Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Diseño y Desarrollo de Videojuegos

Título del Trabajo Fin de Estudios

Trabajo fin de estudio presentado por:	Willian Collaguazo Nicolás Chávez Isaac Chaves
Tipo de trabajo:	Videojuego
URL del repositorio de código	https://github.com/IsaacXB/TFM
URL del ejecutable	https://1drv.ms/u/s!AnwdJOv-pz7SkZh_jtBoTy0vxTgYIA?e=WBQMPl
URL del tráiler de presentación (Youtube)	https://drive.google.com/file/d/16OSqHrZvtp6XxzILCgC5NZJurcq-Mj23/view?usp=sharing
URL del tráiler de presentación (One Drive)	https://drive.google.com/file/d/16OSqHrZvtp6XxzILCgC5NZJurcq-Mj23/view?usp=sharing
Director/a:	Marcelo Fraile
Fecha:	01/28/2022

Resumen

El presente videojuego está orientado a demostrar todo lo aprendido durante el tiempo de estudio de la maestría y crear un prototipo de aventura y plataformas, con elementos de terror.

En el desarrollo del videojuego se utilizó la metodología ágil, llamada *Scrum*, que permite controlar y planificar el desarrollo del juego, por medio de entregables funcionales en poco tiempo, a través de un cronograma de actividades y revisiones mensuales, donde a cada integrante del equipo se le asignan las tareas de acuerdo con su experiencia.

La herramienta de desarrollo seleccionada fue el motor de Unreal, ya que se pueden lograr videojuegos de gran calidad y es uno de los más utilizados en la industria, además, tiene una gran ventaja, ya que por medio de *blueprints* permite la programación visual en bloques y en poco tiempo desarrollar funcionalidades complejas, sin la necesidad de escribir código, adicionalmente se puede programar en C++ y combinar ambas.

El videojuego consta de varios niveles, el jugador debe ir superando acertijos, enemigos, trampas y encontrar llaves u objetos que le permitan avanzar al siguiente nivel y eventualmente el final del juego.

Abstract

This video game is aimed at demonstrating everything learned during the study time of the master's degree and creating an adventure and platform prototype, with elements of terror.

In the development of the video game, the agile methodology, called Scrum, was used, which allows control and planning of the development of the game, through functional deliverables in a short time and a schedule of activities and monthly reviews, where each member of the team gets tasks assigned based on their experience.

The selected development tool was Unreal engine, since high-quality video games can be achieved and it is one of the most used in the industry, in addition, it has a great advantage, through blueprints allows visual programming in blocks and in a short time develop complex functionalities, without the need to write code, additionally you can program in C++ and combine them both.

The video game consists of several levels, the player must overcome puzzles, enemies, traps and find keys or objects that allows him to advance to the next level and eventually the end of the game.

Índice de contenidos

Contenido

1. Introducción 11
1.1. Ficha técnica 12
1.2. Público objetivo 12
1.3. Objetivos del juego 12
1.4. Referencias 13
2. El concepto del videojuego 18
2.1. Ambientación 18
2.2. Jugabilidad (Gameplay) 19
2.3. Mecánicas principales 19
2.4. Dinámicas principales 22
2.4.1. Acertijos 24
2.4.2. Objetivos 30
2.4.3. Trampas 31
2.5. Diseño artístico 33
2.6. Resumen de la Historia 34
3. Planificación del proyecto 34
3.1. Roles en el desarrollo 34
3.2. Actividades 36
3.3. Cronograma de actividades 36
3.4. Necesidades de recursos futuros 38
4. Diseño arquitectónico del software 39
4.1. Diseño arquitectónico 39

4.1.1. Motor de juego - Unreal Engine 40
4.1.2. Lenguaje de programación 46
4.1.3. Inteligencia artificial 47
4.1.4. Uso de interfaces y comunicación entre blueprints: 52
4.1.5. Inkscape / Adobe Illustrator 54
4.1.6. Controlador de versiones - Git / Perforce 54
4.2. Diagrama modular 55
4.2.1. Capa principal 55
4.2.2. Flujo para creación de objetos 3D 58
4.3. Casos de uso 59
4.4. Requisitos técnicos 60
5. GDD de alto nivel 61
5.1. Alcance del juego 61
5.2. Diseño visual 61
5.3. Gameplay y mecánicas 61
5.4. Control 62
5.5. Cámara 63
5.6. Personajes y enemigos 63
5.7. Niveles 65
5.8. Objetos e ítems 67
5.9. Diseño de la UI y Screen flow 67
6. Conclusiones 68
7. Anexos 68
7.1. Análisis de costo 68

7.2. Contenido externo, descargado de la tienda de Unreal Engine 70

7.2.1. Cartoon kid 70

7.2.2. Stitch creature pack 70

8. Postmorten del juego 71

9. Referencias bibliográficas 71

Índice de figuras

Figura 1: Limbo (Playdead, 2018). 14

Figura 2: Inside (Playdead, 2020) 15

Figura 3: Little Nightmares (Tarsier Studios, 2017) 16

Figura 4: Silent Hill (Hobbyconsolas, 2020) 17

Figura 5: Salto normal 20

Figura 6: Salto alto 20

Figura 7: Objeto con peso 21

Figura 8: Objeto sin peso 21

Figura 9: Personaje flotando 21

Figura 10: Personaje volando 22

Figura 11: Blueprint (Activar plataforma) 23

Figura 12: Diseño (Activar plataforma) 23

Figura 13: Funcionalidad (Estado plataforma) 24

Figura 14: Diseño (Puzzle a resolver) 24

Figura 15: Diseño (Puzzle resuelto) 24

Figura 16: Solución (Secuencia de colores) 25

Figura 17: Blueprint (Asignar solución) 25

Figura 18: Solución (Asignar color solución) 26

Figura 19: Blueprint (Actualizar código) 27

Figura 20: Estado (Objeto ganador) 28

Figura 21: Estado (Condición errada) 28

Figura 22: Estado (Condición correcta) 28

Figura 23: Estado (Blueprint referencia) 28

Figura 24: Objeto (Pirámide) 29

Figura 25: Objeto (Llave) 29

Figura 26: Objeto (Puerta) 29

Figura 27: Objeto (Palanca) 29

Figura 28: Objeto (Bloqueo) 29

Figura 29: Blueprint (Activar puerta) 30

Figura 30: Picos (Has Muerto)..... 31

Figura 31: Blueprint (Trampa de picos)..... 32

Figura 32: Trampa (Columnas móviles)..... 32

Figura 33: Diseño artístico (Escenario de nivel) 33

Figura 34: Inside (Meristation, 2017)..... 33

Figura 35: Diseño personaje (Personaje principal)..... 34

Figura 36: Diagrama de gantt (Flujo de actividades)..... 37

Figura 37: Diagrama de flujo (Flujo de funcionalidad) 39

Figura 38: Diseño (Escenario en unreal) 40

Figura 39: Sonido (Asignación de sonido en los pasos)..... 41

Figura 40: Físicas (Físicas en unreal) 42

Figura 41: Gráfico (Visualización de gráficos en unreal) 43

Figura 42: Blueprint (Movimiento del personaje)..... 44

Figura 43: Blueprint (Salto del jugador) 45

Figura 44: Blueprint (Deshabilitar colisión)..... 45

Figura 45: Blueprint (Muerte del personaje) 46

Figura 46: Código fuente (Class Character.h)..... 47

Figura 47: Keys (Comportamietos) 47

Figura 48: Tree (Árbol de comportamientos) 48

Figura 49: Tree (Comportamiento atacando) 49

Figura 50: Tree (Comportamiento buscando)..... 50

Figura 51: Tree (Comportamiento investigador) 51

Figura 52: Tree (Comportamiento mover a investigar)..... 51

Figura 53: Tree (Comportamiento estado patrullaje) 52

Figura 54: Tree (Comportamiento estado activo/inactivo)..... 52

Figura 55: Blueprint (Comunicación entre blueprints)..... 53

Figura 56: Blueprint (Comunicación entre blueprints)..... 53

Figura 57: Blueprint (Animación enemigo) 54

Figura 58: Blueprint (Personaje) 54

Figura 59: Git (Control de código fuente) 55

Figura 60: Classes (Blueprint de clases) 55

Figura 61: Blueprint (Personaje) 56

Figura 62: Personaje (Unrealengine, 2021)..... 57

Figura 63: Capas de juego..... 57

Figura 64: Menú inicio 58

Figura 65: Menú pausa 58

Figura 66: Diagrama de modelado (Personaje)..... 59

Figura 67: Movimiento (Flujo movimiento personaje) 59

Figura 68: Muerte (Flujo muerte personaje) 60

Figura 69: Teclado (Asignación de teclas)..... 62

Figura 70: Enemigo (Diseño propio de un enemigo)..... 64

Figura 71: Diseño enemigo (Diseño esqueleto y malla de enemigo) 64

Figura 72: Nivel 1 (Escenario hospital) 65

Figura 73: Nivel 2 (Escenario casa) 66

Figura 74: Nivel 3 (Escenario mazmorra) 66

Figura 75: Flujo de juego 68

Figura 76: Costo implementación 69

Figura 77: Tiempo de implementación 69

Figura 78: KickStarter 69

Figura 79: Personaje (Unrealengine, 2021) 70

Figura 80: Enemigos (Unreal Engine, 2021) 70

1. INTRODUCCIÓN

Los videojuegos están ligados al continuo desarrollo e innovación tecnológica de la actualidad, la llegada de celulares inteligentes, tabletas, consolas y ordenadores cada vez más potentes en su capacidad gráfica y de procesamiento, han provocado una gran evolución en la cantidad, variedad y calidad de los videojuegos. Es por esta razón que hoy en día la industria de los videojuegos es considerada una de las más importantes del mundo, ya que la mayoría de la población tiene acceso a alguna de estas herramientas, así como acceso a internet, por lo que se convierten en clientes potenciales, a un click de poder descargar y jugar a nuestro juego.

Este gran crecimiento de la industria ha llegado acompañado del desarrollo de varias herramientas tecnológicas que facilitan la creación de juegos y permiten a pequeñas empresas poder alcanzar grandes mercados por medio de una distribución digital con costos muy bajos e incluso gratuita en muchos casos.

A partir de los conocimientos adquiridos en el máster de diseño y desarrollo de videojuegos, el objetivo es crear un prototipo jugable que pueda llamar la atención del público, que gustan de videojuegos de suspenso y generar sensaciones de terror al tratar de resolver diferentes retos que se presentan al jugador, inspirado en juegos de desarrolladores independientes como estudio PlayDead (Inside y Limbo) y Tarsier (Little Nightmares).

La propuesta plantea el juego con perspectiva de desplazamiento lateral con gráficos en 3D, en algunos escenarios también se va a utilizar la cámara en tercera persona.

El juego está ambientado en las pesadillas que sufre nuestro personaje, quién es un pequeño adolescente que perdió a su madre en un trágico accidente. El personaje sufre de estrés post traumático y tiene memorias reprimidas que complican considerablemente su capacidad de dormir y razonamiento, ya que siente que debe escapar de grandes peligros que su mente es capaz de generar avanzando por el escenario, saltando, manipulando objetos y evadiendo sus pesadillas.

1.1. FICHA TÉCNICA

A continuación, detalla la ficha técnica del videojuego a realizar.

Tabla 1. *ficha técnica del videojuego.*

Género	Plataforma, Aventura
Jugadores	Uno
Potenciales plataformas	PC - Xbox
Perspectiva	Desplazamiento lateral (Perspectiva 2D) y en tercera persona.
Tipo de gráficos	3D
Motor de juego	Unreal Engine
Modelado 3D	Blender y 3DS Max
Jugabilidad	Esconderse, escapar e interactuar con objetos del juego para resolver acertijos presentes en los escenarios

Fuente: American Psychological Association, 2020e.

1.2. PÚBLICO OBJETIVO

El juego está orientado al público entre los 16 a 35 años ya que la historia que se plantea y los escenarios pueden ser inapropiados para audiencias más jóvenes.

Jugadores que gusten del género de plataformas, terror y suspenso se identifican como nuestro público objetivo.

1.3. OBJETIVOS DEL JUEGO

El juego tiene como objetivo superar obstáculos y acertijos hasta lograr escapar del lugar que su mente ha sido capaz de generar, todo empieza en un lugar parecido a su habitación, pero en una realidad alternativa y distorsionada, dando lugar a entes y escenarios creados en la mente del personaje y que representan los miedos del personaje, los cuales tendrá que superar y encontrar la salida de esta pesadilla.

La experiencia que se espera transmitir al jugador es la sensación de estar perdido y atrapado. Esto se logrará con pasillos pequeños y parecidos entre sí. También se utilizará iluminación limitada, efectos de sonido y música para transmitir este sentimiento.

1.4. REFERENCIAS

En una primera aproximación a la problemática por desarrollar, se analizaron los siguientes juegos como referencias e inspiración, estos pertenecen a géneros de plataformas y aventuras de terror, con la intención de tomar estos de base ya que han sido exitosos en este tipo de videojuegos y que se consideran como referentes de la comunidad.

Estas referencias incluyen a 2 juegos desarrollados por la empresa independiente danesa Play Dead, creadora de los juegos Inside (2016) y Limbo (2010). Adicionalmente, el juego Silent Hill (1999) desarrollado por Konami y Little Nightmares (2017) desarrollado por Tarsier Studios.

Estos juegos nos permiten tomar algunas de sus características principales y sobresalientes, que vamos a detallar a continuación:

- **Limbo - Juego de referencia**

Limbo es un videojuego de lógica y plataformas, con vista lateral 2D. La trama trata sobre las experiencias de un niño que busca a su hermana en un entorno siniestro.

Con el fin de crear un ambiente de tensión, el juego utiliza una estética en escala de grises con diferencia en el valor para generar capas de profundidad. Gracias a esto, le da una tonalidad oscura y de soledad por la cual el jugador deberá avanzar para superar el nivel.

El personaje puede moverse de izquierda a derecha, saltar, trepar, empujar o jalar objetos que le ayudan a alcanzar lugares con facilidad.

Con la ayuda del escenario sombrío, hay ciertos enemigos u obstáculos que son difíciles de observar. Algunos enemigos se pegarán al jugador hasta que llegue a una fuente de luz.

Mientras avanza el juego, el jugador se encontrará con acertijos que requieren el uso de otras mecánicas como magnetismo o gravedad.

Su estilo de juego se podría describir como “prueba y muerte”, es decir, el jugador debe experimentar con objetos dentro del escenario para resolver acertijos, pero no recibe prácticamente ningún tipo de ayuda visual, aunque sí presenta algunos efectos de sonido que ayudan a identificar si el jugador está haciendo lo correcto o no. Con esto se busca que el jugador deba ir probando con diferentes elementos disponibles en el escenario, hasta

encontrar la respuesta al acertijo, en la mayoría de estas pruebas el jugador termina muriendo cuando se equivoca.

A continuación, podemos observar imágenes de referencia del juego Limbo.

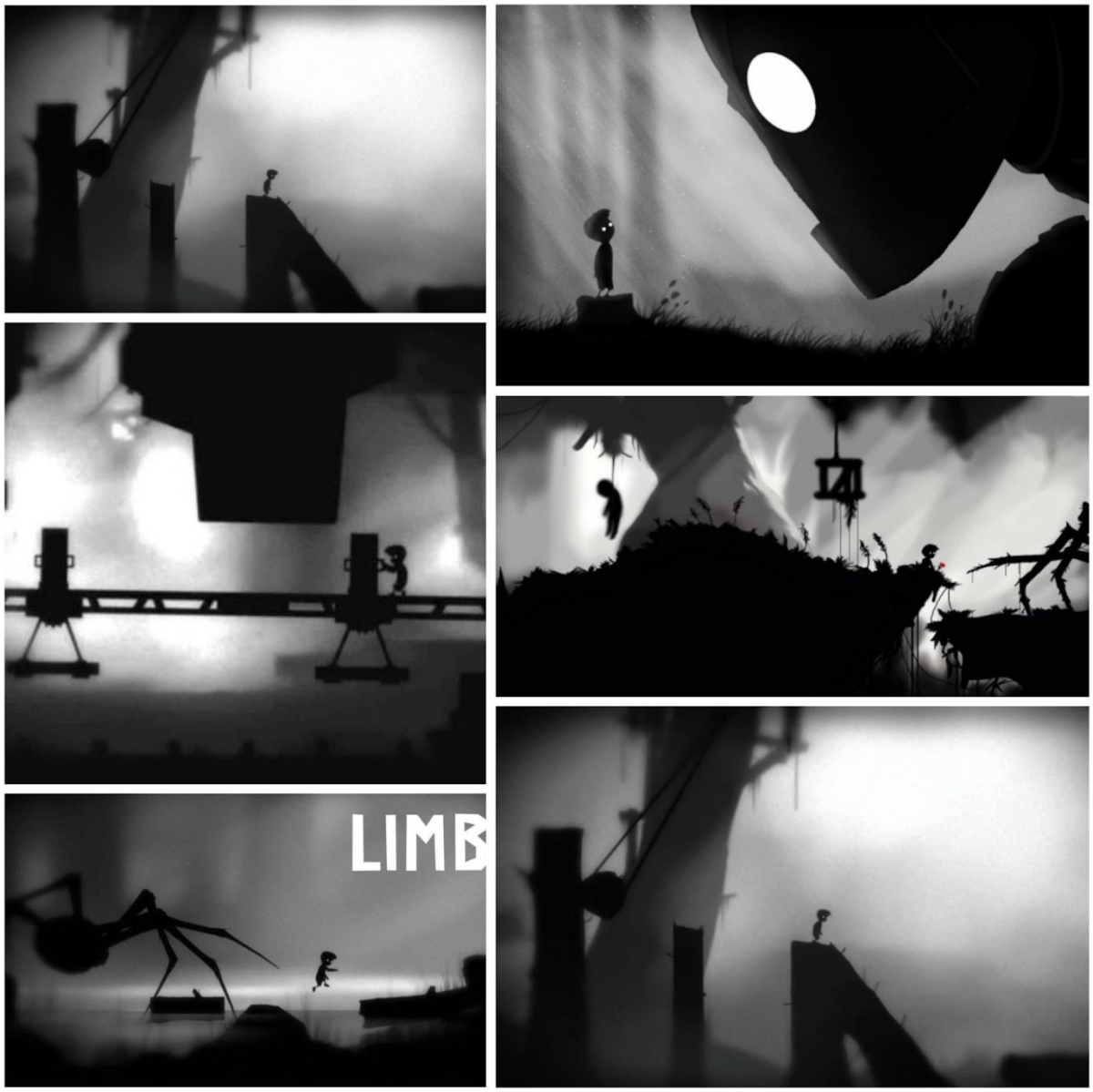


***Figura 1**: Limbo (Playdead, 2018).*

- **Inside - Juego de referencia**

Inside es un videojuego desarrollado por Playdead en 2016 y es el sucesor en Limbo. La trama en esta entrega no es explícita y deja libre interpretación al jugador.

La historia se desarrolla en un mundo postapocalíptico, se infiltra en un laboratorio donde están experimentando con humanos.

Las mecánicas son muy similares a Limbo, el personaje debe moverse de izquierda a derecha para resolver acertijos y avanzar. El juego tiene una estética 2.5D, es decir movimiento lateral, pero con gráficos 3D, los modelos son tipo lowpoly.

***Figura 2**: Inside (Playdead, 2020)*

- **Little Nightmares - Juego de referencia**

Little nightmares es un videojuego desarrollado por Tasier Studios en 2017 y trata de una niña atrapada que tiene que escapar evadiendo a las criaturas que se encuentran.

