el juego de Ralph e instalando este en una máquina tragamonedas con un televisor, siendo este tan famoso que se empezaron a crear 100 máquinas de "Pong" por día (Pong es el nombre del juego de la maquinita), dos años después en 1974 se empezó a vender este producto para el uso doméstico, mediante una consola de video, esta contando con un micro ship que permitía una mejor experiencia para los usuarios.

Para 1978 fue lanzado al mercado "Space Invaders" el cual es el juego de matar marcianos mas conocido, juego que fue muy revolucionario pero a su vez opacado en 1980 por "PACMAN" de la compañía Namco el cual batió récords e inclusive consiguió el récord Guinness al videojuego de arcade más exitoso, en este momento de la historia los videojuegos estaban en su mayor auge, pero debido a la competencia y empresas que sacaban videojuegos de mala calidad se empezó a perder la confianza de los usuarios haciendo que grandes empresas ya no supieran qué hacer con su producción, empezaron a surgir dudas sobre la viabilidad de los videojuegos y no fue hasta 1983 que una consola japonesa empezó a dominar el mercado haciendo surgir nuevamente esa confianza en los videojuegos, consola que dos años después la conoceríamos como Nintendo. [13]

Hoy en día se puede decir que los videojuegos son parte importante del día a día, ya sea como medio de entretenimiento o de forma educativa, un ejemplo de esto es Daniel Pajuelo un sacerdote que enseña clases de religión mediante "Minecraft", videojuego en el cual se conectan los estudiantes a construir templos y así aprender sus partes [3], ahora bien, Minecraft no es un juego educativo, aún así se logró enseñar mediante este lo cual indica un gran potencial por otros juegos y las aplicaciones que se le pueden dar a estos.

1.2.2. Videojuegos en Colombia

Los videojuegos en Colombia iniciaron cerca de 1985 con la consola Creative la cual fue de las primeras en estar en nuestro país, ya para los años 90 la consola Family fabricada en china llegó a Colombia y gracias a sus bajos precios logró masificarse permitiendo a las personas jugar un gran catálogo de juegos. Tres años después llegan juegos como Mario Bros, Contra y Super Star Soccer incluídos en la consola de Super Nintendo y para 1996 la nintendo 64 y la consola de Sony dieron paso a que los juegos pasaran solamente de presentarse en cassette a CD's también. A partir de este punto otras consolas siguieron llegando al país hasta el día de hoy. [18]

"Actualmente, con una cascada de reconocimientos y nuevos títulos, el impresio-

nante salto que ha dado la industria colombiana de videojuegos durante los últimos años no pasa desapercibido. Con participación en juegos de gran popularidad en Steam (Ark: Survival Evolved), premios en el Indie Prize Awards (World War Doh) y reconocimientos por parte de gigantes de la industria como Epic Games (Decoherence), es más común encontrar nuevas y aplaudibles historias sobre los videojuegos hechos en Colombia." [9]

"La plataforma de ofertas, promociones y descuentos destaca que Colombia ocupa el cuarto lugar con más ingresos por videojuegos en todo América Latina, justo detrás de México, Brasil y Argentina, respectivamente" [17]

1.2.3. Motor gráfico

"Todo motor gráfico ha de ofrecer al programador una funcionalidad básica, proporcionando normalmente un motor de renderizado ("render") para gráficos 2D y 3D, un motor que detecte la colisión física de objetos y la respuesta a dicha colisión, sonidos y música, animación, inteligencia artificial, comunicación con la red para juegos multijugador, posibilidad de ejecución en hilos, gestión de memoria o soporte para localización (traducción de los textos y audios del juego según idioma)" [16]

"Las capacidades gráficas de motor gráfico son una de las claves para su elección, destacando motores gráficos como CryEngine. Pero también es importante la facilidad de desarrollo y la plataforma para la que se va a desarrollar. Describir todas las funciones de un motor gráfico llevaría miles y miles de palabras, pero en esencia, un motor gráfico está ahí para que los desarrolladores no tengan que reinventar la rueda y se puedan centrar en lo importante: su juego" [16]

Al momento de querer iniciar en el desarrollo de videojuegos es crucial elegir un motor para facilitar las cosas. En el mercado se puede encontrar diferentes motores de los cuales se hace mención de tres de los más conocidos en la actualidad:

Unreal: *"Unreal Engine se ha posicionado como uno de los motores de juegos mas completos y que incluso ha trascendido a otras industrias del entretenimiento. Si bien comenzó como el motor con el que se desarrolló el juego de Unreal hoy ya ha evolucionado para brindar diferentes herramientas tales como: editor con renders fotorealistas, físicas complejas y dinámicas de simulación, animación fluida y un sistema de manejo de assets. Además puede ser usado para crear juegos de distintos géneros como plataformas, FPS, juegos de pelea, RPGs y más"* [5]

Unity: *"Unity es un motor de juego multiplataformas desarrollado por Unity Technologies. Si bien cuando fue anunciado en 2005 se dijo que sería exclusivo para juegos de Mac OS-X, en la actualidad se ha expandido considerablemente para dar soporte a más de 25 plataformas. Es un programa bastante versátil, permitiendo desarrollar videojuegos en 2D y 3D, así como en realidad virtual, realidad aumentada y simulaciones. Cuenta con una interfaz principal de programación en C así como funcionalidades de arrastrar y soltar elementos. Para el desarrollo de juegos 2D, Unity permite importar sprites y hacer uso de un avanzado renderizador. Por otro lado, para videojuegos 3D, cuenta con especificaciones para la compresión de texturas, mipmaps y ajustes de resolución para cada una de las plataformas con las que es compatible"* [5]

GameMaker: *"Gamemaker Studio es un motor para el desarrollo de videojuegos 2D creado por YoYo Games. Se destaca por brindar la posibilidad de crear juegos de distintos géneros y para una variedad de plataformas, utilizando una interfaz personalizada que cuenta con lenguaje de programación visual así como lenguaje de script, el cual es denominado en conjunto como 'idioma Gamemaker'"* [5]

1.2.4. ¿Por qué escoger Unity?

Al momento de escoger el motor gráfico se tuvieron en cuenta muchos factores como viene siendo:

Precio: el factor de precio se puede decir que fue el que menos afectó ya que todos manejan tarifas parecidas y se basan en la facturación, por lo cual salía de forma gratuita en este aspecto.

Aprendizaje: Unity cuenta con una documentación muy completa, proyectos de prueba y una comunidad enorme en la cual se puede encontrar desde tutoriales hasta solución de errores que se puedan llegar a tener al momento de realizar un proyecto.

Gráficos: Unity al ser un motor menos estricto que los demás no cuenta con gráficos demasiado realistas(comparado con otros motores), no obstante esto no afecta el proyecto ya que el proyecto no requiere estos resultados.

Rendimiento: Unity es un motor bastante ligero por lo cual los requerimientos de máquina no llegan a afectar significativamente durante el desarrollo.

Programación: En todos los motores se tiene que programar pero algunos presentan facilidades gráficas, ahora bien, Unity utiliza C el cual es un lenguaje multiparadigma sencillo e intuitivo, además que usar un lenguaje permite más flexibilidad con respecto a la programación del juego.

Multiplayer Online: En este caso no es utilizado por lo cual no afecta que motor se hubiera elegido.

Editores: Unity es una opción más flexible en este tema ya que no se le va a sacar mucho rendimiento al juego en temas de arte y sonido.

Estabilidad: Debido a la duración del proyecto se requiere una estabilidad en versiones, a pesar de que Unity constantemente se esté actualizando los componentes estos son parecidos por lo cual no permite gran complejidad al momento de migrar a la versión siguiente, no obstante el proyecto se maneja en una versión estática por lo cual esto no genera mayor complejidad. [15]

1.2.5. C#

"C# (léase C Sharp), es una evolución que Microsoft realizó de este lenguaje, tomando lo mejor de los lenguajes C y C++, y ha continuado añadiéndole funcionalidades, tomando de otros lenguajes, como java, algo de su sintaxis evolucionada.

Lo orientó a objetos para toda su plataforma NET (tanto Framework como Core), y con el tiempo adaptó las facilidades de la creación de código que tenía otro de sus lenguajes más populares, Visual Basic, haciéndolo tan polivalente y fácil de aprender como éste, sin perder ni un ápice de la potencia original de C. En la versión de .NET Core, se ha reconstruido por completo su compilador, haciendo las aplicaciones un 600 % más rápidas" [10] Algunas de las características de este lenguaje son:

"Sencillez: *C# elimina gran cantidad de elementos que son innecesarios en .NET. Por ejemplo, no se incluyen elementos pocos útiles como macros, herencias múltiples o la necesidad de un operador distinto del punto*

Modernidad: *C# Incorpora de forma automática e intuitiva en su lenguaje elementos que se han demostrado con el paso de los años que han sido muy útiles para el desarrollo de aplicaciones*

Seguridad: *Incorpora mecanismo para asegurar que los accesos a tipos de datos se lleven a cabo de forma correcta, por lo que se evita que generen errores difíciles de detectar*

Sistemas de tipos unificados: *Todos los datos que obtenemos al programar C# se guardan en una base para que se puedan volver a utilizar posteriormente. Extensibilidad: puedes agregar tipos de datos básicos, operadores y modificadores cuando se vaya a programar*

Versionable: *Dispone de actualización y mejora continua, permitiendo crear versiones de tipo sin tener miedo a que, con la incorporación de nuevos integrantes, provoquen errores complicados de detectar*

Compatible: *C# mantiene una sintaxis muy parecida a C, C ++, Java y muchos otros lenguajes de programación, para facilitar el trabajo del programador*

Eficiente: *A pesar de las restricciones que tiene C# en todo el código, se puede saltar estas restricciones utilizando objetos a través de punteros"* [10]

1.2.6. Conceptos

Unity: *"Unity ofrece recursos de alta calidad para ayudarlo mientras desarrolla su contenido interactivo. Nuestra documentación completa, bien organizada y fácil de leer explica cómo utilizar cada componente de Unity, con una guía de referencia aparte que explica cómo crear y usar los scripts"* [19]

Visual Studio: *"Visual Studio es un conjunto de herramientas y otras tecnologías de desarrollo de software basado en componentes para crear aplicaciones eficaces y de alto rendimiento, permitiendo a los desarrolladores crear sitios y aplicaciones web, así como otros servicios web en cualquier entorno que soporte la plataforma"* [12]

1.3. Metodología

"El modelo incremental de gestión de proyectos tiene como objetivo un crecimiento progresivo de la funcionalidad. Es decir, el producto va evolucionando con cada una de las entregas previstas hasta que se amolda a lo requerido por el cliente o destinatario.

La principal diferencia del modelo incremental con los modelos tradicionales es que las tareas están divididas en iteraciones, es decir, pequeños lapsos en los cuales se trabaja para conseguir objetivos específicos. Con los modelos tradicionales no pasaba esto; era necesario esperar hasta el final del proceso" [2]

La metodología incremental o modelo incremental fue la utilizada en este proyecto ya que esta permite construir el proyecto por etapas, donde cada etapa agrega funcionalidad al proyecto y dependiendo de esta se puede moldear el resultado final. Esta metodología se puede observar más claramente en el paso de las escenas presentadas en el documento, escenas que durante el desarrollo fueron cambiando diversas funcionalidades con respecto al juego y así mismo moldeando los niveles presentados después de las escenas. En el documento se puede evidenciar la estructura que va tomando cada escena con la cual se moldea un nivel y a partir de este nivel, su diseño, codificación y pruebas se produce retroalimentación para el siguiente.

1.4. Cronograma

"El cronograma es una herramienta esencial para elaborar calendarios de trabajo o actividades. Un documento en el que se establece la duración de un proyecto, la fecha de inicio y final de cada tarea; es decir, una manera sencilla de organizar el trabajo" [11]

A continuación se presenta el cronograma de las actividades realizadas durante el proyecto, este cronograma abarca un total de 52 semanas las cuales se dividen en investigación y desarrollo, contando con 28 y 24 semanas respectivamente.

Investigación, la investigación recae sobre la parte mayormente teórica en la cual se fundan las bases para el desarrollo del proyecto y así poder dar inicio al desarrollo:

- Investigación videojuegos, 4 semanas
- Investigación creación de videojuegos, 3 semanas
- Creación videojuegos sin motores, 6 semanas
- Investigación motores, 2 semanas

- Uso y selección de motores, 1 semana
- Revisión documentación, 3 semanas
- Aprendizaje del motor, 4 semanas
- Prueba del motor, 5 semanas

Desarrollo, en el desarrollo del proyecto se presentan las actividades que fueron necesarias para presentar un resultado:

- Diseño de personajes, 1 semana
- Diseño entorno, 2 semanas
- Control de personaje, 2 semanas
- Creación de escenas, 5 semanas
- Desarrollo de animaciones personajes, 1 semana
- Animación del entorno, 3 semanas
- Control lógico a objetos del entorno, 3 semanas
- Animaciones de objetos del entorno, 1 semana
- Animación de enemigos, 1 semana
- Eventos lógicos de los enemigos, 2 semanas
- Creación del menú, 1 semana
- Eventos del menú, 2 semanas

Para una mejor comprensión de los tiempos se presenta a continuación el diagrama de Gantt de la investigación y el desarrollo:

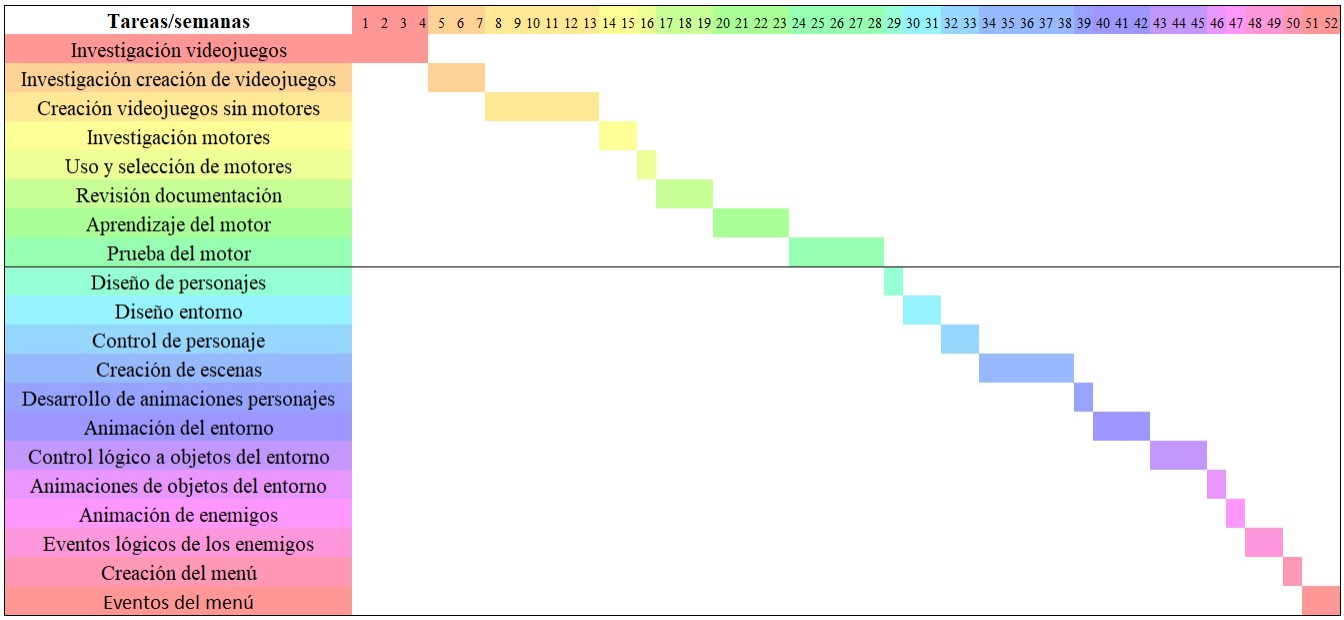

Figura 1.1: Cronograma de actividades mediante diagrama de Gantt

Capítulo 2

Diseño

En este apartado se presenta el diseño del videojuego, aquí se pueden encontrar con los requerimientos tanto funcionales como no funcionales, los casos de uso, los cuales muestran la descripción de las interacciones del juego, además de los diagramas de flujo, estado y/o movimiento del personaje junto con los diseños de escenario.

2.1. Requerimientos funcionales

Los requerimientos funcionales son las especificaciones para el correcto funcionamiento del sistema, estos requerimientos definen la función del sistema de software o de sus componentes.

