Brave//Soulless: Juego de Lucha

Elías Maidanik

Resumen — Este proyecto consiste en el diseño, implementación y testing de una parte de un videojuego en 3D que hemos llamado *Brave//Soulless*, desarrollado sobre un motor estándar en la industria de los videojuegos: Unity3D. El objetivo del proyecto es la creación de un prototipo de videojuego en el cual probar las mecánicas de juego tales como combate con entidades controladas por una inteligencia artificial, movimiento en un escenario 3D y el accionar interruptores que abran nuevos caminos para el jugador.

Palabras clave — Videojuego, Unity3D, C#, inteligencia artificial, Blender, 3D, PC, tercera persona, gameplay, diseño de videojuegos

Abstract— This proyect consists on designing, implementing and testing a part of *Brave//Soulless*, a 3D videogame developed with Unity3D, an engine that's become standard in the videogame industry. The project's objective is to create a videogame prototype in which to test the gameplay mechanics such as fighting against AI-controlled entities, movement in a 3D environment and interacting with switches that open new paths for the player to take.

Index Terms— Videogame, Unity3D, C#, artificial intelligence, Blender, 3D, PC, third person, gameplay, game design

1 Introducción

El género de videojuegos de lucha es uno de los más antiguos de la industria. Tuvo sus comienzos en juegos que pretendían simular combates de boxeo pero pronto se ha expandido a todo tipo de escenarios y temáticas diferentes, siendo los más populares los relacionados a las artes marciales o a la simulación de deportes de lucha reales tales como la lucha libre o el kick-boxing.

Originalmente, los videojuegos de lucha tan solo enfrentaban a un jugador contra una inteligencia artificial pero pronto el modo multijugador de uno contra uno se volvió el más prominente. Aún así, algunos juegos de lucha han experimentado con mayor cantidad de jugadores enfrentándose todos contra todos o por equipos.

La mayoría de mecánicas y convenciones actuales de los juegos de lucha, tales como la posibilidad de cancelar la animación de un ataque para inmediatamente activar otro para crear un combo, la disposición de la interfaz de usuario y la variedad de luchadores con estilos de combate, han surgido de Street Fighter II [1].

Brave//Soulless se inspira principalmente en este género de videojuegos, sumando algunos elementos de exploración y aventuras para generar una experiencia dinámica y diferente.

El jugador puede recorrer diferentes pantallas en sucesión lineal, cada uno conteniendo una combinación diferente de desafíos que pueden ser de tres tipos:

- **Desafíos de Destreza**: El jugador deberá correr y saltar para esquivar obstáculos
- **Desafíos de Astucia:** El jugador deberá encontrar interruptores o resolver algún tipo de problema con las herramientas que tenga disponibles para poder avanzar en el juego.
- **Desafíos de Combate:** El jugador usará las habilidades de lucha de su personaje para derrotar a enemigos controlados por una inteligencia artificial.

A medida que el juego avance, el jugador será recompensado con nuevos estilos de lucha y formas de afrontar los desafíos de pantallas futuras, debiendo aprender a combinar todas estas habilidades para poder atravesar todos los obstáculos que encuentre.

Shigeru Miyamoto, creador de algunos de los videojuegos más innovadores, exitosos y aclamados de la industria de videojuegos [2] reconoció que durante la creación del revolucionario videojuego Mario 64 pasó meses con un prototipo básico que tan sólo contaba con las mecánicas de juego y un escenario sencillo en el cual poder probarlas.

Tan sólo comenzó con el desarrollo formal del videojuego una vez estuvo seguro de que todas las piezas encajaban y que la actividad de desplazarse por el entorno (la mecánica de juego principal en Mario 64) resultase divertida en sí misma [3].

Es por eso que el principal objetivo de este proyecto es desarrollar el motor de jugabilidad de *Brave//Soulless* para poder prototipar y experimentar sobre las mecánicas básicas de juego de modo tal que se pueda refinar el "Game Feel" o la sensación que causa el juego sobre su jugador. Para esto, el prototipo ha de incluír:

- Un sistema de combate
- Un sistema de desplazamiento
- Una inteligencia artificial que sea capaz de poner a prueba al jugador.

Este proceso de iteración sobre las mecánicas básicas ayudaría a determinar cuales son los puntos más divertidos e interesantes del juego, lo que lo diferencie de otros títulos similares. El contenido de pantallas, personajes jugables adicionales, variedad de enemigos y gráficos serán algo a desarrollar a futuro.

Mi motivación personal para desarrollar este proyecto surge de mi pasión por los videojuegos y por contar historias. Siempre he sido un narrador por naturaleza y

mi fascinación por los medios interactivos, tales como los videojuegos, me lleva a querer no sólo desarrollar este proyecto sino dedicar mi vida a crear tantos juegos como pueda. Este no es mi primer paso en esta dirección pero sí que es el primero que he desarrollado con profesionalidad, siguiendo estándares de la industria y asegurando mantener una buena documentación en vez de improvisar. Desde el punto de vista narrativo, los videojuegos de lucha siempre me han fascinado por el esfuerzo necesario para caracterizar un personaje cuando usualmente sólo se le ve luchando.

Adicionalmente, también me atraen debido a la experiencia que ofrecen en entornos sociales, donde varios jugadores se reúnen para ver quien es el mejor.

El desarrollo de este proyecto conlleva una gran oportunidad de aprendizaje sobre las metodologías de trabajo a nivel profesional en el ámbito de la creación de contenido interactivo y de las aplicaciones a tiempo real.

Mi objetivo es seguir con el desarrollo de este videojuego una vez el proyecto sea entregado para incorporar una experiencia guiada para un jugador que cuente la historia de *Brave*, el protagonista de juego, y el mundo que habita.

La principal herramienta de trabajo para el desarrollo de este proyecto ha sido Unity3D [4], una combinación de motor gráfico, herramienta de diseño de pantallas y contenido de juego así como entorno de programación, todo ello diseñado con el objetivo de facilitar y acelerar el proceso de creación de contenido interactivo multiplataforma, orientado especialmente al ámbito de los videojuegos.

Unity implementa una gran variedad de utilidades básicas tales como cámaras, efectos visuales, rendering y un editor que permite visualizar los cambios efectuados en tiempo de ejecución. También incluye un servicio llamado Asset Store [11], el cual ofrece una variedad de diversos recursos gratuitos o de pago. Para este proyecto se han utilizado solo dos recursos de la Asset Store, ambos gratuitos: El primero, una librería llamada Bitstrap [5] que contiene utilidades varias para facilitar el prototipado en Unity3D, el segundo ha sido una serie de animaciones ofrecidas gratuitamente en conjunto con un modelo de personaje (el modelo no ha sido utilizado para el proyecto), llamado Taichi Character Pack [6]. También se han utilizado animaciones de un paquete llamado Huge FBX Mocap Library [7], construido en base a animaciones gratuitas de motion capture ofrecidas por la universidad de Carnegie-Mellon. Estos últimos dos paquetes han sido utilizados debido a la necesidad de acortar el tiempo invertido en arte.

Para la programación de la lógica del juego, inteligencia artificial y de las herramientas necesarias para implementar el contenido presente en el prototipo, se ha utilizado el lenguaje C# sobre el editor MonoDevelop, integrado en Unity3D.

El modelo de personaje fue creado en Blender [8], una herramienta de modelado 3D. Esta herramienta ha sido seleccionada debido a ser gratuito, tener alta calidad calidad y prestigio además de la gran facilidad para encontrar recursos educativos sobre su uso.

Para mantener un seguimiento del proyecto se ha utilizado Asana [9], un servicio web diseñado para la gestión colaborativa de tareas de un proyecto.

Como contenido de este artículo, en la sección 2 se explica el diseño del videojuego, comenzando por un breve resumen de la historia y las mecánicas de juego.

En la sección 3 se detalla el proceso de desarrollo del prototipo.