En esta entrega se ven gráficos con más detalle haciendo que los enemigos sean más perturbadores.

Comparte las mecánicas con Limbo e Inside, movimiento lateral resolviendo acertijos y poder sobrevivir a los enemigos. De esta entrega se tomará como referencia el tipo de movimiento y la ejecución de los acertijos. Así como el hecho de utilizar entornos 3D con vista lateral.

***Figura 3**: Little Nightmares (Tarsier Studios, 2017)*

- **Silent Hill - Juego de referencia**

Silent Hill es un juego de terror psicológico desarrollado por Konami en 1999. Hace uso de gráficos en 3D y una cámara que sigue al jugador. El juego trata de un hombre que pierde a su hija al entrar al pueblo de Silent Hill el cual está desierto y plagado de monstruos. A medida que el personaje avanza, se enfrenta a distintos enemigos para encontrar a su hija.

Los monstruos son una manifestación de los miedos y culpas del personaje, esto se puede ver en el enemigo final “Pyramid head” que actúa como un juez por los errores que ha cometido el personaje. Este aspecto se tomará como referencia para los enemigos y entes que genere la mente del personaje principal del videojuego.

***Figura 4**: Silent Hill (Hobbyconsolas, 2020)*

Analizando estos juegos de referencia, se logra determinar una serie de elementos particulares de este tipo de juegos, que serán parte importante de nuestro trabajo, por ejemplo, podemos observar que el uso de personajes vulnerables sin entrenamiento en

combate genera una sensación de vulnerabilidad e identifica al jugador con el personaje por ser más real y genera una conexión emocional.

Por esta razón, en nuestro juego, el personaje no es capaz de atacar a los enemigos, únicamente tiene la capacidad de esconderse, evadir y superar los enemigos utilizando su agilidad y analizando el comportamiento de los enemigos, al mismo tiempo que debe resolver acertijos para superar los diferentes niveles.

Otra característica que podemos identificar es el uso de colores fríos también ayuda al ambiente lúgubre y reforzar sentimientos de abandono, depresión y pérdida, así como utilizar ambientes oscuros y presentar al jugador ante escenarios de tensión y adversidad.

Un elemento muy importante que también se identifica, es que los enemigos que utilizan especialmente en Silent Hill son representaciones de conflictos internos de los personajes dando más información sobre los mismos, así como el uso de sonidos y audio que ayuden a generar sentimiento de inmersión. Estas son características que también se implementan en nuestro juego.

2. EL CONCEPTO DEL VIDEOJUEGO

2.1. AMBIENTACIÓN

El videojuego está ambientado en la mente de nuestro personaje y su imaginación, se busca recrear las pesadillas y temores que sufre nuestro personaje.

Todos los niveles iniciaran en el cuarto del personaje, el cual empezará como un lugar vacío y color con tonalidad roja demostrando la soledad y falta de emociones en el mundo que lo rodea.

Al salir de la habitación el escenario cambia a un lugar más oscuro y lúgubre, cada nivel estará ambientado en un lugar relacionado con los miedos del personaje y los enemigos serán sus pesadillas con forma de entidades que buscan al jugador. Una vez que el jugador salga de la habitación se encontrará con amenazas que recorren sus sueños.

La iluminación será limitada y proporcionada por pequeños focos que ayudan al jugador a encontrar el camino y objetos importantes. Adicionalmente, ciertos objetos proporcionan iluminación como un vehículo accidentado y aún prendido.

El ambiente estará cubierto de neblina que ayudará a cubrir partes del mapa que el jugador no puede acceder aún.

2.2. JUGABILIDAD (GAMEPLAY)

El objetivo del juego es escapar de las pesadillas de nuestro personaje. El jugador deberá interactuar con diferentes objetos para resolver acertijos y poder avanzar al siguiente nivel y tendrá que utilizar las mecánicas básicas como caminar, correr y saltar para superar obstáculos presentes en el escenario, adicionalmente tendrá que utilizar el sigilo para evitar ser visto por nuestros enemigos.

Existen varias condiciones que pueden provocar la muerte de nuestro personaje como las siguientes:

- **Caer al vacío.**
- **Activar alguna de las trampas existentes en el escenario.**
- **Un enemigo atrapa al jugador.**

Por otra parte, la condición de victoria se da cuando se desbloquea el recuerdo al final del escenario logrando superar los obstáculos como trampas, buscar entre los objetos que se encuentran en el escenario.

2.3. MECÁNICAS PRINCIPALES

Las mecánicas básicas del juego serán moverse, saltar, esconderse, activar y empujar objetos que le servirán al jugador para resolver acertijos o puzzles para escapar o sobrevivir a sus sueños.

Mientras el jugador descubre sus memorias reprimidas, tendrá más control sobre sus sueños desbloqueando habilidades como:

- *Saltar más alto*: Al descubrir un pequeño control sobre su mente, el jugador podrá saltar más distancia y tener una caída lenta.

***Figura 5**: Salto normal*

***Figura 6**: Salto alto*

- *Quitar el peso a objetos*: El personaje podrá hacer que ciertos objetos pierdan su peso y empiezan a flotar para subirse a ellos y alcanzar lugares más altos para descubrir objetos o habitaciones ocultas y resolver los acertijos que necesita para llegar a la memoria de ese nivel.

***Figura 7**: Objeto con peso*

***Figura 8**: Objeto sin peso*

- *Flotar*: con más memorias desbloqueadas el personaje podrá flotar por tiempo limitado.

***Figura 9**: Personaje flotando*

***Figura 10**: Personaje volando*

2.4. DINÁMICAS PRINCIPALES

Las dinámicas principales serían:

Cruzar los cuartos de forma sigilosa, búsqueda de pistas u objetos que le sirvan para poder avanzar en el nivel.

También existen acertijos de memoria, al finalizar el acertijo obtendrá recompensas como llaves, piezas u objetos que le sirvan para avanzar en el nivel.

A continuación, explicamos las dinámicas implementadas en nuestro juego:

- **Sigilo**

El juego se ha diseñado de forma que los enemigos pueden percibir sonidos, ver objetos y tomar decisiones en base a lo que ha percibido, en caso de detectar al jugador, el enemigo cambiará a estado agresivo y perseguirá al jugador con el fin de atacar, para evitar esto, el jugador deberá caminar despacio y agachado para evitar ruidos que llamen la atención del enemigo, así como esconderse detrás de objetos, para evitar que sea visto por el enemigo. En caso de ser visto, el jugador deberá correr e intentar perder al enemigo de su capacidad sensorial. En la sección 4.1.3 de inteligencia artificial, se explica más a detalle la implementación que hemos realizado para que los enemigos se comporten de esta forma.

- **Plataformas**

Se utilizarán 2 blueprints principales, 1 switch que activa plataformas que se mueven de 1 punto a otro y la plataforma que se mueve.

  - **Activar Plataforma**

Este blueprint cuenta con una caja de colisión y un evento que se activa al colisionar con otro objeto, en este momento se valida si es el personaje y en caso de serlo, se activa la plataforma de referencia y se reproduce el sonido de activación.

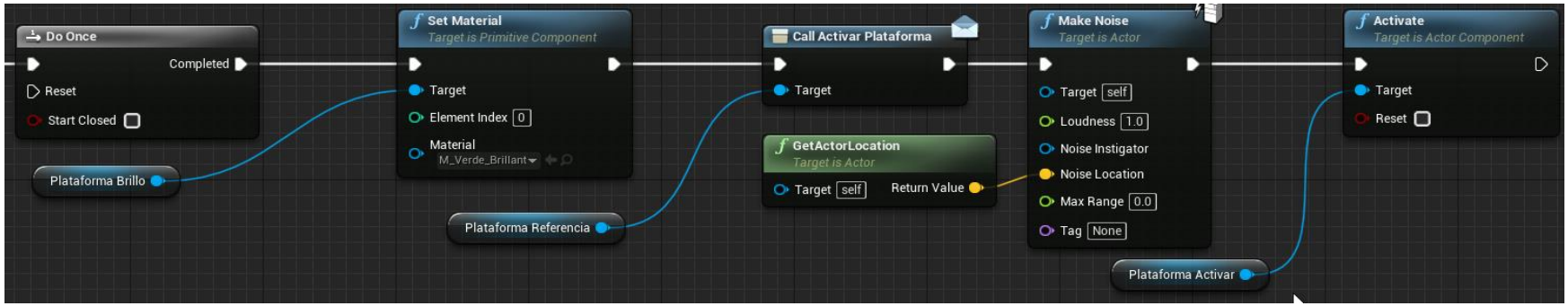


***Figura 11**: Blueprint (Activar plataforma)*

El blueprint de activación (1) tiene asignada la plataforma de referencia (2) que debe activar, como se puede observar en la siguiente imagen.

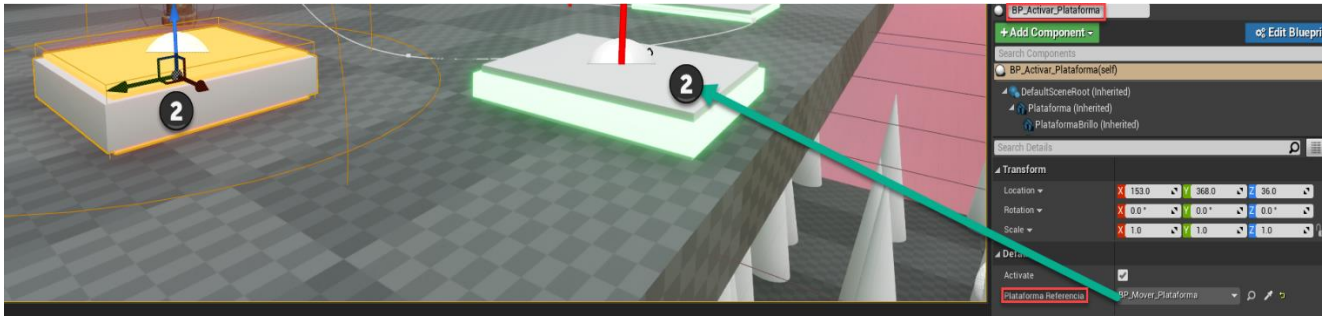


***Figura 12**: Diseño (Activar plataforma)*

- **Mover Plataforma**

Este blueprint cuando se encuentra en estado activo, se encarga de mover la plataforma de un punto de inicio (1) a un punto final (2), como podemos ver en la imagen de referencia.

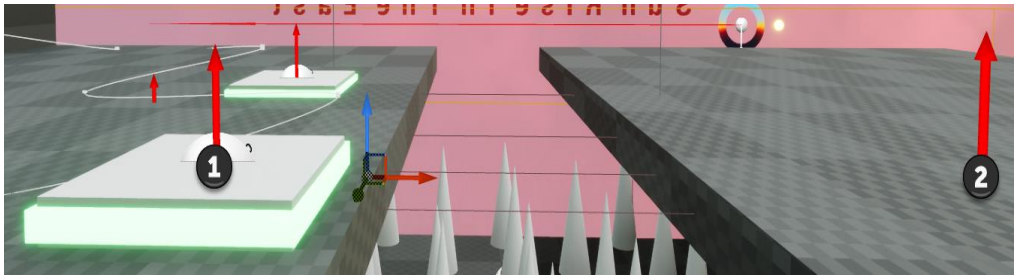


***Figura 13**: Funcionalidad (Estado plataforma)*

2.4.1. ACERTIJOS

El jugador deberá resolver ciertos acertijos para poder avanzar en el juego a siguientes niveles, hemos implementado 2 acertijos, que vamos a detallar a continuación:

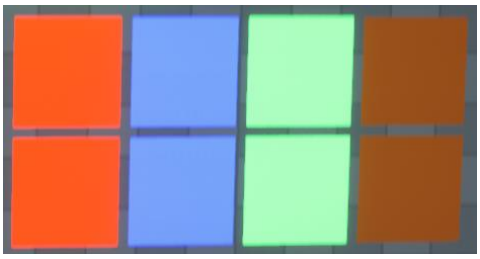
- **Activar objetos en la secuencia correcta**

La idea de este acertijo es que el usuario, debe activar los colores en la secuencia correcta, en este caso de ejemplo, sería 1-rojo, 2-azul, 3-verde y 4-marrón, que se puede observar en la fila superior de la siguiente imagen de referencia.

La idea, es que dentro del juego se va a encontrar alguna imagen o pista, que le ayude al jugador a determinar el orden en que se deben activar los colores.

***Figura 14**: Diseño (Puzzle a resolver)*

Conforme va activando cada color en la secuencia correcta, la fila inferior mostrará los colores



que ha activado el jugador de forma correcta. Si el jugador se equivoca, el acertijo se reinicia. Una vez solucionado el acertijo correctamente, el blueprint procederá a abrir la puerta de referencia que tenga asignado en el nivel.

***Figura 15**: Diseño (Puzzle resuelto)*

Adicionalmente, podemos configurar fácilmente el orden del acertijo, al asignar el orden de los colores de la siguiente forma.

***Figura 16**: Solución (Secuencia de colores)*

En la siguiente imagen podemos ver la función encargada de validar y activar cada color, se usa otra función llamada activar color solución, que se puede ver en la segunda imagen.

Asignar Solución, su función principal es ir actualizando las variables de la solución del acertijo activadas por el usuario.

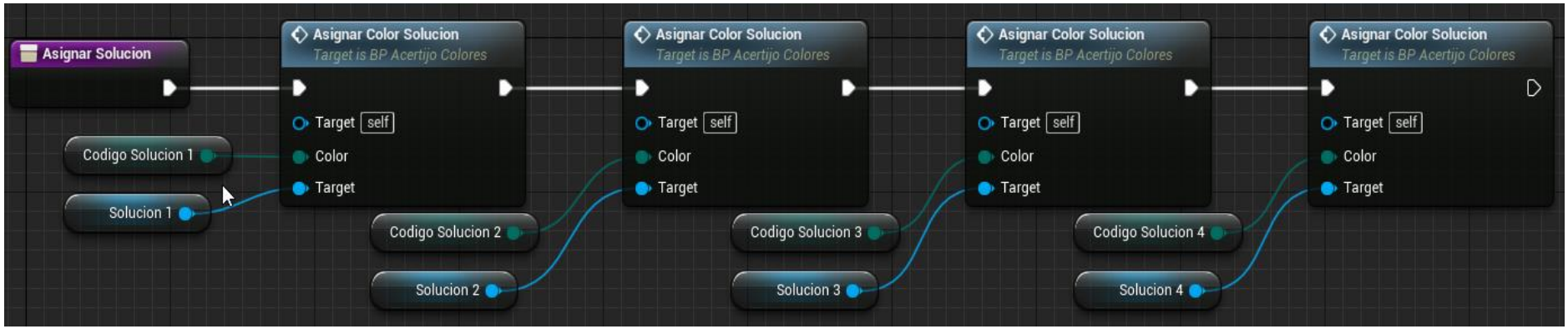


***Figura 17**: Blueprint (Asignar solución)*

- **Asignar Color:** Encargado de actualizar los colores que se han activado correctamente.

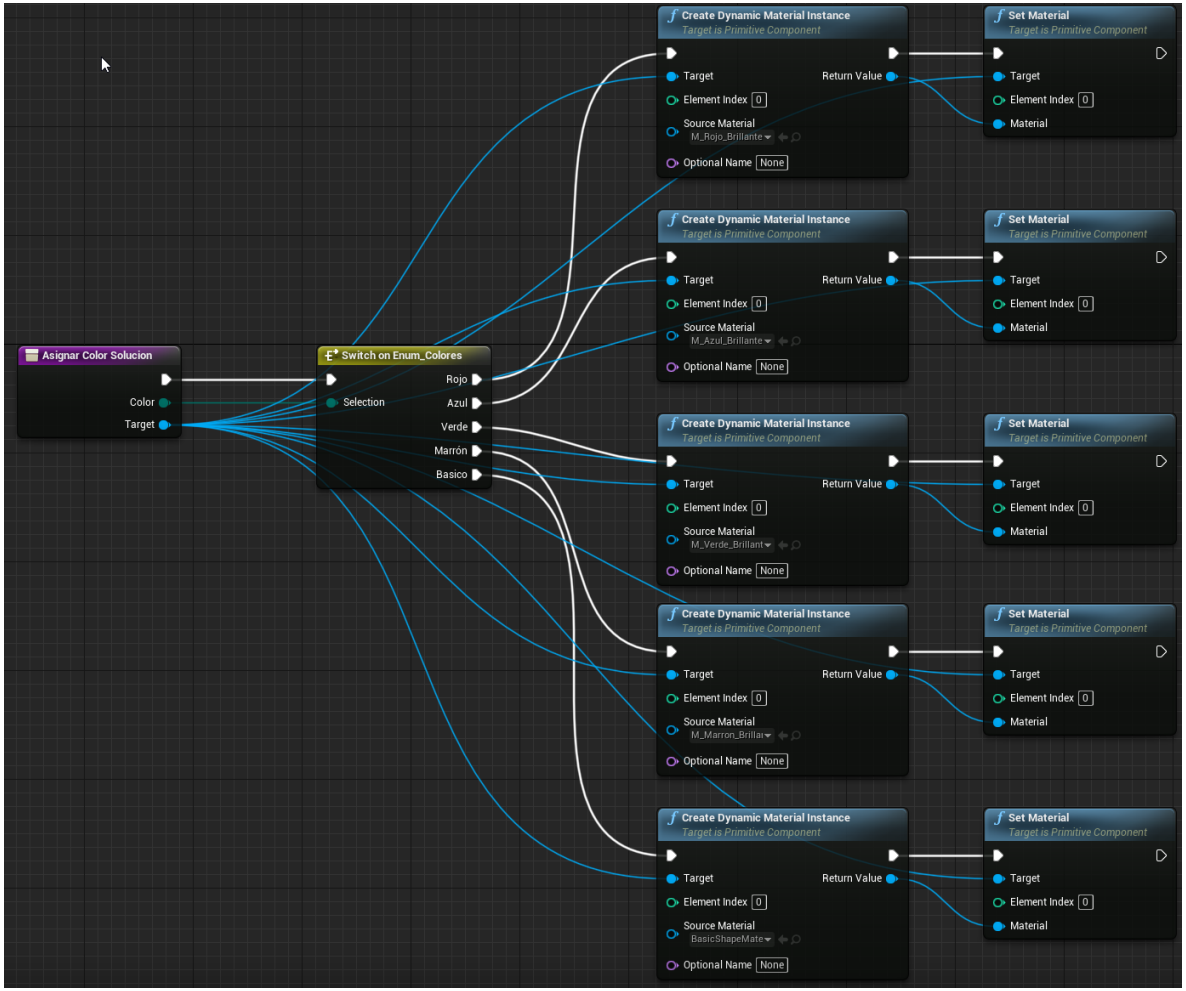


***Figura 18**: Solución (Asignar color solución)*

Dependiendo del color seleccionado, así será el material asignado.

- **Actualizar Código:** Finalmente esta función valida que color asignado sea el correcto y llama la función asignar color.