Identificación del requerimiento	RF 01
Nombre del requerimiento	Movimiento de personaje del jugador
Características	El jugador debe poder moverse a través de los niveles.
Descripción del requerimiento	El jugador deberá poder mover el personaje para interactuar con los niveles y así mismo superar cada uno de estos, para esto se utilizan las flechas del teclado y las letras A, D y W.
	Con la flecha izquierda o A se mueve a la izquierda
	Con la flecha derecha o D se mueve a la derecha
	Con la flecha arriba o W el jugador puede saltar.
Requerimiento NO funcional	RNF 09
	RNF 10
Prioridad del requerimiento	Alta

Tabla 2.1: Movimiento de personaje del jugador RF01

Identificación del requerimiento	RF 02
Nombre del requerimiento	Enemigos
Características	Implementación de enemigos en los niveles.
Descripción del requerimiento	Se implementan tres tipos de enemigos los cuales se interponen en la correcta ejecución de los niveles, estos enemigos se dividen en tres tipos:
	Masa: Hace un movimiento repetitivo para que el jugador se haga daño al pasar
	Roca: Detecta el enemigo a una distancia prudente y le dispara cuando entra en su rango
	Puntas: Al igual que la Masa hace un movimiento repetitivo para que el jugador busque el momento preciso para poder pasar.
Requerimiento NO funcional	RNF 09
	RNF 10
Prioridad del requerimiento	Media

Tabla 2.2: Enemigos RF02

Identificación del requerimiento	RF 03
Nombre del requerimiento	Monedas
Características	Hay monedas en los niveles para que el jugador consiga.
Descripción del requerimiento	Las monedas de los niveles ayudan para que los usuarios obtengan mayores puntuaciones además de fomentar la competitividad.
Requerimiento NO funcional	RNF 09
	RNF 10
Prioridad del requerimiento	Media

Tabla 2.3: Monedas RF03

Identificación del requerimiento	RF 04
Nombre del requerimiento	Niveles
Características	Variedad de niveles en el juego.
Descripción del requerimiento	Hay diferentes niveles disponibles, desde los niveles de adaptación a los controles hasta los niveles donde se pone a prueba las habilidades adquiridas en el juego.
Requerimiento NO funcional	RNF 01
	RNF 02
	RNF 03
	RNF 10
Prioridad del requerimiento	Alta

Tabla 2.4: Niveles RF04

Identificación del requerimiento	RF 05
Nombre del requerimiento	Salir del juego
Características	Cerrar la aplicación.
Descripción del requerimiento	Se requiere cerrar la aplicación para finalizar el proceso sin tener que minimizar para hacer este proceso.
Requerimiento NO funcional	RNF 01
	RNF 02
	RNF 07
Prioridad del requerimiento	Media

Tabla 2.5: Salir del juego RF05

Identificación del requerimiento	RF 06
Nombre del requerimiento	Ingresar al juego
Características	Ingresar a la aplicación para utilizar el juego.
Descripción del requerimiento	El ingreso al juego se da mediante el ejecutable generado por Unity, este se genera con el mismo nombre del proyecto.
Requerimiento NO funcional	RNF 01
	RNF 02
	RNF 07
Prioridad del requerimiento	Alta

Tabla 2.6: Ingresar al juego RF06

Identificación del requerimiento	RF 07
Nombre del requerimiento	Recibir daño
Características	El jugador puede recibir daño de su entorno.
Descripción del requerimiento	El jugador está expuesto a los diferentes componentes que generan daño como lo son enemigos, además de esto abrir cofres para pasar niveles también genera daño y caer del nivel(genera daño y reinicia al punto inicial).
Requerimiento NO funcional	RNF 04
	RNF 09
	RNF 10
Prioridad del requerimiento	Media

Tabla 2.7: Recibir daño RF07

Identificación del requerimiento	RF 08
Nombre del requerimiento	Cambio de nivel
Características	Terminar el nivel actual del jugador.
Descripción del requerimiento	Terminar un nivel es requerido ya que sin este proceso no se pueden experimentar nuevas experiencias en el juego, se necesita el RF 04 para este apartado ya que sin niveles no es posible completar este requerimiento.
Requerimiento NO funcional	RNF 01
	RNF 03
	RNF 09
	RNF 10
Prioridad del requerimiento	Alta

Tabla 2.8: Cambio de nivel RF08

Identificación del requerimiento	RF 09
Nombre del requerimiento	Perder el juego
Características	El jugador puede perder el juego.
Descripción del requerimiento	Cuando la vida del jugador llega a 0 será pasado a una vista de fin de juego para salir o volver a iniciar el juego, además de eso se puede finalizar el juego desde el menú de pausa.
Requerimiento NO funcional	RNF 09
	RNF 10
Prioridad del requerimiento	Alta

Tabla 2.9: Perder el juego RF09

Identificación del requerimiento	RF 10
Nombre del requerimiento	Ganar el juego
Características	El jugador puede ganar el juego.
Descripción del requerimiento	Para ganar el juego se necesitan superar todos los niveles por lo cual se hace uso de RF 08 mediante el cual se finalizan los niveles y se llega al último apartado.
Requerimiento NO funcional	RNF 09
	RNF 10
Prioridad del requerimiento	Alta

Tabla 2.10: Ganar el juego RF10

Identificación del requerimiento	RF 11
Nombre del requerimiento	Conteo de puntuación
Características	Conteo de monedas conseguidas por el jugador.
Descripción del requerimiento	El FR 03 está relacionado con este requerimiento ya que las monedas de los niveles ayudan para que los usuarios obtengan mayores puntuaciones además de fomentar la competitividad. Pero se necesita un contador de todas las que se consiguen con el paso de los niveles.
Requerimiento NO funcional	RNF 09
	RNF 10
Prioridad del requerimiento	Media

Tabla 2.11: Conteo de puntuación RF11

Identificación del requerimiento	RF 12
Nombre del requerimiento	Estadísticas
Características	Estadísticas con la finalización del juego.
Descripción del requerimiento	Haciendo uso de RF 11 y RF 3 se mostrarán estadísticas al finalizar el juego, ya sea mediante los requerimientos 09 o 10. Lo anterior para que el jugador quiera volver a jugar para superar su anterior puntuación.
Requerimiento NO funcional	RNF 09
	RNF 10
Prioridad del requerimiento	Media

Tabla 2.12: Estadísticas RF12

2.2. Requerimientos no funcionales

Los requerimientos no funcionales no afectan el correcto funcionamiento del sistema, estos requerimientos definen la estética, rendimiento, disponibilidad, estabilidad, entre otros.

Identificación del requerimiento	RNF 01
Nombre del requerimiento	Menú principal
Características	El sistema permite al usuario visualizar un menú principal desde el cual puede ingresar a los demás elementos del juego.
Descripción del requerimiento	El menú principal facilita al usuario entrar a los demás elementos del juego permitiendo así que la experiencia de usuario sea más fácil.
Prioridad del requerimiento	Alta

Tabla 2.13: Menú Principal RNF01

Identificación del requerimiento	RNF 02
Nombre del requerimiento	Menú de opciones
Características	El sistema permite al usuario visualizar un menú de opciones desde el cual puede ingresar a las opciones del juego.
Descripción del requerimiento	El menú de opciones permite que el usuario pueda modificar las características por defecto del juego, para que mediante esto se pueda disfrutar sin tener que usar programas externos.
Prioridad del requerimiento	Media

Tabla 2.14: Menú de Opciones RNF02

Identificación del requerimiento	RNF 03
Nombre del requerimiento	Menú de niveles
Características	El sistema permite al usuario visualizar un menú de niveles desde el cual puede seleccionar el nivel que se desea.
Descripción del requerimiento	El menú de niveles permite que el usuario pueda navegar de forma más rápida por los niveles ya explorados y así si llegase a perder puede ingresar sin tener que volver a empezar todo el juego.
Prioridad del requerimiento	Media

Tabla 2.15: Menú de niveles RNF03

Identificación del requerimiento	RNF 04
Nombre del requerimiento	Cambiar volumen
Características	El sistema permite al usuario cambiar los niveles de volumen.
Descripción del requerimiento	El nivel del volumen del juego puede ser modificado desde el juego permitiendo que no se hagan modificaciones externas.
Prioridad del requerimiento	Baja

Tabla 2.16: Cambiar volumen RNF04

Identificación del requerimiento	RNF 05
Nombre del requerimiento	Cambiar calidad de texto
Características	El sistema permite al usuario cambiar la calidad del texto.
Descripción del requerimiento	La calidad del texto puede ser modificado desde el juego permitiendo que el usuario tenga mayores rendimientos de máquina.
Prioridad del requerimiento	Baja

Tabla 2.17: Cambiar calidad de texto RNF05

Identificación del requerimiento	RNF 06
Nombre del requerimiento	Cambiar resolución
Características	El sistema permite al usuario cambiar la resolución del juego.
Descripción del requerimiento	La resolución del juego puede ser modificada desde el sistema permitiendo que el usuario tenga mayores rendimientos de máquina.
Prioridad del requerimiento	Baja

Tabla 2.18: Cambiar resolución RNF06

Identificación del requerimiento	RNF 07
Nombre del requerimiento	Cambiar a pantalla completa
Características	El sistema permite al usuario cambiar la resolución del juego.
Descripción del requerimiento	El juego puede ser usado en pantalla completa o no, esto se hace para mejorar la experiencia de usuario.
Prioridad del requerimiento	Baja

Tabla 2.19: Cambiar a pantalla completa RNF07

Identificación del requerimiento	RNF 08
Nombre del requerimiento	Cambiar brillo
Características	El sistema permite al usuario cambiar los niveles de brillo.
Descripción del requerimiento	El nivel del brillo del juego puede ser modificado desde el juego permitiendo que no se hagan modificaciones externas.
Prioridad del requerimiento	Baja

Tabla 2.20: Cambiar brillo RNF08

Identificación del requerimiento	RNF 09
Nombre del requerimiento	Estética del juego
Características	El juego permite que la estética sea amigable.
Descripción del requerimiento	El juego cuenta con varios niveles los cuales cuentan con distintos escenarios y tareas a realizar permitiendo que estos sean atractivos y variables para el jugador, además de que se puede identificar las diferentes estructuras y conocer su funcionamiento.
Prioridad del requerimiento	Alta

Tabla 2.21: Estética del juego RNF09

Identificación del requerimiento de cada nivel	RNF 10
Características	El juego cuenta con componentes por nivel.
Descripción del requerimiento	El juego cuenta con diferentes niveles que tienen componentes que lo acompañan y así no se vean solamente plataformas, haciendo que cada nivel tenga un ambiente diferente.
Prioridad del requerimiento	Media

Tabla 2.22: Componentes de cada nivel RNF10

Identificación del requerimiento	RNF 11
Nombre del requerimiento	Penalización ingreso a nivel desde el menú
Características	Se puede ingresar a los niveles desde el menú con una penalización.
Descripción del requerimiento	Si el jugador quiere ingresar a un nivel en concreto lo puede hacer desde el apartado del menú RNF 03, permitiendo así que se salte todos los niveles anteriores pero ganando penalizaciones al ingresar con menos vida y sin puntos.
Prioridad del requerimiento	Baja

Tabla 2.23: Penalización ingreso a nivel desde el menú RNF11

2.3. Casos de uso

Un caso de uso es la definición de las acciones o actividades que realizan los actores. Los casos de uso describen las actividades que tiene que realizar alguien para cumplir ciertas tareas o realizar satisfactoriamente un proceso.

Identificador CU 01	
Nombre	Iniciar Juego
Descripción	El usuario requiere ingresar al juego para poder disfrutar de la verdadera emoción y frustración de no poder pasar el nivel 9
Actor	Jugador
Precondiciones	Ninguna
Pos condiciones	Se crearán archivos referentes al juego para guardar el proceso en cada cambio de nivel
Escenario	Archivo de descarga del juego y pantalla inicial del juego

Tabla 2.24: Identificador CU 01

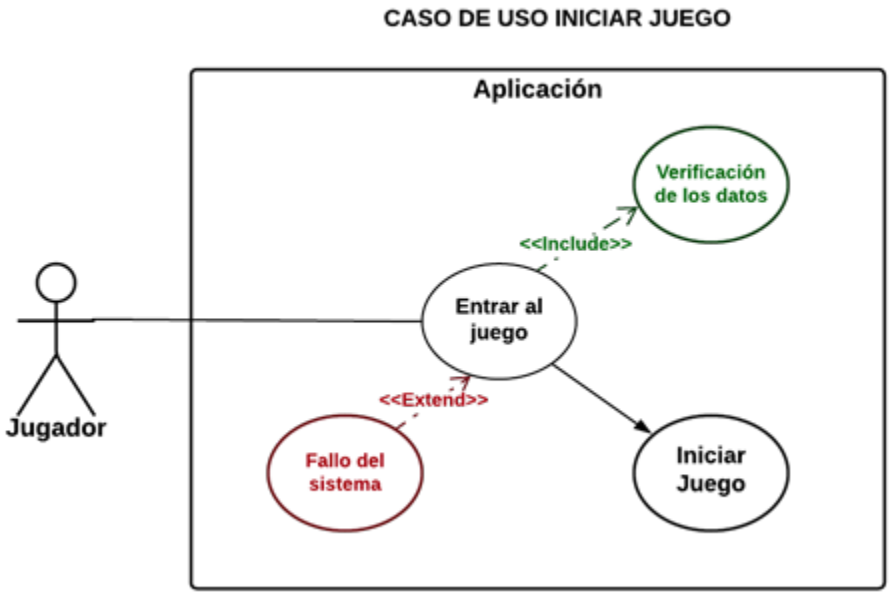

Figura 2.1: Caso de uso Iniciar Juego

Identificador CU 02	
Nombre	Seleccionar niveles
Descripción	El usuario requiere seleccionar niveles para no tener que recorrer todos los niveles ya superados
Actor	Jugador
Precondiciones	Iniciar juego
Pos condiciones	Pos condiciones Se crearán archivos referentes al juego para guardar el proceso en cada cambio de nivel, además de desbloquear los nuevos niveles
Escenario	Menú principal del juego

Tabla 2.25: Identificador CU 02

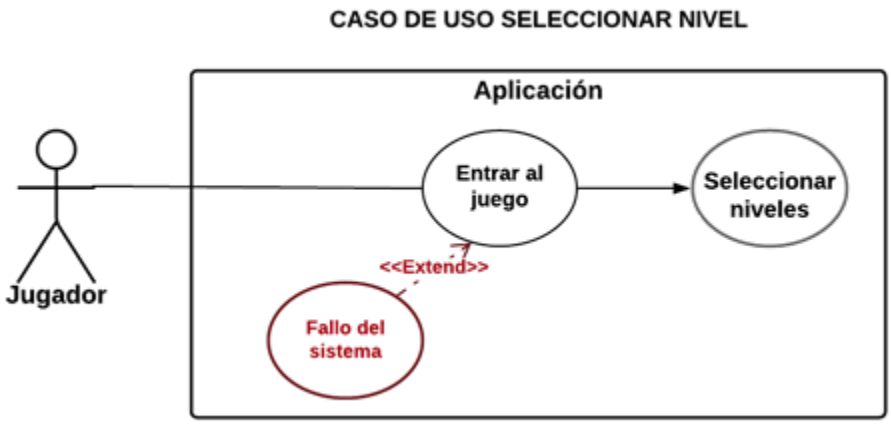

Figura 2.2: Caso de uso Seleccionar niveles

Identificador CU 03	
Nombre	Opciones del juego
Descripción	El usuario requiere modificar las opciones iniciales del juego tales como el nivel de brillo y volumen, la calidad del texto, la resolución de pantalla o si está en pantalla completa o no
Actor	Jugador
Precondiciones	Iniciar juego
Pos condiciones	Se crearán archivos referentes al juego para guardar el proceso en cada cambio de nivel
Escenario	Menú principal del juego

Tabla 2.26: Identificador CU 03

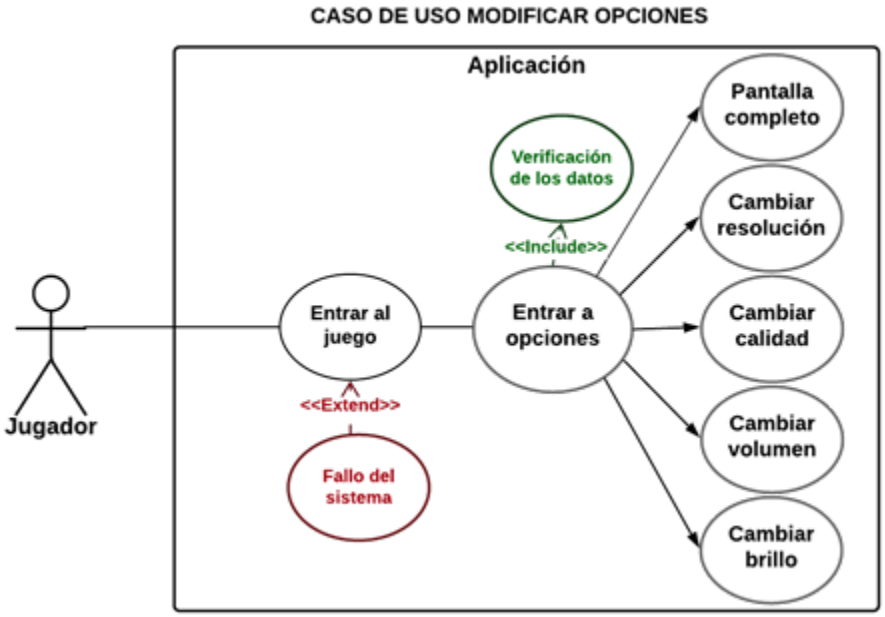

Figura 2.3: Caso de uso Opciones del juego

2.4. Diagramas de flujo

Los diagramas de flujo ayudan a representar la secuencia de actividades en los procesos realizados durante el juego, en este caso se presentan los diagramas de flujo que siguen los enemigos.

2.4.1. Enemigo Masa

Diagrama de flujo de Masa, teniendo en cuenta que este es un movimiento repetitivo sin importar la interacción del jugador, solo se tienen en cuenta los límites.

