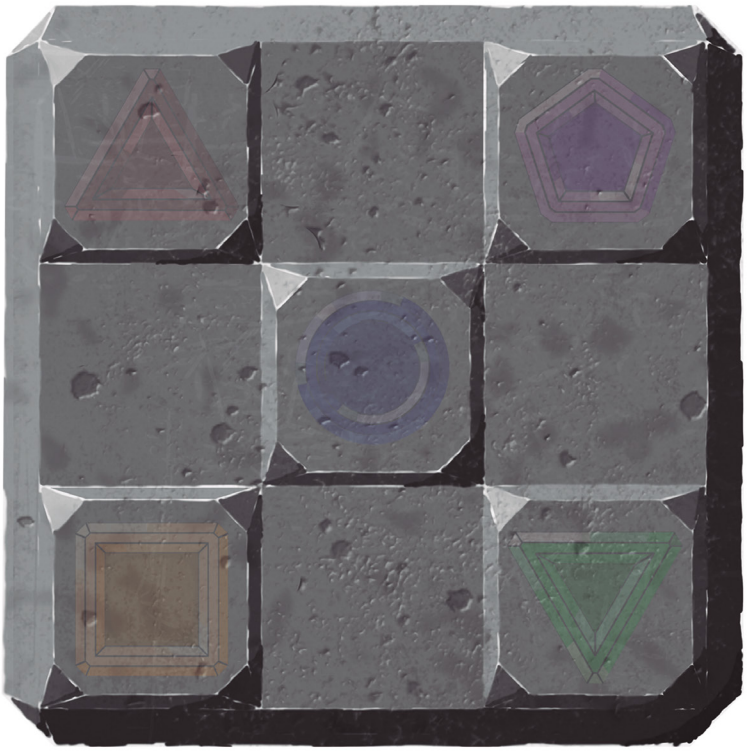
SOUL BOUNDER



Autor del proyecto: Javier Lorenzo Santiago
Directora del proyecto: Agnés Borràs Angosto
Título: Grado en Multimedia 3/10/14

ÍNDICE

1.- Introducción. 3
2.- Glosario. 5
3.- Historia de los roguelike. 7
3.1.- Principios. 9
3.2.- Actuales. 10
4.- Historia de los juegos puzzle. 13
4.1.- Principios. 13
4.2.- Actuales. 14
5.- Descrpción proyecto. 17
5.1.- Idea inicial. 17
5.1.1.- Puzzle. 18
5.1.2.- Personaje. 18
5.2.- Evolución. 18
5.2.1- Puzzle. 18
5.2.2- Personaje. 19
5.3.- Idea final. 19
5.3.1.- Puzzle. 19
5.3.2.- Personaje. 20
6.- Inspiracion del juego. 21
6.1.- Modo laberinto. 21
6.2.- Modo puzzle. 22
7.- Motor de videojuego. 23
7.1.- El motor Unity. 24
7.2.- Alternativa al motor unity. 25
8.- El juego. 27
8.1.- Historia. 27
8.2.- Propiedades del juego. 27
8.2.2.- Salvado y muerte. 27
8.2.2.- Aleatoriedad. 28
8.2.3.- Dificultad. 28
8.2.4.- Escenas del juego 28
8.3.- La torre del laberinto. 29
8.3.1.- Interfaz y control. 32
8.4.- Objetos interactivos. 33
8.5.- Juego puzzle. 35
8.5.1.- Interfaz del juego puzzle. 36
8.5.2.- Reglas juego puzzle. 37

8.6.- Personaje. 39
8.6.1.- Estadísticas del personaje. 39
8.6.2.- Habilidades del personaje. 40
8.7.- Enemigo. 41
8.7.2.- Características del enemigo. 41
8.7.2.- IA enemigo. 42
8.8.- Estilo y diseño. 44
8.9.- Recursos externos. 46
8.9.1.- Efectos de sonido. 46
8.9.2.- Modelos y animación. 47
8.10.- Problemas de programación y resolución de los mismos. 49
9.- Conclusiones. 51
10.- Agradecimientos. 53
11.- Bibliografía. 55

1.- INTRODUCCIÓN

EL proyecto se trata de un videojuego tanto en versión ejecutable para Windows, Mac OS y Linux, como para ser jugado mediante un navegador Web. El proceso abarca tanto el ámbito de la programación, diseño gráfico, modelado 3D y audio digital.

El principal reto propuesto era el de crear un sistema de juego combinando dos de los géneros que están teniendo mucho éxito en la actualidad, Roguelike y Puzzle. Ambos son géneros muy distintos y que por lo general atraen a público opuesto. Los juegos Roguelike atraen más a jugadores experimentados porque suponen un gran reto, sin embargo los juegos de tipo Puzzle atraen a un público muy casual. En la actualidad los juegos tipo puzzle son muy comunes en las redes sociales o en juegos para móviles. El reto es combinar ambos géneros de forma coherente, divertida y diferente a los juegos existentes en el mercado.

El objetivo principal es la de crear un videojuego robusto, adictivo, y listo para el ámbito comercial creando desde cero todos sus componentes. El idioma elegido para el juego es el Inglés, pero la habilidad requerida por el jugador en ese idioma es casi nula. Los únicos textos que hay en el juego se encuentran en los botones, “New Game“ (Nueva partida), “Continue” (Continuar).

Para la creación del juego se han usado una variedad de programas y tecnologías:

-UnityScript como lenguaje de programación, similar a JavaScript.
-Cinema 4D para el modelado y texturizado.
-Photoshop para el retoque de texturas e interfaz.
-Adobe Audition para los arreglos de sonido.
-Unity como motor gráfico.

2.- GLOSARIO

-ASCII: Código de caracteres basado en el alfabeto latino.

-Bonus: Bonificación.

-Combo: Combinación especial. En el caso del juego es una ventaja adquirida al cumplir ciertos requerimientos.

-Compilar: En el ámbito de la informática, es traducir un lenguaje de programación a código máquina, un lenguaje que la computadora es capaz de interpretar.

-Función: En informática, es una rutina dentro del código que se encarga de hacer una tarea específica.

-Hardware: Componentes físicos de un ordenador, procesador, tarjeta gráfica, memoria, disco duro...

-IA: Inteligencia Artificial.

-Indie: Independiente. En ámbito de los videojuegos, se define por juegos desarrollados por individuos o pequeños estudios que no cuentan con financiación por parte de una productora.

-Interfaz: Medio por el cual el usuario se comunica con la máquina o viceversa.

-Matriz: En programación, es una estructura que contiene un conjunto de variables del mismo tipo, los elementos se organizan por filas y columnas.

-Procedural: Generación de contenido a partir de algoritmos en vez de hacerlo de forma manual.

-Puzzle: Género de juegos de ingenio o lógica.