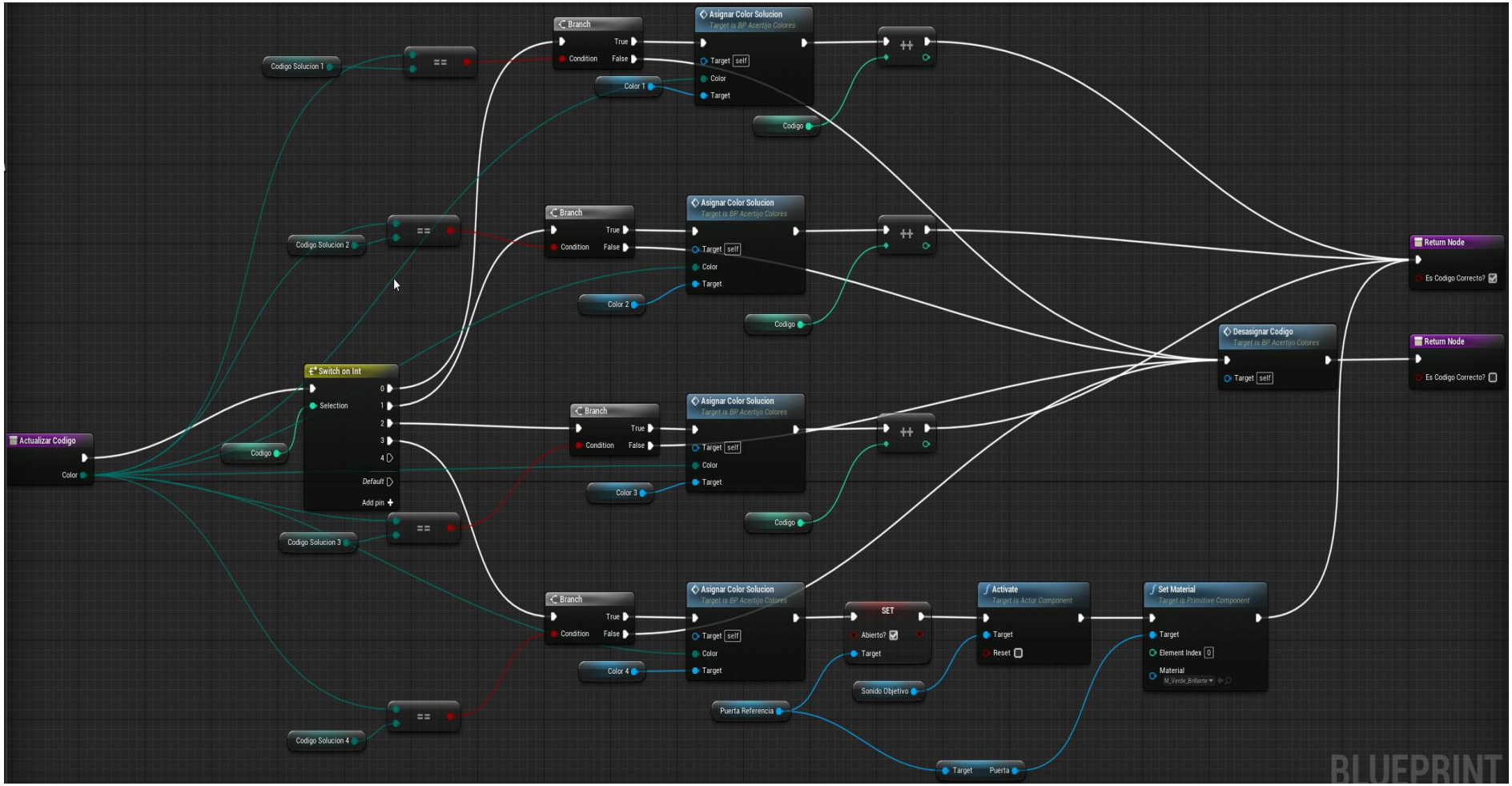


***Figura 19**: Blueprint (Actualizar código)*

- **Mover objetos a lugar específico**

Este acertijo se basa en que tendremos una serie de cubos movibles, que el usuario debe empujar y llevarlos a un punto en el piso que valida si el cubo es correcto o ganador, para esto utilizamos una variable pública, que fácilmente permite asignar el cubo ganador al diseñador de niveles.

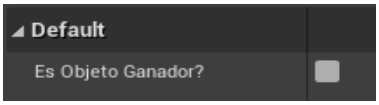


***Figura 20**: Estado (Objeto ganador)*

Si se encuentra activo este valor booleano, el cubo es el ganador.

Cubo no ganador. Color rojo, no resuelve el acertijo

***Figura 21**: Estado (Condición errada)*

Cubo ganador. Color verde, resuelve el acertijo y abre la puerta de referencia.

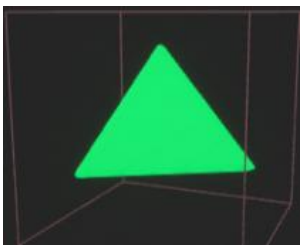
***Figura 22**: Estado (Condición correcta)*

Asignación de blueprint de objeto a activarse.

***Figura 23:** Estado (Blueprint referencia)*

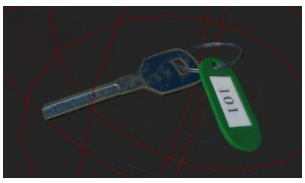
- **Objetos que se utilizan en los niveles**

Dentro de los diferentes niveles, podremos ir tomados objetos que interactúan con el personaje, estos objetos son activadores para poder abrir puertas y poder avanzar en los niveles.



Al tomar esta pirámide sirve para poder activar la puerta final del nivel.

Figura 24: *Objeto (Pirámide)*



Esta llave puede abrir puertas que se encuentren cerradas, estas puertas se asignan cuando se está diseñando el nivel.

Figura 25: *Objeto (Llave)*

Este tipo de puertas se encuentran bloqueadas, por lo que se necesita de una llave para poder abrir.

Figura 26: *Objeto (Puerta)*

Esta palanca sirve para poder destruir bloqueos que existan en las puertas.

Figura 27: *Objeto (Palanca)*

Bloqueo de puertas, este objeto se puede destruir con la palanca.

Figura 28: *Objeto (Bloqueo)*

2.4.2. OBJETIVOS

Otra dinámica que se ha implementado en el juego es la de que el jugador debe encontrar ciertos objetos para desbloquear puertas o desbloquear accesorios o habilidades.

- **Encontrar objeto**

Este blueprint, es un objeto que tiene una caja de colisión, que, al entrar en colisión con un objeto, se valida si es el personaje y en caso afirmativo, procede a desbloquear una puerta u objeto de referencia.

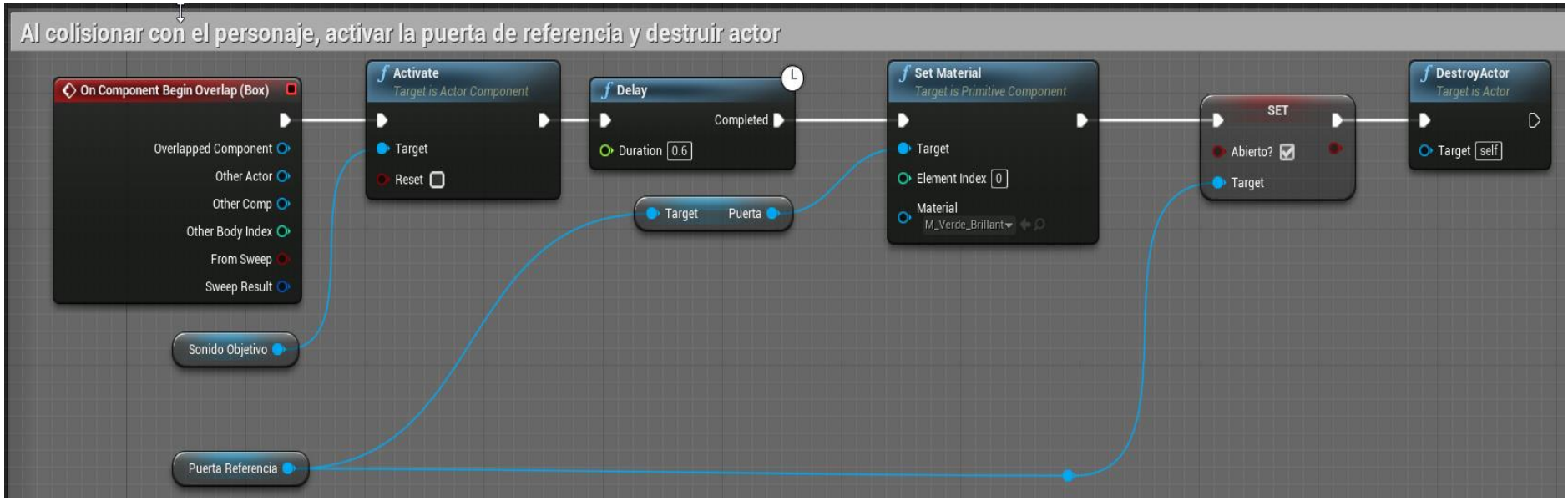


***Figura 29**: Blueprint (Activar puerta)*

- **Llaves**

Las llaves son necesarias para abrir puertas que inicialmente se encuentran cerradas, una llave puede abrir 1 o varias puertas, pero no necesariamente abren todas las puertas, en los niveles se encuentran diferentes llaves, cada una tiene asignadas las puertas que abre específicamente.

- **Palanca**

La palanca permite quitar madera que bloquea el acceso a nuevas áreas dentro del nivel.

2.4.3. TRAMPAS

Otro elemento importante que vamos a desarrollar en el juego, son trampas que se van a encontrar en el escenario, con el fin de dificultar el avance del jugador.

- **Picos**

La idea de esta trampa es que cuando el jugador deba saltar de una plataforma a otra, en caso de no llegar, el jugador cae sobre la trampa de picos y por ende muere.

***Figura 30**: Picos (Has Muerto)*

Para implementar esta trampa, el blueprint tiene una caja de colisión, la cual, al entrar en contacto con un objeto, valida que sea el jugador y en caso afirmativo, procede a notificar la muerte del jugador en pantalla, activa el sonido de muerte, deshabilita el movimiento del jugador, y permite reiniciar el nivel al presionar enter.

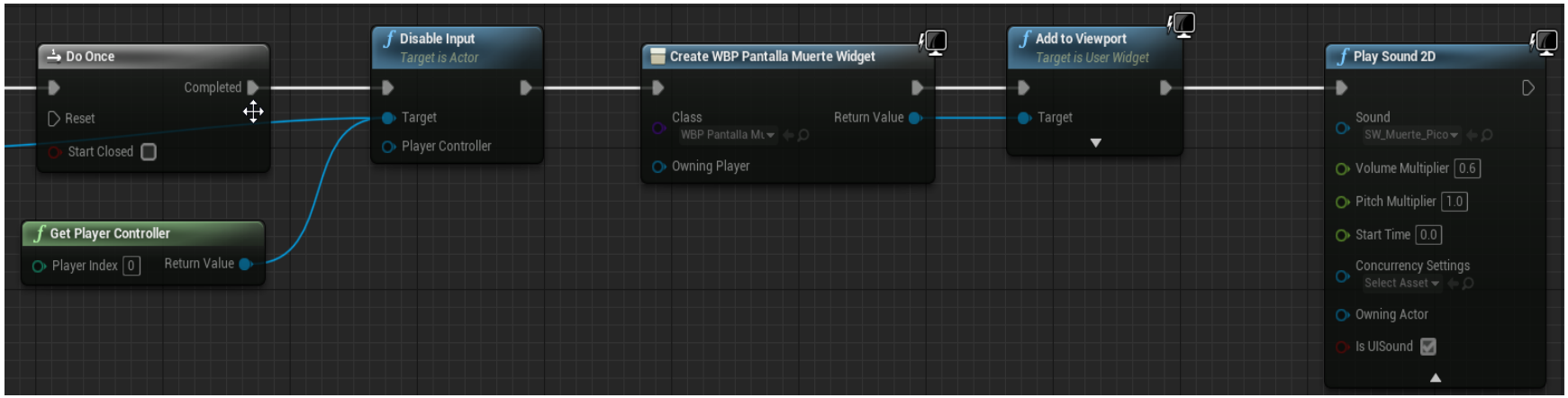


***Figura 31**: Blueprint (Trampa de picos)*

- **Columna movible con picos:** Similar a la trampa de picos, pero la principal diferencia, es que en este caso tenemos una columna que puede moverse dentro del escenario de un punto a otro, girando sobre su propio eje y que, al entrar en contacto con el jugador, procede a notificar la muerte del jugador en pantalla.

***Figura 32**: Trampa (Columnas móviles)*

2.5. DISEÑO ARTÍSTICO

El diseño artístico está enfocado a un ambiente lúgubre y surreal representando la etapa en la que se encuentra el personaje, donde predomina la oscuridad, el jugador debe ser sigiloso, el personaje es representado por un niño, con un rostro triste, temeroso, que se encuentra desesperado de salir de este sitio.

***Figura 33**: Diseño artístico (Escenario de nivel)*

Se utilizará el *low poly* para generar recursos eficientemente y mezclar con iluminación para dar el ambiente deseado como se puede ver a continuación:

***Figura 34**: Inside (Meristation, 2017)*

En uno de los escenarios el personaje deberá encontrar la memoria en un hospital en el cual estuvo atrapado por su estado de salud. En este ambiente se utilizarán espacios cerrados para dar la sensación de claustrofobia que siente el personaje en este lugar.

2.6. RESUMEN DE LA HISTORIA

El juego está ambientado en las pesadillas que sufre nuestro personaje, un adolescente que perdió a su madre en un trágico asalto y desde entonces sufre de diferentes trastornos mentales que le atormentan su imaginación, afectando considerablemente su capacidad de dormir y razonamiento.

El personaje empieza su viaje despertando de sus pesadillas sin saber porque no puede dormir con normalidad. Su madre aparece seguida en sus sueños y él puede verla fuera de ellos también sin saber que ya lleva tiempo fallecida.

Gracias a este evento él genera un mundo nuevo en la mente donde debe buscar la manera de poder sobresalir el impacto psicológico que tuvo a la muerte de la madre y pueda volver a la realidad.

***Figura 35**: Diseño personaje (Personaje principal)*

3. PLANIFICACIÓN DEL PROYECTO

En la planificación del desarrollo del video juego, se plantea de la siguiente manera.

3.1. ROLES EN EL DESARROLLO

- Diseñador de niveles y programación. (Isaac Chaves)

En este proyecto el rol principal fue de programador y diseñador de niveles, mediante el uso de blueprints como lenguaje de programación.

Tareas desarrolladas están:

- Mecánicas básicas de juego
  - Movimiento, salto del personaje y animaciones.
- Creación y programación de acertijos
- Inteligencia artificial
  - Rutas de patrullaje
  - Implementación de capacidad visual y auditiva
- Diseño de niveles
  - Dormitorio tenebroso
  - Tutorial 1 y 2
  - Mazmorra
- Cinemática de introducción
- Implementación de sonidos FX y música.

- Diseñador de personajes. (Nicolás Chávez)

Este rol se usó para el diseño e implementación de un enemigo, también realizó nuevos niveles y nuevas mecánicas a partir de los blueprints ya diseñados, también realizando una ambientación realista al nivel.

Tareas desarrolladas están:

- Creación del nivel 2 (casa).
- Acertijo de luces.
- Búsqueda de llaves.
- Ubicación de enemigo.
- Mecánica de palanca.

- Diseñador de mecánicas de juego y acertijos. (Willian Collaguazo)

El rol que fue usado fue de diseñador de nivel, donde utilizando los blueprints ya diseñados anteriormente, se inició a dar forma a los diferente acertijos y puzles que tiene cada nivel, también ambientando de una forma tétrica cada escenario, ubicando de manera estratégica cada objeto, enemigo, etc.

Tareas desarrolladas están:

- Creación del nivel 1 (hospital).
- Búsqueda de llaves.
- Ubicación de varios enemigos.
- Ubicación de cuchillas en movimiento.

3.2.ACTIVIDADES

Las actividades principales fueron divididas en 4 etapas principales.

- Diseño de niveles
- Inteligencia Artificial
- Mecánicas de juego y acertijos.
- Programación de personaje y cámara.

Estas actividades se realizan en paralelo para ir avanzando en el desarrollo del video juego más rápido y poder corregir errores en el caso que se dieran en el desarrollo.

3.3. CRONOGRAMA DE ACTIVIDADES

De acuerdo con todas las actividades que son necesarias para el desarrollo de un videojuego, estás se dividieron en diferentes hitos, para poder asignar de mejor manera a cada integrante lo que tiene que ir desarrollando e implementado.

Por lo que al implementar cada uno de los niveles se asignó a cada integrante que nivel debe implementar y finalizarlo.

Estos niveles deben contener, la ambientación, los enemigos, acertijos, mecánicas de juego, etc.

Por lo que se implementó el siguiente cronograma para la implementación del videojuego.

	Nombre	Duracion
1	**Fase inicial**	46 days
2	Creación del proyecto en Unreal	1 day
3	Creación de nivel de prueba	15 days
4	Implentación de IA enemigos	5 days
5	Implementación de blueprint de personaje	5 days
6	Implementación de plueprints a utilizarse (objetos)	5 days
7	Implementación de puzles	10 days
8	Implementación de mecánicas	5 days
9	**Nivel 1**	20 days
10	Creación de escenario nivel 1	4 days
11	Ambientación escenario 1	4 days
12	Jugabilidad nivel 1	7 days
13	Mejoras nivel 1	5 days
14	**Nivel 2**	20 days
15	Creación de escenario nivel 2	4 days
16	Ambientación escenario 2	4 days
17	Jugabilidad nivel 2	7 days
18	Mejoras nivel 2	5 days
19	**Nivel 3**	20 days
20	Creación de escenario nivel 3	4 days
21	Ambientación escenario 3	4 days
22	Jugabilidad nivel 3	7 days
23	Mejoras nivel 3	5 days
24	**Documentación**	66 days
25	Desarrollo de la documentación	66 days

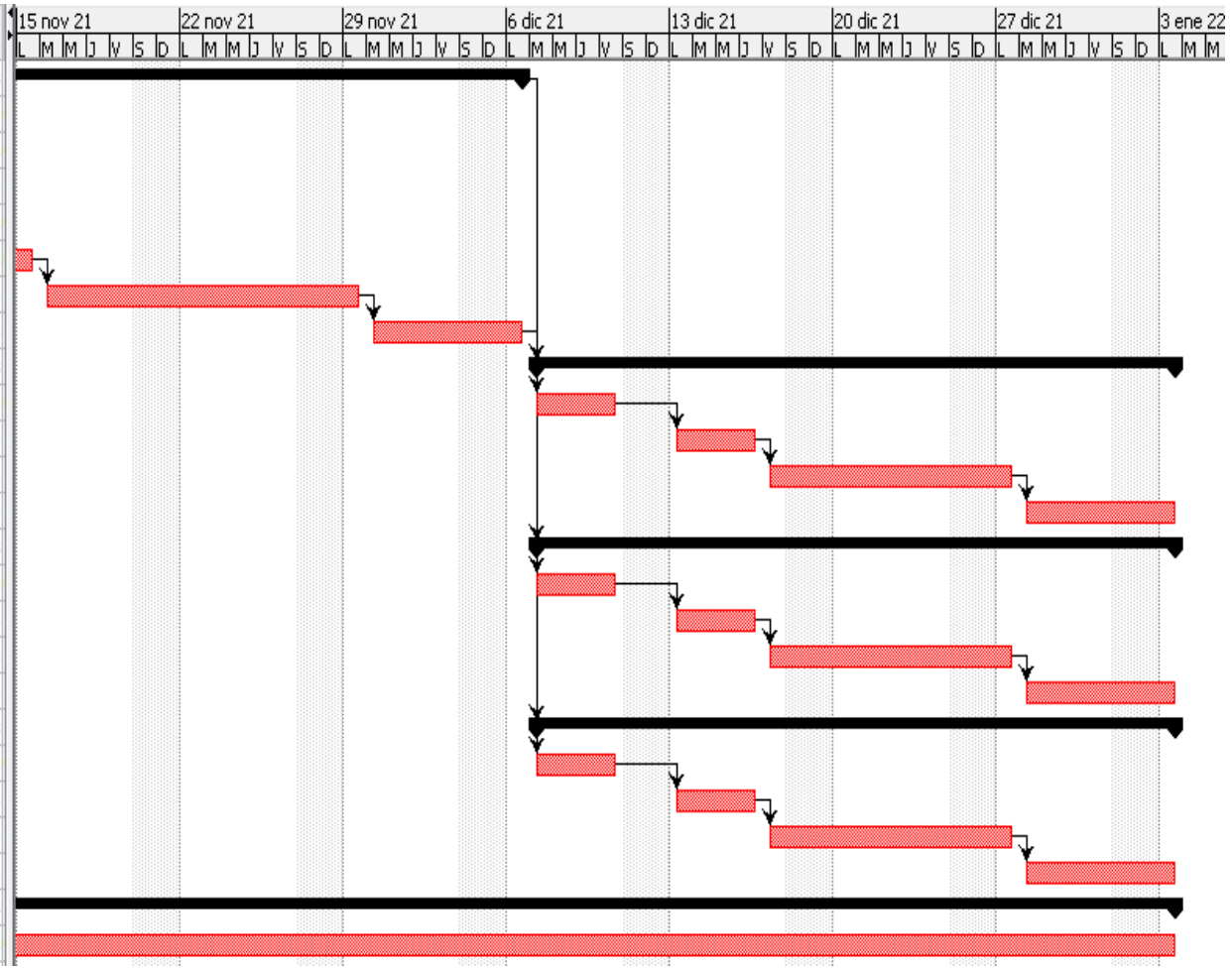


***Figura 36**: Diagrama de gantt (Flujo de actividades)*

3.4. NECESIDADES DE RECURSOS FUTUROS

Los recursos que se necesitarán a futuro serían: edición de videos, soundtracks, nuevos artes para los escenarios, añadir más personajes, mecánicas mucho más llamativas.