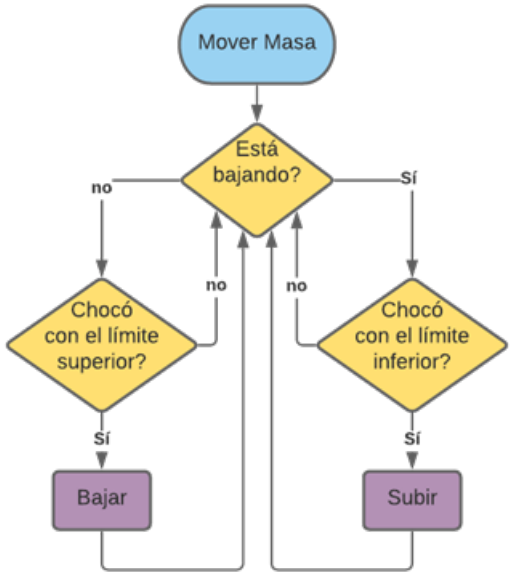

Figura 2.4: Diagrama de flujo enemigo Masa

2.4.2. Enemigo Roca

Diagrama de flujo de Roca, teniendo en cuenta que este es un movimiento que se ejecuta cuando se cumple la condición de que el jugador entra en el campo de visión.

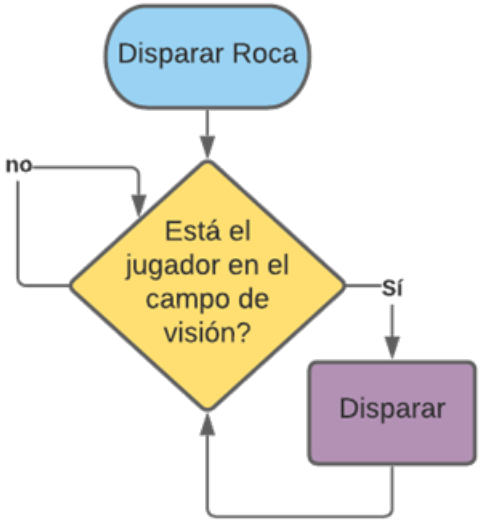

Figura 2.5: Diagrama de flujo enemigo Roca

2.4.3. Enemigo puntas

Diagrama de flujo de Puntas, teniendo en cuenta que este es un movimiento repetitivo

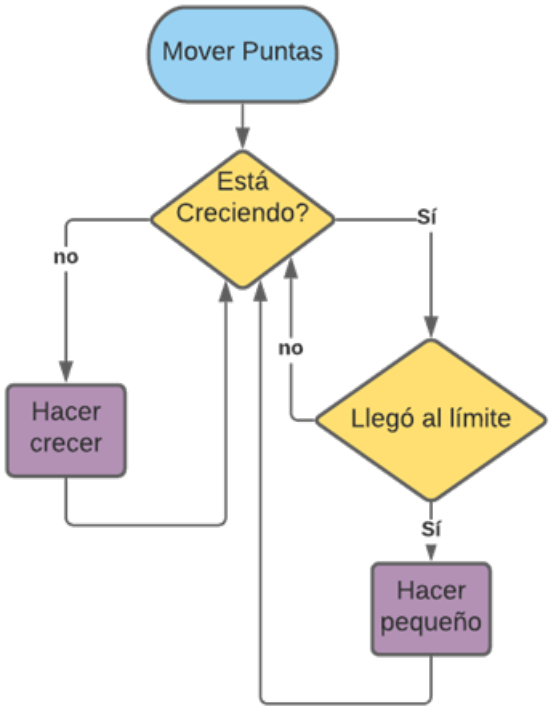

Figura 2.6: Diagrama de flujo enemigo Puntas

2.4.4. Enemigo Cerdo Furioso

Diagrama de flujo de Cerdo Furioso, teniendo en cuenta que este es un movimiento repetitivo sin importar la interacción del jugador, solo se tienen en cuenta los límites.

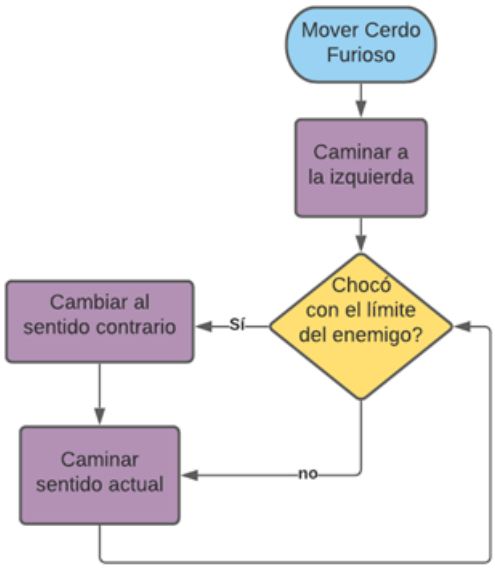

Figura 2.7: Diagrama de flujo enemigo Cerdo Furioso

2.5. Diagramas de estado o movimiento del personaje

Un diagrama de estado representa los estados de la máquina como estados finitos, es un modelo de comportamiento conformado con acciones transitorias a otras acciones. El diagrama de movimiento de personaje es muy parecido al diagrama de estados con la diferencia que solamente se muestra la relación que tiene el jugador al momento de moverse o interactuar con el entorno, en este caso se mostrarán los diagramas de enemigos, objetos del mapa y del jugador.

2.5.1. Diagrama del jugador

El diagrama de jugador representa las animaciones que se presentan al realizar acciones como por ejemplo saltar, quedarse quieto, recibir daño o caminar.

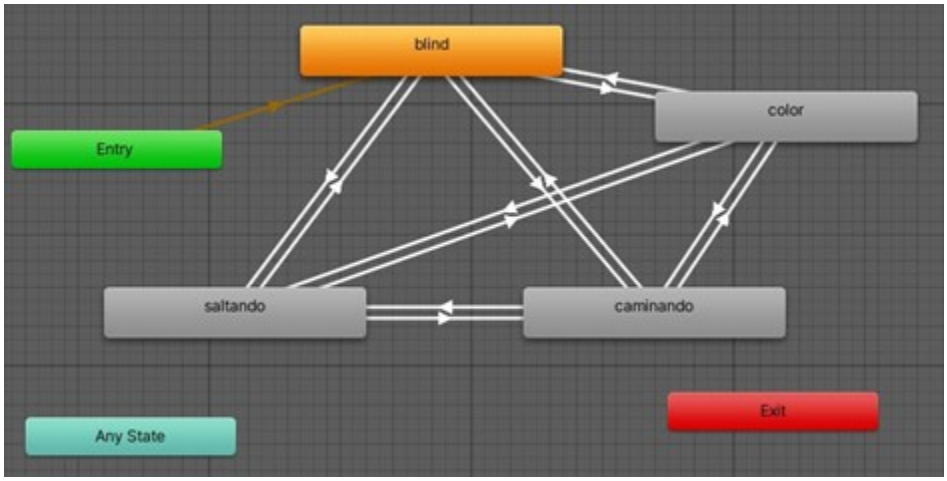

Figura 2.8: Diagrama de jugador

2.5.2. Diagrama del cofre

El diagrama del cofre representa las animaciones que se presentan al abrirlo y abrirlo totalmente.

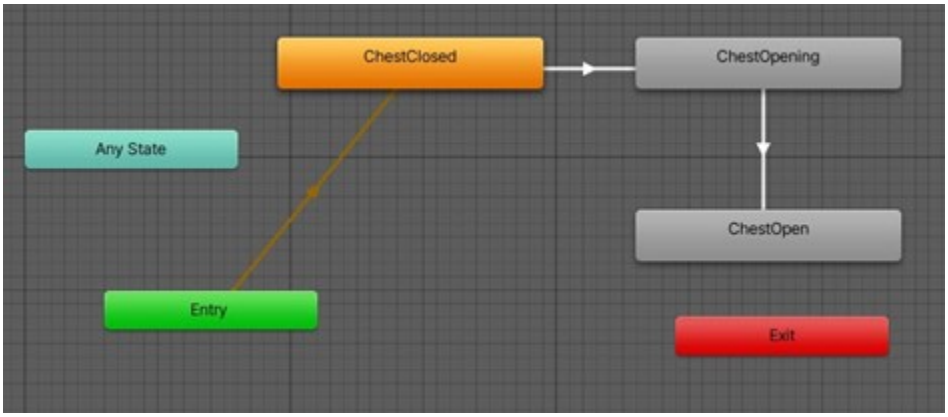

Figura 2.9: Diagrama de cofre

2.5.3. Diagrama de las nubes

El diagrama de las nubes representa las animaciones constantes de estas, en las cuales van recorriendo la pantalla siempre con el mismo movimiento.

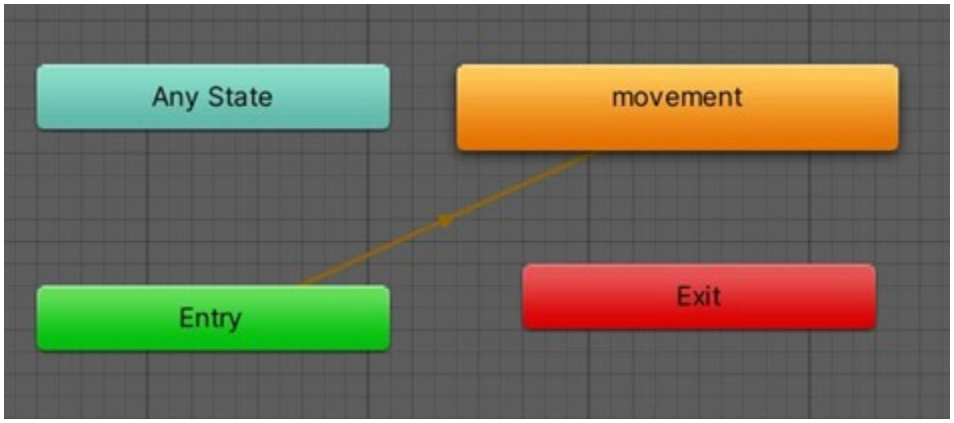

Figura 2.10: Diagrama de las nubes

2.5.4. Diagrama de las monedas

El diagrama de las monedas representa las animaciones constantes de estas, en las cuales tiene una animación de arriba abajo siempre con el mismo movimiento.

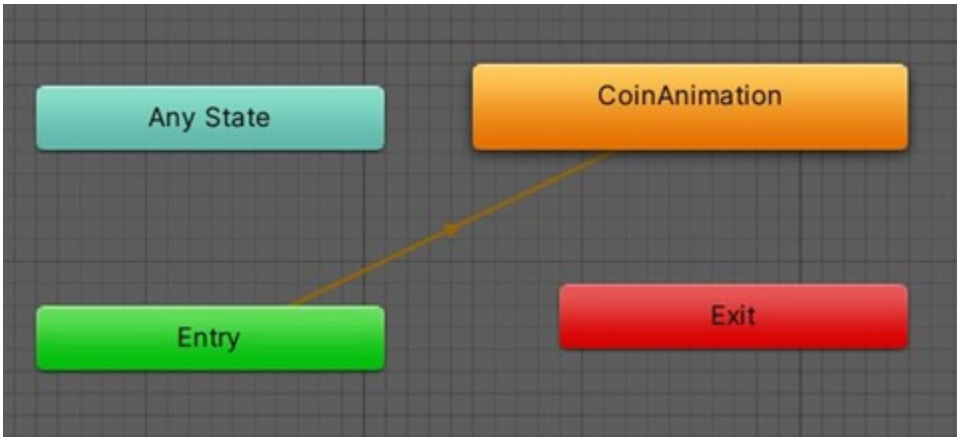

Figura 2.11: Diagrama de las monedas

2.5.5. Diagrama del enemigo Puntas

El diagrama del enemigo Puntas se conforma de dos partes como vienen siendo crecer y rotar, al finalizar la rotación se sale de la animación y vuelve a iniciar.

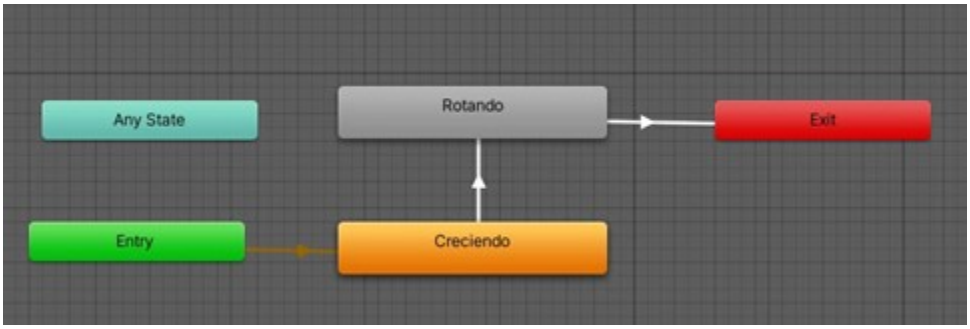

Figura 2.12: Diagrama del enemigo puntas

2.5.6. Diagrama del enemigo Roca

El diagrama del enemigo Roca se conforma de dos partes como viene siendo la animación de la roca(pasivo) y la animación de disparo que se ejecuta por eventos, además de eso tiene la animación de muerte que se puede ejecutar en cualquier momento.

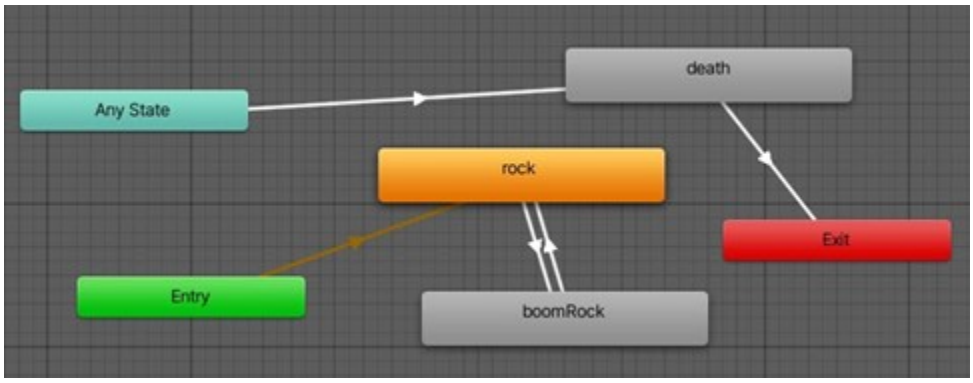

Figura 2.13: Diagrama del enemigo Roca

2.5.7. Diagramas del pájaro azul

El diagrama del pájaro azul representa las animaciones constantes, en las cuales tiene una animación recorriendo la pantalla siempre con el mismo movimiento.

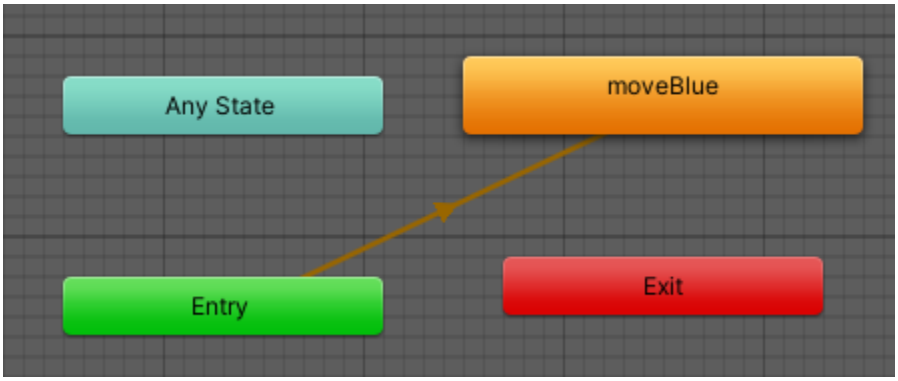

Figura 2.14: Diagrama del pájaro azul 1

Se diferencia de las nubes ya que internamente cuenta con una animación auxiliar que da el efecto de caída y hace más realista la animación, además de la animación propia del diseño.

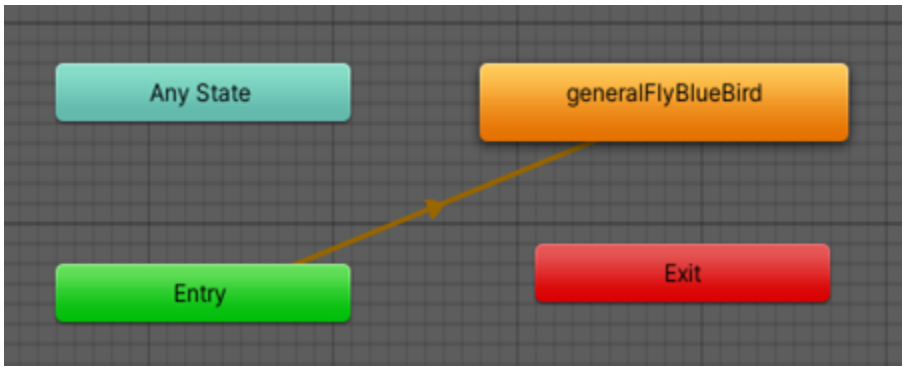

Figura 2.15: Diagrama del pájaro azul 2

2.5.8. Diagrama del murciélago

El diagrama del murciélago representa las animaciones constantes de este, en las cuales tiene una animación de cruce de pantalla siempre con el mismo movimiento, además de la animación propia del diseño.