-Roguelike: Género de juegos basados en una alta dificultad y aleatoriedad.

-Vector: Estructura unidimensional que contiene un conjunto de variables del mismo tipo, los elementos se organizan de forma lineal.

3.- JUEGOS ROGUELIKE

Roguelike *[1]* es el término utilizado para un sub género de los videojuegos basados por lo general en historias de fantasía. Normalmente son juegos para un jugador y se da más importancia a la jugabilidad que al diseño estético. Están regulados por turnos, esto implica que el estado del juego será el mismo hasta que el jugador haga una acción. Como elementos principales nos encontramos con una serie de laberintos y mazmorras en los cuales es común encontrar enemigos u objetos de distinta índole que son generados aleatoriamente. Una característica importante es que el jugador, cada vez que muere, vuelve a empezar, esto se conoce como muerte permanente. El objetivo principal de los juegos Roguelike es avanzar a través de los diferentes niveles hasta llegar al último.

A mediados de los años 80, el número de juegos Roguelike había aumentado considerablemente, los elementos por los cual se rigen este tipo de video juegos, fueron definidos en la “International Roguelike Development Conference 2008”, también conocida como “Berlin Interpretation” *[2]*. Dichos elementos se diferenciaron en dos categorías, factores de alta importancia y factores de baja importancia.
Dentro de los factores de alta importancia nos encontramos los siguientes:

- Entorno generado de forma aleatoria: Esta característica favorece que el jugador pueda volver a jugar repetidas veces sin encontrarse los mismos escenarios, enemigos u objetos de la misma forma que los encontró previamente.

- Sistema por turnos: El jugador realizará un turno, independientemente del tiempo que tarde en hacerlo, y seguidamente el juego hará el correspondiente turno.

- Sistema de rejilla: El mundo esta representado por una malla de celdas uniformes Tanto el jugador como los monstruos ocupan una casilla, independientemente del tamaño del tamaño relativo de los monstruos.

- No modal: Tanto el movimiento, combate u otras acciones tienen lugar en el mismo modo, es decir que tienen que estar siempre disponibles en cualquier punto del juego.

- Muerte permanente: Los juegos Roguelike están pensados para que el personaje muera repetidas veces con la posibilidad de empezar el juego nuevamente.

- Gestión de recursos: Es imprescindible que el jugador gestione los diferentes recursos que ofrece el juego, tales como pociones, comida, etc. ya que estos son limitados.

- Complejidad: La gran variedad de enemigos y objetos permite que el juego adquiera un nivel de complejidad el cual asegura diferentes soluciones para completar los objetivos.

- Hack'n'slash: El concepto del juego es jugador contra el entorno aunque una de las tareas más importantes es matar monstruos o enemigos.

- Exploración y descubrimiento: También es una parte importante del juego el explorar mazmorras y descubrir objetos. Este proceso ha de repetirse cada vez que el jugador empieza de nuevo el juego.

Los factores secundarios corresponden a los siguientes:

- Personaje único: El jugador controla únicamente a un solo personaje.

- Los enemigos son semejantes a los jugadores: Ambos enemigos y personajes se rigen por las mismas reglas tales como equipamiento, hechizos, uso de objetos, etc.

- Mazmorras: Son los escenarios del juego y cada una representa un nivel diferente.

- Reto táctico: Con el fin de progresar y avanzar en el juego, es necesario que el jugador conozca y aprenda las diferentes tácticas que le ayudarán a ganar.

- Números: Los puntos de daño así como vida, mana restante, etc. están representados a través de números.

- Representación ASCII: El método tradicional de representar un juego Roguelike es usar caracteres ASCII como representación gráfica

Existe cierta controversia sobre los puntos definidos en la "Berlin Interpretation". Darren Grey, los califica como desfasados, poco precisos y que no representan un género abierto y dinámico en su artículo "Screw the Berlin Interpretation!". A pesar de dicha controversia, se denominan juegos Roguelike a aquellos que poseen una o más de las características mencionadas anteriormente pertenecientes a la "Berlin Interpretation" pero no es necesario que cumplan todas ellas.

3.1.- Principios

El inicio de los juegos Roguelike se da en los años 80 con un juego llamado Rogue *[3]* creado por Michael Toy y Glenn Wichman en el cual el jugador tenía que ir descendiendo de niveles desde el más alto hasta el más bajo. A través de una serie de mazmorras sin mapa con incontables enemigos y tesoros el héroe debe recuperar el Amuleto de Yendor para posteriormente ascender a la superficie. En la Figura 3.1 *[4]* podemos ver una imagen de Rogue de un ordenador IBM en color.

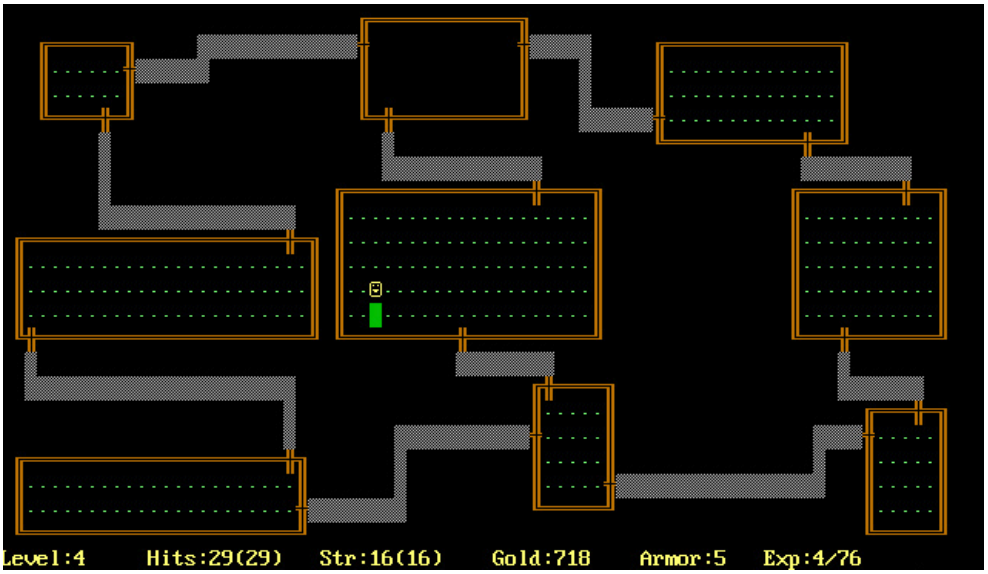


fig. 3.1

En la Figura 3.2 *[5]* vemos el juego con gráficos ASCII. podemos apreciar aquí la ausencia de color así como una pérdida de elementos reduciendo el juego a la mínima expresión gráficamente hablando ya que "@" representa por lo general el jugador. El ".", "#" y "-" hacen referencia a los suelos y las paredes de las mazmorras. Los enemigos u objetos eran representados con otras letras o símbolos.