También se requiere una inversión monetaria para el desarrollo del videojuego.

Coste aproximado para el desarrollo ($ dólares), tiempo aproximado de desarrollo de 3 a 4 meses:

Actividad	**Coste**	
Desarrollo de escenario	$	6,000.00
Desarrollo de personajes	$	7,000.00
Desarrollo de mecánicas	$	7,000.00
Desarrollo de soundtracks	$	2,000.00
Equipos de computación	$	5,000.00
Licencias de software a utilizar	$	4,000.00
Total, de personas a desarrollar		3
Total:	$	31,000.00

Fuente: Elaboración Propia

El cuadro anterior define el costo que se valora al implementar el videojuego, también detalle que se necesita para la implementación, sus diferentes desarrollos, también los valores en los equipos de trabajo que se va a ocupar, la cantidad de personas que están involucradas en este videojuego, con esto se determina una aproximación al costo real que tendría el videojuego.

4. DISEÑO ARQUITECTÓNICO DEL SOFTWARE

4.1. DISEÑO ARQUITECTÓNICO

La arquitectura del software se basa en las entradas y salidas de cada uno de los eventos que tiene cada componente del juego.

El usuario inicia la aplicación y se levanta la interfaz de usuario la cual será utilizada para ejecutar y mostrar las diferentes acciones que tendrá nuestro juego.

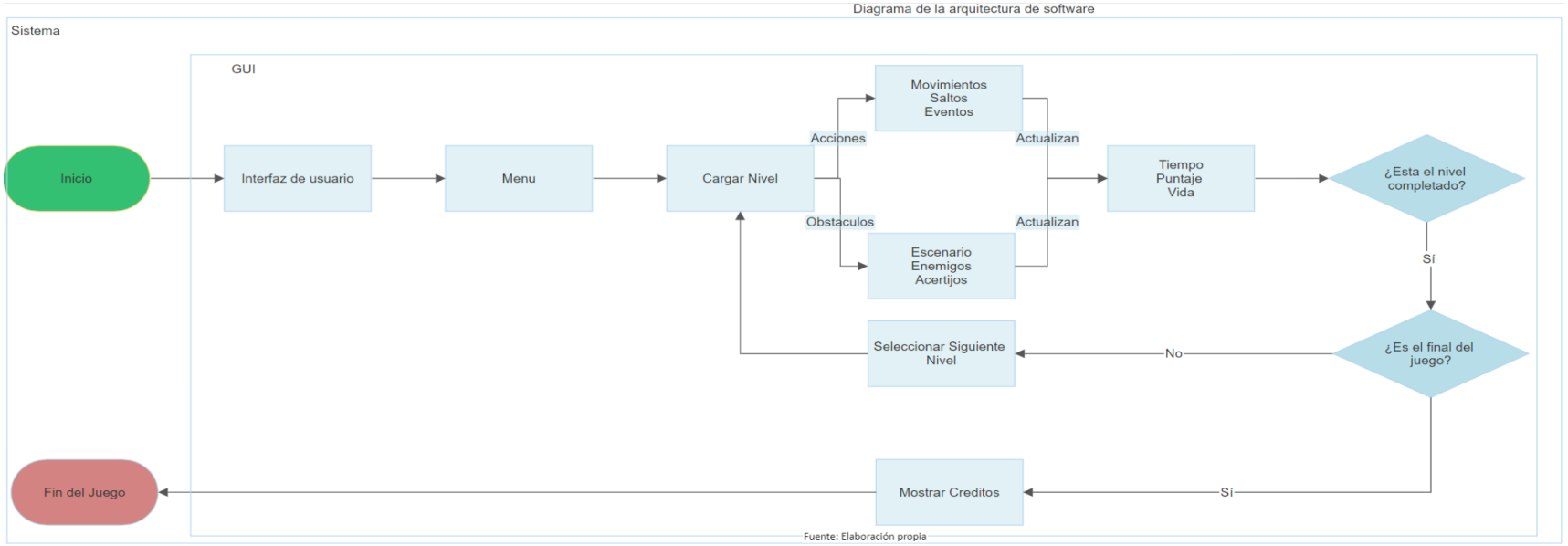


***Figura 37**: Diagrama de flujo (Flujo de funcionalidad)*

La primera interacción del será en el menú principal, donde escogeremos el nivel del juego, por medio del uso del mouse y un click en un botón de inicio, en este momento nuestro sistema deberá levantar todos los objetos y componentes del nivel.

Una vez que tenemos el nivel cargado el jugador podrá empezar a jugar utilizando mouse y teclado y ejecutando las diferentes acciones de movimiento, salto e interacción con objetos en el escenario, las cuales debemos utilizar para superar enemigos, obstáculos y enemigos, el resultado de estas interacciones va a afectar y actualizar nuestro puntaje, la vida del personaje y la resolución de acertijos que nos permitan completar el nivel. Alternativamente se puede usar un mando de control, por ejemplo, el control de xbox.

Una vez completado el nivel actual, el sistema validará si existen nuevos niveles por completar, en caso afirmativo, procederemos a cargar el siguiente nivel y así sucesivamente hasta llegar al final del juego, donde se procede a mostrar los créditos y el final del juego. Herramientas de software a utilizar.

4.1.1. Motor de juego - Unreal Engine

Unreal Engine es un motor que se utiliza principalmente para crear juegos, se caracteriza por facilitar y agilizar el desarrollo de videojuegos, ya que, cuenta con una serie de componentes que trabajan en conjunto para ser utilizados en nuestro desarrollo de una manera fácil.

En nuestra implementación, toda la lógica y programación de se ha desarrollado mediante el uso de blueprints, el cuál es el principal componente diferenciador de Unreal Engine, ya que facilita la programación, encapsulando el código C++ en blueprints, que son más fáciles de utilizar y entender incluso para profesionales de otras áreas y que no son programadores necesariamente.

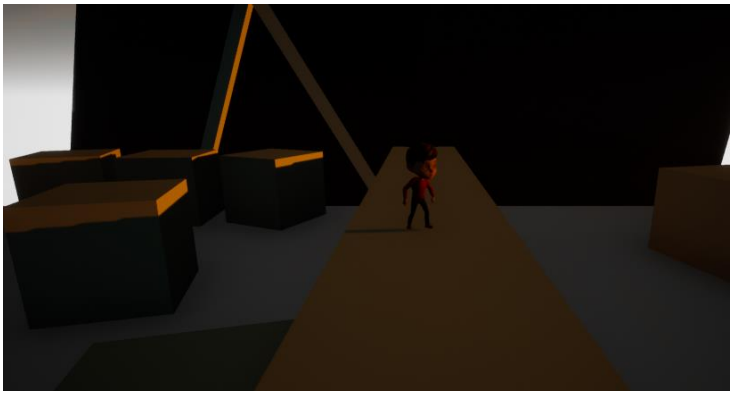


***Figura 38**: Diseño (Escenario en unreal)*

- **El motor de sonido**

El motor de sonido es el responsable de administrar y reproducir los diferentes efectos sonidos y música que tengamos para nuestro videojuego.

Los efectos de sonido se pueden disparar por medio de eventos y acciones que genera el jugador como, por ejemplo, los sonidos de los pasos al caminar. En la imagen (39) se puede observar cómo podemos sincronizar el momento de dar un paso en la animación, con un sonido de paso, que se representa con el color morado.

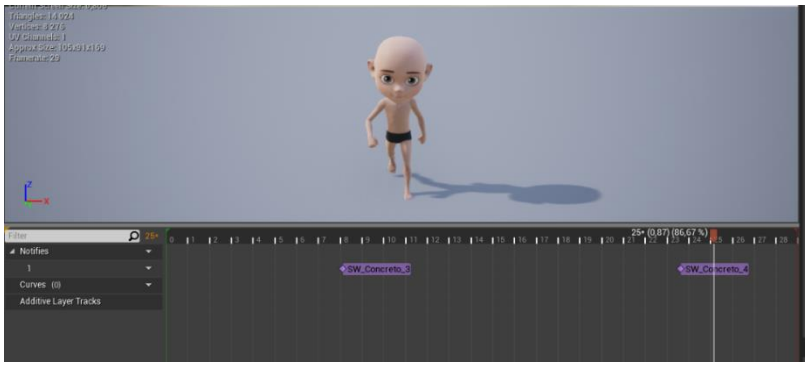
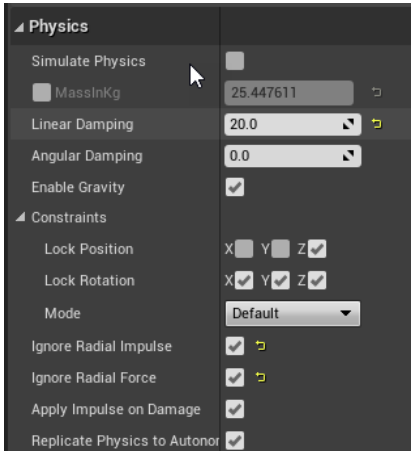


***Figura 39**: Sonido (Asignación de sonido en los pasos)*

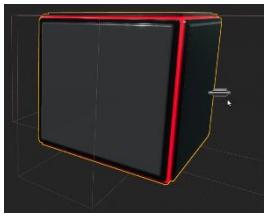
La música y efectos de sonidos que se utilizan en nuestro juego, fueron descargados de la tienda de Unreal y se detallan a continuación.

- Sonidos FX o de ambiente:
  - Footsteps Sounds with Blueprint Setup:
    - Efecto de sonido de los pasos (cuando camina el personaje) y salto
    - Efecto de sonido al activar un objetivo
- Clips de música
  - Complete Horror:
    - Creepy Music Box
    - Lonely house
    - Short Mystery
    - Distorted Ambience
- Fantasy Orchestral:
  - Elven Harmonies
  - Land of mistery

- **El motor de físicas**



Nos permite ajustar una serie de características, para modificar el motor de acuerdo con las necesidades de nuestro juego, por ejemplo, en el caso de los cubos que el jugador debe empujar para resolver algunos acertijos, se le aplica una modificación a la física para que su rotación no se pueda modificar, de esta forma podemos empujar el cubo, sin que el mismo rote. Adicionalmente, podemos modificar estos parámetros en tiempo de ejecución usando blueprints, de forma que podemos activar y desactivar la simulación de físicas únicamente cuando el jugador esté cerca del cubo, en caso contrario lo desactivamos para optimizar el rendimiento y uso de memoria.



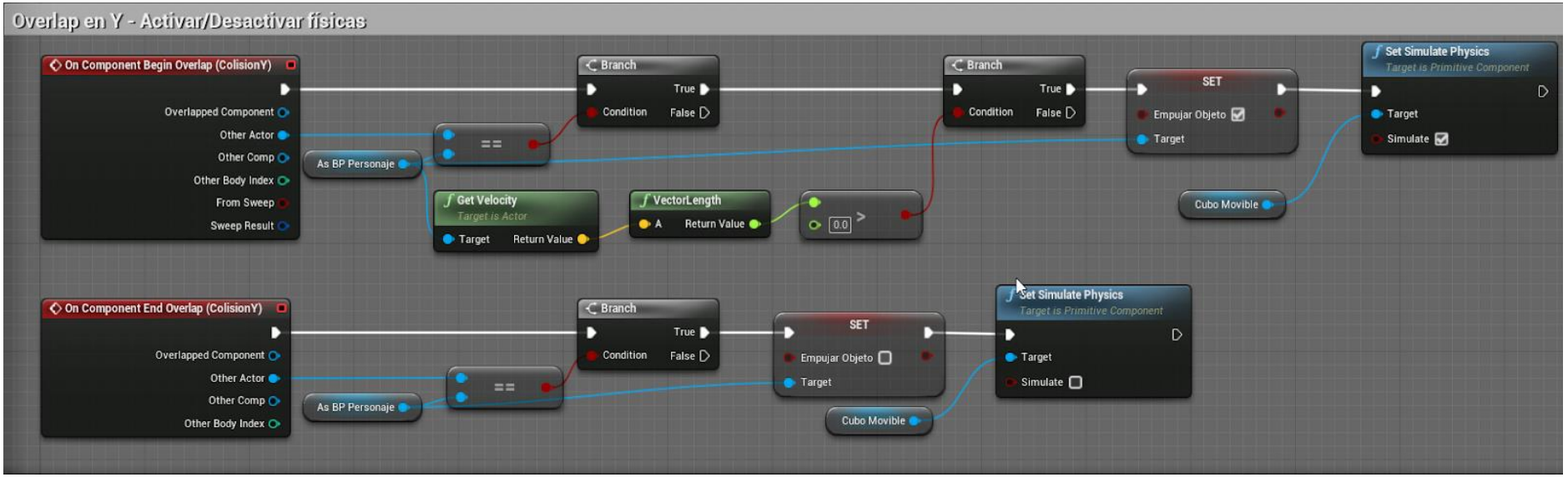


***Figura 40**: Físicas (Físicas en unreal)*

- **El motor gráfico**

Es el encargado de renderizar los recursos gráficos del videojuego, basándose en la información de nuestro nivel o escena, como pueden ser los colores, las texturas, las sombras y geometría. En nuestro juego hemos diseñado los niveles tratando de optimizar el uso de recursos para mejorar el rendimiento de nuestro juego, por ejemplo, se hace uso de interfaces y herencia de objetos, para minimizar los llamados al procesador y se cargan variables al iniciar el nivel, las cuales son compartidas entre los diferentes actores.

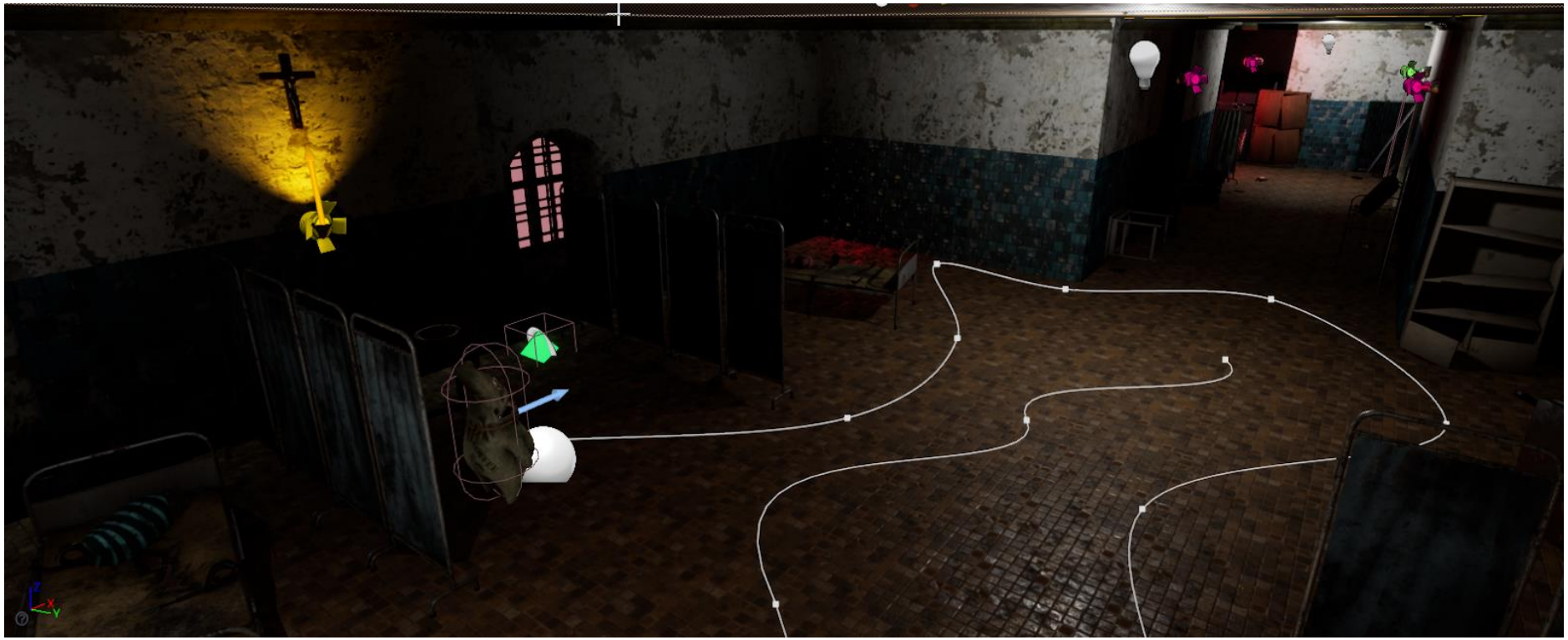


***Figura 41**: Gráfico (Visualización de gráficos en unreal)*

- **Controlador de entradas (Input)**

En Unreal tenemos un sistema que se encarga de controlar las entradas que generan los usuarios como, por ejemplo, al hacer click a un botón o tecla y generar o disparar un evento y ejecutar alguna acción o lógica dentro de nuestro sistema.