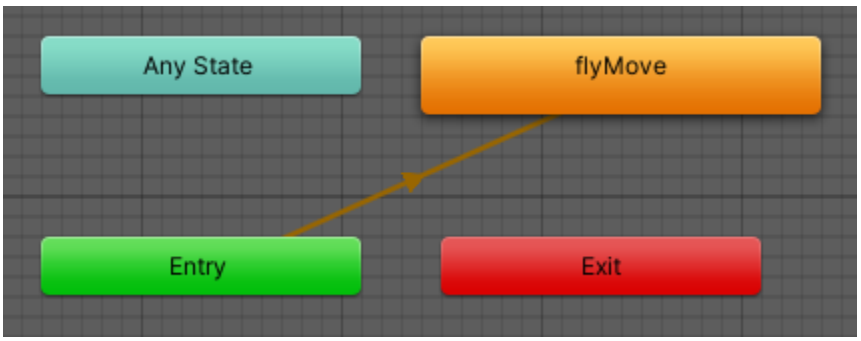

Figura 2.16: Diagrama del murciélago

2.5.9. Diagrama del enemigo cerdo furioso

El diagrama del enemigo Cerdo Furioso se conforma de dos partes como viene siendo la animación del cerdo (pasivo) y la animación de movimiento que se ejecuta por eventos dependiendo de las zonas de movimiento, además de eso tiene la animación de muerte que se puede ejecutar en cualquier momento. Es necesario recalcar que el movimiento de eventos se maneja mediante colisiones en los Scripts.

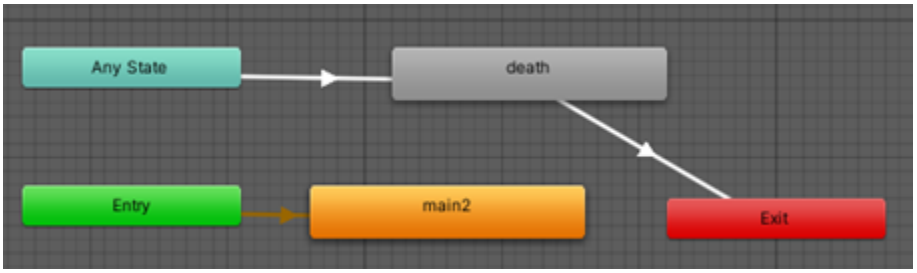

Figura 2.17: Diagrama del enemigo cerdo furioso

2.5.10. Diagrama del enemigo Fantasma

El diagrama del fantasma representa las animaciones constantes de este, en las cuales tiene una animación de cruce de pantalla siempre con el mismo movimiento, además de la animación propia del diseño.

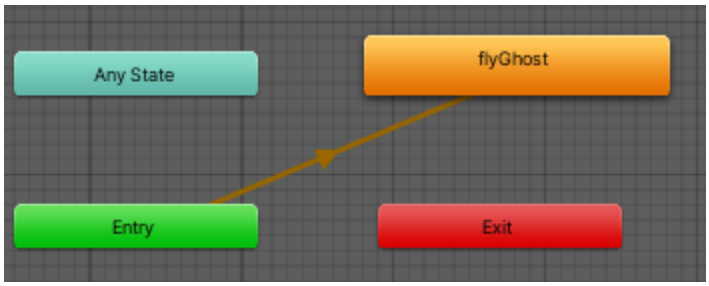

Figura 2.18: Diagrama del enemigo Fantasma

2.6. Diseño de escenario

Para el diseño del juego hay que tener en cuenta que se necesita un escenario presente, sobre el cual se encontrarán diferentes plataformas que permiten al usuario recorrer los niveles, lo primero a tener en cuenta va a ser la base, sobre la cual el usuario se va a mantener de pie, para hacer esto se utilizará el software GIMP el cual permite crear libremente imágenes 2D. Hay que tener en cuenta que con el tiempo se cambiarán los diseños, ya sea por medio de material suministrado por la tienda de Unity o factores externos.

Todas las imágenes presentadas acontinuación se utilizarán para dar un espacio mas amigable a los usuarios, hay que tener en cuenta que estos diseños son propios,

pero Unity nos brinda la oportunidad de encontrar diferentes recursos que pueden variar entre precios, inclusive contando con algunos gratuitos.

2.6.1. Césped

Para el césped se creó un diseño rectangular el cual consta de una base de tierra café(543023) con partículas del mismo color pero un poco mas oscuras(280a02), además de esto el tope es verde claro(66a642) con detalles más oscuros pero del mismo tono(36671f), a continuación se puede ver el resultado.

Figura 2.19: Diseño césped

2.6.2. Tierra

El diseño de la tierra es muy similar al del césped, este diseño tiene variaciones al anterior ya que la capa de césped desaparece dando paso a un café(543023) con partículas del mismo color pero un poco mas oscuras(280a02), además de que se añaden puntos grises(705b52) en la parte inferior simulando pequeñas piedras.

Figura 2.20: Diseño tierra

2.6.3. Arena

Al igual que con el diseño de la tierra la arena tiene un fondo amarillo(af8e46) con partículas grises(7e7b6e) en el interior, simulando las piedras nuevamente.

Figura 2.21: Diseño arena

2.6.4. Planta Verde

para la planta verde se realizaron 5 variaciones de imagen, con diferentes tonos de verde y diferentes formas, esto para permitir tener un escenario más variado, las plantas realizadas son las siguientes:

- Color: 589640
  Detalle: 6fa730

Figura 2.22: Diseño planta verde 1

- Color: 6fa730
  Detalle: 368030

Figura 2.23: Diseño planta verde 2

- Color: 368030
  Detalle: NA

Figura 2.24: Diseño planta verde 3

- Color: 63a135
  Detalle: 589640

Figura 2.25: Diseño planta verde 4

- Color: 61a130
  Detalle: 589640

Figura 2.26: Diseño planta verde 5

2.6.5. Piedras

Al igual que las plantas, las piedras tienen diferentes variaciones para que el escenario sea más variado, en este caso se tienen 7 variaciones las cuales llevan una mezcla de grises claro(a2a2a2), medio(666666) y oscuro(4d4d4d), además de detalles en verde(83a52a) para simular césped.

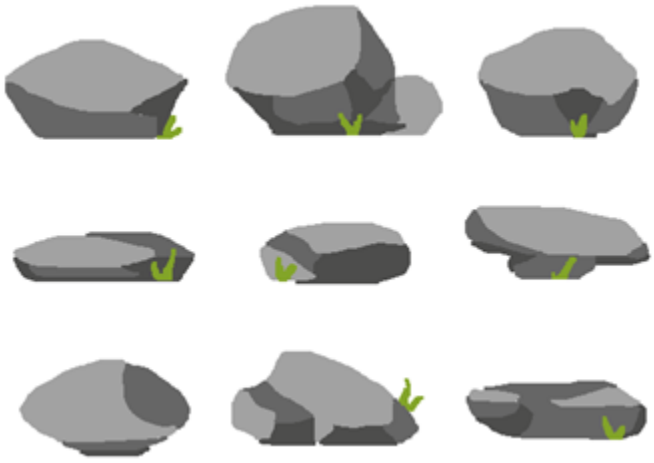

Figura 2.27: Diseño piedras

2.6.6. Rocas

Las rocas se diferencias a las piedras ya que en el escenario van a constar de un mayor tamaño, los diseños realizados tienen tres tonos de color piel como lo son claro(e7d7ca), medio(cfbeae) y oscuro(ab998d), a diferencia de los anteriores diseños estas rocas tienen un diseño geométrico un poco más notable.

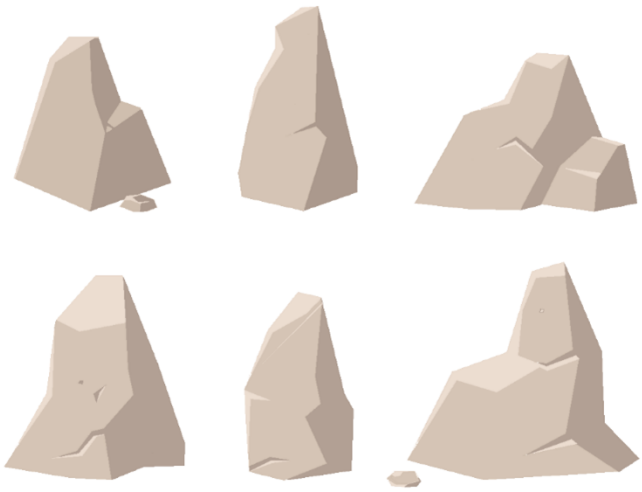

Figura 2.28: Diseño rocas

2.6.7. Árboles

En el apartado de árboles se tienen 11 diferentes modelos, los cuales estarán divididos en tres sesiones, con hojas, sin hojas y especiales:

Con hojas

Los árboles con hojas cuentan con 4 colores característicos los cuales son verde claro(83b917) y oscuro(597d24), y café claro(513311) y oscuro(1c0809), a continuación se pueden ver los diferentes modelos.

Figura 2.29: Diseño árboles con hojas

Sin hojas

Los árboles sin hojas básicamente son árboles que se secaron y solamente cuentan con el tronco, estos tienen un único color como lo es el café(513311).

Figura 2.30: Diseño árboles sin hojas

Árboles especiales

En los árboles especiales se encuentra una palma que cuenta con los mismos colores que los árboles con hojas teniendo una pequeña variación en el tono del café que se encuentra en cada coco.

Figura 2.31: Diseño palmera

Pino

Luego de este se tiene un pino verde(537a05) con nieve(b7ebe0) en sus hojas.

Figura 2.32: Diseño pino 1

Pino 2

Para finalizar el mismo pino(verde claro(237c12) y oscuro(194b0f)) pero sin nieve.

Figura 2.33: Diseño pino 2

Capítulo 3

Desarrollo

En este apartado se presenta el desarrollo del videojuego, aquí se puede encontrar la creación y movimiento de los personajes, las interacciones de los elementos, la tienda de Unity, la estructura del juego dadas las escenas y los niveles que se desarrollaron.

3.1. Creación y movimiento de personajes

Para la primera fase del proyecto se realizó un juego con el cual pudiéramos adaptarnos al entorno Unity, hay que tener en cuenta que este diseño se inició netamente para la adaptación; y con el tiempo se ha venido moldeando de tal forma que pueda evolucionar en algo más llamativo e intuitivo.

Primero que todo se inició por el diseño del escenario y el personaje. Para el escenario se planteó una plataforma de 187 píxeles de ancho por 13 de alto

Figura 3.1: Diseño plataforma de juego

esta plataforma cumpliría la función de dar un soporte al personaje el cual tiene unas dimensiones de 9 píxeles de ancho por 13 de alto.

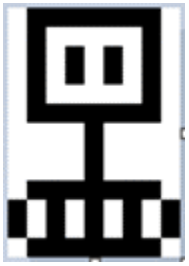

Figura 3.2: Diseño personaje del juego [14]

Al momento de crear un nuevo proyecto 2D en Unity se abrirá el entorno de trabajo el cual permite la manipulación de los objetos a utilizar durante el desarrollo, a continuación se presentan los elementos del entorno de trabajo.

1. Escenario de desarrollo
2. Cámara de usuario
3. Lista de objetos de la escena
4. Explorador de archivos de Unity
5. Inspector: permite la manipulación de objetos

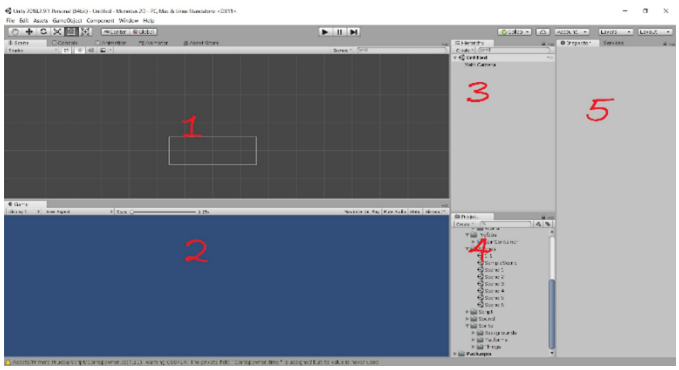

Figura 3.3: Entorno de trabajo

Lo primero que se hace es arrastrar las imágenes creadas anteriormente hacia el explorador de archivos de Unity, luego de esto se proceden a ser ubicadas en la escena, ya sea creando un objeto vacío desde nuestro Hierarchy (lista de objetos en la escena).

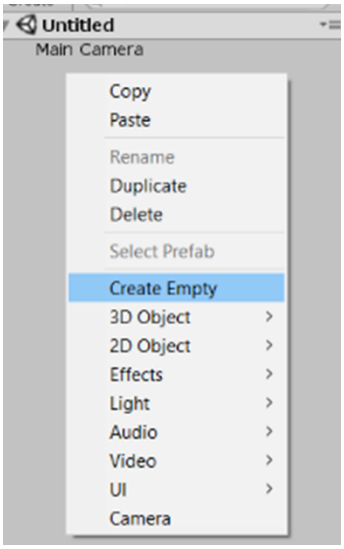

Figura 3.4: Escenario con los personajes

Hay que tener en cuenta que esto también se puede realizar arrastrando la imagen directamente al escenario.

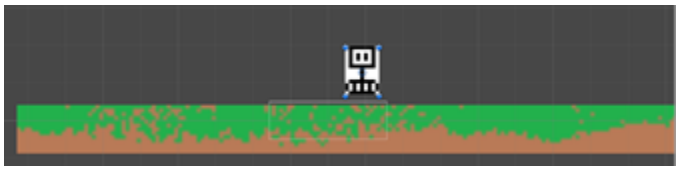

Figura 3.5: Escenario con los personajes

Al momento de arrastrar estas imágenes a nuestro escenario se nos mostrará en el Inspector sus atributos, los cuales corresponderá a un Transform (el cual es la posición de cada objeto en la escena) y un Sprite Renderer que es el atributo que contiene la imagen creada anteriormente.

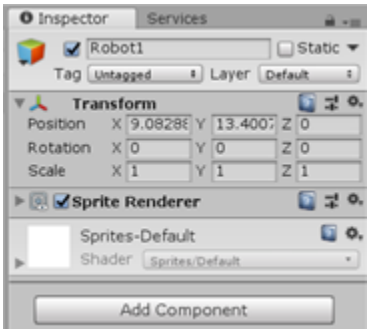

Figura 3.6: Inspector de los objetos recién creado

Tanto el personaje como a la plataforma se le añaden los componentes de Box collider 2D (tiene la función de hacer que nuestros objetos choquen entre los objetos con

este mismo atributo), luego a el personaje se le añadirá el componente Rigidbody2D (Nos permite agregarle físicas al personaje, gravedad, masa, etc), además de que se añadirá un Script previamente creado.

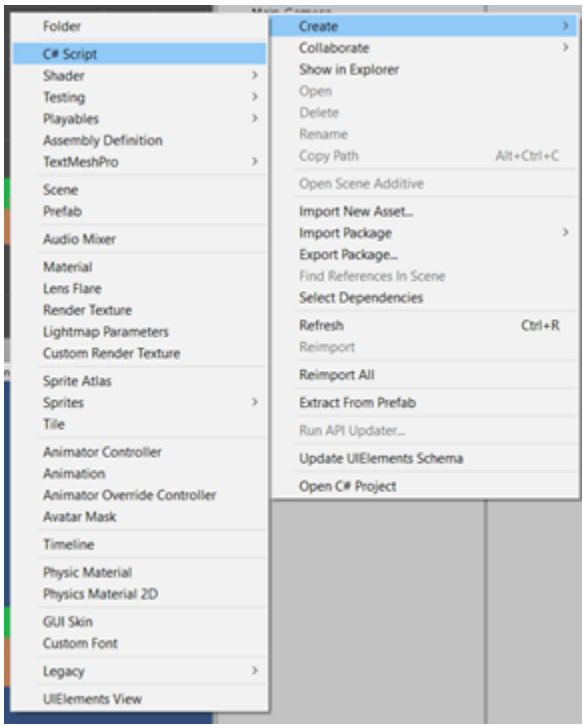

Figura 3.7: Creación de Script en el explorador de Unity

A continuación se muestra un ejemplo de las características del personaje, el cual cuenta con los atributos añadidos anteriormente.

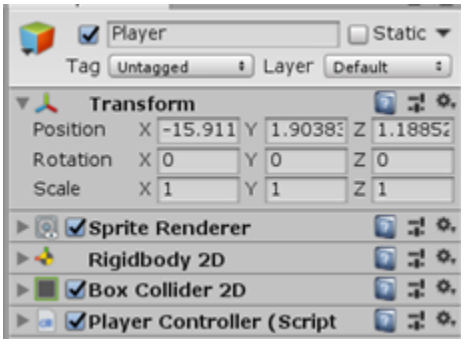

Figura 3.8: Componentes del personaje

Ahora bien, para empezar a darle movimiento al personaje que se tiene se necesita modificar el script creado anteriormente Figura 7 para esto se entra a este script mediante visual Studio, le vamos a añadir un nuevo vector3 para modificar el movimiento en X, además de esto vamos a modificar nuestro personaje de tal forma que cuando

se mueva a la izquierda este se da la vuelta, esto con el comando Flip, el código se puede ver a continuación. Algo importante a recalcar es que el movimiento se puede asignar directamente a una tecla en específico o a un conjunto de estas.

```
private void movimientoX()
{
    if (Input.GetKey("left"))
    {
        movePlayer(new Vector3(-500f * Time.deltaTime, 0, 0), true, true);
    }
    if (Input.GetKey("right"))
    {
        movePlayer(new Vector3(500f * Time.deltaTime, 0, 0), true, false);
    }
}
private void movePlayer(Vector3 move, bool moving, bool flipX)
{
    gameObject.GetComponent<Rigidbody2D>().AddForce(move);
    gameObject.GetComponent<SpriteRenderer>().flipX = flipX;
}
```

Figura 3.9: Movimiento del personaje

Después de que el personaje se mueva se tiene que hacer que este salte al presionar una tecla, para lo cual se utiliza el mismo comando del movimiento solo que ahora en el eje Y, añadiremos un Tag a nuestro objeto de suelo.