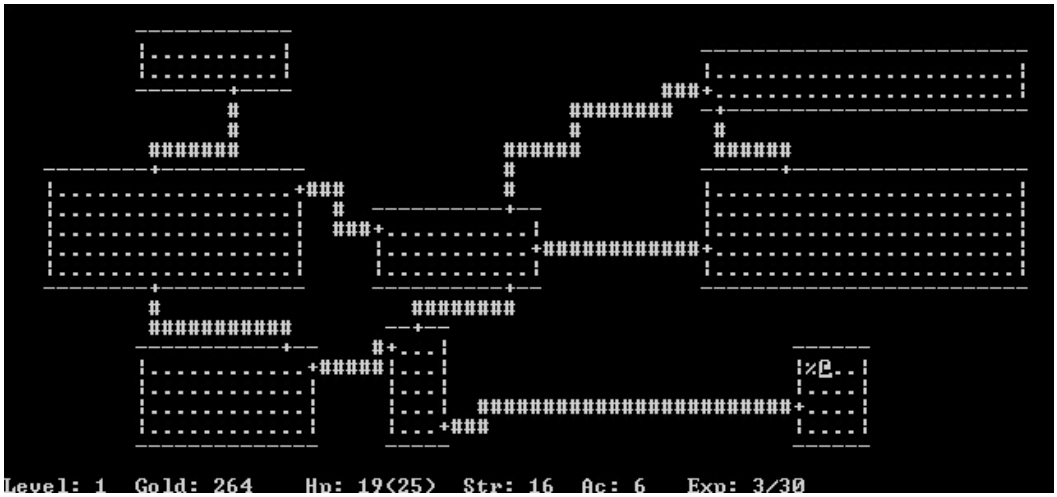


fig 3.2

3.2- Actualidad

A lo largo de los años se fueron creando juegos Roguelike con alguna que otra variación y aprovechando las mejoras gráficas que ofrecía la tecnología. Un ejemplo el videojuego Diablo (1996) de Blizzard Entertainment.

A partir del año 2000, el número de juegos Roguelike fue aumentando considerablemente gracias a los desarrolladores independientes. Uno de los primeros juegos fue Strange Adventures in Infinite Space (2002) (Figura 3.3) *[6]* y su correspondiente secuela Weid Worlds: Return to Infinite Space (2005) creados por Digital Eel. En ambos casos, el jugador tiene que explorar planetas y enfrentarse a enemigos generados aleatoriamente.

fig. 3.3

Así mismo, en Japón, títulos como Pokémon Mystery Dungeon (basado en Pokemon) y Chocobo series (basado en Final Fantasy) (Figura 3.4) *[7]* contribuyeron al resurgir de los juegos Roguelike.

figura 3.4

Otro ejemplo a destacar es “FTL” (Faster than Light) (Figura 3.5) *[8]* .Tenemos en esta ocasión un juego de tipo Roguelike ambientado en una nave espacial que va explorando la Galaxia. El jugador tendrá que manejar a la tripulación asignándole diferentes tipos de trabajos dentro de la nave como por ejemplo ingeniero de armas, ingeniero de motores, piloto. Tiene que ir avanzando por diferentes planetas de cada uno de los sectores de los que se compone la Galaxia mencionada anteriormente. El objetivo del juego es regresar a casa para avisar del peligro inminente de los rebeldes, antes de que alcancen tu nave. A medida que avanzas en los sectores, la dificultad aumenta. Cada vez que inicias el juego todo el mapa se genera aleatoriamente y cada vez que se entra en un planeta o asteroide, existe la posibilidad de que ocurra un evento tal como la aparición de una tienda, un enemigo u otro tipo de evento. Asi mismo, el concepto de muerte permanente también está presente en el juego.

figura 3.5

4.- HISTORIA DE LOS JUEGOS PUZZLE

Los videojuegos puzzle se conocen también como videojuegos de lógica o videojuegos de inteligencia. En ellos se propone al jugador resolver diferentes tipos de problemas de lógica, estrategia, puzzles, organizar figuras, etc.

4.1- Principios

Dos de los videojuegos puzzle más conocidos son el Tetris (Figura 4.1) *[9]*, creado por Alekséi Pázhitnov a principios de los años 80 y el Buscaminas, creado por Robert Donner en 1989.
El Tetris consiste en varias figuras geométricas de diferentes colores las cuales pueden ser rotadas en ángulos de 90º, 180º y 270º para ser agrupadas formando líneas horizontales del mismo color con la finalidad de eliminar esos cuadrados de la pantalla permitiendo que sigan saliendo piezas de la parte superior de la pantalla. A medida que avanza el juego, los bloques salen cada vez más rápido haciendo que el jugador tenga menos tiempo para pensar donde colocar la pieza. El juego termina cuando la pantalla se llena de dichas piezas porque el jugador no ha sido capaz de agruparlas correctamente.

fig 4.1

El Buscaminas (Figura 4.2) es un videojuego para un solo jugador cuyo objetivo es despejar un campo de minas sin que estalle ninguna. El juego se compone de un tablero con todas las casillas ocultas en las que podemos encontrar tres elementos diferentes:

-Minas: Son las casillas que debemos evitar. Si se desvela una casilla que contenga una mina, la partida terminará (Figura 4.3).

-Números: Las casillas con números proporcionan información para que el jugador vaya desvelando el tablero sin tocar ninguna mina. El número que aparezca indicará cuantas minas hay alrededor.

-Casillas vacías: Indican que no hay ninguna mina alrededor.

figura 4.2

figura 4.3

4.2- Actualidad

Hoy en día el número de videojuegos puzzle, inteligencia o lógica, ha umentado considerablemente. Al igual que pasara con los Roguelike, el avance de las tecnologías han permitido desarrollar videojuegos más complejos. Un buen ejemplo de esto es la saga de El Profesor Layton, desarrollado por la compañía de videojuegos independiente japonesa Level-5. El primer juego de la saga data de 2007, a partir de entonces, ya cuenta con varios títulos.

En El Profesor Layton (Figura 4.4) *[10]*, el jugador tiene que ir resolviendo puzzles que le proponen tanto los personajes del juego como el entorno con el fin de avanzar en la historia. Dichos puzzles pueden ser tanto simples problemas matemáticos como de deducción, organizar un rompecabezas, etc. El jugador tiene la opción de utilizar un número determinado de pistas como ayuda para la resolución de los puzzles planteados. En El Profesor Layton no existe el concepto de muerte del personaje con su consecuente terminación de la partida, simplemente la no resolución de ciertos puzzles impiden que nuevos contenidos aparezcan, quedando el juego en el mismo punto indefinidamente.