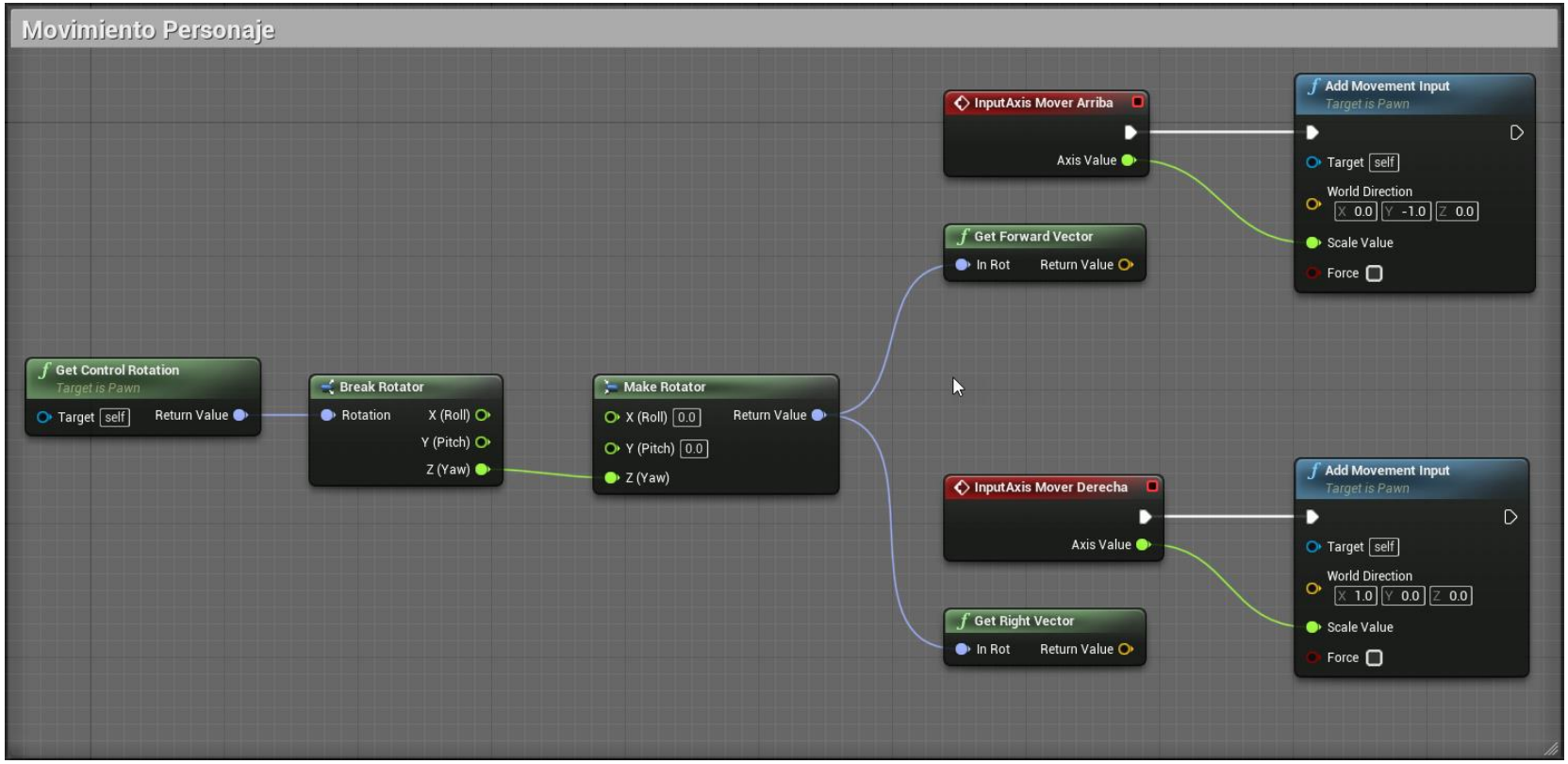


***Figura 42**: Blueprint (Movimiento del personaje)*

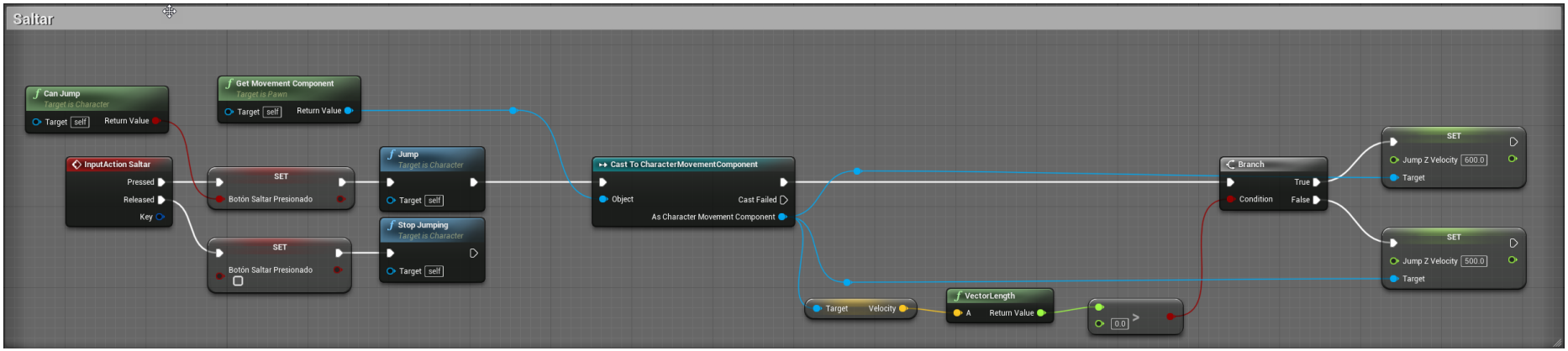


***Figura 43**: Blueprint (Salto del jugador)*

- **Framework de Jugabilidad**

El framework de jugabilidad es el encargado de llevar el control del avance y reglas que están definidas en nuestro juego.

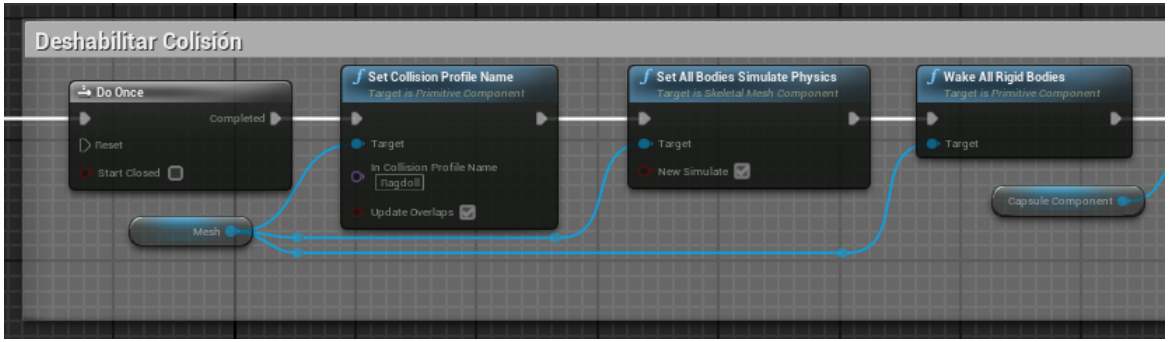


***Figura 44**: Blueprint (Deshabilitar colisión)*

4.1.2. Lenguaje de programación

Unreal Engine nos permite programar nuestro video juego de diferentes formas, una de las más populares es mediante el uso de plantillas (Blueprints), otra para usuarios con mayor conocimiento de programación usando C++ o incluso una combinación de ambas, ya que los blueprints son compilados a C++ y por ende se pueden utilizar ambos en Unreal Engine.

- **Plantillas (Blueprints)**

Es una metodología de desarrollo visual, que le permite al usuario programar juegos conectando algo similar a un diagrama de flujo, uniendo objetos y componentes por medio de líneas, es decir, les permite a usuarios que no tienen mucho conocimiento en programación una alternativa más sencilla de entender y que agiliza los tiempos de desarrollo enormemente. Como podemos observar en la siguiente imagen, se basa en la conexión de nodos o bloques de lógica, sin tener la necesidad de escribir código directamente.

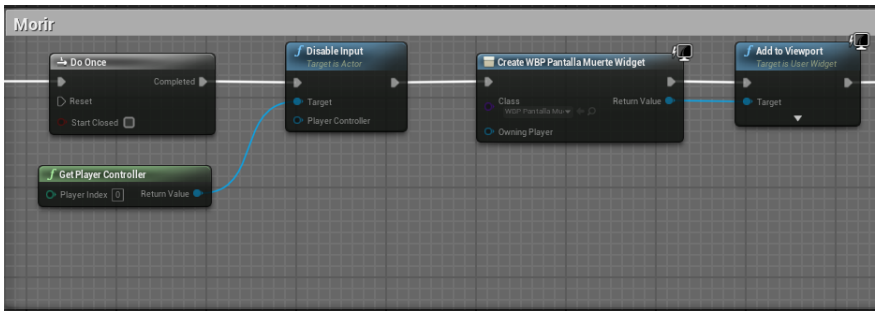


***Figura 45**: Blueprint (Muerte del personaje)*

- **C++**

Como alternativa a los blueprints o incluso para personalizar blueprints a un más bajo nivel, en Unreal podemos utilizar C++ como lenguaje de programación, el cual contiene un framework que permite exponer cualquier clase de C++ a un blueprint e interactuar con él, adicionalmente por medio de C++ se puede optimizar el rendimiento del sistema y manejar escenarios complejos de una forma más robusta.

```
Character.h
585     virtual void GetSimpleCollisionCylinder(float& CollisionRadius, float& CollisionHalfHeig
586     virtual UActorComponent* FindComponentByClass(const TSubclassOf<UActorComponent> Compone
587     virtual void TornOff() override;
588     virtual void NotifyActorBeginOverlap(AActor* OtherActor);
589     virtual void NotifyActorEndOverlap(AActor* OtherActor);
590     //~ End AActor Interface
591
592     template<class T>
593     T* FindComponentByClass() const
594     {
595         return AActor::FindComponentByClass<T>();
596     }
597
598     //~ Begin INavAgentInterface Interface
599     virtual FVector GetNavAgentLocation() const override;
600     //~ End INavAgentInterface Interface
601
602     //~ Begin APawn Interface.
603     virtual void PostInitializeComponents() override;
604     virtual UPawnMovementComponent* GetMovementComponent() const override;
605     virtual UPrimitiveComponent* GetMovementBase() const override final { return BasedMoveme
606     virtual float GetDefaultHalfHeight() const override;
607     virtual void TurnOff() override;
608     virtual void Restart() override;
609     virtual void PawnClientRestart() override;
610     virtual void PossessedBy(AController* NewController) override;
611     virtual void UnPossessed() override;
612     virtual void SetupPlayerInputComponent(class UInputComponent* PlayerInputComponent) over
```

***Figura 46**: Código fuente (Class Character.h)*

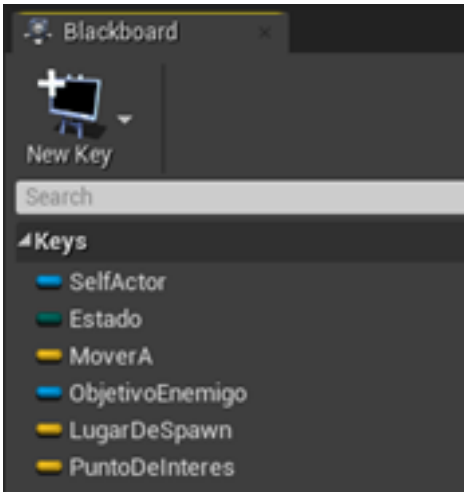
El código fuente lo pueden localizar y descargar en la siguiente dirección:

https://github.com/IsaacXB/TFM

4.1.3. Inteligencia artificial

Para implementar inteligencia artificial a los enemigos, se han utilizado una serie de sistemas y componentes que trabajan en conjunto, analizan una serie de variables, condiciones y reglas, para determinar el comportamiento que debe tener el enemigo en relación con lo que este sucediendo en el juego y principalmente en relación con las acciones del jugador. A continuación, explicaremos los principales componentes utilizados:

4.1.3.1. **Árbol de Comportamiento (Behaviour Tree) y Blackboard:**



El árbol de comportamiento se utiliza para ejecutar ramas con bloques de lógica y condiciones, para determinar cuál rama debe ejecutar, se usa un componente llamado “Blackboard”, que se usa como controlador de las principales variables o llaves. Las llaves utilizadas en nuestro juego son las siguientes:

***Figura 47**: Keys (Comportamietos)*

Self Actor: Componente base del blackboard, utilizado por el motor gráfico de Unreal.

Estado: Es un enumerador que permite controlar el estado de nuestro enemigo el cual puede ser:

- ***Pasivo*:** Se utiliza cuando los sensores de la inteligencia artificial no han detectado al jugador, sonidos u otros enemigos. Tiene una ruta de patrullaje, la cual seguirá ejecutando mientras no se activen los sensores. La velocidad de movimiento es lenta*.*
- ***Buscando:*** Se utiliza cuando el sensor de sonido o visión escucha algún ruido cercano o ve algún objeto, para determinar si debe moverse a dicho punto en búsqueda del personaje.
- ***Investigador:*** Se utiliza cuando se ha detectado un punto de interés, se incrementa la velocidad de movimiento y el enemigo se mueve hacia el área de interés, en caso de confirmar el personaje, actualizar el movimiento a este punto.

Mover A: Vector 3D que almacena el punto donde se va a mover el objeto dentro del mapa.

Objetivo Enemigo: Se asigna el actor del personaje cuando lo tiene dentro de su capacidad sensorial.

Lugar de Spawn: Vector 3D Especifica un punto en el mapa en el que hizo spawn el enemigo.

Punto de Interés: Vector 3D que almacena el punto de interés donde se debe analizar algún evento.

Con estas llaves establecidas dentro de nuestro blackboard, las mismas pueden ser consultadas o actualizadas, en el árbol de comportamiento para tomar decisiones, mediante el uso de tareas o “tasks” que implementan un segmento de lógica con un fin específico, por ejemplo, determinar el tipo de objeto o actor que percibe el sensor de percepción, para determinar si es de interés o no para nuestra lógica y tomar decisiones en base a ello.

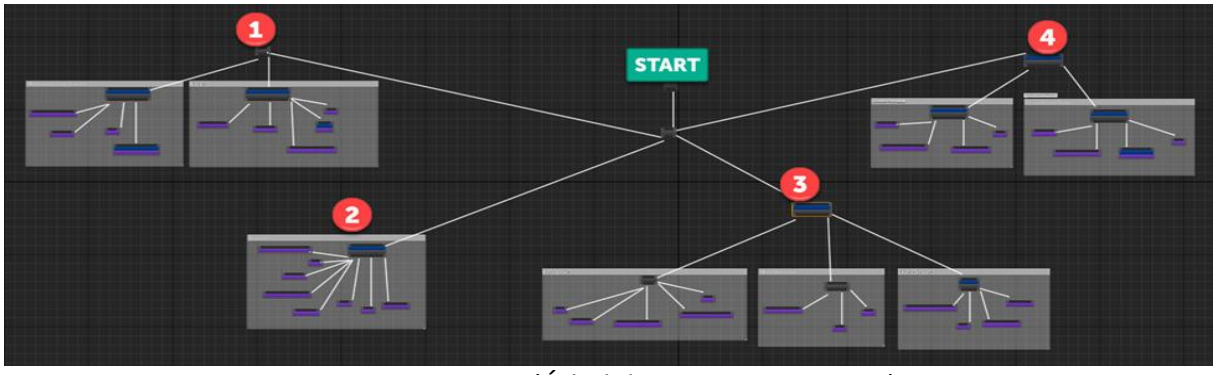


***Figura 48**: Tree (Árbol de comportamientos)*

El árbol de comportamiento de nuestro enemigo de tipo guardián se compone de 4 ramas principales en nuestro juego, cabe recordar que el orden de análisis del árbol siempre se ejecutará o tendrá prioridad, de izquierda a derecha (1 => 2 => 3 => 4):

- ***Atacando (1):*** En este estado lo primero que vamos a consultar es si tenemos asignado al actor del blackboard “Objetivo Enemigo”, en caso de no tenerlo, que sería lo normal al iniciar el juego, vamos a cambiar el estado del blackboard a pasivo, en caso de que si tengamos un objetivo asignado, procedemos a mover el enemigo hasta el rango de ataque, cuando llegamos a estar dentro del rango de ataque o arma, procedemos a cambiar el estado a agresivo y enfocar hacia el personaje para posteriormente realizar el ataque.

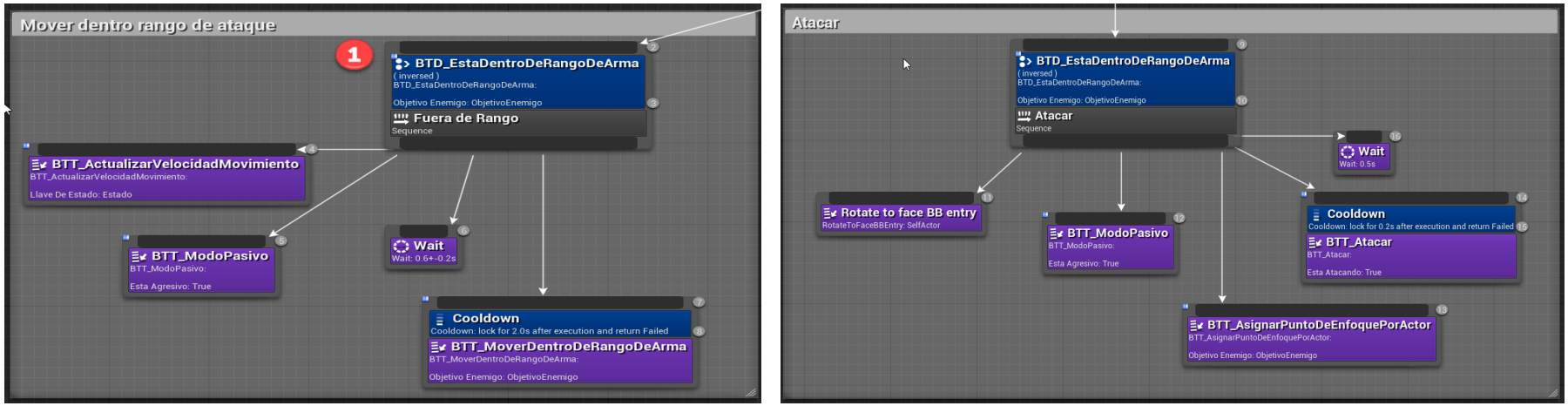


Figura 49*: Tree (Comportamiento atacando)*

- ***Buscando (2)*:** Se activa cuando el estado del blackboard se encuentra asignado a buscar y se asigna el punto de interés detectado por los sensores de percepción (sonido y visión).