La sección 4 se encuentran los resultados y las validaciones realizadas.

La sección 5 muestra las conclusiones del proyecto y las lineas de desarrollo futuro.

Finalmente se expondrán los agradecimientos, una reseña de las referencias utilizadas así como un apéndice que contendrá con capturas de pantalla del juego en ejecución.

A lo largo de este documento se utilizarán los conceptos que definimos a continuación:

- **Pantalla :**Un conjunto de desafíos o contenidos de juego englobados dentro de un escenario en concreto. El límite entre escenas está marcado por la necesidad de cargar nuevos datos de escenario a memoria.
- **Personaje:** El conjunto de un modelo 3D y las mecánicas de juego necesarias para que un jugador o inteligencia artificial pueda controlarlo.

2 Diseño del juego

Se partió de la base que el juego sería eventualmente desarrollado en su completitud y por lo tanto se han planteado y diseñado todos los aspectos necesarios para que el juego completo provea una experiencia atractiva.

Primero veremos la historia y ambientación del juego, el contexto del mundo que visitará el jugador durante su experiencia.

Seguidamente serán introducidos los diferentes documentos de diseño que fueron desarrollados. Nótese que esto sólo pertañe al diseño de juego y no al del código o estructura del proyecto en sí.

Finalmente se explicarán las mecánicas de juego a resaltar.

2.1 Contexto

El primer paso para crear un diseño de juego interesante es definir el marco contextual que englobará toda la acción: ¿Cómo es el mundo, cuales son sus reglas? ¿En qué se diferencia del nuestro? ¿Cuál es el conflicto que le da contexto a lo que el jugador haga o pueda hacer durante su experiencia con el juego?

- *E-mail de contacto: eliasmaidanik@gmail.com*
- *Mención realizada: Computación*
- *Trabajo tutorizado por: Enric Martí Godia (departament)*
- *Curso 2015/16*

En el caso de *Brave//Soulless*, la ambientación circula alrededor de un mundo de fantasía con magos, guerreros, reyes, artefactos místicos y dragones: El juego pleantea un mundo en el cual los villanos ya han sido victoriosos y han hecho caer el mundo en la oscuridad y locura.

El jugador tomaría el papel de *Brave*, un humano creado mediante alquimia y tecnología mágica con el propósito de restaurar la paz al mundo. Por este motivo, *Brave* no posee un alma o personalidad propia pero es capaz de absorber la de quienes derrote, obteniendo tanto sus poderes como parte de la humanidad que han perdido.

2.2 Documentos de Diseño

Para el desarrollo de *Brave//Soulless* se han escrito y desarrollado una variedad de extensos documentos que describen como podría ser el juego en su versión completa, más allá del ámbito de este proyecto. Todos estos documentos siguen el estándar de la industria, utilizados en cualquier proyecto de videojuegos que quiera considerarse profesional.

2.2.1 Game Concept

Un documento de 28 páginas destinado a la descripción conceptual del juego. Habla de ideas, historias, personajes y mecánicas desde un punto de vista abstracto. Describe a 11 personajes que pueden ser controlados tanto por el jugador o por una inteligencia artificial hostil, 4 personajes secundarios de relevancia para la trama pero que no participan en combate, 2 enemigos que no pueden ser controlados por ningún jugador pero pueden enfrentarse a él y modos de juego de uno o más jugadores. Adicionalmente, resume la idea de 12 posibles pantallas y la historia que contarían.

Describe también el tipo de entornos que podrían verse a lo largo de estas pantallas y como sería la jugabilidad.

2.2.2 Story Bible

Un documento de 15 páginas que trata exclusivamente sobre la historia del juego. Toma cada personaje, pantalla y detalle sobre la historia que se encuentre presente en el Game Concept o el Game Design Document y lo explora desde un punto de vista narrativo.

Habla no sólo de la historia que experimentará el jugador sino del pasado de cada personaje y de como llegó el mundo al estado en el que está.

Parte de la historia de los diversos personajes no ha sido acabada, por lo demás se encuentra prácticamente completo.

2.2.3 Game Design Document

Un documento de 25 páginas que explica el diseño de los aspectos jugables del videojuego. Se lo considera básico e imprescindible en el mundo del desarrollo de videojuegos. Este documento detalla mecánicas de juego, habilidades de los personajes, diseño de los pantallas, controles, estilos gráficos, recursos visuales necesarios, diseño de la inteligencia artificial y modos de juego.

En cuanto al diseño de pantallas, habilidades y personajes, tan sólo se ha escrito lo relevante a las primeras pantallas del juego. El resto de secciones ha sido desarrollado al cien por ciento.

2.3 Mecánicas de juego

De los documentos de diseño se han extraido una serie de mecánicas de juego que fueron consideradas básicas para poder crear un prototipo representativo del juego final. A lo largo de esta sección se discutirán las más relevantes: Los cambios en una escena, las habilidades que puede ejecutar el jugador o una inteligencia artificial, los controles y su relación con los personajes en pantalla y finalmente las condiciones de victoria y derrota.

2.3.1 Cambios en la Escena

Una parte de las necesidades para representar este mundo se encuentra en el contenido artístico: Música suave y triste para ambientar la desolación; colores apagados y fríos para reflejar la depresión que envuelve el mundo. Otra parte ha de ser cubierta por los sistemas y mecánicas de juego.

Para empezar, es importante que el éxito del jugador sea recompensado por un cambio en la tonalidad de la escena: Una compuerta cerrada puede tener un color azul o mientras esté bloqueada pero en cuanto el jugador consiga desbloquearla, debería iluminarse en un brillante amarillo, simbolizando la esperanza resurgiendo a medida que el protagonista se acerca a la victoria. Este concepto puede extrapolarse a muchos otros aspectos de la parte estética.

2.3.2 Habilidades

Si *Brave* es capaz de absorber esas almas y obtener nuevos poderes en nuestra historia, el jugador debería notar esas diferencias: Cada cierto tiempo, el jugador debe recibir nuevas habilidades como recompensa de sus éxitos, haciéndolo más poderoso y dándole más herramientas a utilizar en futuros desafíos. Esto también incrementa la dificultad de los mismos ya que se volverían más complejos y requieran del uso de más cualidades.

Dado que las habilidades nuevas están asociadas a las almas de otros personajes, lo más adecuado es que cuando el jugador quiera utilizar una de las habilidades nuevas, *Brave* pase a moverse y actuar como el personaje derrotado. Para esto se utilizará una mecánica similar a la que puede verse en los juegos de Megaman [10], en los que tras derrotar al enemigo más poderoso de una cierta pantalla el jugador obtiene acceso a su arma, pudiendo en cualquier momento equiparla y utilizarla en futuras pantallas.

Brave necesita estas almas para subsistir, recuperar vitalidad consistirá en derrotar enemigos menores y recoger pickups.

2.3.3 Controles

Debido a la variedad de habilidades que estarían en el juego, es importante que una misma combinación de inputs cumpla con un propósito similar sin importar qué alma se encuentra equipada. Para conseguir esto, se limitan los controles a los siguientes:

- Movimiento
- Salto
- Voltereta (Rodar)

- Ataque
- Poder

Los efectos de estas teclas dependen tanto del personaje actual (o en el caso del personaje protagónico *Brave*, las habilidades equipadas) como del estado actual del jugador, como se puede apreciar en la figura 1.

Los recuadros representan estados del personaje, las flechas representan las posibles transiciones entre estados y los círculos asociados a cada estado representan los ataques y poderes que pueden ser ejecutados desde ese estado. Las líneas que conectan los círculos con los recuadros representan un input que puede ser la tecla de Ataque o la de Poder. Existe la posibilidad de que un personaje se encuentre en un estado desde el cual no sea capaz de ejecutar ningún tipo de ataque tras recibir daños: El período de recuperación no está representado en esta figura.