Otro videojuego actual del género puzzle es el Bejeweled (Figura 4.5) *[11]*, desarrollado por PopCap Games en 2001. Una serie de gemas de diferentes colores aparecen en la pantalla. El jugador debe de hacer grupos de 3 gemas del mismo color (como mínimo) intercambiando la posición de una gema por otra en cada turno. Los cambios pueden ser tanto verticales como horizontales. Si se combinan 4 o más gemas se creará una gema especial que hará desaparecer más gemas de lo normal. También es posible combinar a la vez gemas colocadas tanto en horizontal como en vertical.

fig. 4.4

fig. 4.5

5.- DESCRIPCIÓN DEL PROYECTO

El juego es una combinación de dos estilos muy diferentes, los juegos Roguelike y los de Puzzle. El estilo Roguelike se usa en el apartado de exploración, y el de Puzzle en los combates.

El apartado de exploración consta de un escenario formado por habitaciones, las cuales pueden contener objetos interactivos o enemigos. Cuando el personaje encuentra un enemigo se activa el combate, y se abre el modo puzzle.

Lo mas importante de los juegos Roguelike es la aleatoriedad, por eso se ha tenido muy en cuenta este punto en el desarrollo de todo el juego. Cada vez que se crea una partida o que el personaje supere un nivel, se genera de forma aleatoria un laberinto de habitaciones.

En el juego tomas el papel de un brujo que controla a un esqueleto. El método de control es mediante un tablero y unas gemas, lo que da sentido a usar el juego de puzzle en el modo de combate. El título es una combinación de las palabras inglesas “Soul” alma, y “bounder”, persona sin escrúpulos. A su vez la palabra “bounder” tambien es una nominalización del pasado del verbo “bind“ atar o obligar. Este término encaja perfectamente con la idea del juego del control del brujo sobre el personaje y sobre la historia.

5.1.- Idea inicial

El juego ha evolucionado a lo largo de su desarrollo. Pero siempre teniendo en cuenta El concepto de la parte del laberinto de las habitaciones siempre ha sido la misma, prácticamente no ha evolucionado, pero el resto del juego si que lo ha hecho.

5.1.1.- Puzzle

En un principio el modo puzzle iba a estar formado por una cuadricula de 8x8 y en cada celda una pieza, como se puede ver en la Figura 5.1. El objetivo era que el jugador tenia que mover una fila entera de piezas para intentar combinar las figuras iguales en las columnas (Figura 3.1.1.2).

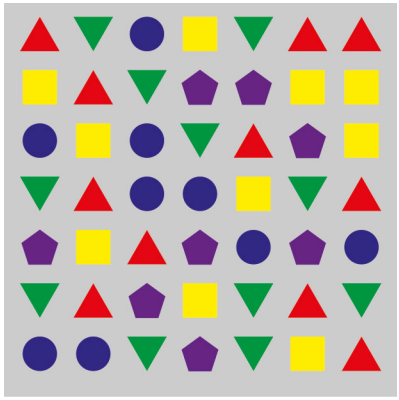


fig. 5.1

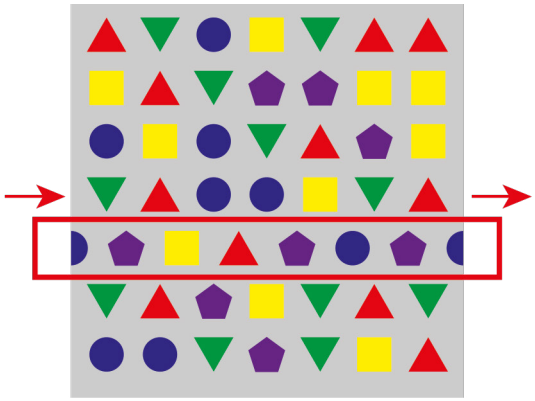


fig. 5.2

5.1.2.- Personaje

La idea inicial del personaje era la de no haber un personaje en concreto, sino que el jugador tendría la opción de escoger entre diferentes tipos de personaje. Cada personaje tendría unos atributos diferentes lo que haría que el jugador adaptase su forma de juego a ese personaje creando una experiencia de juego diferente. También contaría con un inventario en el que pudiese organizar y equiparse con diferentes objetos.

5.2.- Evolución de la idea inicial

Conforme iba transcurriendo el proceso de la creación del juego, Algunas de las ideas iniciales tuvieron que sufrir un proceso de cambio. Este proceso tuvo que ser necesario tanto por la extensión del juego, como por una evolución mas madura de la idea inicial.

5.2.1.- Puzzle

El sistema de juego de puzzle aunque no era exactamente igual, era similar a otros ofrecidos por diferentes juegos, así que se desecho la idea tanto del tipo de tablero, y de cómo el jugador tenia que combinar las piezas para conseguir un resultado.

5.2.2.- Personaje

Tanto el hecho de tener que hacer varios personajes como un sistema de objetos e inventario, iba a hacer que el desarrollo del juego se hiciese demasiado largo. Aun así, en su lugar había que pensar una manera de hacer que el jugador tuviera una experiencia diferente con el personaje cada vez que empezase una partida.

5.3.- Idea final

5.3.1.- Puzzle

Al haber desechado la idea inicial, la dificultad se encontraba en intentar ofrecer al jugador una experiencia de juego diferente a las existentes, y eso implicaba analizar juegos similares. Casi todos los juegos de este género tienen varios puntos en común, muchas fichas y de mover para conectarlas. Para buscar que el jugador experimentase con tener algo diferente, había que alejarse de la idea de tener un gran tablero y de la idea de mover fichas. Con esos conceptos en mente el resultado final fue el agrupar piezas en pequeños bloques, y que el jugador tenga que girar los bloques (aun así si podría decir que las fichas se mueven dentro del bloque) para conectar las figuras con las de los bloque adyacentes. En la Figura 5.3 se muestra de forma esquemática el diseño final del modo puzzle.