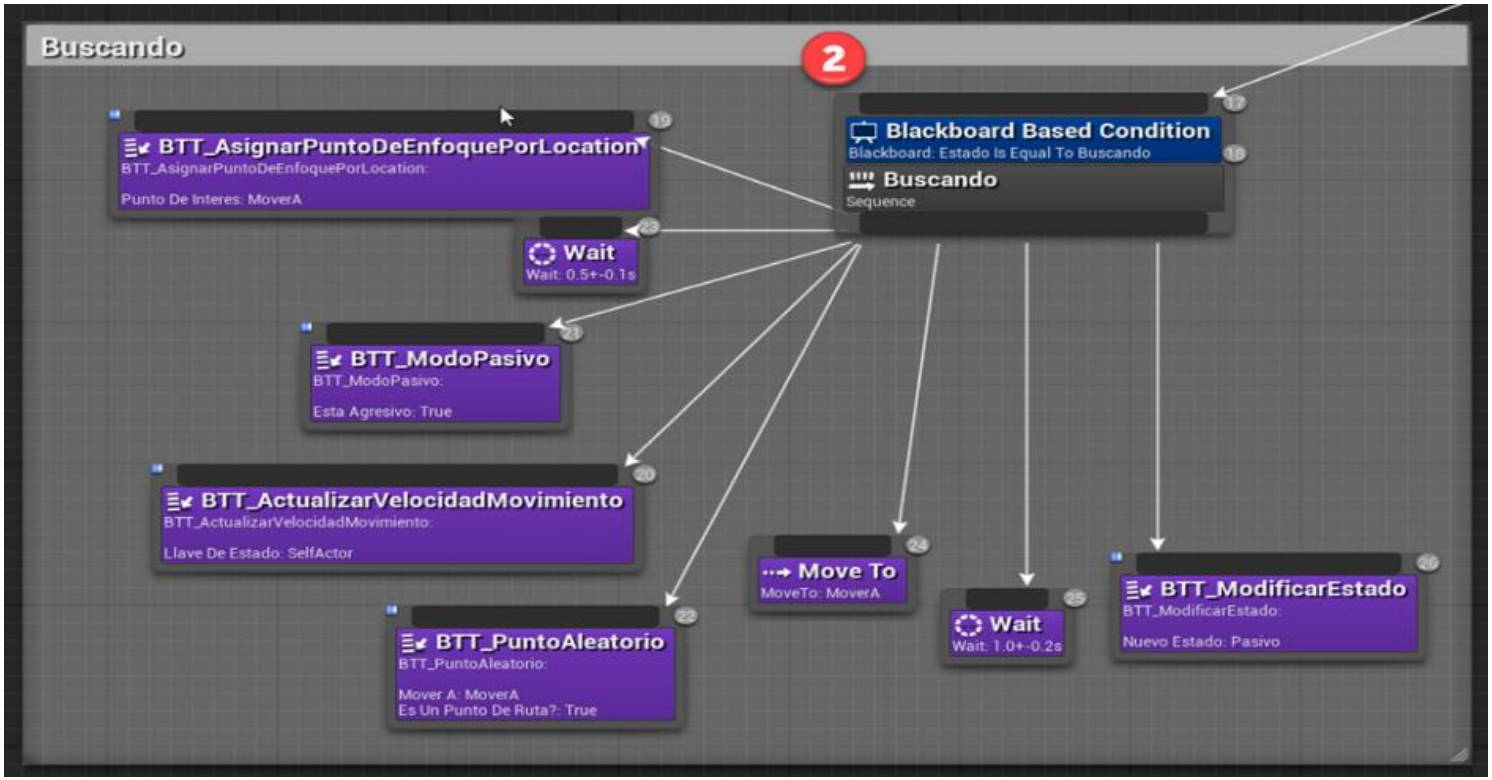


***Figura 50**: Tree (Comportamiento buscando)*

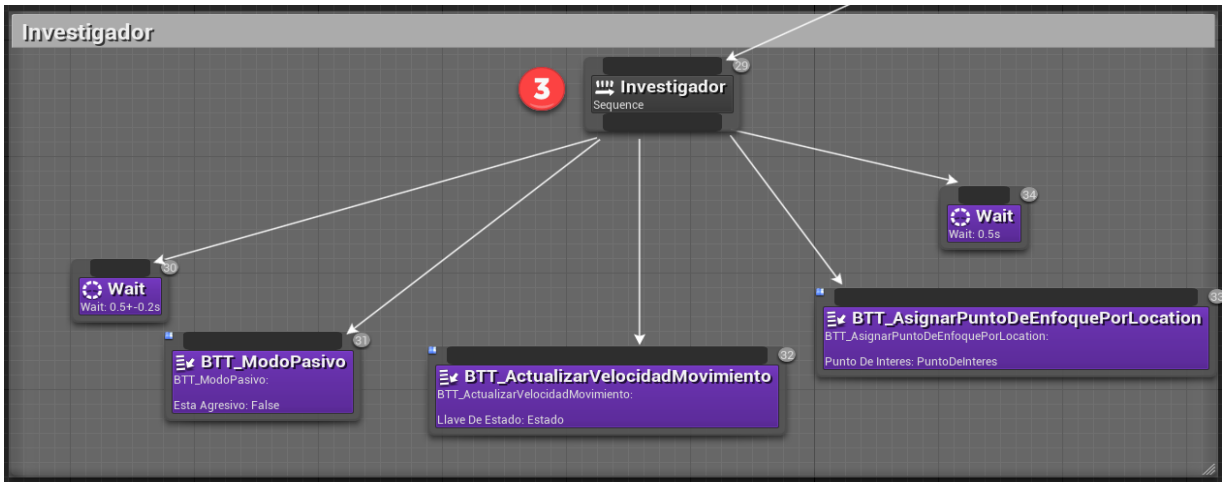


***Figura 51**: Tree (Comportamiento investigador)*

- ***Investigador (3)*:** Se activa cuando el estado del blackboard es igual al investigador y se mueve al enemigo al punto de interés.

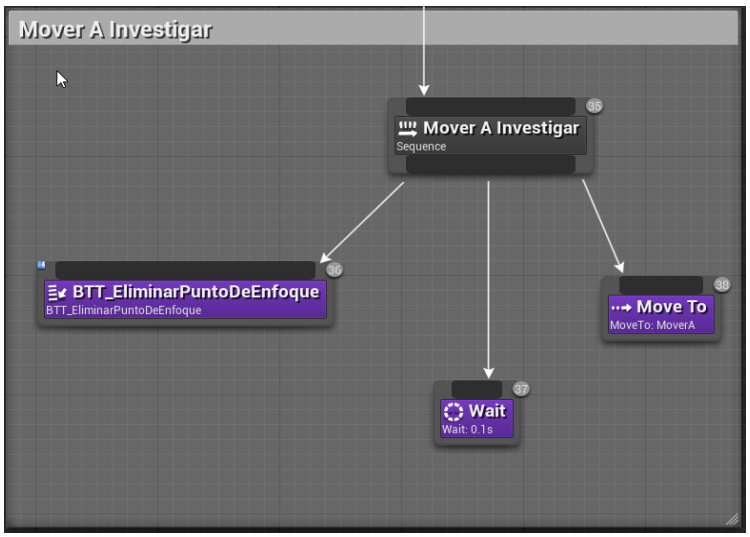


***Figura 52**: Tree (Comportamiento mover a investigar)*

- ***Pasivo/Patrullaje (4)*:** Cuando el estado del enemigo es pasivo, el enemigo recorrerá su ruta de patrullaje.

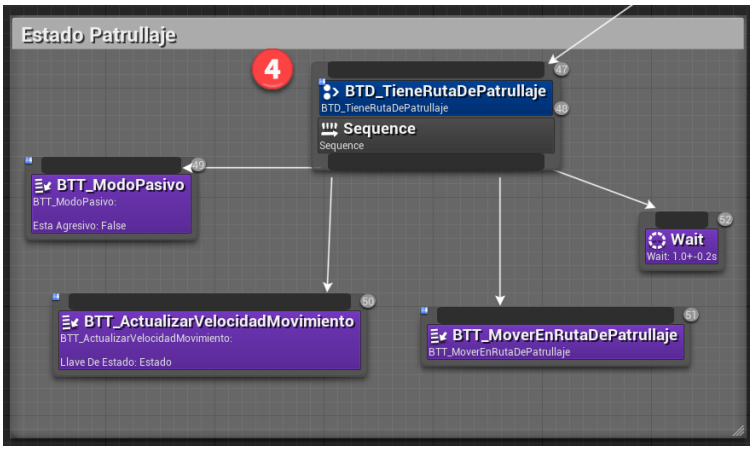


***Figura 53**: Tree (Comportamiento estado patrullaje)*

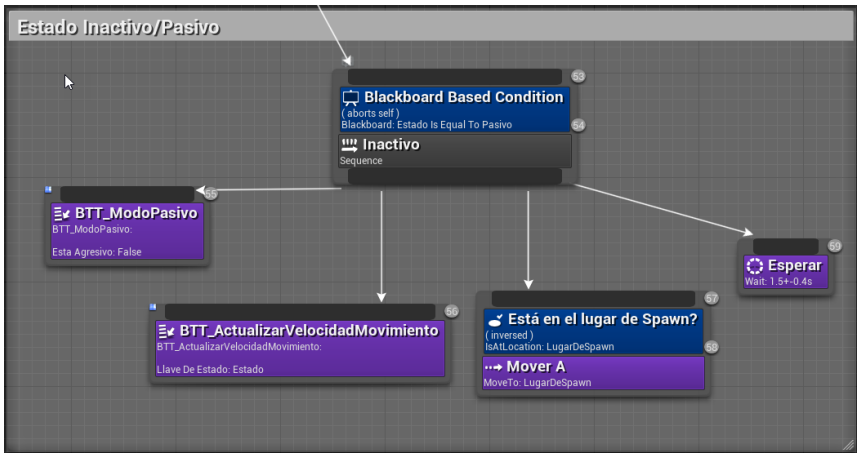
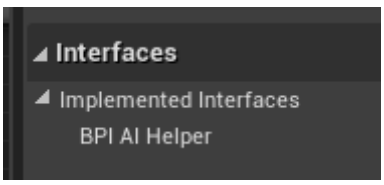


***Figura 54**: Tree (Comportamiento estado activo/inactivo)*

4.1.4. Uso de interfaces y comunicación entre blueprints:

Para optimizar el uso de memoria y recursos para correr el video juego, se han implementado interfaces y disparadores de eventos para la comunicación entre los diferentes componentes y blueprints. Adicionalmente, se ha evitado utilizar la función de cast por requerir muchos recursos, en su lugar se obtienen los objetos referenciados en el blackboard o mediante el uso de interfaces y disparadores de eventos.

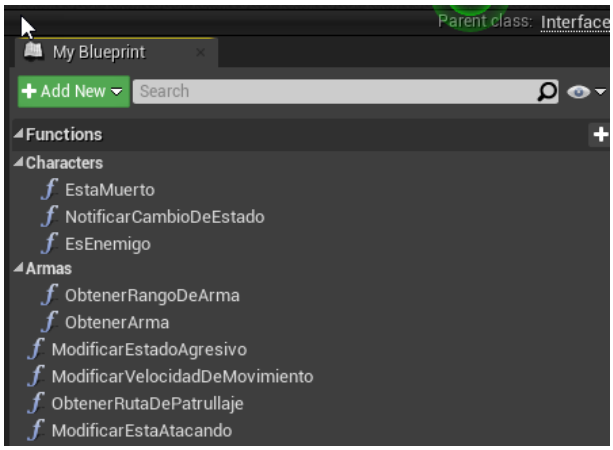
- **Interfaz *de* ayuda para la inteligencia artificial (BP_AI_Helper):**



En esta interfaz se especifican métodos que pueden ser implementados en los blueprints, de forma que el mismo método tendrá una implementación diferente en cada

blueprint. Adicionalmente, permite la comunicación y uso de los métodos disponibles en la interfaz.

En este caso se han creado los siguientes métodos o funciones



Estos métodos permiten la comunicación entre diferentes blueprints, de una forma eficiente. Por ejemplo, el blueprint de animación, el blueprint del actor y el blueprint del personaje.

***Figura 55**: Blueprint (Comunicación entre blueprints)*

A continuación, podemos ver las diferentes implementaciones que tienen los métodos en cada blueprint, a pesar de ser diferentes tipos de blueprints.

- **Blueprint del *actor* o enemigo:** Utiliza el método que notifica el cambio de estado al blueprint de animación, para que este pueda utilizar la animación correcta ante ciertos eventos, también utiliza el método de modificar la velocidad del movimiento en base al estado del objeto en el blackboard.

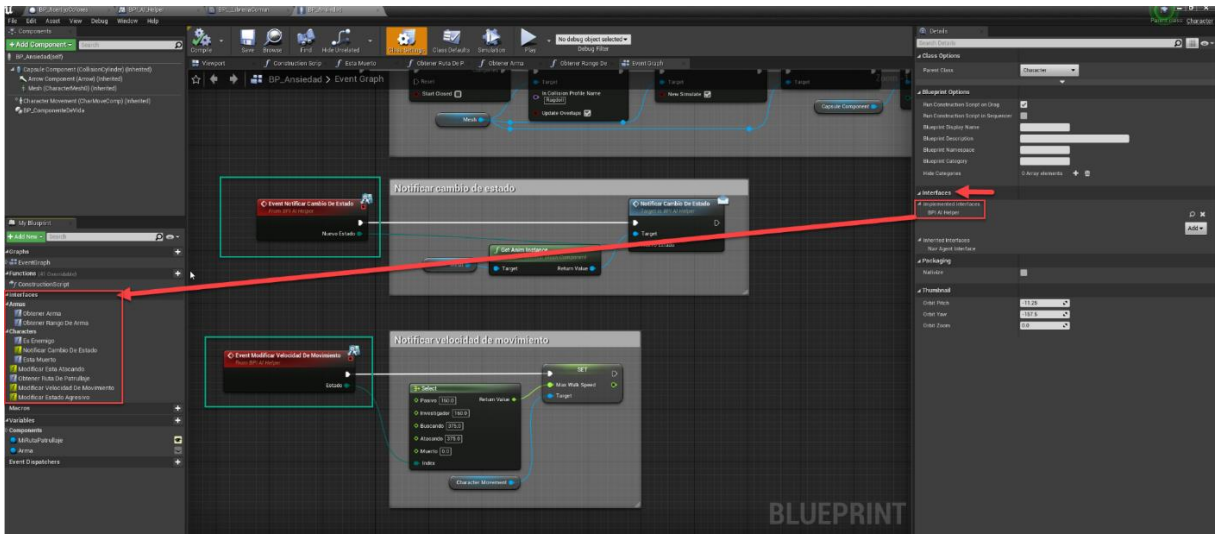


***Figura 56**: Blueprint (Comunicación entre blueprints)*

- **Blueprint de animación del enemigo:** Cuando recibe notificación de cambio de estado, modifica las variables respectivas para que se muestre la animación correcta.

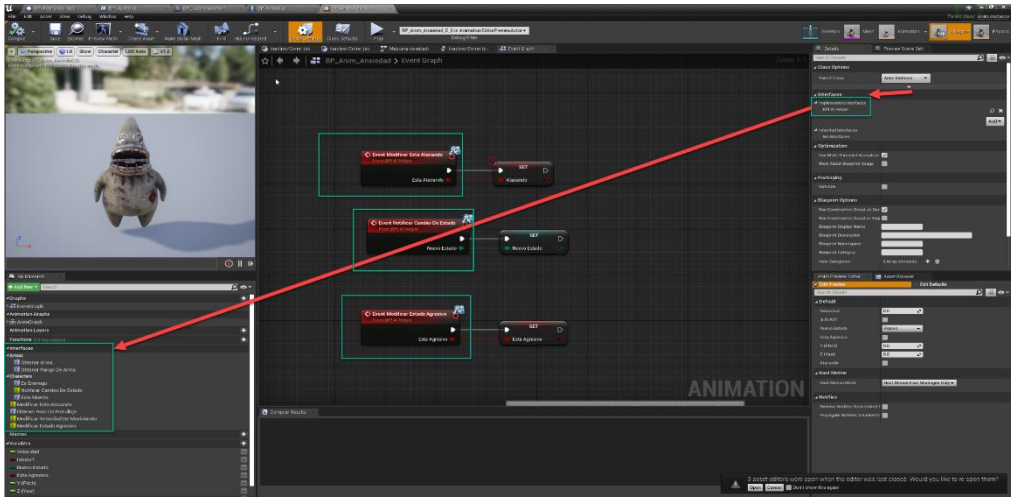


***Figura 57**: Blueprint (Animación enemigo)*

- **Blueprint del personaje:** Utiliza la interfaz para validar si el objeto que ha entrado en contacto nos puede hacer daño.

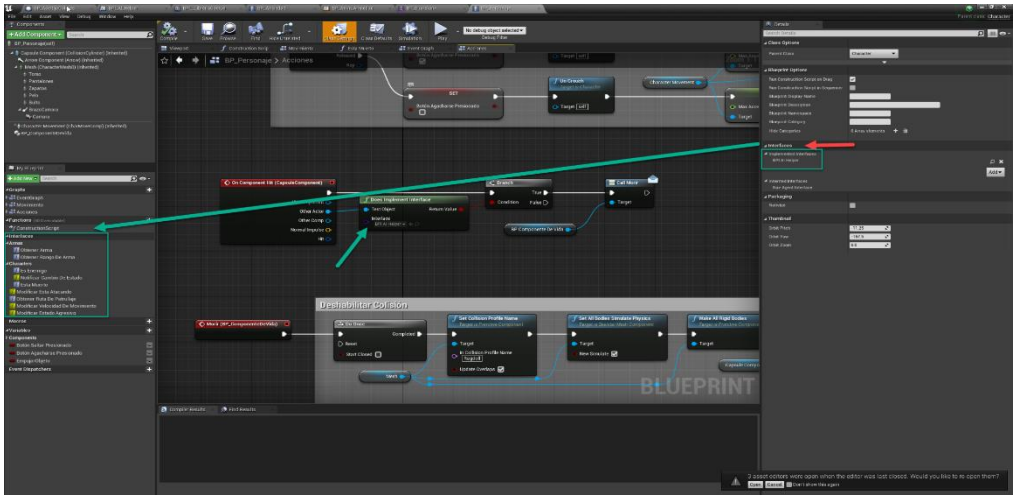


***Figura 58**: Blueprint (Personaje)*

4.1.5. Inkscape / Adobe Illustrator

Para la creación de artes digitales e interfaz de usuario se va a usar Inkscape o adobe illustrator.

4.1.6. Controlador de versiones - Git / Perforce

Para tener un repositorio de cambios de nuestro proyecto vamos a utilizar Github.

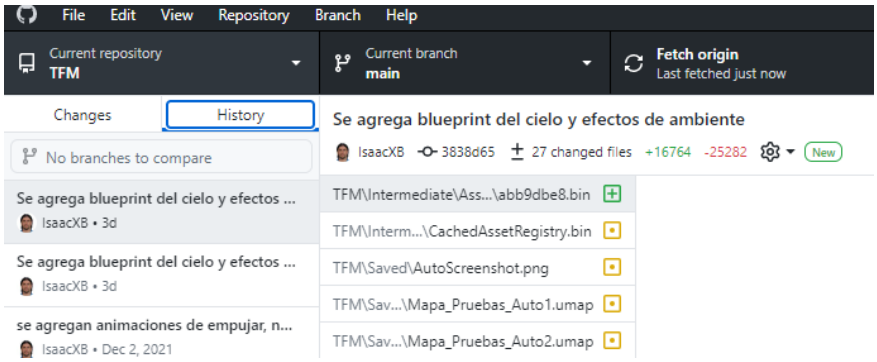


***Figura 59**: Git (Control de código fuente)*

Al tener el código en la nube, es más fácil la integración o unión del proyecto en un mismo repositorio y no tener varios repositorios para luego estar uniendo.

4.2. DIAGRAMA MODULAR

4.2.1. CAPA PRINCIPAL

En este proyecto utilizaremos la metodología de programación visual, utilizando plantillas (Blueprints) a las cuales vamos a identificar en nuestro proyecto con las letras BP al inicio del nombre del objeto.

La capa principal de blueprints contará con los siguientes blueprints:

- BP Modo de Juego: Este blueprint administramos los componentes principales, por medio de referencias a clases y componentes bases de nuestro juego, entre ellos utilizamos los siguientes:

***Figura 60**: Classes (Blueprint de clases)*

  - Sesión de juego: Será el encargado de administrar la sesión de juego.

- Estado del juego: Nos permite administrar los diferentes estados que puede tener nuestro juego, como pueden ser:
  - Juego no iniciado
  - Juego en progreso
  - Juego finalizado
- Estado del jugador: Nos permite identificar el estado de salud de nuestro personaje, para determinar si la entrada de comandos debe estar activa o no, así como disparar el evento de fin en caso de que la vida de nuestro personaje llegue a 0.
- BP Controlador del jugador (BP Personaje): Este es el blueprint donde encapsulamos la malla estática, que representa a nuestro personaje en un modelo 3D, la cual suele tener las animaciones, cápsulas de colisión, cámaras, controlador de acciones del jugador y otros elementos que definen las características y componentes que tendrá nuestro personaje.