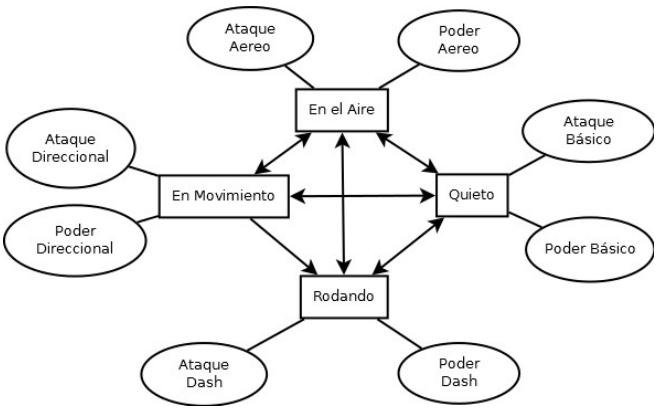

Figura 1: Acciones disponibles según el estado

Presionar la tecla de Ataque mientras se está en movimiento y en el suelo siempre ejecutará algún tipo de habilidad ofensiva que desplazará al jugador hacia adelante pero la habilidad en cuestión puede tener propiedades o animaciones completamente diferentes.

2.3.4 Victoria/Derrota

Para vencer en una pantalla siempre será necesario llegar a un punto en concreto, el cual será señalizado de alguna manera para que el jugador sepa identificarlo. Por el camino habrán diversos desafíos que no permitirán al jugador avanzar hasta que sean superados. Estos desafíos consisten en derrotar a una cierta cantidad de enemigos menores, derrotar a un enemigo poderoso, ejecutar algún tipo de desplazamiento por un terreno peligroso, encontrar una manera de abrir una compuerta o una combinación de las anteriores. En caso de tener éxito y llegar al objetivo final, el jugador sería recompensado y luego llevado a la siguiente pantalla.

- *E-mail de contacto: eliasmaidanik@gmail.com*
- *Mención realizada: Computación*
- *Trabajo tutorizado por: Enric Martí Godia (departament)*
- *Curso 2015/16*

Para ser derrotado, tan solo será necesario que la vitalidad de *Brave* se agote, obligando al jugador a recomenzar la pantalla desde el principio. Los errores que cometa el jugador a lo largo de la pantalla se traducirían en una pérdida de vitalidad.

3 Desarrollo del Prototipo

Esta sección está dedicada a la explicación de los pasos por los cuales se han llegado a obtener el prototipo funcionando. El orden de las secciones imita el orden en el cual se desarrolló cada elemento del prototipo: Arte, mecánicas de desplazamiento, mecánicas de combate, mecánicas de exploración y finalmente inteligencia artificial.

3.1 Arte

El prototipo contiene un personaje *(Brave)*, el cual cuenta con un modelo de pruebas que será utilizado para representar tanto al jugador como a sus enemigos. Este modelo ha sido desarrollado específicamente para el juego utilizando Blender.

El proceso de creación del modelo comenzó por el diseño conceptual del personaje (presente en el documento "Game Concept") hasta el desarrollo del concept art (visible en la figura 2), realizado por un artista externo al proyecto.

Figura 2: Concept Art de Brave

Tras eso, se progresó con el modelado del personaje (work in progress visible en la figura 3) hasta llegar a un resultado relativamente aceptable (figura 4).

Se han modificado los shaders ofrecidos por defecto por Unity3D para obtener un efecto de Cel-Shading, visible en la figura 4. Este modelo no es perfecto ni completo pero fue necesario recortar el tiempo dedicado al arte para poder enfocar más tiempo en el desarrollo de otras partes del proyecto.

Como assets externos, el juego contiene una serie de figuras geométricas 3D simples para poblar los escenarios y un conjunto de animaciones.

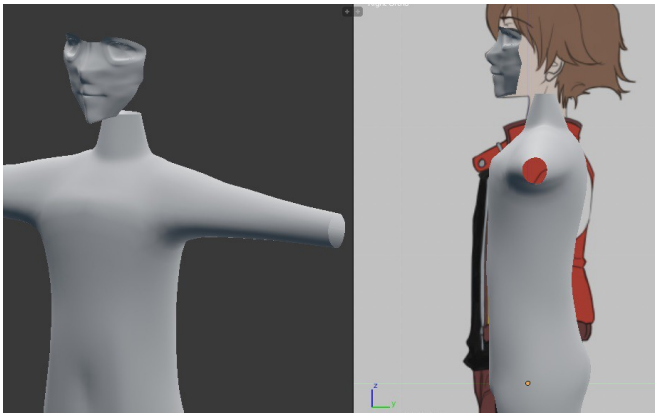

Figura 3: Modelado de Brave en Blender

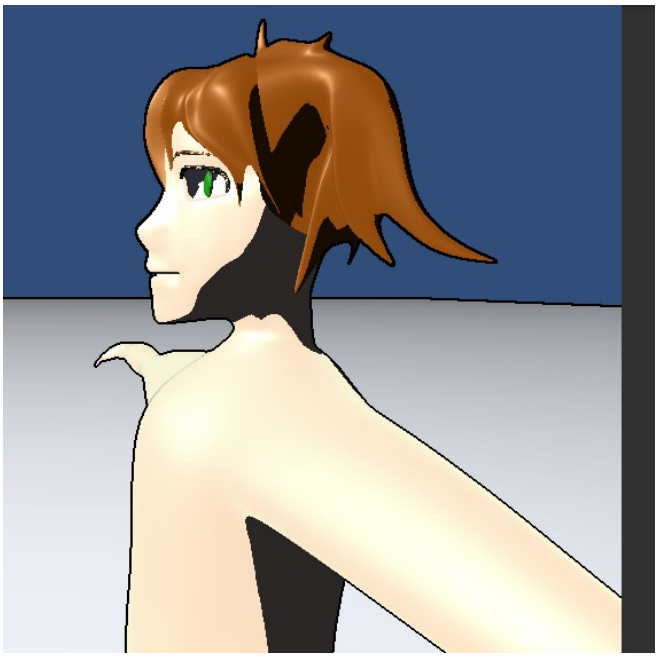

Figura 4: Modelo visualizado en Unity3D con Cel-Shading aplicado

3.2 Gameplay

Esta sección trata sobre las diferentes piezas que fueron desarrolladas para hacer jugable el prototipo.

Primero se explicará el funcionamiento general del motor de gameplay desarrollado. Seguidamente habrán varias sub-secciones explorando diversas piezas del funcionamiento del juego.

El motor de juego está dividido en 7 módulos principales, visibles en la figura 5.

Unity3D recibe los comandos de teclado o mando y los envía al módulo de Player Input, el cual procesa los comandos del jugador y los envía al personaje correspondiente.

El módulo Character Control se encarga de procesar los comandos recibidos, el estado actual del personaje y los movimientos y ataques que se encuentren en ejecución, ofreciendo esta información a los módulos de Game Logic y World para que puedan reaccionar acordemente.

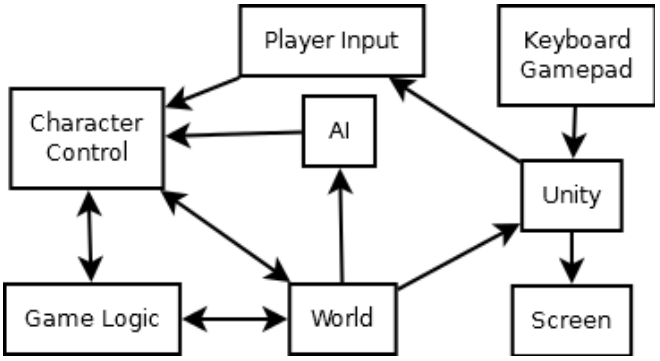

Figura 5: Módulos del motor de jugabilidad

El módulo de Game Logic procesa cambios de escena, cambios de modo de juego, final de partida, mostrar o no y actualizar la interficie de usuario.