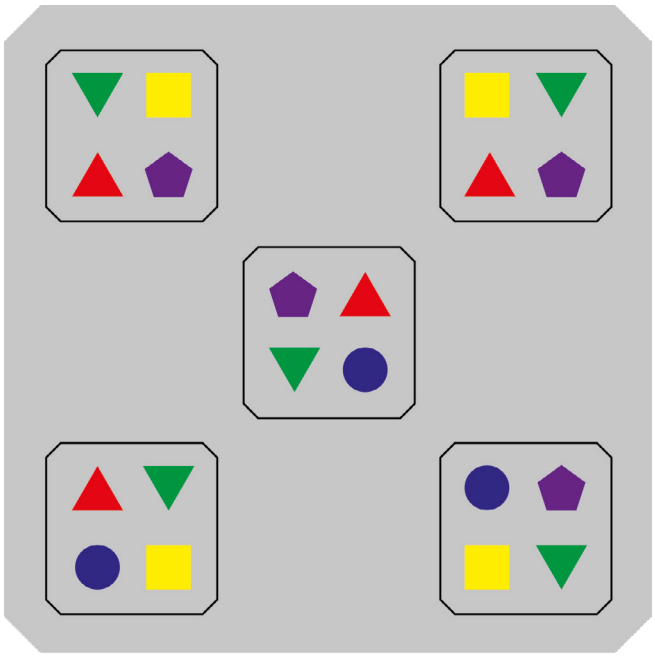


fig. 5.3

5.3.2.- Personaje

Para poder ofrecer una experiencia diferente en cuanto a darle al jugador el poder experimentar con diferentes tipos de personaje (Figura 5.4), se opta por dar al jugador la posibilidad de subir los atributos del personaje cada vez que suba de nivel (Figura 5.5). De este modo la evolución del personaje corre a cuenta del jugador determinando así tanto los puntos fuertes como los flacos de su personaje. Cada figura es un atributo, son iguales a las gemas del tablero. Subir un atributo es útil porque aumenta el valor de las gemas del tablero, haciendo al personaje más fuerte. Si el jugador no sube los atributos, no podrá avanzar en el juego, porque los enemigos lo derrotarían fácilmente. Como se puede ver en la Figura 5.5, la figura del triángulo rojo (gema de ataque), tiene un valor de 2, el “0/3” indica que el jugador debe de gastar 3 puntos para subir el valor del atributo de 2 a 3, y cada vez tendrá que gastar mas puntos para subir el atributo. Esto se hace para evitar que el jugador suba un atributo demasiado rápido.

fig. 5.4

fig. 5.5

6.- INSPIRACIÓN DEL JUEGO

El concepto del juego siempre ha sido el de mezclar de forma coherente ambos estilos de juego, Roguelike y puzzle, y para ello lo mas sensato era usar las características de los juegos Roguelike, aleatoriedad y exploración para formar un laberinto de habitaciones, y el puzzle al modo de combate. El juego también tiene un toque de RPG

6.1- Inspiración roguelike

Los juegos Roguelike siempre han atraído sobre todo a jugadores veteranos que buscan un juego dificil que rete sus habilidades. No han sido juegos muy populares en toda la comunidad de jugadores, pero han tenido un reciente auge, tanto en cantidad como en acercamiento al jugador. Esto se debe sobre todo a la popularidad de los juegos independientes (en ingles son conocidos como indie games), juegos creados por pequeños estudios independientes, grupos de amigos o incluso por un solo individuo. Estos juegos independientes se dan a conocer gracias a plataformas de venta digitales como Steam.

La principal inspiración en este género viene del juego independiente 2D “The Binding of Isaac“ (Figura 6.1), creado por Edmund McMillen y Himsl Florian en 2011 *[12]*. En el juego, el jugador controla a Isaac, un niño encerrado en el sótado de su casa. Isaac tiene que avanzar por las habitaciones disparando a enemigos y recolectando items para ayudarle a superar los niveles.
El nivel de dificultad es bastante alto y como en casi todos los Roguelike, la muerte es permanente. Gracias a la popularidad que alcanzo el juego ha recibido una expansión, y a finales de 2014 saldrá a la venta una revisión completa del juego, añadiendo mucho mas contenido y un modo multijugador.

fig. 6.1

6.2.- Inspiracion puzzle

Los juegos de tipo puzzle son muy variados. El objetivo del jugador puede ser muy diferente según el género, colocar piezas de diferente forma para formar una figura, mover fichas para encadenar fichas similares o usarlas de diferente forma para resolver un enigma. Pero lo que siempre tienen en común es que hacen pensar al jugador. Este tipo de juegos son muy populares en la actualidad, sobre todo en juegos web para redes sociales y en juegos para móviles. Por ese simple hecho de ser tan populares era una gran oportunidad el incluir este estilo en el juego.

De todos los juegos de puzzle, solo son interesantes para este proyecto los que tienen cierto toque RPG. Entre los mas famosos de este estilo está el “Puzzle Quest” (Figura 6.2) *[13]*. Originalmente lanzado en 2007 para las consolas portátiles nintendo DS y Playstation Portable, pero que gracias a la repentina popularidad del título ya ha salido tanto en diferentes consolas de sobremesa como en dispositivos móviles. En el apartado de combate del juego, el jugador debe de juntar 3 o mas fichas iguales, según el tipo de ficha que junte ataca al enemigo o obtiene recursos para gastar en habilidades.

fig 6.2

7.- MOTOR DE VIDEOJUEGO

El motor de videojuego es un entorno de software que permite a los diseñadores de software y artistas digitales la creación de los videojuegos. El motor puede estar diseñado para una plataforma en concreto, una consola, una plataforma móvil, o multiplataforma. Dentro del mismo entorno es donde todos los componentes de un videojuego, gráficos, audio, etc... se combinan para formar el producto final. Los componentes principales de un motor son:

-Motor de render: Determina tanto la forma de representar los gráficos del juego, 2D o 3D como la calidad de las texturas y efectos visuales. La cantidad de polígonos que puede contener la escena también depende de este componente y de la potencia del hardware de la consola, PC, o dispositivo móvil para la que el juego este desarrollado.

-Motor de sonido: Encargado de reproducir la música y efectos sonoros. Este motor también puede ser 2D o 3D. La diferencia es que la versión 3D puede ajustar el volumen o dirección del objeto que emita el sonido según a la distancia o lugar en relación a la posición del personaje.

-Motor de físicas: Para dar mas realismo al juego, este motor se encarga de los cálculos necesarios para dar a los objetos propiedades físicas reales, como peso, gravedad, velocidad... y como se comportarían en caso de colisionar entre ellos.

-Script: Es el lenguaje de programación en el que el juego está programado. La programación es una de las partes mas importantes porque es el que se encarga de la jugabilidad del programa, de como se comporta tu personaje con el entorno o como los enemigos actúan ante ti.

Un motor de videojuego puede ser desarrollado unicamente para un videojuego, pero lo mas común es que las compañías que desarrollan este tipo de software lo hagan para que el motor pueda ser usado para crear diferentes tipos de juegos y así poder cubrir los costes del desarrollo del software. Muchos de los motores son de uso privado dentro de la propia compañía, otros ofrecen la posibilidad de ser vendidos a otras compañías para que puedan desarrollar juegos usando ese motor. Pero hay una pequeña cantidad de ellos que ofrecen la posibilidad de ser usados tanto de manera libre como para enseñanza académica.

7.1.- El motor Unity

Unity es el motor usado para desarrollar este proyecto. Es uno de los pocos motores que hay en el mercado que ofrecen la posibilidad de ser usados tanto por desarrolladores independientes y particulares como por grandes compañías.