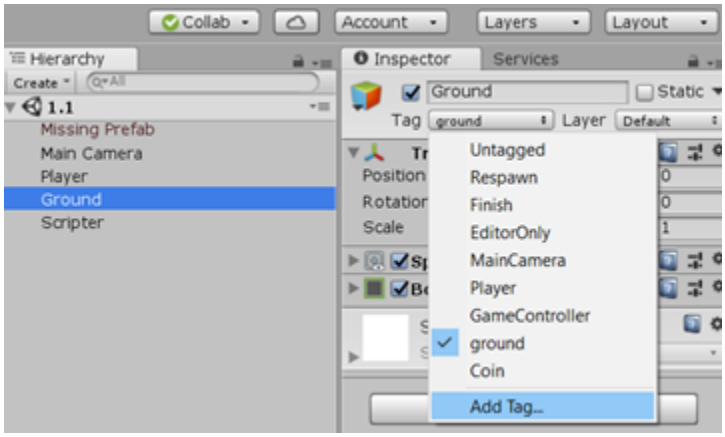

Figura 3.10: Añadir Tag a un Script

Luego de esto se procede a añadir el salto a nuestro personaje, con la restricción de que solamente podrá hacerlo mientras que esté en contacto con el suelo, hay que tener en cuenta que desde este punto se va a tener en cuenta la lógica del programa ya que no es lo mismo un salto de un conejo al de un robot.

```
private void jumpPlayer()
{
    if (Input.GetKeyDown("up") && canJump)
    {
        gameObject.GetComponent<Rigidbody2D>().AddForce(new Vector3(0, 100f, 0));
        canJump = false;
    }
}
private void OnCollisionExit2D(Collision2D collision)
{
    if (collision.transform.tag == "ground")
    {
        canJump = false;
    }
}
```

Figura 3.11: Salto del personaje

Ya teniendo el movimiento del personaje, lo siguiente es añadir animación a este, para lo cual se crea un nuevo personaje, que va a constar de varias imágenes de este mismo simulando el movimiento.

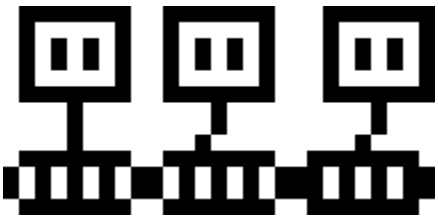

Figura 3.12: Diseño de la simulación del movimiento

Para su uso se arrastra a la carpeta de archivos de Unity y se cambia su "Sprite Mode" de single a múltiple, esto se hace ya que se presentan varias imágenes de un mismo personaje, lo cual tiene que ser dividido por capas o imágenes.

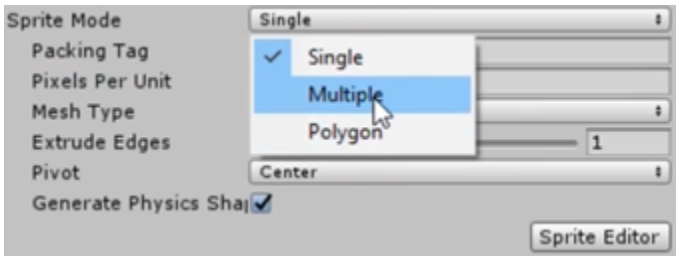

Figura 3.13: Cambio del Sprite Mode

Luego se le da en Sprite Editor y se editan las dimensiones de la imagen, en este caso 9*13, ya que fue lo estipulado anteriormente, si se contara con más imágenes que cumplieran esta misma proporción permanece dividido en la misma cantidad de pixeles, si las imágenes presentan diferentes tamaños ya sea por altura o ancho se puede utilizar el modo automático de división o el manual.

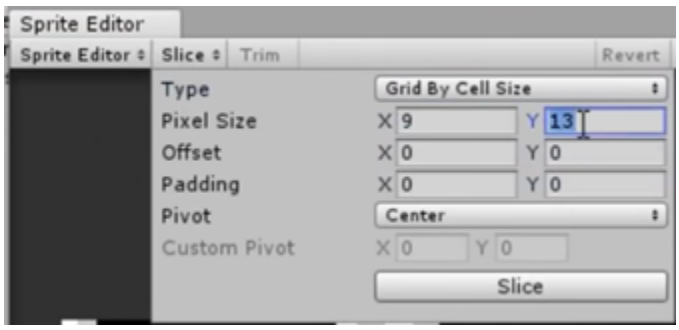

Figura 3.14: Edición del Sprite por separado.

Después de lo anterior se tiene que arrastrar este nuevo Sprite al objeto que contiene el Sprite del personaje original. Ahora bien para crear la animación se tiene que dirigir al espacio del proyecto, donde se procede a dar click derecho, create y Animation, creando así la animación del Script, lo siguiente es seleccionar lo que se creó y yendo al inspector se selecciona la casilla "loop time", esto se hace ya que va a ser una animación que se repita constantemente mientras nos estamos moviendo, por lo tanto arrastraremos esta animación al objeto player en la escena Figura 2.

Para empezar a crear el movimiento del personaje se selecciona y dirige a la pestaña "Animation" y dando en el botón rojo que se encuentra en la parte superior,

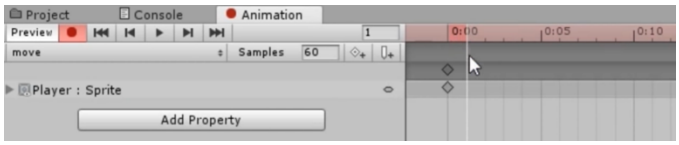

Figura 3.15: Pestaña de Animation para añadir imágenes

después de esto se dirige al inspector y selecciona las imágenes creadas del personaje Figura 15, cada una de estas en un instante de tiempo diferente Figura 16, además de crear una en movimiento también se tiene que crear una animación con el personaje quieto(esto ya sea para respiraciones o acciones de descanso), luego de esto en la pestaña de "Animator" se selecciona la última creada como nuestra animación por defecto.

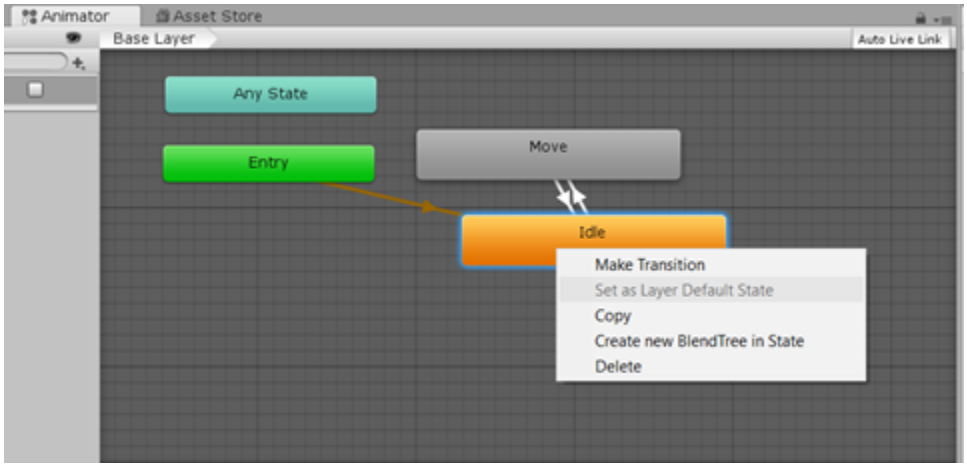

Figura 3.16: Selección de animación por defecto

Al momento de tener todo creado se dirige a crear las transiciones entre las animaciones (dando click derecho sobre cada animación) Figura 17, además de esto la creación un parámetro en la misma pantalla que se llame "moving", el cual se modifica mediante el código para que así se puedan hacer las transiciones entre el movimiento y el estado en reposo.

```
private void movimientoX()
{
    if (Input.GetKey("left"))
    {
        right = false;
        movePlayerX(new Vector3(-500f * Time.deltaTime, 0, 0), true, true);
    }
    if (Input.GetKey("right"))
    {
        right = true;
        movePlayerX(new Vector3(500f * Time.deltaTime, 0, 0), true, false);
    }
    if (!Input.GetKey("left") && !Input.GetKey("right"))
    {
        gameObject.GetComponent<Animator>().SetBool("moving", false);
    }
}
private void movePlayerX(Vector3 move, bool moving, bool flipX)
{
    gameObject.GetComponent<Rigidbody2D>().AddForce(move);
    gameObject.GetComponent<Animator>().SetBool("moving", moving);
    gameObject.GetComponent<SpriteRenderer>().flipX = flipX;
}
```

Figura 3.17: Movimiento del personaje

3.2. Interacción de elementos

Para la primera fase del proyecto se realizó un juego con el cual pudiéramos adaptarnos al entorno Unity, hay que tener en cuenta que este diseño se inició netamente para la adaptación; y con el tiempo se ha venido moldeando de tal forma que pueda evolucionar en algo más llamativo e intuitivo, en este apartado se va realizar la interacción de elementos tales como monedas, puntos u objetos varios del entorno.

Los puntos son un componente importante, ya que permite monitorear el proceso o desarrollo que se lleva en el juego, para esta ocasión se optó por un diseño de monedas bastante simple el cual permite diferenciar los objetos que se obtienen para ganar dichos puntos.

Figura 3.18: Diseño de la moneda del juego

Siguiendo con lo anterior; las monedas van a estar situadas dentro de un controlador en la escena el cual será un objeto vacío, esto para poder generar movimiento sobre el objeto y a su vez situarlo en diferentes partes del escenario. Lo primero que se hace es arrastrar el objeto(moneda) sobre el objeto vacío.

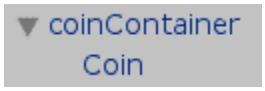

Figura 3.19: Objeto "moneda" añadido a objeto vacío

Hay que tener en cuenta que a esta moneda se le añade un BoxCollider2D, en el cual se activa la casilla "Is Trigger", así el personaje no se chocará con la moneda, solamente la traspasará y detectará la colisión.

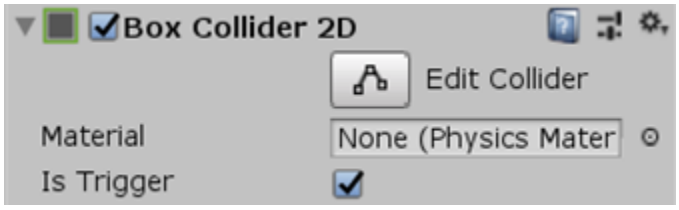

Figura 3.20: Activación de "Is Trigger"

Ahora bien para la detección de la colisión se crea un nuevo script el cual será el controlador de la moneda, este script llevará el evento "OnTriggerEnter2D" el cual se activará cuando entre en contacto con otros objetos(Collider), además de eso

el “padre” de este objeto(el objeto vacío donde se encuentra la moneda) tendrá que desaparecer, ya que fue otorgado el punto al jugador. Todo esto se hará dentro del controlador de la moneda (CoinController).

```
public class CoinController : MonoBehaviour {


    private void OnTriggerEnter2D(Collider2D collision)
    {

        Destroy(transform.parent.gameObject);
    }
}
```

Figura 3.21: Destrucción objeto al chocar con un Collider

Ahora bien, cada vez que se seleccione una moneda tendrá que aparecer una nueva en otro lugar, esto lo manejaremos por medio del padre de nuestra moneda, para esto se utiliza el siguiente código,

```
public static CoinSpawner coinSpawner;
float timer;
public GameObject coinPrefab;
// Update is called once per frame
void Update () {
    if (!GameObject.Find("Coin"))
    {
        //Llamado de la función generar
        Generar();
    }
}
public void Generar()
{
    timer = 0;
    // genera un float random para la
    //ubicación de una nueva moneda
    float x = Random.Range(-100f, 70f);
    // vector posición de la moneda nueva
    Vector3 position = new Vector3(x, 0, 0);
    // vector rotación de la nueva moneda
    Quaternion rotation = new Quaternion();
    // Instancia de la moneda
    Instantiate(coinPrefab, position, rotation);
}
```

Figura 3.22: Spawner de monedas

en el cual se explican las diferentes líneas, ahora bien, al tener la moneda apareciendo cada vez que se selecciona una; se puede dar movimiento, para esto se utiliza la misma técnica del capítulo anterior en el cual animamos al personaje al moverse.

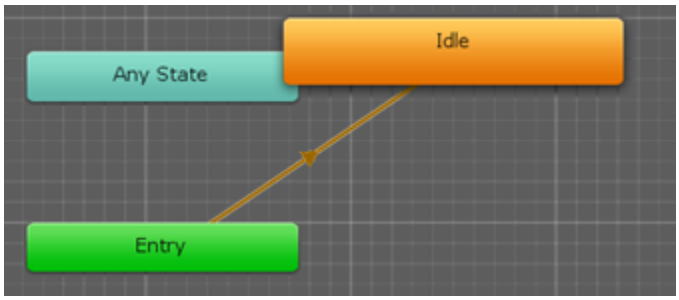

Figura 3.23: Animación de la moneda (Objeto Coin)

Para la realización de la animación se creó dentro de la carpeta Animations una nueva carpeta llamada “Coin” además de su respectivo “Idle”, el cual se le activará la casilla “Loop time” y se arrastrará sobre el objeto coin(Sobre el objeto y no sobre el padre) Figura24 luego de esto se selecciona el objeto “Coin” y se presiona sobre “Aply”, esto para permitir que los cambios realizados se apliquen al prefab

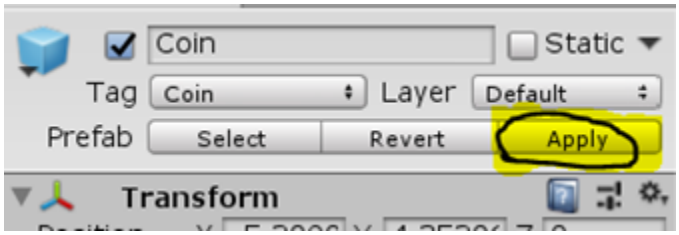

Figura 3.24: Aplicación de cambios al Prefab

a partir de este punto se realiza la animación de la moneda para lo cual se selecciona esta, luego yendo al apartado “Animation” se añade una nueva propiedad de tipo Transform-Position,

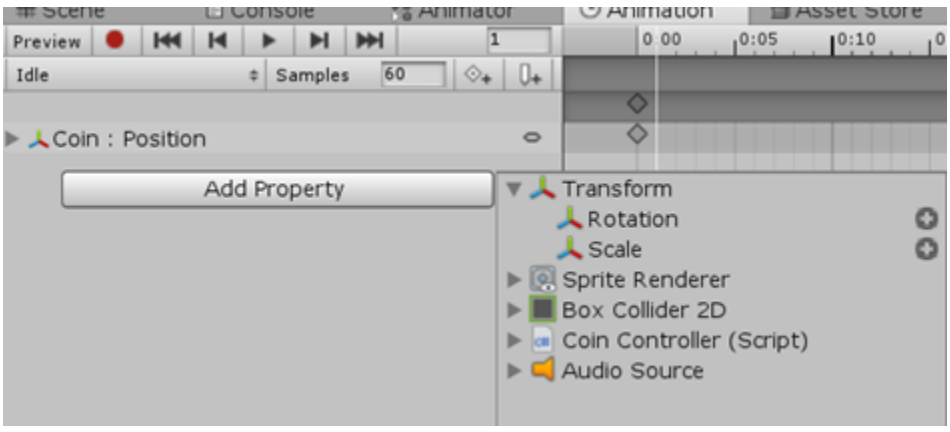

Figura 3.25: Añadir propiedad de Posición

para la cual se ajustará la posición de la moneda. A partir de este punto se añade una “Keyframe” en el medio de la animación,

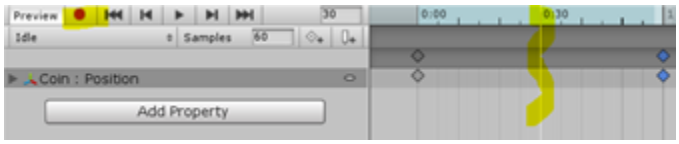

Figura 3.26: Añadir Keyframe

para esto nos situamos en el centro de la línea temporal y presionamos sobre el botón Stop, luego movemos la moneda ligeramente hacia arriba,

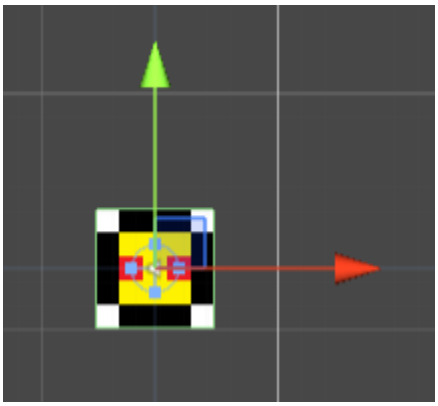

Figura 3.27: Movimiento de la moneda

y automáticamente se creará la animación hacia ese punto que movimos la moneda.