El módulo World recibe información del control de jugador y de la lógica del juego para efectuar cambios sobre la geometría del escenario. También se encarga de actualizar el mapa del entorno que utiliza la inteligencia artificial para buscar caminos y de retener el estado actual de interruptores o puertas.

El módulo de AI recibe información del mundo para detectar por donde ha de mover a los agentes inteligentes y reconoce qué peligros existen. Utiliza esa información para actualizar el personaje del agente.

Finalmente, Unity recibe la información del estado del mundo y la envía a pantalla. Además, recibe las interrupciones de teclado o mando para enviarselas al módulo de Player Input.

A continuación, explicaremos algunos de los componentes presentes en el módulo Character Control y World.

3.2.1 Character Control

Debido a la gran cantidad de movimientos, ataques y habilidades que deberían ser posibles en *Brave//Soulless*, se ha desarrollado una herramienta que simplifica la creación de acciones a ejecutar por un personaje, ya sea jugador o enemigo.

La herramienta consiste en añadir al editor de Unity3D un nuevo comando que pueda crear "movimientos" y "efectos de movimiento", siguiendo el patrón de "Factory Method" [12].

Un movimiento es una acción que ejecuta un jugador. Se compone de una duración, una animación y una cantidad variable de efectos de movimiento, siguiendo el patrón de diseño "Decorator" [13]. Los efectos de movimiento son piezas que contienen algún tipo de acción que se ha de aplicar sobre el personaje que activa el movimiento. Estas piezas pueden ser aplicarle un movimiento o permitirle herir a otros personajes durante el período de tiempo que dure un ataque.

Existen múltiples tipos de efectos de movimiento, todos descendientes de una misma clase abstracta. Cada efecto de movimiento cumple una función diferente y puede ser configurado de diversas maneras.

Algunos de los efectos de movimiento desarrollados son los siguientes:

- **Desplazar Usuario:** Utilizado para crear saltos o movimientos que empujen al personaje que use el movimiento en una cierta dirección. Puede o no prohibir el uso de saltos o volteretas mientras se desarrolla esta habilidad.
- **Crear Esfera de Colisión:** Dado un margen de tiempo (normalmente dentro de la duración del Movimiento en sí) y una referencia a uno de los huesos del modelo de personaje, crea una esfera de colisión sobre el hueso que puede aplicar ciertos efectos sobre quienes toque. Utilizado para puñetazos y patadas, por ejemplo. La esfera de colisión sigue al hueso creado.
- **Crear Objeto:** Similar a la creación de una esfera de colisión pero en vez de eso crea un objeto nuevo en el mundo. Utilizado para el lanzamiento de proyectiles.
- **Encadenar Movimiento:** Dado un tipo de input y un margen de tiempo, si el jugador presiona esa tecla dentro del margen indicado mientras se ejecuta este movimiento, el movimiento actual será cancelado y uno nuevo se activará. Utilizado para doble saltos y para cadenas de ataques con animaciones diferentes.
- **Cancelación del Movimiento:** Dado un tipo de input y un tiempo de retardo, cancela el movimiento actual si se lo recibe tras el retardo especificado. Dado que uno de los tipos de inputs es el evento generado cuando el jugador toca el suelo, este efecto es utilizado para cancelar saltos.

En la figura 6 puede apreciarse la interface de creación de un ataque aéreo.

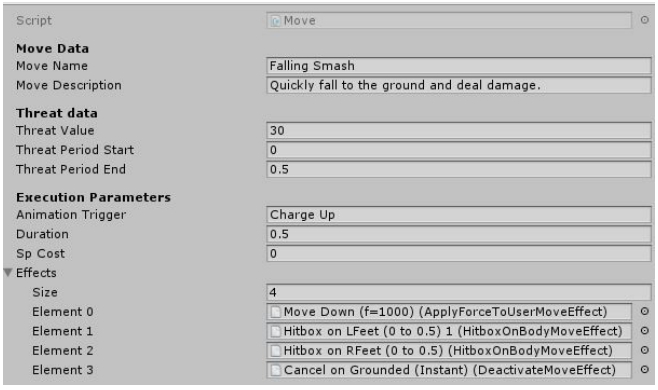

Figura 6: Formulario de creación de movimientos

La sección de "Move Data" del formulario contiene datos cuyo propósito es ser mostrados al jugador para que comprenda qué se supone que hace esa habilidad o acción. En este caso tenemos el nombre de un ataque en caída llamado "Falling Smash" y una pequeña descripción de su funcionamiento.

En la sección de "Threat Data" se define un período en el cual se ajusta el qué tan peligrosa es considerada la entidad que ejecuta este movimiento. Esta información la utiliza la inteligencia artificial para decidir si optará por defenderse o no. En nuestro ejemplo, el peligro del usuario incrementa en 30 puntos desde que el movimiento inicia hasta que transcurra medio segundo.

En la sección de "Execution Parameters" ejecución se define la animación asociada a este movimiento (parámetro "Animation Trigger", en este caso la animación llamada "Charge Up"), qué tanto tiempo ha de transcurrir hasta que el jugador pueda tomar otras acciones o moverse (Definido según "Duration", en este caso es de medio segundo) y un coste en un recurso llamado SP ("SP Cost", si el jugador no tiene suficientes puntos de estos para activar la habilidad, no funcionará. De tenerlos, se los quitará). Finalmente, tiene una lista de efectos (En nuestro caso, 4 elementos dentro del array "Effects").

Cada efecto es un objeto serializado que define como ha de accionar. En nuestro ejemplo tenemos uno que aplica una fuerza hacia abajo en cuanto se usa la habilidad ("Move Down") y crea dos cajas de colisión ("Hitbox on..."), una en cada pie del personaje. Esas cajas de colisión tendrán programada la lógica de detección de enemigos y aplicarán el daño y retroceso correspondiente a quien toquen. También pueden activar interruptores.
Finalmente, el último efecto define que si el jugador toca el suelo, la habilidad se cancela inmediatamente y el jugador es libre de ejecutar las acciones que quiera ("Cancel on Grounded").

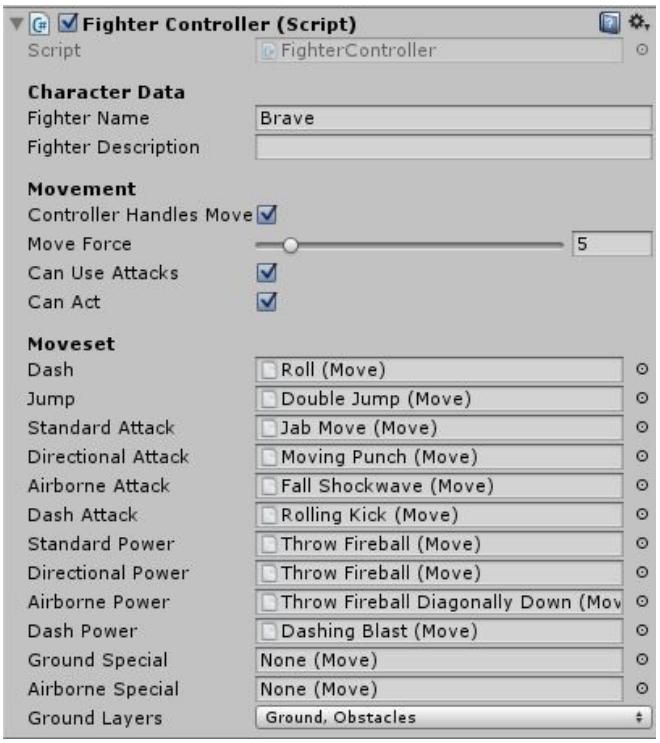

Figura 7: Formulario de un Fighter Controller

En la figura 7 podemos apreciar la configuración del protagonista del juego, *Brave*: Posee un "Name" y "Description" que serán utilizados para informar al

jugador de las características del personaje durante la selección del mismo. A continuación contiene la velocidad a la cual se mueve ("Move Force" de 5 metros por segundo) y si puede o no actuar y atacar. Tras eso está su "Moveset", una colección de movimientos que puede ejecutar mediante la ejecución de los inputs correspondientes. Como se definió anteriormente, existen dos teclas de ataque ("Poder" y "Ataque"): Dependiendo del estado del personaje, los inputs activan movimientos diferentes. Por ejemplo, si el jugador que controla a *Brave* presiona la tecla asociada a utilizar un "Poder" mientras que *Brave* está ejecutando una voltereta (o "Dash"), se activará el movimiento "Dashing Blast".