A principios del año 2000 en Dinamarca, tres amigos decidieron que querían realizar su propio motor de videojuegos *[14]*. Se inspiraron en el software Final Cut de Apple, aportar herramientas profesionales a usuarios aficionados. En 2005 se lanzó la primera versión en dos tipos de licencia, la Indie para pequeños desarrolladores, y la Profesional para grandes compañías, pero solo estaban disponibles para los ordenadores de Apple. Los juegos solo se podían compilar para ser jugados en esa misma plataforma, pero en poco tiempo añadieron la posibilidad de poder compilar juegos para plataformas Windows y para navegadores web.

En 2008 el motor ya gozaba de cierta popularidad, pero hubo un gran punto de inflexión que ayudo a la compañía a despegar. Apple anunciaba la tienda de aplicaciones para sus recientemente lanzados teléfonos iPhone. Con el auge que estaban sufriendo los juegos para teléfonos móviles, la compañía vio una gran oportunidad en ser los primeros en ofrecer compatibilidad con los teléfonos iPhone, y así lo hicieron a finales de ese año.

En 2009 con la salida de la versión 2.5 del motor, se dio compatibilidad para poder usar el programa en PCs con sistema operativo Windows. A su vez ese mismo desaparece la versión Indie del programa para convertirse en una licencia gratuita. En 2010 se lanza la “Asset store“ o tienda de recursos, donde los usuarios pueden vender toda clase, objetos, sonidos, modelos, animaciones... a otros desarrolladores.

Una de las grandes ventajas que ofrece el motor, es la posibilidad de compilar los juegos para diferentes plataformas, actualmente son, IOS, Android,Windows Phone, Blackberry, Windows, Mac, Linux, navegadores web, PS3, PS4, PSVITA, Playstation Mobile, Xbox 360, Xbox One y Wii U. Gracias a este amplio abanico de posibilidades y a la capcidad del motor de crear tanto juegos en 2D como 3D, muchas grandes compañías como Ubisoft, Square Enix, Microsoft o Electronic Arts han usado Unity como herramienta de desarrollo. Incluso Nintendo ha deignado a Unity como herramienta oficial para el desarrollo de su consola Wii U. Los lenguajes de programación que soporta Unity son, UnityScript, C#, y Boo.

Unity cuenta con dos tipos de licencias *[15]*, la gratuita y la profesional. Ambas son muy similares pero la gratuita está limitada, sobre todo en que la versión gratuita no es capaz de reproducir vídeos, la calidad gráfica es menor y que no puede ser usado por compañías con ganancias superiores a 100000$. El precio de la licencia profesional es de 1500$ y el desarrollo profesional para IOS y Android se han de comprar aparte como módulos por 1500€ cada uno.

7.2.- Alternativa al motor Unity

Unity es la mejor opción para hacer el proyecto, tanto por la gran comunidad de desarrolladores que tiene, como por su versatilidad. Hay una gran multitud de motores de videojuego disponibles en el mercado, pero ninguno es tan completo ni tan sencillo de usar por el valor de la licencia gratuita de Unity. Sin embargo, en el mercado hay un motor muy conocido en el sector, el Unreal Development Kit *[16]*, que aunque es de pago, da la posibilidad de usarse gratuitamente con ámbito no comercial.

El Unreal Development Kit (UDK) , es un entorno de desarrollo basado en el motor Unreal Engine. Este motor se lleva usando en el ámbito profesional desde 1998 con la salida del juego Unreal por parte de la compañía Epic Games. Siempre ha sido uno de los motores más importantes del sector, ya que cada versión suponía un importante salto gráfico y tecnológico. Una de las ventajas de este motor, es que los juegos en los que se usaba, siempre han sido relativamente fáciles de modificar. Esta característica llamada modding, siempre ha sido muy popular, porque permitía a los usuarios experimentados el poder cambiar aspectos del juego a su gusto e incluso hacer que el juego cambiase completamente. Gracias a la gran comunidad de usuarios en este sector, en 2009 Epic Games decidió sacar el UDK para el público en general. Actualmente hay dos versiones del motor:

-Unreal Development Kit: Para desarrollar juegos para ordenador y consolas de la pasada generación, PS3, XBOX 360, etc...

-Unreal Engine 4 *[17]*: Para las consolas de nueva generación, PS4 y Xbox One, y ordenador. Es la versión mas actual del motor, la que ofrece mayor calidad y las mas avanzada tecnológicamente.

Aun cuando el Unreal Engine 4 es la versión mas avanzada, ambos motores conviven en el mercado porque aun se están desarrollando juego para la previa generación de consolas. Si se quiere lanzar al mercado un juego desarrollado con esos motores, se ha de pagar una cuota y un porcentaje del dinero recaudado por el juego.

8.- EL JUEGO

8.1.- Historia

“ -No esperes estar al mando de las fuerzas del bien y que tu objetivo sea el de acabar con los villanos que se crucen en tu camino. Tomas el papel de un malvado brujo que busca derramar la sangre de la princesa para poder crear poderosos hechizos y maldiciones. ¿Cuanto de poderosos?, lo suficiente para que el reino se arrodille a tus pies. El principal problema es que el Rey sabe de tus peligrosas intenciones, y se ha adelantado a tus maquiavélicos planes. Ha mandado encerrar a la princesa en una enorme torre llena de laberínticos pasillos y está custodiada por valientes caballeros que no dudarán en dar su vida por ella. Pero lo que ellos no sospechan es que tienes en tus manos un poderoso artefacto mágico que puede cambiar las tornas de esta misión tan peligrosa a tu favor . Con el poder del tablero eres capaz de resucitar y controlar a los muertos, eso si, solo a uno. Cada vez que activas el tablero, unas gemas aparecen mágicamente en su superficie, y notas que su poder está enlazado a algo mas oscuro, un muerto viviente. Un esqueleto se levanta en el interior del primer piso de la torre, esta bajo el control del tablero, combinando las gemas tu decides sus acciones, su fuerza y sus debilidades. ¿Seras tan hábil como para conseguir llegar hasta la princesa?- ”

8.2.- Propiedades del juego

Como se ha dicho con anterioridad, uno de los pilares del juego es el de intentar hacer una mezcla entre el género roguelike y el de puzzle. Al encontrarse con un enemigo se activa el modo puzzle, es en ese modo en el que el jugador se pasará la mayor parte del juego y el que supondrá el mayor reto para el jugador. Acabar con enemigos dará experiencia al jugador que le servirá para subir de nivel, y así a su vez conseguir puntos para asignarlos a diferentes habilidades. Aunque el jugador se pase la mayor parte en el modo puzzle que en el modo laberinto, son las bases de los juegos roguelike los que primarán en ambos.

8.2.1.- Salvado y muerte.