Como se puede observar en la siguiente imagen, el personaje tiene mallas estáticas por aparte para el cuerpo (torso), pantalones, zapatos, pelo y el bulto, los cuales se asignan a la pose máster del componente.

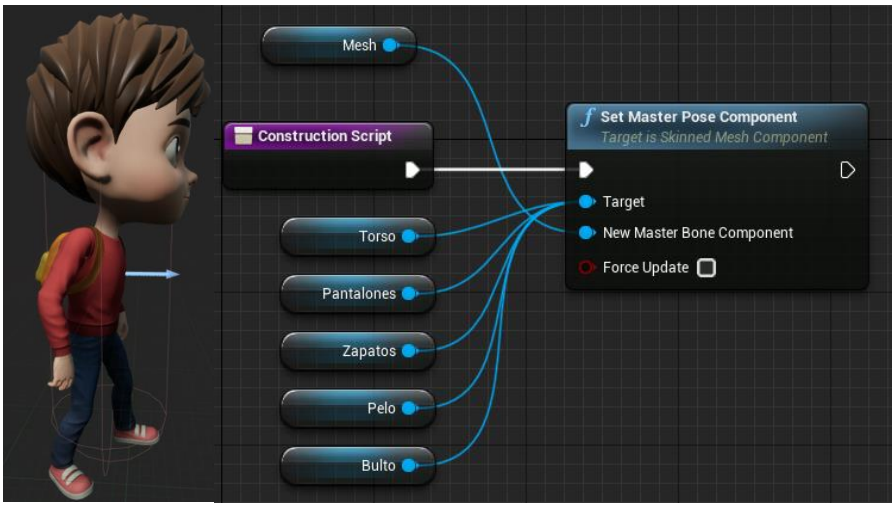


***Figura 61**: Blueprint (Personaje)*

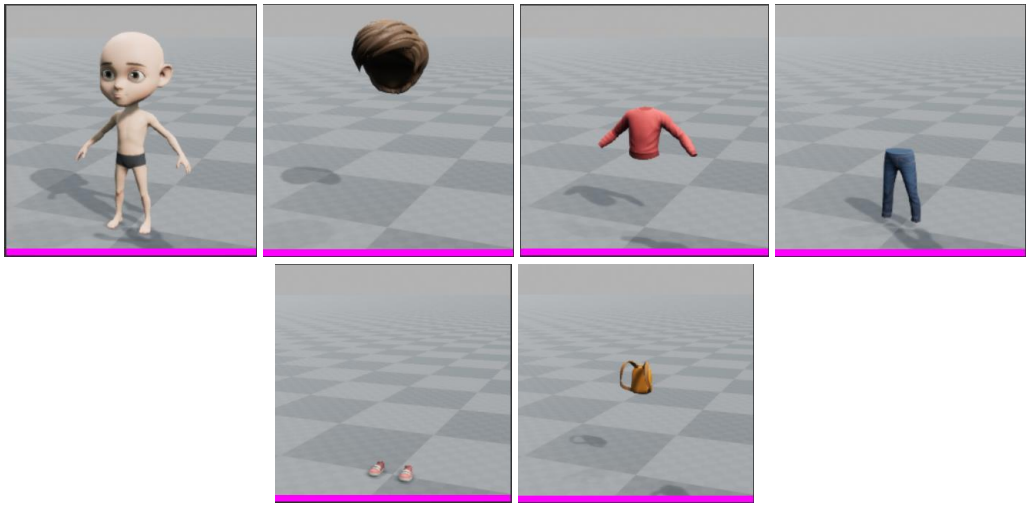


***Figura 62**: Personaje (Unrealengine, 2021)*

- BP Salvar Juego: Este blueprint nos permite guardar y cargar los archivos de guardado, para que el jugador pueda almacenar un estado de avance dentro de nuestro videojuego.

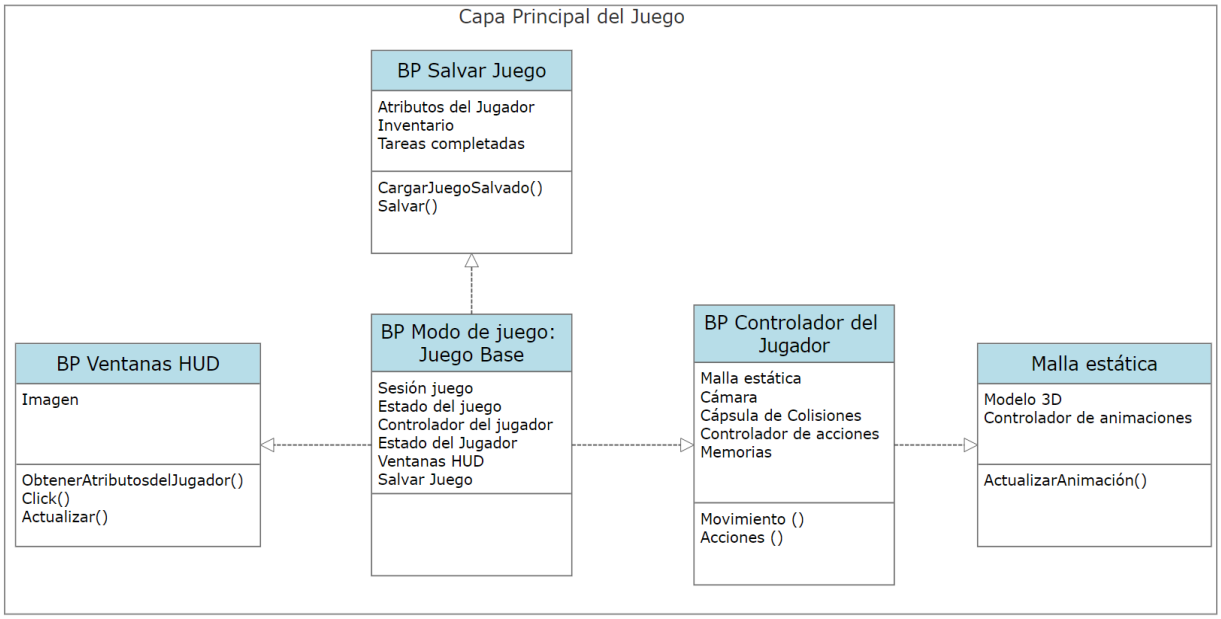


***Figura 63**: Capas de juego*

- BP Ventanas HUD (elementos visuales presentes en la interfaz gráfica): Este blueprint nos permite mostrar al usuario los elementos que se sobreponen a la cámara para mostrar información relevante al jugador como suelen ser:
    - Menú Principal: Es el primer contacto que tendrá el jugador con nuestro juego y desde aquí se puede iniciar el juego o salir del mismo.

***Figura 64**: Menú inicio*

- Menú Pausa: El menú de pausa es utilizado para pausar el juego en tiempo de ejecución, también permite al usuario salir del juego.

***Figura 65**: Menú pausa*

4.2.2. FLUJO PARA CREACIÓN DE OBJETOS 3D

Algunos modelos en 3D los vamos a desarrollar utilizando las herramientas de software Blender o 3DS Max, las cuales permiten exportar los objetos a Unreal Engine, adicionalmente, en Blender se pueden realizar animaciones 3D, lo cual será utilizado para crear algunas animaciones, principalmente de movimientos básicos.

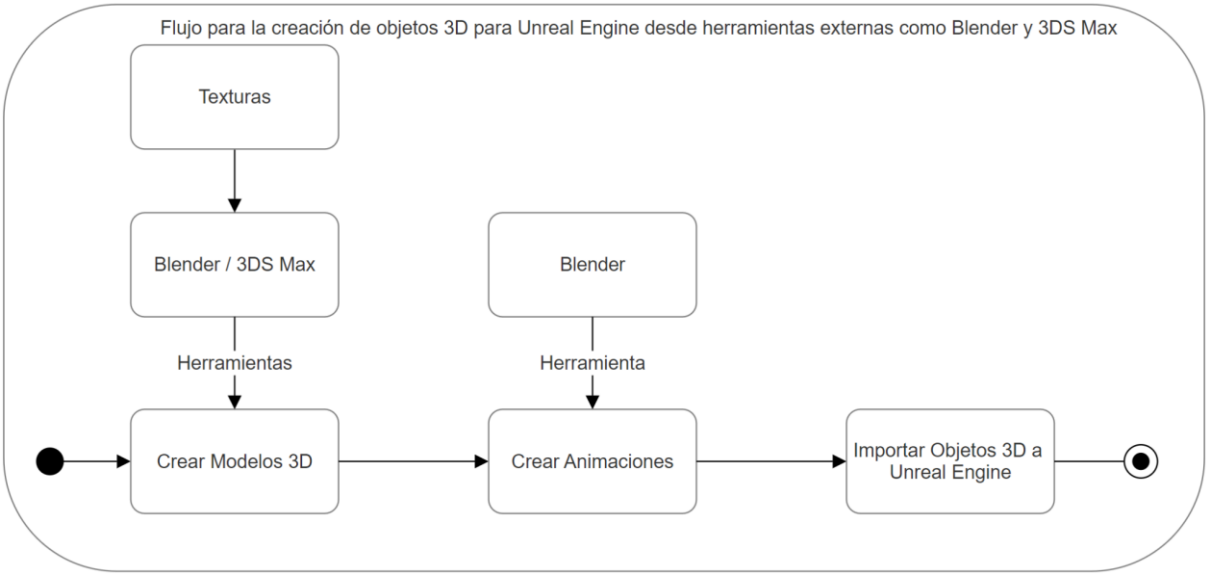


***Figura 66**: Diagrama de modelado (Personaje)*

4.3. CASOS DE USO

4.3.1. Movimiento del Personaje

Cuando el usuario presiona una de las teclas de movimiento el personaje debe reproducir la animación de movimiento en la dirección deseada y empezar a desplazarse en el escenario del nivel. El personaje debe reproducir una animación de reposo cuando el jugador no esté presionando ninguna tecla o acción de movimiento.

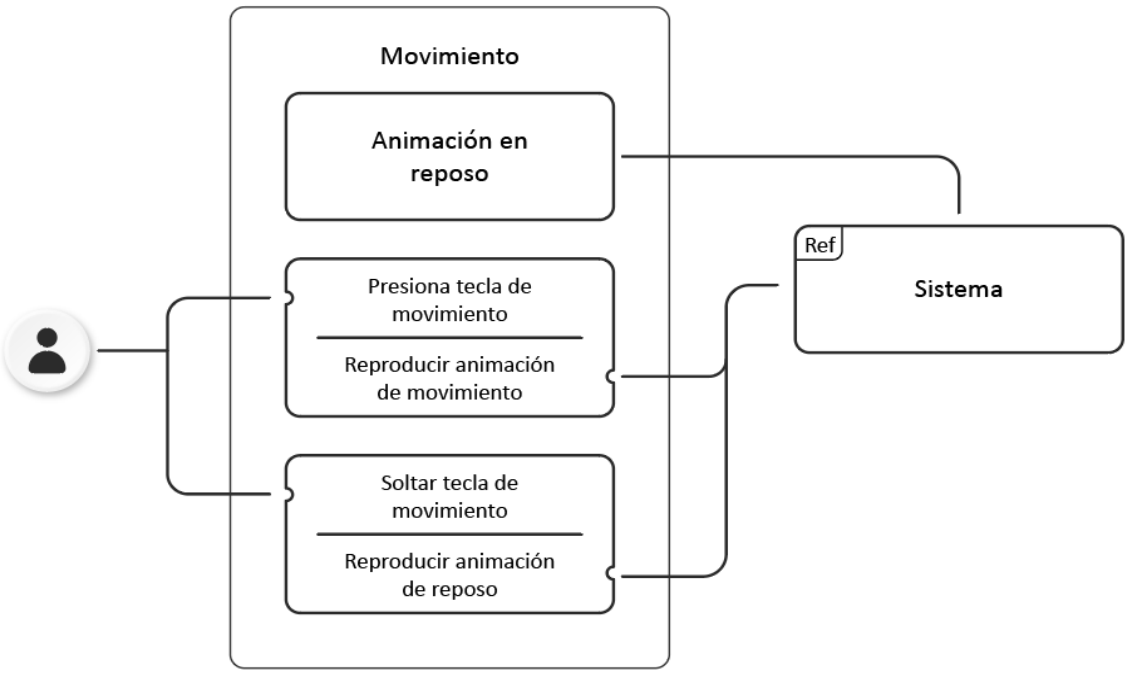


***Figura 67:** Movimiento (Flujo movimiento personaje)*

El movimiento del personaje está diseñado de forma que tenga una mayor velocidad que los enemigos para que el jugador pueda escapar de los enemigos que lo persiguen, así mismo, esta velocidad varía de acuerdo con el estado y acción que está realizando el jugador

4.3.2. Muerte del personaje

Cuando nuestro personaje sea capturado por alguno de nuestros enemigos, caiga en algún precipicio o trampa, nuestro personaje debe mostrar una animación de muerte.

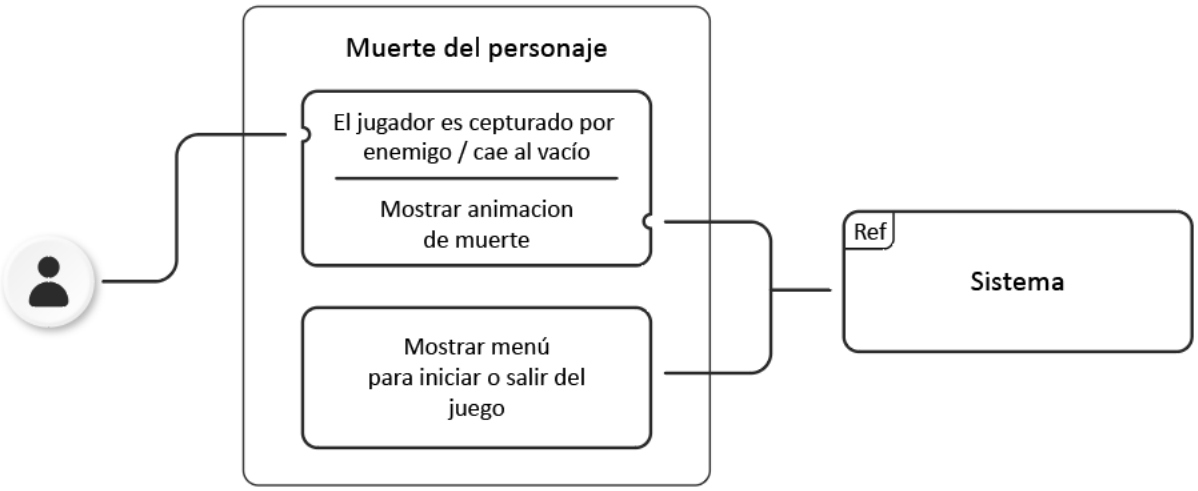


Figura 68: *Muerte (Flujo muerte personaje)*

4.4. REQUISITOS TÉCNICOS

Los requisitos técnicos para correr este juego son los siguientes:

Requisitos mínimos:	Requisitos recomendados:
Procesador: 64-bit. Intel CPU Core i3 Sistema operativo: Windows 7 (64 bit) Memoria: 4 GB RAM Tarjeta gráfica: Nvidia GTX 460 DirectX: Version 11 Almacenamiento: 3.5 GB	Procesador: 64-bit. Intel CPU Core i7 Sistema operativo: Windows 7 (64 bit) Memoria: 8 GB RAM Tarjeta gráfica: Nvidia GPU GeForce GTX 660 DirectX: Version 11 Almacenamiento: 3.5 GB

Fuente: Elaboración Propia

5. GDD DE ALTO NIVEL

5.1. ALCANCE DEL JUEGO

El videojuego contará con dos niveles los cuales representarán los sueños del personaje principal, se espera incrementar el número de niveles en el futuro. En este, el jugador deberá encontrar la salida resolviendo acertijos y encontrando memorias que estarán repartidas por el nivel. una vez encontradas todas las memorias el personaje desbloqueara un recuerdo y parte de su historia.

Los niveles contarán con tres tipos de enemigos que van cambiando, dependiendo la zona que nos encontremos. Los enemigos perseguirán al jugador al verlo y representarán los miedos que atormentan al personaje.

5.2. DISEÑO VISUAL

El mundo real será diseñado con colores opacos y apagados para representar la depresión y la visión del personaje.

Inicia el juego el cuarto del joven con objetos comunes: una cama, escritorio, ropa en el piso, algunos platos, entre otros. Al entrar al sueño vemos un cuarto oscuro, sin color y pocos objetos, el cuarto se va extendiendo mientras el personaje va explorando y encontrando con enemigos que él tema. Los personajes tendrán forma antropomórfica distorsionada.

A medida que el jugador avanza, el escenario irá cambiando al igual que los enemigos. Con cada recuerdo recuperado, las cosas serán más claras y tendrán más forma.

5.3. GAMEPLAY Y MECÁNICAS

5.3.1. Mecánicas del personaje

- ***Movimiento***: Puede moverse en el escenario lateralmente, así como hacia arriba o abajo.
- ***Salto***: Permite alcanzar plataformas a mayor altura o distancia entre las plataformas.
- ***Agacharse***: Permite esconderse y producir menos ruido, así como acceder a túneles y secretos en el escenario.

- ***Acción***: Funciona parainteractuar con los diferentes elementos en el escenario.

5.3.2. Mecánicas de escenario

- ***Acertijos***: Los escenarios cuentan con accesos bloqueados o secretos, para lograr acceder a estos lugares, el jugador deberá resolver acertijos que le permitan abrir puertas o acceder por rutas alternativas a nuestros objetivos.
- ***Plataformas***: En algunos momentos el usuario al pisar las plataformas o bloques las mismas empezarán a caer, para evitar que el jugador pueda regresar y que obligue al jugador a seguir avanzando rápidamente para evitar caer al precipicio.

5.3.3. Mecánicas de enemigos:

- ***Patrullando***: Los enemigos estarán patrullando el escenario en busca de nuestro personaje, si entramos en su campo de visión el enemigo empezará una persecución con el objetivo de eliminarnos.
- ***Persecución***: Se activa cuando el personaje entra en el campo de visión del enemigo, en este momento el enemigo tomará una actitud agresiva en contra de nuestro personaje y buscará aniquilarlo.

5.4. CONTROL

El juego utilizará teclado y ratón para la interacción con el usuario, de la siguiente forma:

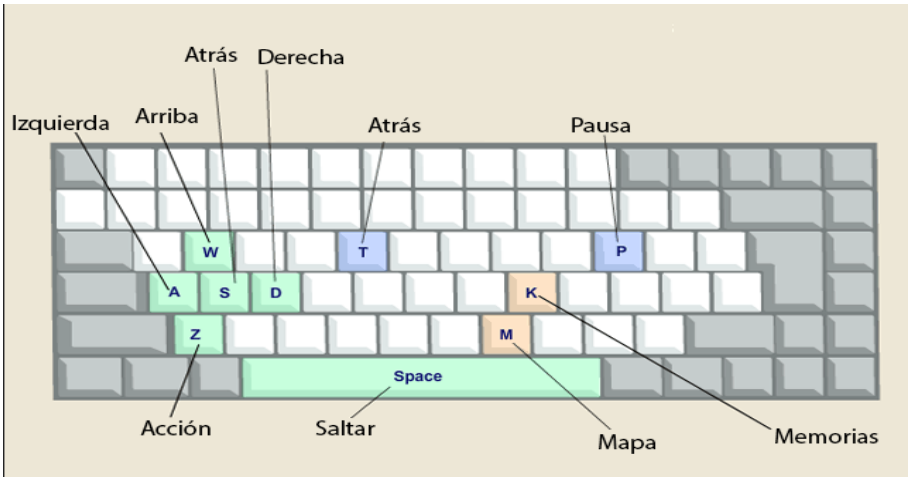


***Figura 69**: Teclado (Asignación de teclas)*

Tecla	Acción
w	Moverse arriba
A	Moverse izquierda
S	Moverse atrás
D	Moverse derecha
Z	Acción
P	Pausa
M	Mapa
K	Memorias

Fuente: Elaboración Propia

5.5. CÁMARA

Se utiliza perspectiva lateral 2D en algunos escenarios y se desplaza lateralmente conforme avanza el jugador a la izquierda o derecha o incluso hacia arriba o abajo, sin embargo, los escenarios, personajes y objetos al estar desarrollados en 3D, se le presentará al usuario una perspectiva muy interesante y con una profundidad fácilmente visible al ojo humano. En la mayoría de los escenarios, la vista estará desde atrás del personaje.