Existe otro componente que acompaña al "Fighter Controller", llamado "Character Resources", visible en la figura 9. Se encarga de definir el estado actual del personaje ("Hp" representando su vitalidad, "Sp" su energía para ejecutar ataques especiales y "Dead" si ha sido derrotado o no).

Los recursos del personaje no pueden ser inferiores a 0 y sus máximos son marcados por este formulario.cantidad de daño que puede recibir un personaje u objeto antes de "morir" como la cantidad de puntos que posee para ejecutar sus Poderes ("Max HP", "Max SP").

También define el funcionamiento de esta muerte en la sección "On Death", desde la cual se puede especificar un efecto sonoro cuando es destruido, un efecto de particulas que es creado durante esa destrucción ("SFX"), una cantidad de puntos que gana o pierde el jugador cuando es destruido ("Victory Point Modification"), si el objeto entero ha de ser destruido en cuanto muera ("Destroy Self") y si ha de enviar una señal de destrucción a otros componentes del objeto ("Broadcast Kill"). También permite solicitar que se cree una interfaz de usuario para que el jugador sea capaz de ver sus recursos, visible en la figura 8 (la barra naranja representa la vida o "HP", la barra verde para la energía o "SP"). Esta barra está animada tanto cuando crece como cuando decrece. Se encuentra siempre en una esquina de la pantalla.

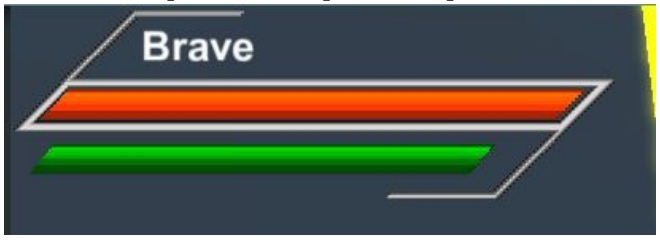

Figura 8: Barras de Vida y Energía

3.2.2 World

El módulo de mundo se ocupa principalmente de factores de la física y posicionamiento de los objetos.

Existen dos componentes a destacar dentro de él: Switch y Door.

El componente Door está programado para simular compuertas, las cuales se mueven en cuanto reciben una señal de activación tras haber sido desbloqueadas. Para desbloquearlas han de recibir una cierta cantidad de señales de desbloqueo. Una vez se activa la secuencia de movimiento, inician una interpolación entre su posición actual a una nueva, apartándose del camino.

El componente Switch, visible en la figura 10, permite regular el funcionamiento de un elemento que puede ser activado para abrir ("Opens Door") o desbloquear ("Unlocks Door") una puerta en concreto ("Target Door"). También define las formas que tiene el jugador de interactuar con él:

- **Trigger on Contact:** Cuando sea tocado por un objeto sólido como el modelo del protagonista.
- **Trigger on Interact:** cuando sea tocado por un objeto no sólido como la caja de colisión de un ataque. "Interaction Layers" sirve para definir el la capa en la que se encuentran los objetos no-sólidos que admite.
- **Trigger on Kill:** Se activa tan solo si el interruptor o uno de sus objetos padre también tiene un componente de Character Resources. Cuando el Character Resources envía una señal de destrucción, el interruptor se considera activado.

Finalmente, el interruptor puede cambiar el material ("Chance Material") del objeto que posee este componente, pudiendo así darle un feedback visual al jugador para que sepa que el interruptor ha sido activado.

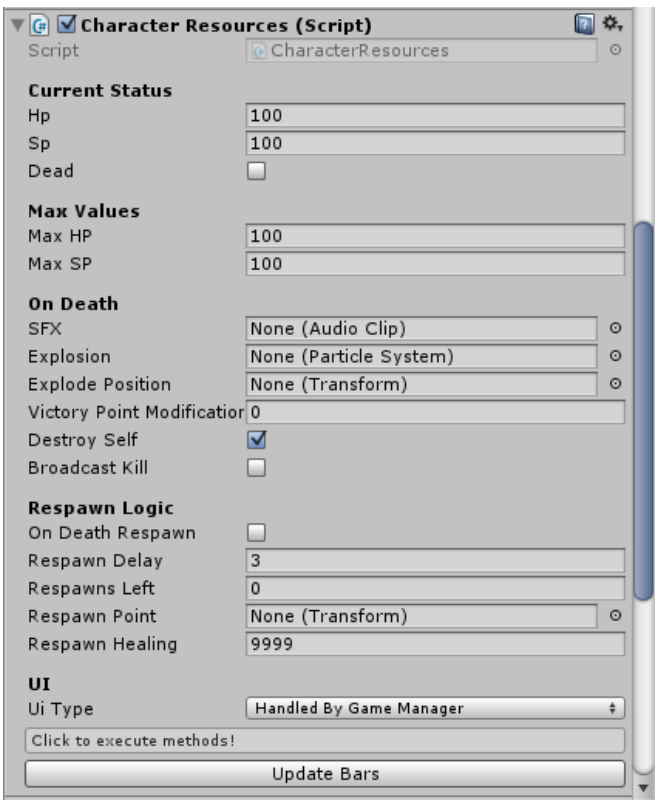

Figura 9: Formulario de Character Resources

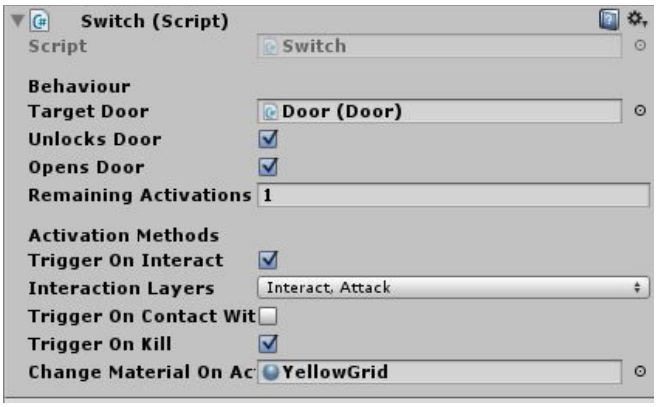

Figura 10: Formulario de un Switch

3.3 Inteligencia Artificial

Esta sección ofrece una vista general del funcionamiento de la inteligencia artificial que fue implementada para el prototipo de *Brave//Soulless*. Se explicará el funcionamiento general de los dos componentes principales de la misma: AI Agent y Agent Data.

3.3.1 AI Agent

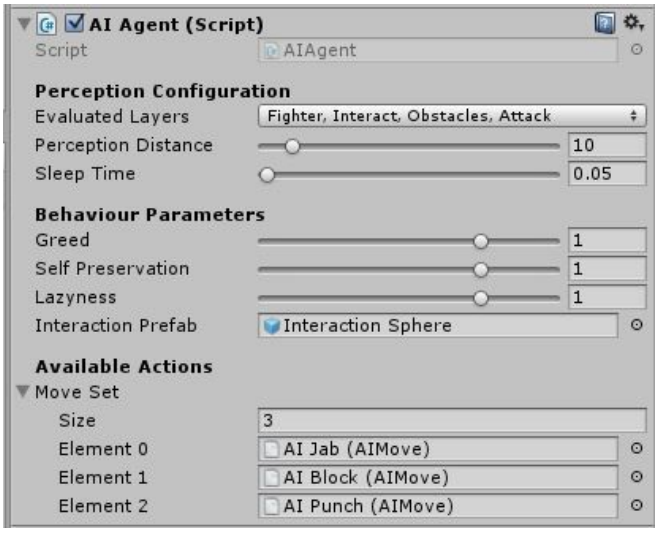

Figura 11: Formulario de configuración de un AI Agent

En la figura 11 se puede apreciar la herramienta de configuración de un enemigo, compuesta de tres secciones principales:

En la sección de Perception Configuration se encuentra el radio de detección del agente en metros (Perception Distance), el tiempo entre cada evaluación de la situación en segundos (Sleep Time) y las capas de objetos que deben ser evaluados (Evaluated Layers).