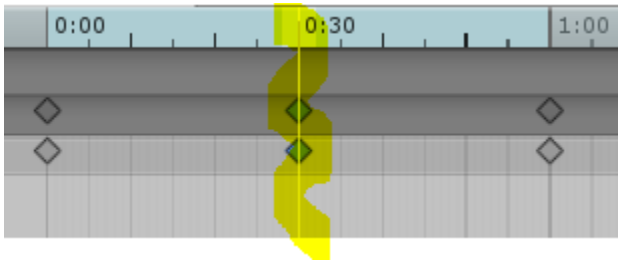

Figura 3.28: Creación automática del Keyframe

Ahora bien, para asignar puntos y que el usuario pueda verlos se crea un Canvas, el cual es un objeto que se ve directamente sobre la pantalla, este permite asignar valores (en este caso los puntos), para esto vamos a crear un Script el cual recibe un valor cada vez que una moneda es seleccionada, estos puntos se almacenan en una variable la cual tendrá una interacción directa con el Canvas permitiendo ver los puntos que el jugador lleva hasta el momento.

3.3. Estructura del juego

El juego va teniendo una serie de variaciones a lo largo de su desarrollo, y donde más se puede apreciar esto es es sus versiones de escenas las cuales presentan cada una variaciones que han surgido a partir de su desarrollo. Hay que tener en cuenta que estas versiones en su mayoría son modificaciones que van surgiendo a lo largo del tiempo, en el cual se añaden nuevas funcionalidades o nuevos personajes para interactuar.

3.3.1. Scene 1.1 (escenario)

Al ser la primera escena con la que se va a contar, se añadirán los objetos principales que van a ir dentro del juego, en este caso solo se incluyeron el personaje principal, dos enemigos y algunos premios que se pueden encontrar alrededor del mapa, además de los objetos externos como lo son el escenario y el agua. Hay que tener en cuenta que estos no cuentan con animación alguna por el momento.

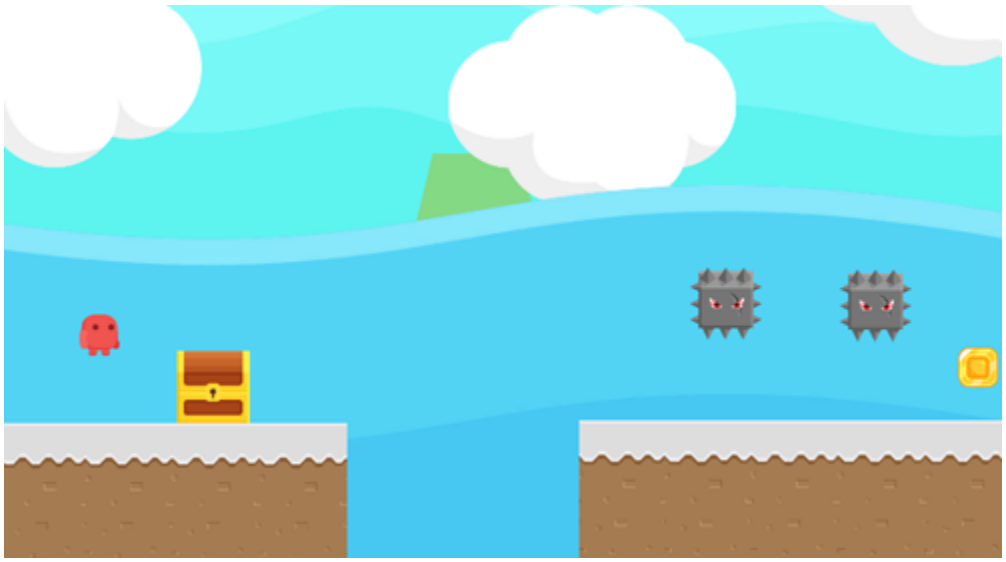

Figura 3.29: Escenario 1

3.3.2. Scene 1.2 (Ajuste de escenario)

Para esta escena se ajustaron errores de la primera en la cual se añadió animación al agua y animación al personaje como lo es moverse de un lado a otro, además de esto se plantean las animaciones al momento de que el personaje camine.

Figura 3.30: Escenario 2

3.3.3. Scene 1.3-1.4 (Ajuste de escenario 2)

Para estas escenas el personaje ya cuenta con movimiento lateral, por lo cual se optó por la animación de los enemigos, para lo cual se asignaron animaciones y zonas específicas de las cuales estos no puedan salir.

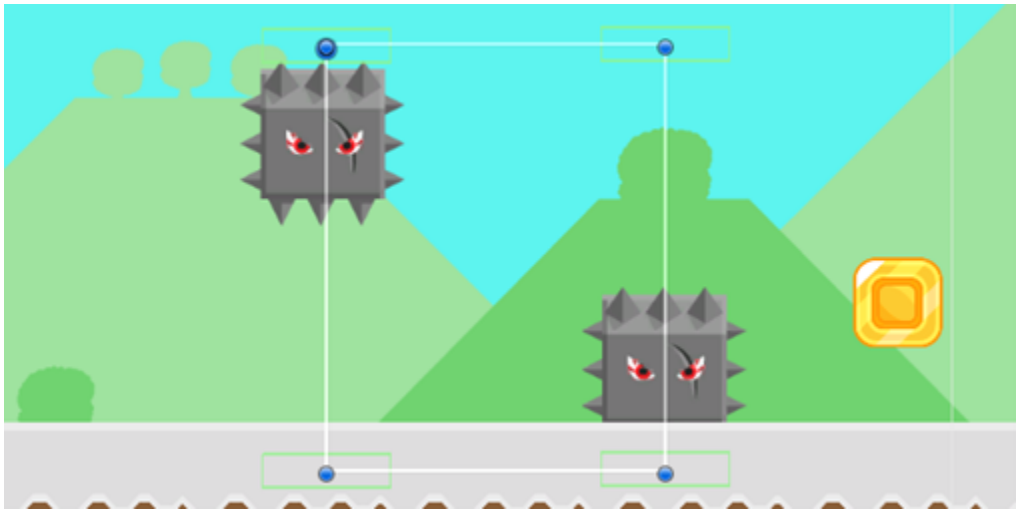

Figura 3.31: Enemigos, movimiento lateral

3.3.4. Scene 1.5 (Interacciones)

Las interacciones del héroe con algunas de los objetos añadidos en el mundo se pueden apreciar en esta escena, ya que se añaden interacciones como parpadeos, abrir cofres y cambios de físicas de los enemigos.

Figura 3.32: Héroe y cofre, interacción con el escenario

3.3.5. Scene 1.6 (Complementos de enemigos y mejoras del héroe)

Para esta escena se agregó la opción para que el personaje pudiera saltar ya que es algo crucial para evitar los obstáculos que presenta el juego, a pesar de que la animación ya estaba la programación aún no se hacía, además de esto se incorporó una bala que sale de la nada a atacar el enemigo, esta bala es un prototipo para una futura interacción con los enemigos.

Figura 3.33: Héroe y cofre, interacción con la bala

3.3.6. Scene 1.7.0-1.8.1 (Nuevos enemigos)

Para la presente escena se añade un enemigo que pueda arrojar balas y no se salga de la temática del juego, en este caso se seleccionó una roca.

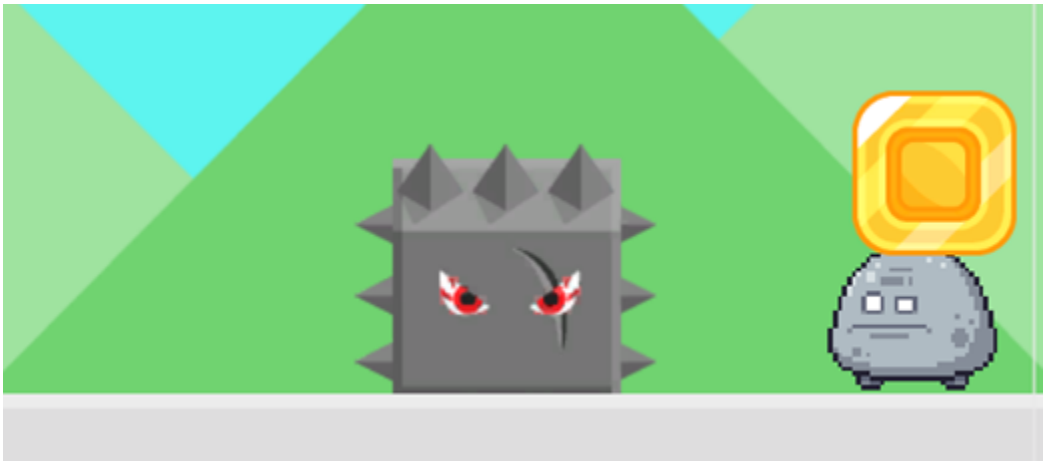

Figura 3.34: Nuevos enemigos

3.3.7. Scene 1.8.1-1.8.3 (Nuevas animaciones)

Las animaciones son cruciales para que se puedan ver cambios y se mantenga un ambiente fluido, en este caso mediante los enemigos.

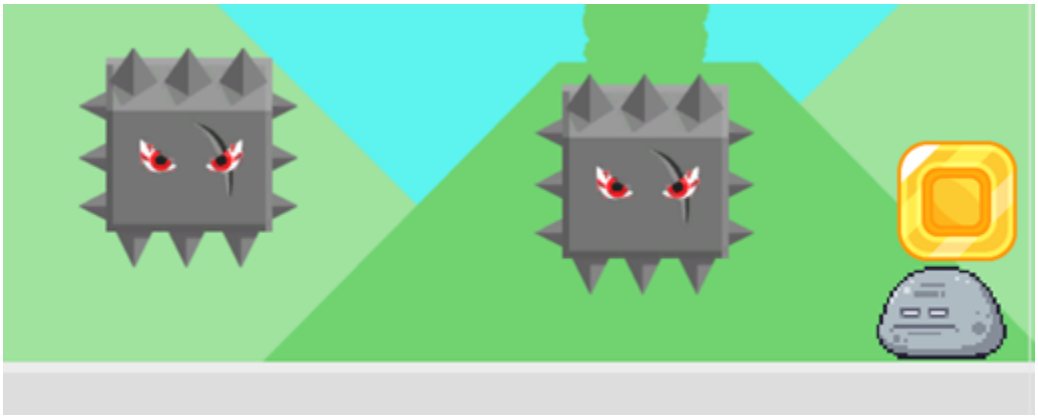

Figura 3.35: Nuevas animaciones para enemigos

3.3.8. Scene 1.8.4 (Nuevas funciones)

En este apartado se añade la bala creada previamente en el apartado 12.5, para lo cual se hace uso de el nuevo enemigo y se relaciona un script que se ejecute cada vez que el personaje se encuentra en el rango de este.

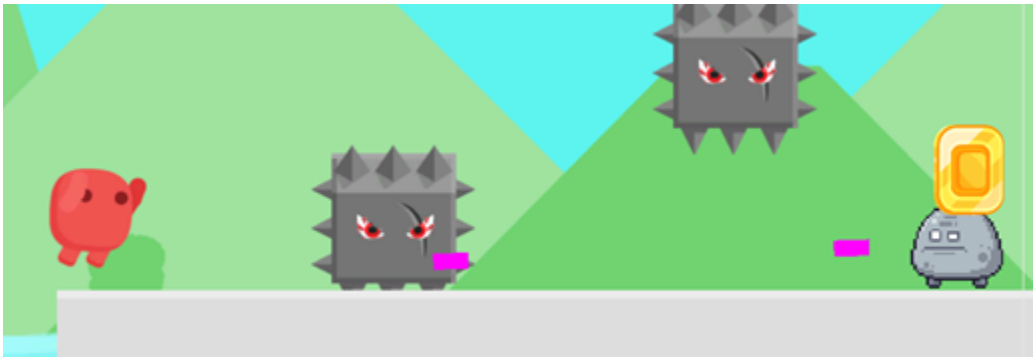

Figura 3.36: Funciones nuevas con la bala

3.3.9. Scene 1.8.5 (Daño en héroe)

El daño en el héroe es algo crucial ya que añade dificultad al juego, para esto se optó por el uso de tres factores en el canvas como lo son un slider, un text y una imagen, esto nos va a mostrar el estado actual del personaje y así mismo si estamos llegando a matar a susodicho.

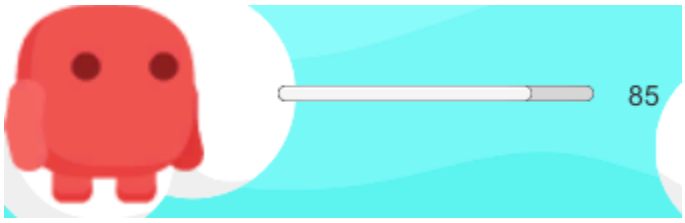

Figura 3.37: Barra de vida del héroe

3.3.10. Scene 1.8.6 (Animación de daño en héroe)

Al momento de contar con indicadores de daño se procede a añadir animaciones para este mismo, para lo cual el personaje principal cambia a un color más rojo cada vez que recibe una bala.

Figura 3.38: Animación de daño al héroe

3.3.11. Scene 1.8.7-1.8.8(Animación y Script de daño en enemigo)

En estas dos escenas se añade una nueva animación que surgirá cuando el enemigo sea golpeado en la cabeza por el héroe, lo cual le causará una hemorragia y lamentablemente la muerte instantánea, esto se maneja por medio de scripts en los cuales tanto el enemigo como el héroe cuentan con uno.

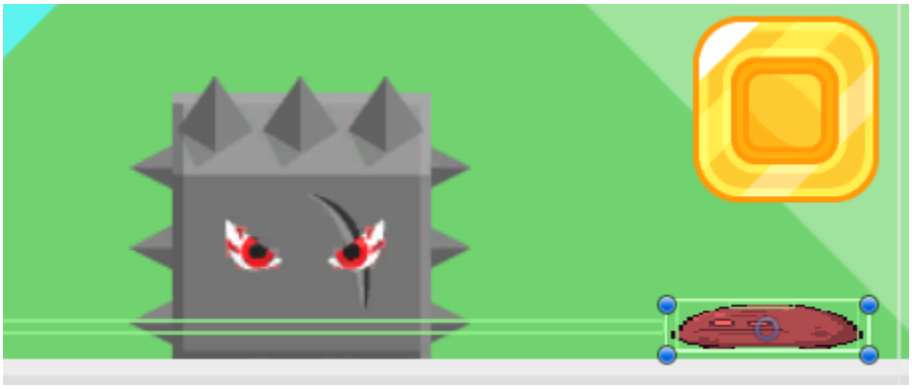

Figura 3.39: Animación de daño en enemigo

3.3.12. Scene 1.9.0 (Ajustes de daño e interacción en enemigos)

Escena pertinente para añadir interacciones con enemigos, en este caso se añade la animación para que cada vez que el personaje colisione con un top este pierda 5 de vida además, de generar una animación de daño y un pequeño salto. Otra cosa son los colliders de cada uno de los lados, con los cuales el héroe no tiene colisión pero si genera daño.

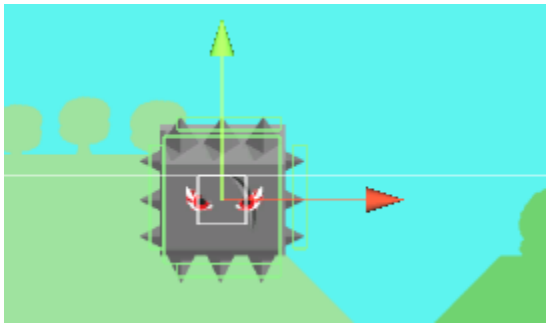

Figura 3.40: Box Collider para los enemigos

3.3.13. Scene 1.9.1 (Ajustes de daño e interacción en enemigos 2)

Para esta escena se ajustaron algunos errores del programa los cuales permitían al jugador pasar a través de el enemigo presentado en la anterior imagen, para lo cual se activó el elemento como Kinematic además de su desactivación de IsTrigger para lo cual activará las colisiones entre este enemigo y el héroe. Además de estas mismas propiedades para el otro enemigo, bola espinosa.

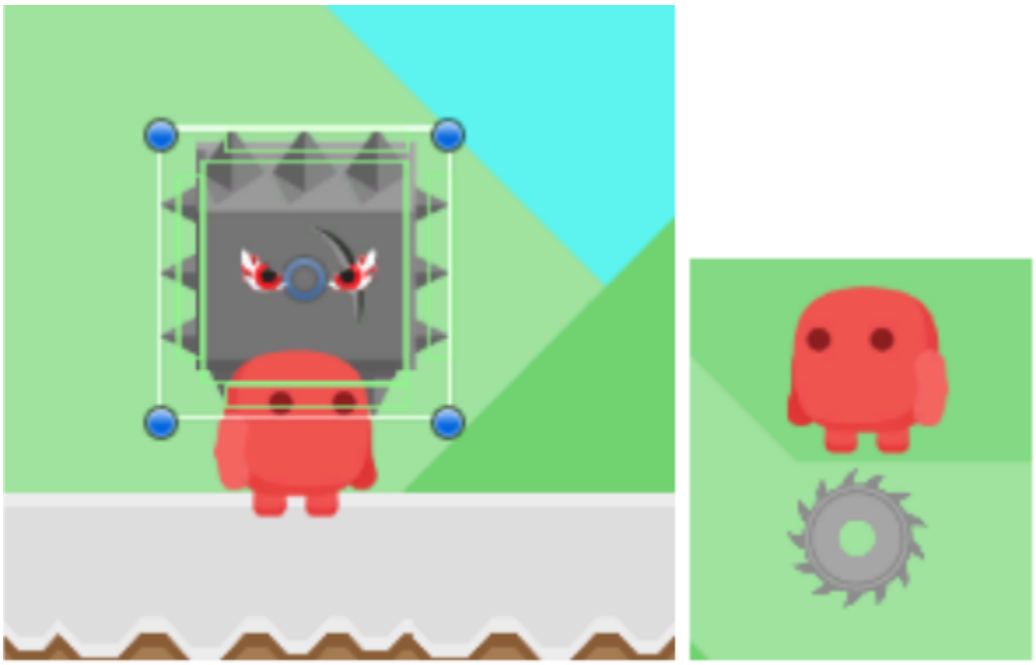

Figura 3.41: Interacción del Box Collider del enemigo con el héroe

3.3.14. Scene 1.9.2 (Asociación de sonidos al héroe y ambiente)

Los sonidos al héroe y en general a la escena se presentaron al momento de realizar diversas acciones, como lo son: saltar, hacerse daño o morir, además de esto se incluye un sonido de fondo para ambientar la escena.