El juego se salva automáticamente cada vez que el jugador sube de experiencia, gana un combate, usa un objeto o supera un nivel de la mazmorra. Si se cierra la partida el jugador podrá seguir jugando con ese mismo personaje. Sin embargo cada vez que el jugador continúe una partida, vuelve a aparecer al inicio del nivel de la mazmorra en el que se encuentre. En ningún momento se da la posibilidad de que el jugador salve la partida manualmente.

Cuando los puntos de vida del personaje llegan a 0, el personaje muere, y la muerte es permanente, es decir no podrá volver a cargar la partida a un estado anterior, los datos de su partida se borran y solo se le da la opción de volver a empezar el juego desde el principio.

Cuando se salva una partida lo hacen tanto sus puntos actuales de vida o magia como su experiencia y de sus puntos de atributo. Este proceso se hace guardando en un vector todos los valores actuales, y este se graba en el disco duro usando ArrayPrefs2 *[18]*. ArrayPrefs2 es un añadido que permite salvar diferentes tipos de vectores en la memoria, es más robusto y eficiente que el propio método que usa el motor Unity.

8.2.2.- Aleatoriedad

Para poder ofrecer al jugador una experiencia nueva cada vez que inicie una partida se ha tomado muy en cuenta el factor de aleatoriedad en muchos de los puntos de desarrollo del juego. El simple hecho de hacer que cada partida sea diferente hace que la rejugabilidad del mismo se haga mayor y que el jugador no pierda el interés por el juego rápidamente. Los valores aleatorios con los que se encuentra el jugador cada vez que juega o que carga su partida son, la configuración del camino a seguir para poder alcanzar la salida hacia el siguiente nivel, los objetos interactivos que encuentre en cada habitación, y tanto la habilidad como las características de cada enemigo que se encuentre en su camino. Mas adelante se habla en profundidad de cada uno de ellos.

8.2.3.- Dificultad

Uno de los puntos más importantes de los Roguelike es su elevada dificultad. Eso hace que el jugador tome el juego no solo como una forma de diversión, sino que también hace que le produzca un reto. El objetivo es que el jugador tenga que planear bien su estrategia y sopesar sus decisiones. Cada vez que el jugador asciende un nivel de la torre, tendrá que enfrentarse a enemigos más poderosos, pero que a su vez dan mayor experiencia para que pueda subir mas rápidamente de nivel y volverse mas fuerte. Un enemigo de nivel superior implica que tiene mayor cantidad de puntos de vida, defensa y ataque, pero eso no implica que todos los enemigos del nivel sean exactamente iguales, hay un cierto factor de aleatoriedad que puede hacerlos mas o menos poderosos.

8.2.4.- Escenas

El juego consta de 4 tipos de escenas:

-Título, Donde se le da la opción de empezar un juego nuevo, continuar una partida anterior o salir del juego.
-Juego, Escena principal, es el juego en sí mismo.
-Derrota, Se abre cuando el jugador es derrotado, y se le invita a jugar de nuevo.
-Victoria, El personaje alcanza la cima de la torre y mata a la princesa.

8.3.- La torre del laberinto.

Cada nivel de la torre está formado por un laberinto de habitaciones, y tanto la configuración del laberinto como los elementos que pueda encontrarse el jugador dentro de cada habitación es aleatorio. El jugador se mueve de habitación en habitación mediante el uso de puertas situadas en los tabiques de las mismas, superior, inferior, derecha e izquierda.

Todos los niveles están formados por 16 habitaciones situadas en una configuración de 4x4, 4 filas por 4 columnas. El laberinto se crea de forma procedural mediante la aleatorización de las puertas que pueda tener la habitación, pero teniendo en cuenta una serie de reglas:

-Las habitaciones situadas en los bordes no pueden contener puertas que den lugar a habitaciones fuera de la malla de 4x4, es decir, una habitación situada en la primera fila no puede contener una puerta superior porque daría lugar a que el jugador se saliera del mapa.

-Se puede permitir que cada nivel tenga habitaciones a las que el jugador no tenga acceso. Esto se permite para que se pierda la noción de estar en una malla regular.

-Siempre tiene que haber un modo de que el jugador tenga acceso a la salida de cada nivel.

En La Figura 8.1 se puede ver un ejemplo de como es una habitación. En el centro las escaleras de entrada del nivel y el personaje, y en el lateral derecho e inferior las puertas. Las puertas inferiores no muestran ni la puerta ni el marco para no entorpecer la visión del jugador, pero se marca su existencia con una marca rectangular en el suelo.

fig. 8.1

Para entender como se hace este proceso, hay que imaginarse que cada nivel es una cuadrícula, y por ejemplo, la primera habitación está situada en la primera columna de la primera fila, la segunda habitación en la segunda columna de la primera fila, etc... La información de las puertas de cada habitación se guarda en una matriz tridimensional, “doorSetup[X,Y,Z]“, donde “X” es el numero de la fila, “Y” es el numero de columna, y Z corresponde a la configuración de puertas de la habitación. Para crear esta aleatorización, el código va en orden habitación por habitación variando la posibilidad de haber puertas en sus tabiques o no, siempre teniendo en cuenta los valores de las habitaciones que ya ha creado. El orden en el que el código va recorriendo las habitaciones, es de fila por fila recorriendo de izquierda a derecha todas las habitaciones. El valor “Z“ de la matriz corresponde a las cuatro posiciones (0,1,2,3), en las que puede haber una puerta en el siguiente orden, superior, derecha, inferior e izquierda. Y para terminar, la existencia de una puerta se marca con un “0“, o un “1“, el “0“ implica que en esa posición no hay una puerta, y el “1” que si la hay.

Expongamos un ejemplo para entenderlo mejor. En la Figura 8.2, se puede apreciar un trozo del código usado para configurar las puertas de la segunda habitación de la primera fila.

```
up=0;
right= Ran();
down= Ran();
left=doorSetup[X,Y-1,1];

Doors (X,Y, up, right, down, left);
SaveDoors ("Room2", up, right, down, left);
```

fig. 8.2

En la primera línea del código se especifica que no hay puerta en la pared superior, porque es una habitación de la primera fila. Sin embargo en las dos líneas siguientes se llama a una función que se encarga de dar un valor aleatorio, o “0“, o “1“, se esta definiendo la existencia o no de una puerta en esa posición. La cuarta línea es totalmente diferente, pero se encarga de determinar la existencia de la puerta izquierda. Lo que hace es mirar en la matriz doorSetup en esa misma fila, “X”, si en la habitación anterior, “Y-1“, el valor de la puerta derecha, “1“. Si ese valor es “0”, quiere decir que no hay puerta en la habitación contigua, y si es 1, sí que la hay. Las dos siguientes lineas se encargan de darle formato a la información anterior y almacenarla. Cada vez que el jugador carga la partida, crea una nueva, o supera un nivel, este proceso genera de misma manera un nuevo laberinto de habitaciones. Para interactuar con una puerta el jugador ha de apretar la tecla “espacio” estando pegado a la puerta.