En la sección de Behaviour Parameters se establece la personalidad del agente, definiendo qué tipo de acciones prefiere ejecutar según su avaricia (Greed), sentido de la auto-preservación (Self Preservation) y pereza (Lazyness) a (desdén a acciones que le hagan desplazarse mucho). El "Interaction Prefab" es un objeto que crea en el mundo para poder interactuar con objetos tales como interruptores.

En la sección de Available Actions se encuentra la lista de movimientos, similar a como se define el Moveset de un Fighter Controller: Cada una de estas es el equivalente a una acción que podría realizar un jugador pero contiene algunos datos adicionales para que la IA sea capaz de reconocer la utilidad de la acción en sí. En este caso, el agente puede ejecutar una defensa (Block) y dos tipos de ataques (Jab y Punch).

En la figura 12 se puede visualizar el radio de detección de un agente.

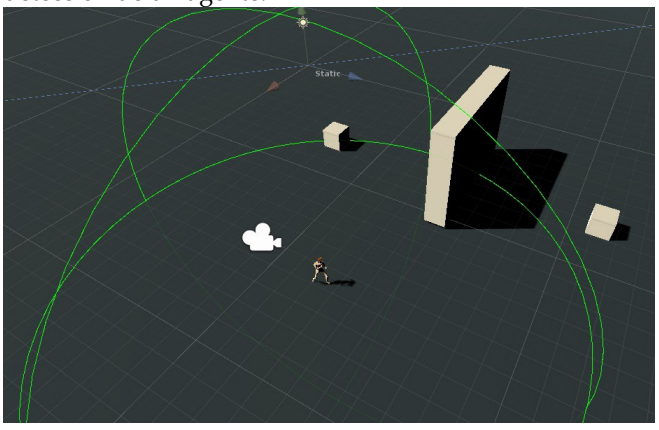

Figura 12:Ejemplo de detección de un Agente IA

En este caso, las cajas son marcadas como Obstáculos. El agente puede detectar lo que esté marcado dentro de la esfera verde: Nuestro agente puede detectar la caja de la izquierda y la pared. Dado que la pared no tiene interacción posible, decidirá ir hacia la primera caja.

En el apéndice A2 puede apreciarse un diagrama de flujo con el funcionamiento del algoritmo que ejecuta este componente.

3.3.2 Datos de Agente

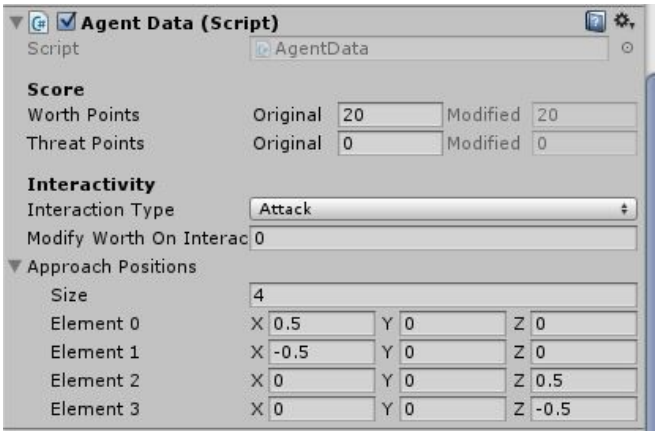

Figura 13: Datos de Agente

Los Datos de Agente, visibles en la figura 13, son un componente asociado a todo objeto con el cual puede interactuar un agente. Definen como se ha de interactuar (sección Interactivity) y que tan valiosa o peligrosa es esta interacción (sección Score).

También definen las posiciones desde las cuales se puede acceder al objeto (Approach Positions), utlizado por el sistema de pathfinding para determinar si un

objeto es alcanzable o no.

Los valores de "worth" y "threat" cambian dinámicamente según según qué acción esté ejecutando el objeto o si se ha interactuado con él antes (Modify Worth on Interact). Los parametros de "Greed" y "Self-Preservation" del AI Agent representan cuanto caso se le hace a estos puntajes.

4 Resultados

En esta sección se describe la metodología de trabajo, los tests realizados y el contenido del prototipo.

4.1 Métodología de trabajo

El proyecto se ha desarrollado con una metodología ágil basada en SCRUM: Durante sus etapas iniciales, se creó un cronograma de todas las tareas a realizar para completar el proyecto y se las han asignado sobre las diversas semanas disponibles.

Los lunes de cada semana se evaluaban las tareas completadas y se analizaba la posibilidad de ajustar la distribución de las mismas o ajustar las tareas opcionales. Puede verse en la figura 14 una gráfica del progreso del proyecto tomada el día 15 de junio del año 2016. La linea azul representa la cantidad de tareas planificadas, la zona verde representa la cantidad de tareas completadas a lo largo del tiempo.

Nótese que este gráfico no es representativo del porcentaje del proyecto completado ya que valora todas las tareas con el mismo peso. Las alteraciones en la cantidad de tareas se debe mayoritariamente a cambios en la granularidad de las mismas de cara al aplicativo, a efectos prácticos solo se han eliminado tareas opcionales.

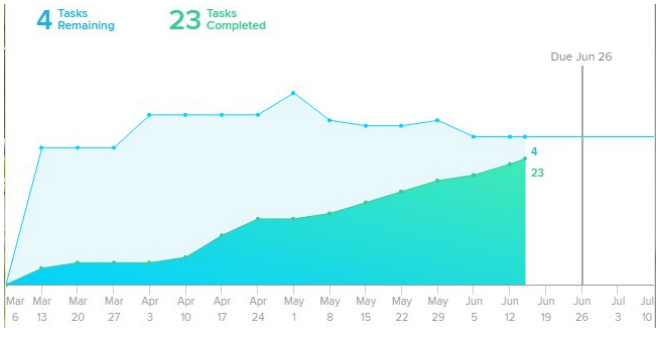

Figura 14: Gráfica de progreso según Asana

Las primeras semanas fueron dedicadas exclusivamente al diseño del juego y al desarrollo del modelo 3D utilizado en el juego, lo cual explica el poco avance respecto a la cantidad de tareas finalizadas.

A continuación se comenzó a implementar los controles de juego y ajustar las físicas de juego para poder manejar al personaje dentro de una escena de prueba.

El siguiente paso consistió en programar diversas partes de la jugabilidad, tales como las herramientas para crear movimientos y habilidades, la funcionalidad de interruptores que puedan abrir puertas tras tocarse, la posibilidad de destruir objetos tras que reciban suficiente daño o una interfaz básica para transmitirle al jugador el estado del personaje que controla

Como último paso se desarrolló una inteligencia artificial capaz de comprender el terreno que le rodea, determinar cual de todos los posibles objetivos es el más importante para él según unos parámetros de rango de detección y de personalidad.

4.2 Test

Gracias al bajo tiempo de compilación y a la posibilidad de manipular variables en tiempo de ejecución que provee Unity3D, fue posible ejecutar una multitud de pruebas sobre cada componente del juego a medida que se iba desarrollando. Las herramientas provistas para debugging de Unity3D permitieron la ejecución de tests de caja negra y caja blanca por igual, facilitando la exploración de cada funcionabilidad tanto en sus casos límite como en sus casos promedio. Se ha determinado, gracias a estas pruebas, que el código funciona como se pretendía y no hay errores a destacar.