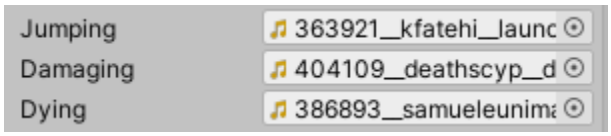

Figura 3.42: Asociación de sonido para el héroe

3.3.15. Scene 1.9.3 (Asociación de sonidos a enemigos)

Los sonidos a los enemigos se asociaron a partir de eventos los cuales se ejecutarán dependiendo de las interacciones que tengan con el personaje principal.

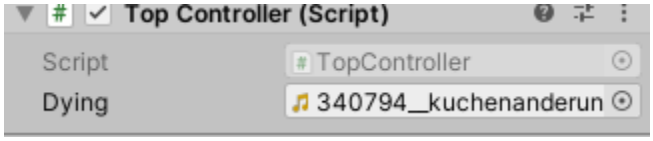

Figura 3.43: Asociación de sonido para los enemigos

3.3.16. Scene 1.9.4 (Asociación de sonidos y animaciones a la moneda)

La animación de la moneda puede ser tomada como un complemento, el cual motivará al jugador a conseguir una mayor cantidad de puntos en el transcurso del juego, esto permitirá que el jugador se sienta motivado y realice los niveles con el fin de obtener puntuaciones mayores.

Figura 3.44: Asociación de sonido y animación para la moneda

3.3.17. Scene 1.9.4 (Asignación de propiedad de puntos al Canvas)

Para la asignación de puntos se seleccionó una parte de la pantalla en la cual serán mostrados los puntos que se han obtenido, estos serán controlados por medio de un script el cual envía señales a los controladores y estos actualizan la puntuación.

Figura 3.45: Canvas con las propiedades de vida y puntos

3.4. Niveles

El funcionamiento de los niveles del juego es muy similar a otros juegos de esta misma temática, Blink, nuestro personaje principal, aparecerá en un mundo con ciertas plataformas y enemigos que dificultan el paso. El objetivo de Blink en cada nivel es encontrar un cofre y abrirlo mediante la tecla E, esto para dar paso al siguiente nivel, hay que tener en cuenta que en el transcurso de los niveles Blink podrá perder vida al igual que ganar puntos.

3.4.1. Menú

El menú principal cuenta con cuatro opciones de inicio como vienen siendo "Start" que permite dar inicio al juego desde el nivel 1, "Levels" que permite que el jugador seleccione el nivel desde el cual quiere iniciar con la penitencia de perder 10 de vida inicial, "Options" que es un apartado para las opciones de juego y "Exit" el cual permite salir del juego.

Figura 3.46: Menú principal

3.4.2. Nivel 1

El primer nivel o nivel de adaptación presenta a un enemigo, un premio y la meta, la función principal de este nivel es mostrar el funcionamiento de los controles y los enemigos presentes.

Figura 3.47: Nivel 1 del juego

3.4.3. Nivel 2

El segundo nivel al igual que el primero presenta la adaptación a los enemigos y las mecánicas como viene siendo el salto, se presentan dos enemigos que pueden ser asesinados pero cuentan con mecánicas diferentes.

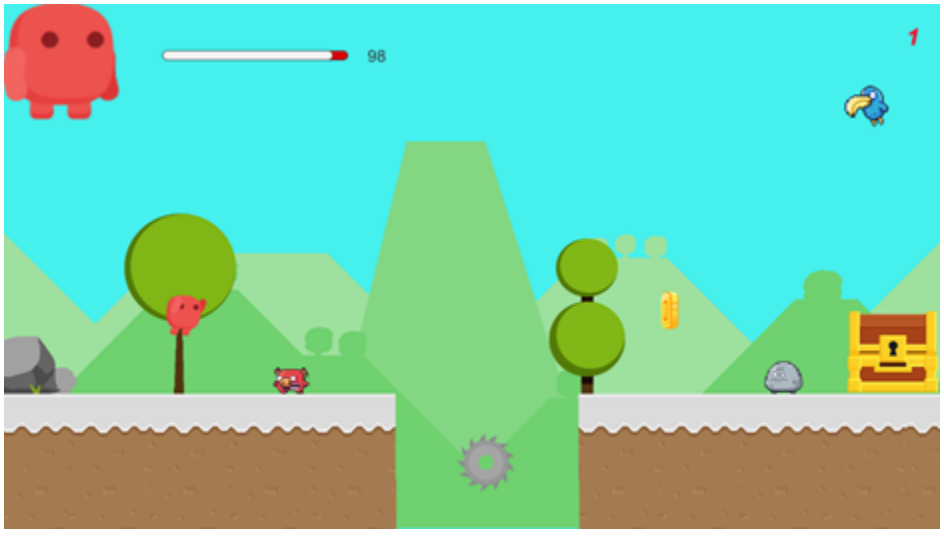

Figura 3.48: Nivel 2 del juego

3.4.4. Nivel 3

El tercer nivel al igual que el primero y el segundo presenta la adaptación a los enemigos y las mecánicas como viene siendo el salto en el aire, se presenta un enemigo inmortal con mecánicas diferentes a los anteriormente vistos.

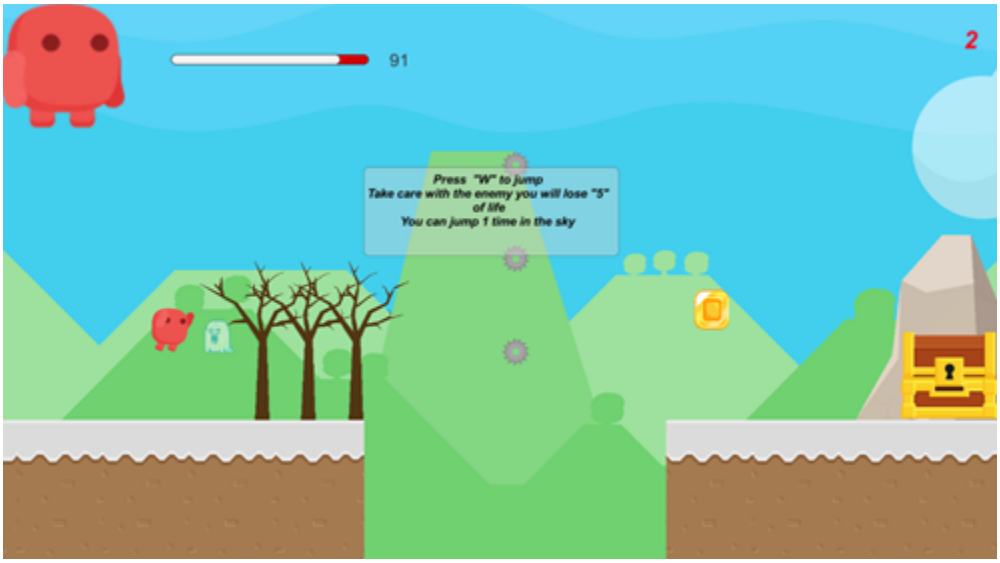

Figura 3.49: Nivel 3 del juego

3.4.5. Nivel 4

El cuarto nivel permite que el jugador ponga a prueba la mecánica de escalar a doble pared, este nivel no cuenta con enemigos que puedan moverse pero si hay obstáculos al pasar el nivel.

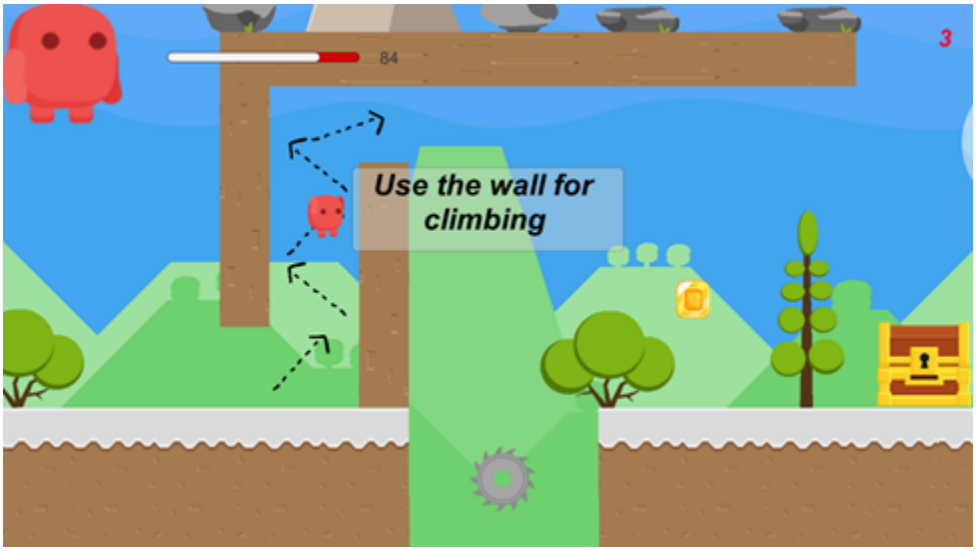

Figura 3.50: Nivel 4 del juego

3.4.6. Nivel 5

El quinto nivel presenta mayor complejidad tanto con los obstáculos como con los enemigos ya que no presenta ayudas y tiene que usar las habilidades que se adquirieron en los niveles anteriores.

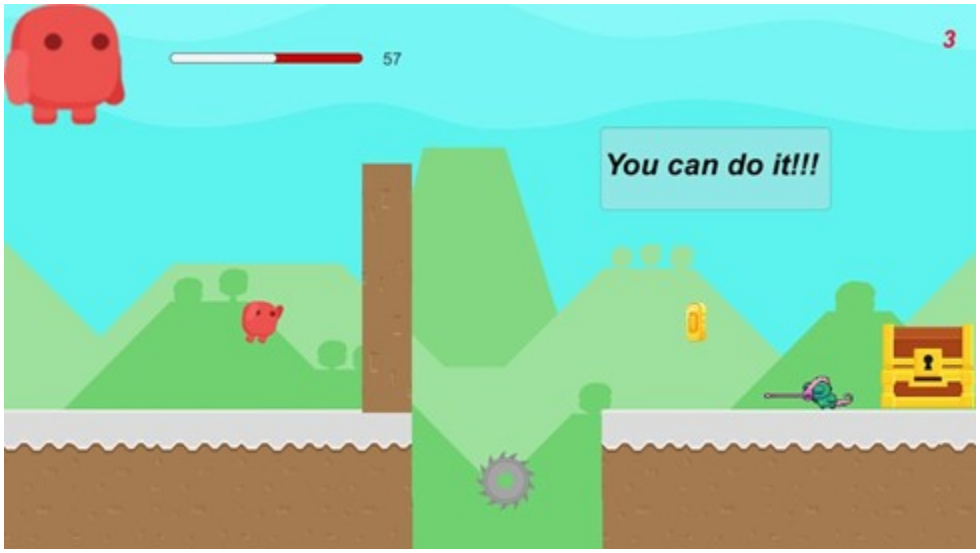

Figura 3.51: Nivel 5 del juego

3.4.7. Nivel 6

El sexto nivel al igual que los tres primeros presenta adaptación con el salto a la pared, o en su defecto la propiedad que permite saltar en el aire, este nivel es transitorio por lo cual su complejidad no es alta.

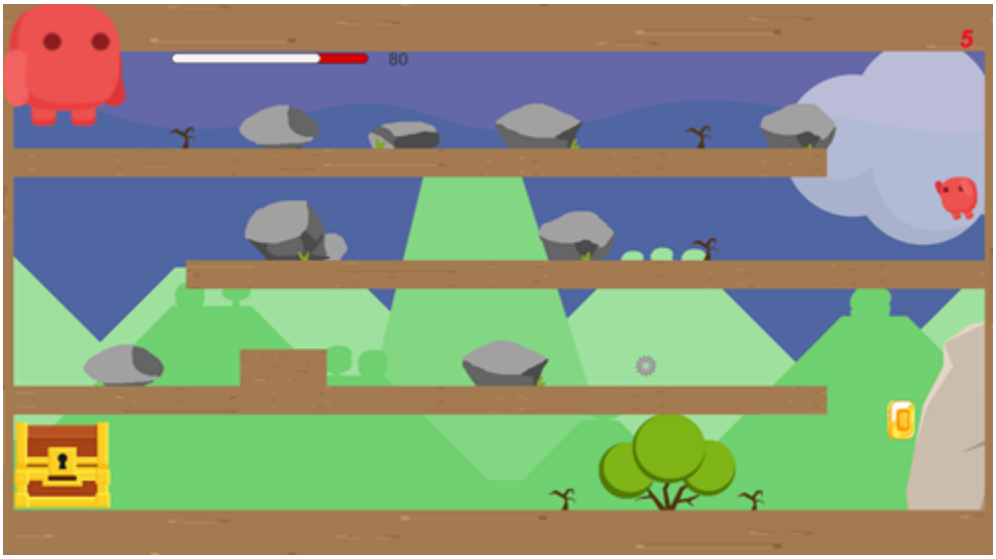

Figura 3.52: Nivel 6 del juego

3.4.8. Nivel 7

El séptimo nivel no presenta enemigos pero sí presenta alta complejidad en término de obstáculos ya que se necesita un dominio medio al momento de escalar y controlar al personaje.

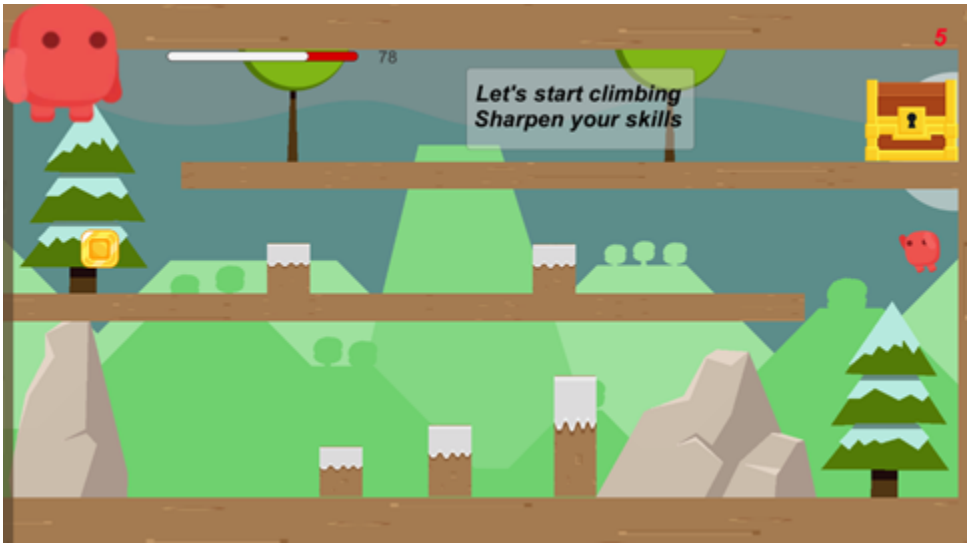

Figura 3.53: Nivel 7 del juego

3.4.9. Nivel 8

El nivel 8 al igual que el anterior no presenta enemigos que puedan afectar el rendimiento del jugador, en vez de esto muestra una recopilación de todo lo que se mostró en los niveles anteriores, para lo cual cuenta con un sistema de plataformas estáticas y móviles.

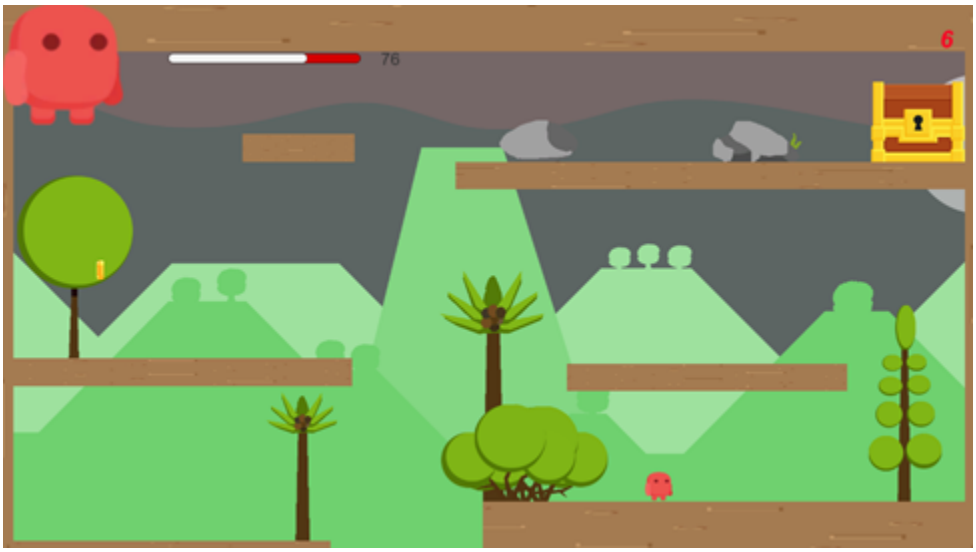

Figura 3.54: Nivel 8 del juego

3.4.10. Nivel 9

El noveno nivel es el nivel más complicado en el juego en términos de plataformas ya que se necesita mucha coordinación y maniobrabilidad al momento de escalar, la mayor parte de su complejidad está centrada en que al ser el noveno nivel el jugador

pudo haber perdido mucha vida para llegar hasta acá, por lo cual cada vez que falle en escalar perderá vida y tendrá que volver a iniciar.