5.6. PERSONAJES Y ENEMIGOS

Para el video juego se utiliza un personaje principal, que se llama Michael, es un adolescente de 13 años, que por el trauma que tuvo sobre la muerte de la madre en frente de su presencia, desarrolló un mundo imaginario en su mente donde queda atrapado, ahora para poder salir de este mundo debe buscar la manera de ir avanzado entre los niveles.

La estética corresponde con un niño que utiliza vestimenta deportiva, su comportamiento es triste, melancólico, sin sentimientos, como si estuviera vacío por dentro.

Michael debe interactuar con los objetos que se presentan en los niveles, no tiene ningún tipo de arma, solo tiene una sola vida, en la que, si es atrapado por los enemigos, debe reiniciar el nivel e intentar superar el nivel.

Enemigos:

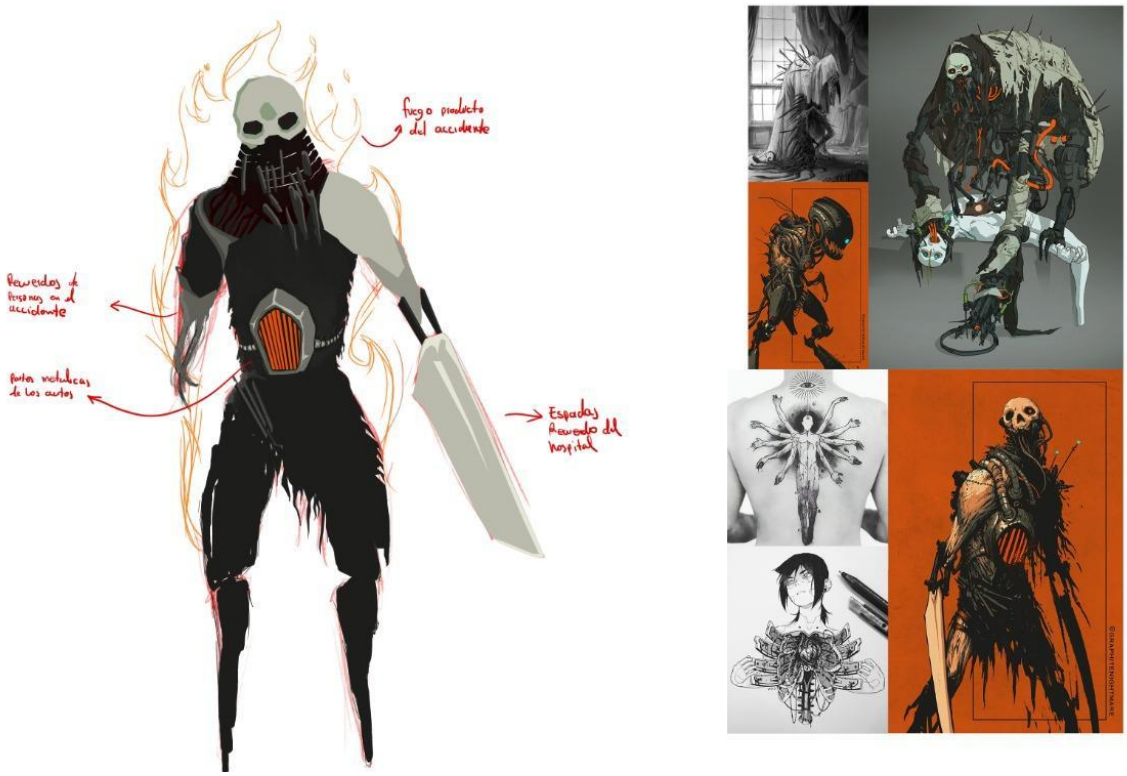


***Figura 70**: Enemigo (Diseño propio de un enemigo)*

De acuerdo como mencionamos anteriormente en el análisis de referentes, entre los recursos a utilizar para producir miedo, se encuentran como instrumentos cortantes que se utiliza en un hospital, restos de vehículos, fuego que se acercaba a Michael y personas con heridas todo producto del accidente.

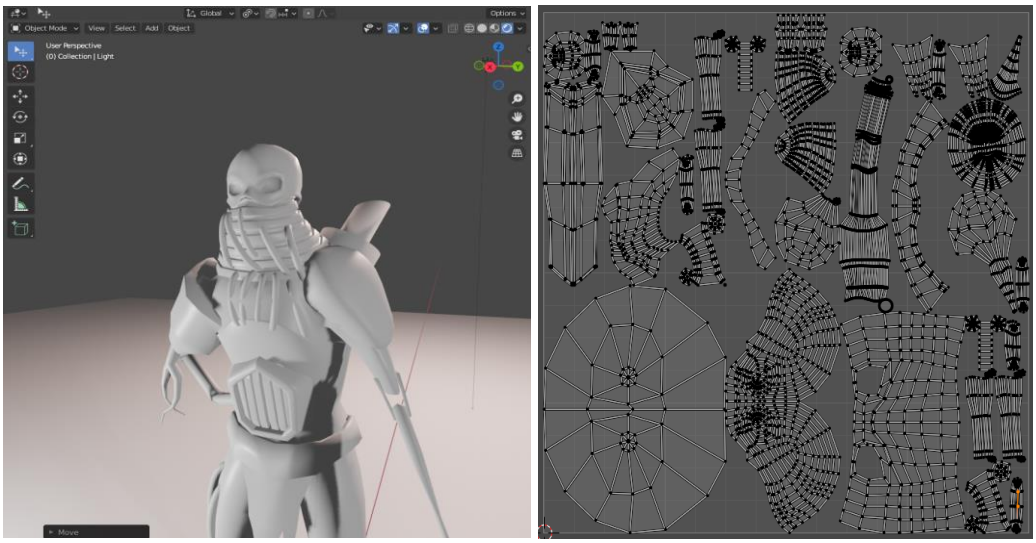


***Figura 71**: Diseño enemigo (Diseño esqueleto y malla de enemigo)*

El modelado 3D se realizó en Blender. La malla se creó a partir de una figura primitiva utilizando la técnica de box modeling y se creó la estructura general de un humanoide. A partir de ese modelo base, se agregaron más detalles como la cabeza, espada en el brazo y elementos metálicos. Una vez terminado el modelado se procedió a sacar los mapas UV para el proceso de texturación.

Adicionalmente, en este juego se utilizaron los enemigos del paquete de la tienda llamado "Stitch Creature Pack", como se puede observar en la sección 7.2.2.

5.7. NIVELES

Para el desarrollo de los niveles inicialmente se desarrollará en 3 niveles.

- Nivel 1 - **Hospital**:

***Figura 72**: Nivel 1 (Escenario hospital)*

Este nivel se encuentra ambientado en un hospital, buscando representar la tristeza que sufrió el personaje por la pérdida de su madre, donde falleció tras el accidente ocurrido.

En este nivel, se debe buscar los siguiente objetos llaves y una pirámide, estos objetos se encuentran esparcidos en todo el escenario.

En este nivel existen enemigos que estarán patrullando, por lo que no se podrá realizar mucho ruido para no ser detectado y tampoco ser atacado por estos. Debemos ser sigilosos, buscar en todos los rincones los objetos necesarios para superar este nivel.

- Nivel 2 - **Casa**:

***Figura 73**: Nivel 2 (Escenario casa)*

Este nivel representa la soledad que nuestro personaje siente en su interior, por lo que al estar en su casa solo sin su madre, tiene la desesperación de salir de aquí, por lo que tendrá que buscar objetos que le ayuden a poder localizar la salida.

En este nivel también existen enemigos que se encuentran patrullando por toda la zona, por lo que tendremos que ser sigilosos y no hacer mucho ruido, para no ser localizado.

Aquí encontraremos un puzle de buscar objetos en un orden específico, para poder desbloquear la salida.

- Nivel 3 - **Mazmorra**:

***Figura 74**: Nivel 3 (Escenario mazmorra)*

Este nivel representa la desesperación de encontrar al culpable por la muerte de su madre, a causa de un desconocido.

En este nivel existen enemigos donde igual que los niveles anteriores, estarán patrullando, por lo que tendremos que ser sigilosos y esquivar a los enemigos que se encuentren aquí.

Se debe buscar algunos objetos que son necesarios para poder superar el nivel, así mismo los diferentes puzles que deben ser resueltos para poder avanzar en el nivel.

- **Niveles extras**: existirán 2 niveles más donde serán representados otros sentimientos del personaje.

5.8. OBJETOS E ÍTEMS

Para el desarrollo del video juego, se establecieron una serie de objetos que puede utilizar el jugador, se encuentran en el entorno del nivel, como pueden ser cajas, donde empuje para poder acceder alguna parte alta, también puede usar cuerdas colgantes, empujar palancas para activar puertas, switch de encendido de luz, lugares para esconderse como mesas.

5.9. DISEÑO DE LA UI Y SCREEN FLOW

5.9.1. UI

De acuerdo con la estética seleccionada, el diseño de la interfaz de usuario o “UI” tendrá un diseño minimalista con iconos estilizados y simples que ayudarán a la correcta lectura y no generar ruido visual para que el jugador se enfoque en la jugabilidad y los escenarios. Se tendrá disponible un diario en donde se guardarán los recuerdos descubiertos para que el jugador pueda acceder a ellos en busca de pistas o la historia completa.

Al obtener un objeto se desplegará un pequeño recuadro detallando brevemente la funcionalidad de este y un icono que lo represente.

No contaremos con vida o recursos ya que sí atrapan al jugador o cae fuera del mapa, regresará a un punto específico del mapa.

El jugador contará con un menú de inventario donde podrá guardar ítems que encuentre en los niveles.

5.9.2. Screen Flow

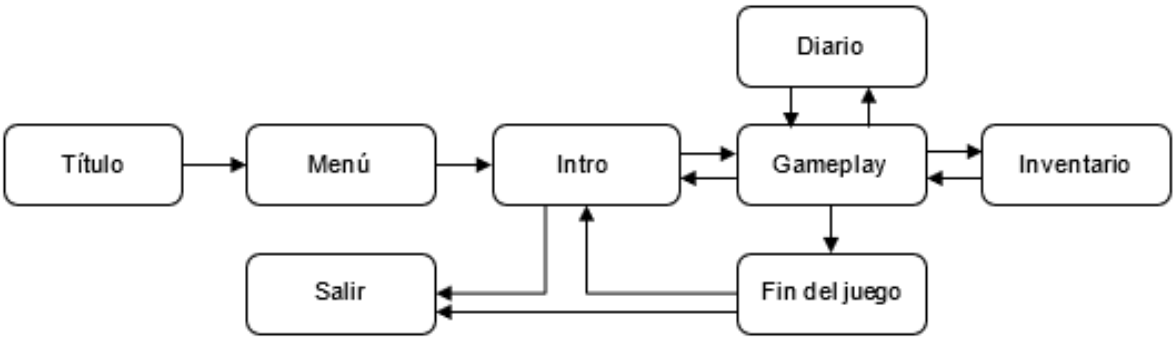


***Figura 75**: Flujo de juego*

6. CONCLUSIONES

Una vez cumplida con la implementación del videojuego, podemos demostrar que a través de los niveles se transmiten los sentimientos que tiene el personaje.

Al escoger correctamente la herramienta de desarrollo del videojuego, se puede demostrar la fácil manipulación de los objetos ya creados y poder reutilizar en la creación de nuevos niveles o escenarios.

Entre las ventajas que podemos encontrar, es la posibilidad de tener el código fuente con un controlador de versiones y que el repositorio se encuentre en la nube, utilizando “Git Hub”, que permite tener un respaldo del código fuente, poder revertir cambios y ver histórico de cambios, así como poder trabajar en equipo sin importar la ubicación geográfica de cada integrante.

El asignarse actividades individuales, es mucho más rápido en el avance de cualquier proyecto que se quiera diseñar o crear.

7. ANEXOS

7.1. ANÁLISIS DE COSTO

De acuerdo con los valores actuales de mercado (febrero 2022), se estima que el costo total que tomará completar el desarrollo del video juego al 100% será de 740 mil euros aproximadamente, como podemos ver en las siguientes imágenes de referencia.

El costo total de implementación sería de 737,446.16 €.

Phases	Concept	Pre-production	Production	Testing	Post Production	TOTAL	Total por team
Months	1	1	5	1	1	9	247
Team							
Artist	1.0	2.0	10.0	2.0	1.0	56	167194
FX	0.0	0.0	5.0	1.0	0.0	26	77626
Animators	0.0	1.0	5.0	1.0	0.0	27	80612
Designer	1.0	1.0	8.0	2.0	1.0	45	134353
Programer	1.0	1.0	8.0	2.0	1.0	45	134353
Audio	0.0	1.0	3.0	1.0	0.0	17	50755
QA	0.0	0.0	0.0	2.0	0.0	2	5971
Producer	1.0	1.0	5.0	1.0	1.0	29	86583
Total # persons	4	7	44	12	4	247	737446.16
MM	2,985.61 €						
Total cost	11,942.45 €	20,899.28 €	656,834.64 €	35,827.34 €	11,942.45 €	737,446.16 €	
Other Costs							
Outsourcing	0.00 €	0.00 €	0.00 €	0.00 €	0.00 €	0.00 €	
Port	0.00 €	0.00 €	0.00 €	0.00 €	0.00 €	0.00 €	
Localization	0.00 €	0.00 €	0.00 €	0.00 €	0.00 €	0.00 €	
OST	0.00 €	0.00 €	0.00 €	0.00 €	0.00 €	0.00 €	
Licensing	0.00 €	0.00 €	0.00 €	0.00 €	0.00 €	0.00 €	
Total Other Costs						0.00 €	
TOTAL (in €)						737,446.16 €	

***Figura 76**: Costo implementación*

Se necesita un total de 67 recursos para desarrollar el video juego, están divididos para cada mes de desarrollo y para cada área específica de desarrollo.

	Concept	Preproduction	Production	Production	Production	Production	Colchón	Testing	Archive
Mes	1	2	3	4	5	6	7	8	9
Artist	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	1.0
Fx Artist	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0
Animator	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0
Designer	1.0	1.0	1.0	1.0	2.0	2.0	2.0	2.0	1.0
Programmer	1.0	1.0	1.0	1.0	2.0	2.0	2.0	2.0	1.0
Audio	0.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0
QA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
Producer	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Total									
67	4	7	7	7	10	10	10.0	12.0	4.0

***Figura 77**: Tiempo de implementación*

También se analizó realizar un kickstarter, como una opción de financiamiento para el video juego.

***Figura 78**: KickStarter*

7.2.CONTENIDO EXTERNO, DESCARGADO DE LA TIENDA DE UNREAL ENGINE

Para poder llevar a cabo el desarrollo de este prototipo y ante la falta de tiempo y recursos para poder desarrollar todos los personajes y enemigos, se han descargado los siguientes contenidos de la tienda de Unreal Engine.

7.2.1. CARTOON KID

El modelo 3D de nuestro personaje principal, utiliza un mapeo y rigging al esqueleto estándar de Unreal Engine, lo que permite importar animaciones que utilicen este esqueleto. Adicionalmente, fue el modelo 3D que más se asimilaba a nuestra idea de juego.

***Figura 79**: Personaje (Unrealengine, 2021)*

7.2.2. STITCH CREATURE PACK

Este paquete de enemigos nos permite tener varios enemigos que tienen un estilo de arte que puede aplicar a nuestro juego, la idea es que cada uno de estos enemigos, representan algunos de los temores y traumas psicológicos de nuestro personaje.

***Figura 80**: Enemigos (Unreal Engine, 2021)*

8. POSTMORTEN DEL JUEGO

Para la creación del videojuego, primero se realizó una lluvia de ideas, con estas lluvias de ideas se escogieron los puntos más fuertes y llamativos.

Los puntos más fuertes fueron el tipo de juego, sería suspenso psicológico, con mecánicas de búsqueda de objetos, pequeños acertijos, etc.

- **Lo que salió bien**.

  Fue la de asignarse a crear un nivel cada integrante, en este caso se crearon 3 niveles, donde cada escenario es diferente, como también su ambientación.

  Las constantes pruebas que realizaba el equipo a los niveles, para poder pulir los aspectos de los escenarios en el que faltase algo, el feedback respectivo de cada diferente mecánica que existan en el nivel.

  Mejoras que se le puede añadir a la jugabilidad y experiencia de usuario.

- **Correcciones**.

  Al iniciar el desarrollo del videojuego se tenía ideas muy grandes, por lo que al intentar desarrollar no alcanzaría el tiempo asignado para el proyecto, por lo que se optó diseñar pequeñas ideas que sean alcanzables en el tiempo asignado.

  Inicialmente los escenarios serían en forma solo de un cuarto o habitación, por lo que obteniendo un feedback externo, en la que, para mejorar la experiencia de juego, debe tener escenarios más grandes, en la que existan muchas más habitaciones que se pueda ir explorando.

9. REFERENCIAS BIBLIOGRÁFICAS

Silent hill two. (01/2020). Hobbyconsolas. Recuperado 1 de febrero de 2022, de https://cdn.hobbyconsolas.com/sites/navi.axelspringer.es/public/media/image/2020/01/silent-hill-two-new-games-1848249.jpg

Inside. (27/12/2017). Meristation. Recuperado 1 de febrero de 2022, de https://as01.epimg.net/meristation/imagenes/2017/12/27/noticia/1514397960_856224_1532076852_sumario_normal.jpg

Playdead. (28/06/2018). Limbo. Nintendo. Recuperado 1 de febrero de 2022, de https://www.nintendo.com/es_LA/games/detail/limbo-switch/

Playdead. (19/11/2022). Inside. Twitter. Recuperado 2 de febrero de 2002, de https://twitter.com/Playdead/status/1329454877085478912/photo/1

Tarsier Studios. (27/04/2017). Little Nightmares. Steam. Recuperado 2 de febrero de 2022, de https://store.steampowered.com/app/424840/Little_Nightmares/

RodionSadykov. (02/04/2021). Stitch Creatures Pack. Unreal Engine. Recuperado 3 de febrero de 2022, de https://www.unrealengine.com/marketplace/en-US/product/stitch-creatures-pack

Taylor Brook Music. (10/11/2015). Sound Phenomenon Fantasy Orchestra. Unreal Engine. Recuperado 4 de febrero de 2022, de https://www.unrealengine.com/marketplace/en-US/product/fantasy-orchestral-music

John Leonard French. (08/05/2018). Complete Horror. Unreal Engine. Recuperado 4 de febrero de 2022, de https://www.unrealengine.com/marketplace/en-US/product/complete-horror

Leshiy3d. (08/08/2021). Cartoon kid. Unreal Engine. Recuperado 4 de febrero de 2022, de https://www.unrealengine.com/marketplace/en-US/product/cartoon-kid?sessionInvalidated=true