Adicionalmente, el juego también fue dado a probar a diversas personas durante diferentes etapas del proyecto, incorporando el feedback ofrecido por ellos para adaptar el diseño y detectar errores. Se incorporó este tipo de testing iterativo sobre el proyecto para imitar el tipo de re-evaluación y prueba continua que es recomendada en los diferentes métodos de desarrollo ágil. Gracias a los usuarios que han asistido en la prueba, se han identificado y solucionado múltiples problemas de diseño, como ajustar la fuerza de la gravedad para que los saltos sean más kinéticos e incrementar la velocidad a la que se ejecutan los ataques para que el jugador vuelva a estar en control lo antes posible.

5 Conclusiones

Se ha desarrollado el prototipo de videojuego de lucha en una versión cien por ciento funcional, incluyendo las mecánicas de combate, motor de jugabilidad, una lógica básica para el desarrollo de puzzles y una inteligencia artificial capaz de reconocer su entorno, reconocer qué acciones son prioritarias según su configuración de "personalidad" e interactuar con el entorno, el jugador o con otras inteligencias artificiales presentes en el mundo.

Se han creado también diversas herramientas que fueron integradas como componentes para Unity3D. Estas herramientas existen para configurar el funcionamiento de los personajes, pantallas, componentes de juego y de la inteligencia artificial. Las herramientas desarrolladas pueden ser

En el prototipo, el jugador puede navegar por una pantalla de menu principal y seleccionar que desea comenzar la partida. Tras eso, será será colocado en una zona segura donde probar las diferentes acciones que puede tomar: Para poder progresar, el jugador debe ser capaz de activar dos interruptores que abren una puerta y le permiten el acceso a una una zona que contiene un enemigo con el cual es posible luchar.

En el apéndice A1 se pueden ver algunas capturas de

pantalla tomadas dentro del juego.

Incluso abstrayendo la necesidad de desarrollar el motor gráfico y de físicas, el desarrollo de un videojuego es una tarea monumental. Hay una gran cantidad de sistemas y mecanismos que han de interactuar entre sí; no es suficiente con que todas las piezas que componen el juego funcionen como deben, también deben resultar en algo divertido e interesante, algo que construya un "Game Feel" atractivo (definido como el cómo se siente el jugador mientras interactúa con los sistemas de juego).

Tras quedar implementadas todas las funcionalidades necesarias para la creación del prototipo, preparadas las herramientas para agilizar el desarrollo de contenido y quedando asegurado que el diseño es escalable, será posible la creación de una versión completa del juego.

Una de las mayores dificultades del proyecto fue el diseño de la inteligencia artificial, requiriendo de mucho tiempo para poder idear una buena forma de conseguir que sea lo más reactiva e inteligente posible.

Como posibles mejoras al prototipo en sí, se destacan las siguientes:

- Añadir un componente multijugador para brindar una experiencia más social y enfocar el juego hacia uno de los elementos motivadores de este proyecto.
- Desarrollar uno o más modelos de personaje adicionales para dar mayor variedad visual al juego y asegurarse de que los enemigos no se vean iguales al jugador.
- Añadir música, efectos sonoros, escenarios y cinemáticas para poder desarrollar el juego en su versión completa.
- Cambiar a una estética low-poly para acelerar la velocidad de desarrollo del arte 3D y obtener más volumen de contenido en menos tiempo.

Agradecimientos

Me gustaría agradecer a Enric Martí Gòdia por tutorizar este proyecto y por la dirección que le ha brindado.

También quisiera agradecer a Silvia Danès Farrarrons por realizar el concept-art del protagonista y el apoyo que me ha brindado. Gracias también a todos los amigos que me han ofrecido ayuda y consejo a la hora de diseñar y probar los diversos elementos del juego.

Finalmente, doy mi agradecimiento al lector de este artículo por brindarme su tiempo.

Referencias

[1] http://www.polygon.com/a/street-fighter-2-oral-history, artículo sobre la historia del desarrollo de Street Fighter 2. (Fecha de último acceso: 15/06/2016).

[2] http://www.giantbomb.com/shigeru-miyamoto/3040-614/, breve artículo sobre la relevancia de Shigeru Miyamoto en la industria. (Fecha de último acceso: 15/06/2016).

[3] http://www.gamasutra.com/view/feature/130734/game_feel_the_secret_ingredient.php?print=1, artículo sobre el proceso de diseño del videojuego Mario 64 (Fecha de último acceso: 15/06/2016)

[4] https://docs.unity3d.com/ScriptReference/, API de Unity3D. (Fecha de último acceso: 15/06/2016)

[5] https://www.assetstore.unity3d.com/en/#!/content/51416, librería con utilidades varias para acelerar la programación de ciertas funciones en Unity3D. (Fecha de último acceso: 15/06/2016)

[6] https://www.assetstore.unity3d.com/en/#!/content/15667, paquete de la Asset Store que combina un modelo de personaje y una gran variedad de animaciones de combate. Se han utilizado solo las animaciones. (Fecha de último acceso: 15/06/2016)

[7] https://www.assetstore.unity3d.com/en/#!/content/19991, primera parte de un paquete de la Asset Store que contiene una gran cantidad de animaciones obtenidas con motion capture. Se han utilizado solo las de saltos y volteretas. (Fecha de último acceso: 15/06/2016)

[8] https://www.blender.org/, página web de Blender, herramienta open-source de modelado 3D. (Fecha de último acceso: 15/06/2016)

[9] https://asana.com/product, página web con la descripción de Asana, una aplicación web destinada a la organización de proyectos. (Fecha de último acceso: 15/06/2016)

[10] https://en.wikipedia.org/wiki/Mega_Man_(video_game), artículo de Wikipedia de Megaman donde se puede encontrar una descripción de la jugabilidad encontrada en ese juego. (Fecha de último acceso: 15/06/2016)

[11] https://www.assetstore.unity3d.com/, página web de la Asset Store de Unity3D. (Fecha de último acceso: 23/06/2016)

[12] https://sourcemaking.com/design_patterns/factory_method, pagina web que contiene una explicación del patrón de diseño "Factory Method". (Fecha de último acceso: 23/06/2016)

[13] https://sourcemaking.com/design_patterns/decorator, página web que contiene una explicación del patrón de diseño "Decorator". (Fecha de último acceso: 23/06/2016)

APÉNDICE

A1. Juego en ejecución

En esta sección del apéndice se podrán observar varias capturas de pantalla del juego en diferentes modos y zonas de prueba. En particular, se observarán tres escenas: Un dialogo, un puzzle y un combate.

a) Diálogos

Figura 15: Caja de Dialogo

En la figura 15 se puede ver al juego en el modo de aventura: En este modo, la interfaz de usuario desaparece y el jugador no puede ejecutar ataques o poderes.

A cambio, puede interactuar con el entorno: Ciertos objetos reaccionan de una u otra manera al ser interactuados.

Algunos pueden ser interruptores que abran puertas, otros pueden hacer aparecer un texto en pantalla. Para comunicarla información al jugador. En este ejemplo se puede apreciar al jugador interactuando con una caja parlante con un ego muy grande.

b) Puzzles

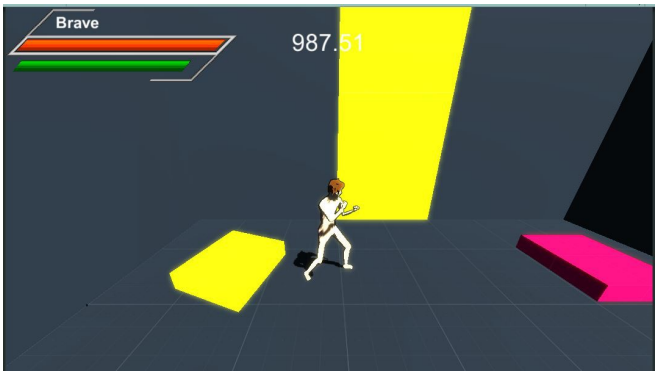

Figura 16: Puzzle sencillo

En la figura 16 podemos ver al jugador frente a una puerta (pared amarilla) que se se abrirá tras presionar dos interruptores rosas. Uno de los interruptores ya ha sido presionado y ha cambiado de color a amarillo, señalizando que está asociado a esa puerta y que ya está activo.

c) Combate

Figura 17: Combate entre dos personajes

En la figura 17 puede apreciarse al jugador siendo golpeado por un enemigo controlado por la inteligencia artificial. Unas partículas azules resaltan el impacto mientras la barra de vida del jugador desciende. El jugador es lanzado al suelo, donde permanecerá incapaz de actuar durante un segundo. Tras eso se pondrá en pie automáticamente y podrá seguir luchando.