Figura 3.55: Nivel 9 del juego

3.4.11. Nivel 10

El último nivel es una mezcla de todo lo visto en los anteriores, donde se implementa un trampolín y plataformas que repelen al jugador, en términos de complejidad es más sencillo que el nivel nueve pero presenta más elementos para poder pasar.

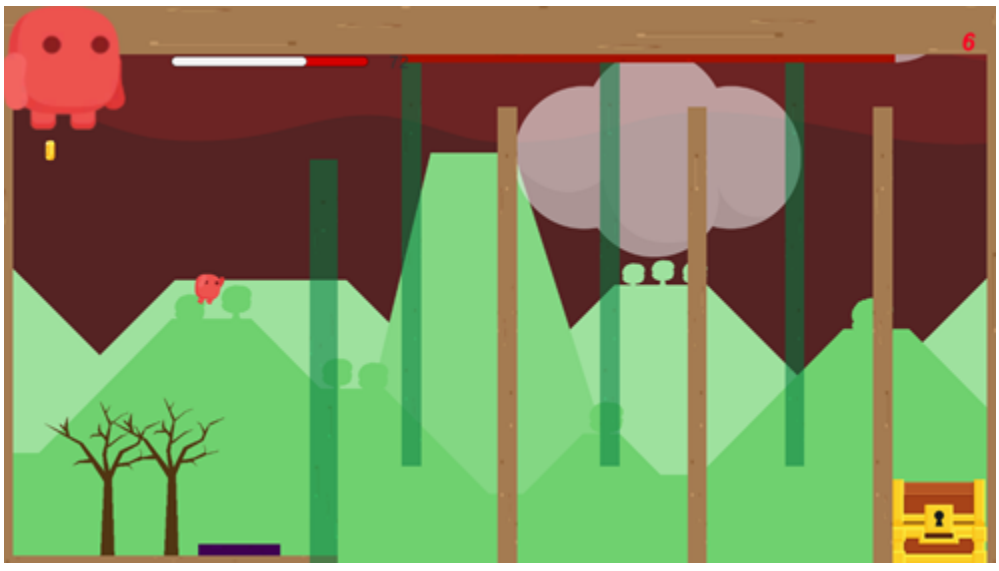

Figura 3.56: Nivel 10 del juego

3.4.12. Nivel de agradecimientos

El nivel de agradecimiento muestra al personaje principal en compañía de un letrero dando las gracias por jugar, en este nivel se encuentran dos casillas que el personaje puede seleccionar y con las cuales podrá dirigirse a su siguiente destino como lo son el menú principal o salir del juego.

Figura 3.57: Nivel de agradecimiento

3.4.13. Nivel Game Over

Cuando el jugador pierda toda su vida se presentará la última pestaña la cual permitirá volver al menú o salir del juego, también se puede mostrar este nivel si llega a fallar el nivel de agradecimientos.

Figura 3.58: Game Over

Capítulo 4

BlockChain

Blockchain es un sistema que facilita el manejo de información mediante cadenas de bloques que son distribuídas por diferentes equipos de cómputo, su principal éxito recae en la seguridad con la cual cuenta este sistema ya que hay dos de las razones fundamentales que hacen esto posible, como viene siendo el Hash, el cual es como un ID que tiene cada bloque de información y no puede ser modificado ya que cambia según la información que contiene, haciendo que los bloques no encajan con la secuencia y la otra razón es que hay muchos ojos sobre cada bloque lo cual genera mayor seguridad ya que se vuelve "sencillo" ver si alguien modificó el código generando una mayor confianza entre usuarios ya que son estos los que permiten que BlockChain funcione de esta forma.

Para la siguiente fase del proyecto se plantea el desarrollo del token, su implementación y la economía de estos mediante los tokenomics, además de la respectiva implementación como la Binance Smart Chain (BSC)

4.1. Tienda de Unity

Al igual que los diseños propios, el uso de diseños de terceros permite al usuario una amplia gama de opciones para elegir, Unity cuenta con su apartado de Asset Store en el cual se publican trabajos que pueden ser gratuitos o de pago, esto permitiendo el enfoque más que todo sobre la programación del juego en vez del diseño:

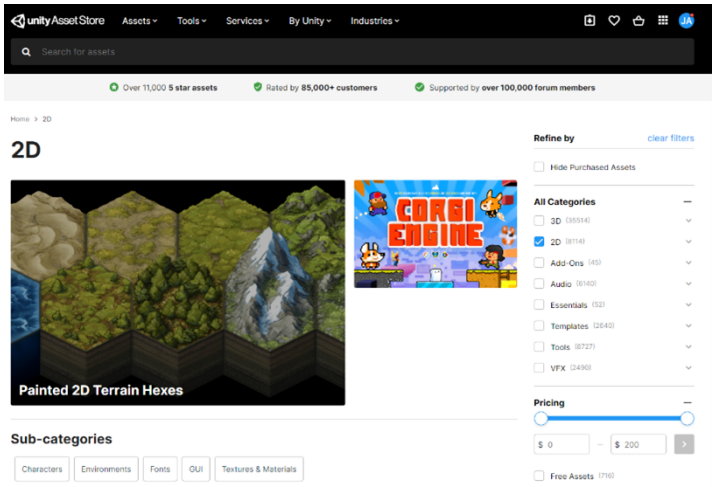

Figura 4.1: Unity Asset Store 2D

Una de las tantas funciones de la tienda de Unity es un Plugin para la manipulación de Ethereum el cual se puede encontrar como "_dLabs Lite: Ethereum Plugin" este plugin permite hacer intercambio de la moneda virtual, permitiendo así que mediante la aplicación de Unity hacer que el juego tenga las propiedades de los juegos NFT o token no fungibles, ahora bien, este plugin requiere una versión del programa actualizada(2019.4.16) con la cual no se cuenta (se cuenta con la versión 2019.3.6) por lo cual no se pudo profundizar en este tema, pero se tienen alojados los plugins en la carpeta de "Plugins" para una futura versión.

Capítulo 5

Alfa y proyecto

Para la estructuración del proyecto se presentará la Alfa la cual muestra el desarrollo tenido durante el semillero de Makers, primero que nada se incluyen los 10 niveles presentados anteriormente ya que cada uno refleja un punto que se ha trabajado, además de esto tiene escenas en el UI las cuales ayudan a una mejor interacción como lo vienen siendo el menú y el sistema de pause.

Hay que tener en cuenta que esta Alfa permitirá ver el proceso realizado y todos los componentes que se han utilizado en el transcurso del presente desarrollo, permitiendo interactuar con ciertos componentes de diferente complejidad que ayudan a presentar un producto claro y conciso, no obstante no se verá el trabajo de arte ya que al ser hecho de forma individual hay algunos componentes en los cuales no se pudo enfatizar. Para ver el proceso y sus componentes se puede hacer uso del proyecto completo el cual se encuentra a continuación.

A continuación se encuentran los enlaces donde pueden encontrar el juego y los archivos del juego.

Beta [7]: `https://github.com/Fenixfo/MakersUnity/tree/master/Start%20Game`

Proyecto [1]: `https://github.com/Fenixfo/MakersUnity/tree/master/Proyecto%20Blink2`

Scripts [6]: `https://github.com/Fenixfo/MakersUnity/tree/master/Proyecto%20Blink2/Assets/Scripts`

La Alfa consta de un menú principal en el cual se puede acceder a componentes como viene siendo el inicio del juego, los niveles, opciones de configuración y el botón de salida:

Figura 5.1: Menú principal

El apartado de niveles cuenta con la totalidad de niveles desbloqueados hasta el momento los cuales se presentan en un menú para que así los jugadores puedan acceder de forma más sencilla a estos.

Figura 5.2: Niveles del juego

El apartado de opciones cuenta con especificaciones como la vista de pantalla completa, la calidad de imagen, el volumen y el brillo, los cuales pueden ser modificados por el usuario y quedarán almacenados en memoria,

Figura 5.3: Opciones del juego

Cada nivel cuenta con un menú de pausa el cual permite volver al menú principal, además de la vida con la que cuenta el jugador y su puntuación

Figura 5.4: Menú de pausa

5.1. Sistema de vida

El sistema de vida es bastante sencillo ya que cuenta con 100 puntos de vida los cuales no se podrán regenerar a lo largo de todo el juego, la única forma de llegar a recuperar vida es mediante el uso de puntos los cuales pueden ayudar en la compra de las habilidades, pero esto no se da a conocer al jugador, ya que el tiene que descubrirlo por sí mismo. La forma de perder vida es mediante el paso de los niveles ya que cada nivel vale 2 de salud, los cuales se restaran del total cuando el jugador abra el cofre además de que si el jugador cae del mapa perderá 5 de vida volviendo al punto de

inicio del nivel, al igual que si es golpeado por un enemigo, pierde 5 de vida y continúa donde es golpeado.

5.2. Sistema de puntos

El sistema de puntos tienen un funcionamiento sencillo en el cual le otorga 1 punto por cada moneda recogida, estas monedas a simple vista parecen solamente algo para medir el rendimiento del jugador al pasar de los niveles pero tienen una función oculta la cual le regenera la vida al jugador restando puntos del total conseguido.

5.3. Sistema de niveles

El sistema de niveles como ya se mencionó anteriormente es algo desde lo cual se puede ingresar desde el menú principal, ahora bien para poder pasar entre niveles se necesita abrir un cofre que puede estar ubicado en cualquier parte del nivel, para esto el jugador tiene que superar los obstáculos y sobreponerse al cofre presionando la tecla E.

5.4. Sistema de movimiento

El sistema de movimiento cuenta con movimientos simples como vienen siendo mover a la derecha (flecha a la derecha o D), mover a la izquierda (flecha a la izquierda o A), saltar (flecha arriba o W) y abrir cofre (E). Además de esto cuenta con mecánicas extra propias del juego como vienen siendo:

5.4.1. Salto en el aire

Una de las mecánicas que se tiene es el salto en el aire, ya que el usuario cuenta con un salto el cual se puede implementar en cualquier momento, y será reiniciado al momento de tocar el suelo. Lo anterior hace que el jugador pueda saltar en el aire permitiendo así que se lleguen a lugarea alejados que sin esta mecánica sería imposible.

5.4.2. Escalado vertical

Se tiene implementado un sistema de escalado el cual funciona de tal forma que el usuario gana o recupera el salto al momento de tocar ciertas superficies como viene siendo el suelo y ciertos muros, lo cual permite golpear un muro, retirarse de este y saltar para repetir el proceso, ganando un bonus de salto.

5.4.3. Escalado horizontal

Se tiene implementado un sistema de escalado horizontal que puede ser asemejado a pegarse a los muros superiores como si fueran un pasamanos, este escalado solamente funciona en algunos niveles que lo requieran haciendo que el material de los techos sea diferente a los techos normales. Para poder escalar horizontalmente se implementa la misma lógica del escalado vertical, solamente que se hace uso de la gravedad para que el personaje caiga y vuelva a agarrar impulso de salto.

5.5. Enemigos

Los enemigos son importantes en cualquier video juego pero acá pasan a ocupar un lugar secundario, ya que durante cada plataforma a la cual el usuario se enfrente tendrá que poner a prueba lo anteriormente aprendido, haciendo que los enemigos sean fáciles de superar pero no pasará lo mismo con los obstáculos, el juego consta de los siguientes enemigos implementados:

5.5.1. Masa

Este enemigo presenta una funcionalidad que se repite en la cual sube hasta cierta distancia y se deja caer haciendo daño con las puntas que lo rodean, la principal característica es su forma cuadrada que lo hace parecer un cubo de daño, además de que el jugador estará perdiendo vida cada vez que se acerque a el enemigo.

Figura 5.5: Enemigo Masa

5.5.2. Roca

A diferencia de los demás enemigos la roca interactúa directamente con el usuario disparando hacia este y generando tanto daño como los demás (5 puntos), este enemigo tiene forma de roca y no se mueve, solamente va a detectar al jugador cuando este entre en su campo de visión, disparando así una vez.

Figura 5.6: Enemigo Roca

5.5.3. Puntas

Puntas tiene un parecido con Masa, solamente que redondo, este enemigo tiene una particularidad y es que se hace cada vez más pequeño para crecer de golpe, permitiendo así que el usuario solamente pase por ciertas zonas si tiene un buen tiempo de reacción para no perder vida.

Figura 5.7: Enemigo Puntas

5.5.4. Cerdo Furioso

El Cerdo Furioso camina por zonas específicas(se limita por barreras que el jugador no ve) este cerdo es pasivo pero genera daño si llega a ser tocado, al igual que la roca el cerdo puede ser asesinado si golpean su cabeza.

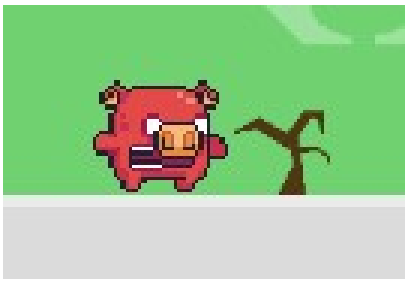

Figura 5.8: Enemigo Cerdo Furioso

5.5.5. Camaleón

El camaleón es un enemigo fijo el cual se mantiene constantemente lanzando golpes con su lengua, cada uno de estos golpes genera 5 puntos de daño. El camaleón puede

ser asesinado golpeando su cabeza.

Figura 5.9: Enemigo Camaleón

5.5.6. Fantasma

El fantasma es un enemigo inmortal levemente diferente a los otros inmortales ya que este recorre el mapa rápidamente golpeando al jugador si este se interpone en su camino.

Figura 5.10: Enemigo Fantasma

Capítulo 6

Conclusiones

A pesar de que el inicio de la creación es bastante complejo debido a que se está incurriendo en un formato algo desconocido, con el tiempo llega a ser bastante intuitivo, el desarrollo del juego permitió profundizar en temas como la programación orientada a objetos y la lógica en los videojuegos permitiendo así obtener un conocimiento útil para la carrera y el desarrollo de este campo.

Para el desarrollo en Unity hay bastantes tutoriales o documentos, pero estos no pueden ser tomados como una guía para el desarrollo personal, ya que cada diseño es único en su complejidad, al inicio del desarrollo se pueden tomar como guías cursos básicos para iniciar en Unity, tales como los que ofrece la misma plataforma o plataformas de terceros, aún así estos no pueden ser tomados como un tutorial, en vez de eso como una base, hay que tener en cuenta también que estos cursos no van a contar con toda la información de Unity, por lo cual es recomendable tomar más de uno.

Las monedas no son solamente algo que de puntos, también son una herramienta la cual permite guiar al jugador por el mapa, aún así el hecho de que es un juego bastante sencillo que permite visualizar cual va a ser el objetivo al inicio de este, dejando a las monedas más como un reto que como una necesidad.

Los trigger permiten la interacción con elementos los cuales no queremos que choquen con el jugador, esto permitiendo una mayor experiencia del usuario y así mismo suministrar más información al juego, tales como alertas, decoraciones, entre otros.

El desarrollo de un juego es un trabajo en conjunto que lleva muchos pasos, algunas veces sencillos y otras veces no tanto, en este caso en particular se tuvieron ciertas complejidades ya que el trabajo se produjo por una sola persona la cual no tenía conocimientos previos en este campo además de la programación, dejando así bacantes como la parte artística, sonora y creativa. Al trabajar de esta manera hay que tener conocimientos o al menos bases en cada uno de los aspectos, de lo contrario el desarrollo tomará más tiempo de lo esperado.

Bibliografía

[1] Build software better, together.

[2] Características y fases del modelo incremental.

[3] El sacerdote que enseña clases de religión con el videojuego 'minecraft'.

[4] Historia de los videojuegos.

[5] Los 25 motores de videojuegos gratis y de paga.

[6] MakersUnity/proyecto blink2/assets/scripts at master · fenixfo/MakersUnity.

[7] MakersUnity/start game at master · fenixfo/MakersUnity.

[8] Roll-a-ball (deprecated).

[9] ¿cómo ha crecido la industria de videojuegos en colombia? | BringITon.

[10] ¿qué es c# en programación y para que sirve?

[11] ¿qué es un cronograma? organiza los proyectos de tu empresa.

[12] ¿qué es y para qué sirve visual studio 2017?

[13] Academia Play. Historia de los videojuegos (1972-1983) parte i.

[14] Alva Majo. Unity para retrasados.

[15] Gius Caminiti. Unity VS unreal engine 4 [motores de videojuegos].

[16] F. Palazuelos. Qué son los motores gráficos y cuáles son los más populares.

[17] Semana. Día mundial del gamer 2021: Colombia tiene todo el potencial en el sector de videojuegos. Section: Empresas.

[18] Semana. La evolución de los videojuegos desde 1980. Section: Novedades.

[19] U. Technologies. Collaborate w/ other creators, get help, or just report a bug | community | unity.