A2. Diseño

En la siguiente página se podrán apreciar las figuras 18 y 19, destinadas a representar un diagrama de flujo de datos en el motor de jugabilidad y la lógica de la IA.

Se desarrollaron diagramas de clases pero por cuestiones de dimensiones no se han añadido a este documento.

Flujo de Datos

Este diagrama representa las clases principales del juego y señala el flujo de las diferentes partes del juego. En la esquina superior izquierda tenemos toda la sección de inteligencia artificial (AgentData, AIAgent, AIMove), en el centro toda la sección de lógica de personajes (CharacterController, Character Resources, Move, Move Effects, Hitbox), en la zona inferior la sección de lógica de escenarios (Door, Switch) y en la zona derecha la sección de control de escena (Game Manager, Scene Manager).

El output a pantalla es realizado por Unity3D de forma independiente a mi diseño y por lo tanto lo he excluido del diagrama.

Lógica de la IA

Este diagrama de flujo explica de forma general el proceso por el cual un agente decide qué acciones tomar: A cada frame se pregunta si puede actuar para determinar si procesarse o no.

Si puede, determinará si debe o no detener su movimiento.

Tras eso y si tiene un objetivo, iterará por todas sus acciones hasta encontrar la mejor. Si puede ejecutar alguna, lo hará. De lo contrario, asumirá que aún debe acercarse más y seguirá con su lógica. Como las acciones pueden empujar al agente mediante físicas, desactiva el

pathfinding del agente y activa las físicas.

Finalmente, si determina que no está ejecutando ninguna acción, se asegura que el pathfinding esté activado y, sí su tiempo de reacción ya ha transcurrido, busca todos los objetos su rango perceptivo, calcula cual es el más conveniente para interactuar (según los parámetros del agente y del objeto en cuestión) y lo señala como siguiente objetivo para el pathfinding y las acciones.

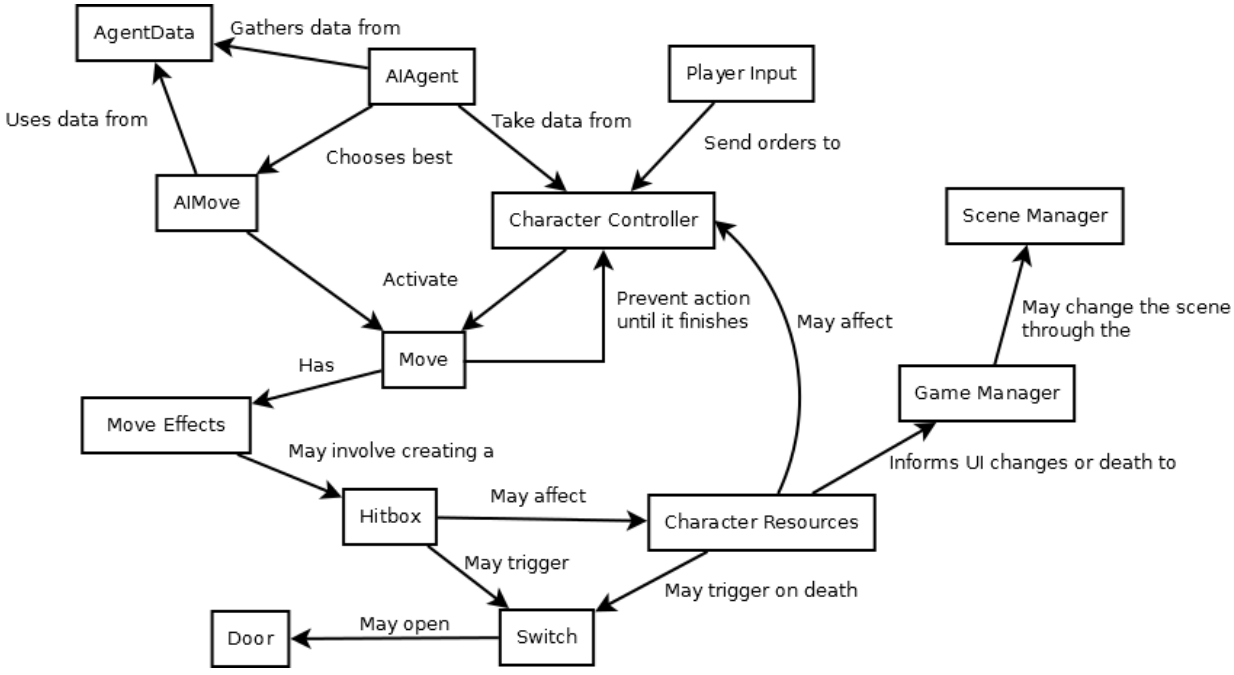

Figura 18: Flujo de Datos

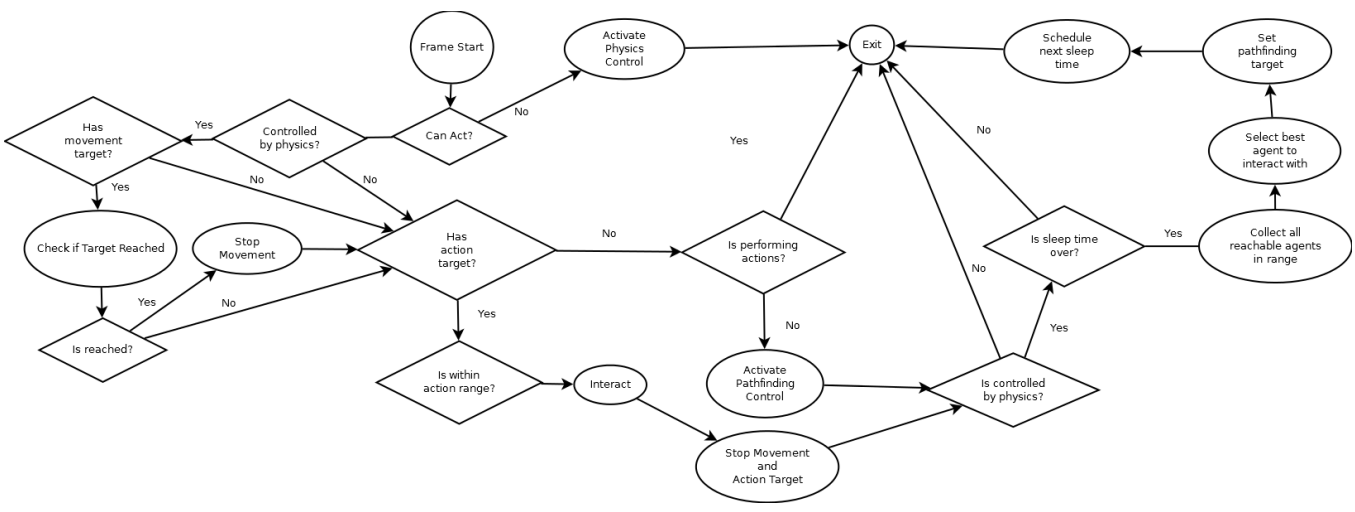

Figura 19: Diagrama de Flujo de la IA