En la Figura 8.3, se muestra de forma esquemática un ejemplo del laberinto generado por el juego. El punto de vista en este ejemplo es de una vista cenital, para apreciar todas las habitaciones del laberinto. Los rectángulos equivalen a una habitación, y las marcas de las paredes son las puertas. El personaje es en cuadrado situado en la habitación superior izquierda.

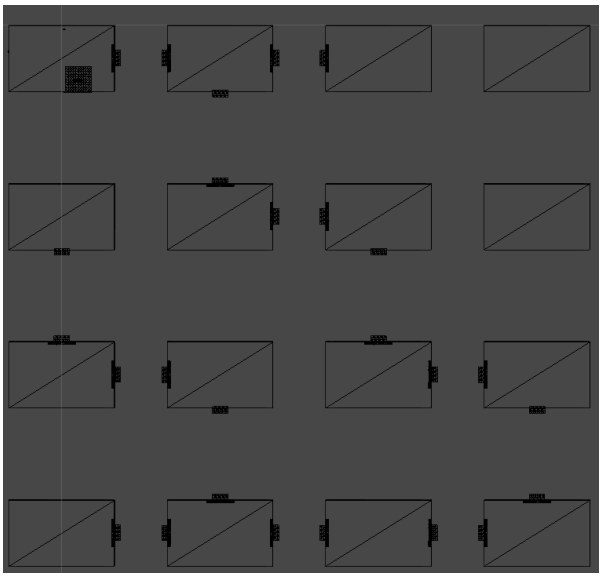


fig. 8.3

Los pisos que conforman la torre no son infinitos, si el jugador es capaz de alcanzar el undécimo nivel gana la partida. La salida de cada nivel viene determinada por unas escaleras ascendentes (Figura 8.4), cuando el jugador activa la salida asciende al siguiente nivel, pero es un camino de no retorno porque no es posible que el jugador vuelva al nivel anterior .

fig. 8.4

8.3.1 Interfaz y control

La interfaz esta formada por dos zonas principales (Figura 8.4). La parte izquierda muestra la información del personaje, vida magia y energía, y sus atributos. En la parte superior de esta zona, se encuentra la imagen del esqueleto y un número que indica su nivel. En la parte derecha, y mucho mas grande, se muestra el escenario y el nivel del piso de la torre.

fig. 8.4

El método de control del personaje es muy sencillo:

-Movimiento: Mediante el método de control de teclado más usado en la actualidad, las teclas “W, A, S, D”. Corresponden al moviemiento en las direcciones, arriba, izquierda, abajo y derecha, respectivamente.

-Interacción: Tecla “espacio“, para interactuar con las puertas u otras partes del escenario.

-Botones: Cuando el jugador tiene puntos de atributo para gastar, lo hace mediante el ratón clickando en los botones “+” (Figura 8.5) .

fig. 8.5

8.4.- Objetos interactivos.

En cada una de las habitaciones puede haber un objeto que resulta beneficioso para el jugador, un enemigo, la salida y entrada del nivel o simplemente puede estar vacía. Las probabilidades de encontrarse con un enemigo es mayor que la de encontrar algo beneficioso. El proceso de escoger lo que hay en cada habitación también se hace de manera aleatoria, pero de una forma mucho mas sencilla. En un vector se guarda una cantidad de los objetos y enemigos mencionados anteriormente, y se desordena el vector. Luego es importante definir la entrada y la salida del nivel, la entrada es la habitación en la que aparece el personaje, y la salida, la habitación que da acceso al siguiente nivel. Para evitar que la salida y la entrada se encuentren muy juntas, se limita que la entrada solo pueda existir en la primera fila, y la salida en la última. Para finalizar, se asigna un valor de cada posición del vector a cada una de las habitaciones del nivel y se colocan en el centro de las mismas.

En el juego, los objetos que ayudan al jugador en su hazaña para llegar hasta el último nivel, se denominan altares. El jugador es capaz de interactuar con ellos acercándose presionando la tecla de “espacio“. Cada altar solo se puede activar una vez. Existen cuatro tipos de altares, y sus efectos son:

-Altar de curación: Sana al personaje una pequeña cantidad de salud. Figura 8.6.

fig 8.6

-Altar de magia: Otorga 5 puntos de magia. Figura 8.7.

fig 8.7

-Altar de energía: Otorga 5 puntos de energía. Figura 8.8.

fig 8.8

-Altar de experiencia: Otorga cierta cantidad de experiencia. No es una cantidad fija, es similar a la experiencia de otorgada por los enemigos, depende del nivel de la torre en el que se encuentre el jugador. Figura 8.9.

fig 8.9

En la Figura 8.10, se muestra un ejemplo del código encargado de activar un altar. En la primera línea están las condiciones para que el jugador active el altar, que apriete la tecla “espacio“, que esté pegado al altar, y que este no estuviese usado con anterioridad. Si los requisitos se cumplen, la cuarta línea se encarga de activar una función en el código del personaje que le aumenta la energía en 5 puntos. Por último se desactiva el altar para que el jugador no lo pueda volver a utilizar.

```
if (Input.GetKeyDown ("space")&&touchcolider==true&&deactivated==false){
    if(gameObject.tag=="energy"){
        touchcolider=false;
        playr.GetComponent(playerskelstats).UpdateStats ("penergy",5);
        DeactivateShrine();
    }
```

fig. 8.10

Aparte de los altares, existen dos objetos más con los que el jugador se puede encontrar, las escaleras de entrada, y las escaleras de salida. Las escaleras de entrada no son interactivas, simplemente marcan por donde entra el personaje al nivel. En cambio las de salida sirven para ascender al siguiente nivel de la torre.

8.5.- Juego puzzle

Cuando el jugador entra en una habitación y hay un enemigo (Figura 8.11), inmediatamente se inicia el modo de combate, el juego puzzle. En ese mismo instante el juego cambia completamente, pasa de ser 3D a 2D y el sistema de control se cambia del teclado al ratón. Ante el jugador aparece el tablero y la información del enemigo con el que esta luchando. El jugador no tiene la posibilidad de salir de este modo de juego hasta que se acabe el combate. El combate acaba cuando los puntos de vida de cualquiera de los dos llega a cero. Si el jugador gana, volverá al mismo lugar en el que se encontraba en la torre, pero si pierde, deberá de empezar el juego de nuevo desde el principio.

fig. 8.11

8.5.1.- Interfaz del juego puzzle

La nueva interfaz que se le presenta al jugador se puede dividir en tres partes (Figura 8.12), una zona central con el tablero, y dos laterales con información del jugador y del enemigo.

fig. 8.12

La parte central es la más importante de todas, es donde se encuentra el tablero y las gemas que el jugador debe conectar. Esta zona está dividida en cinco grupos de gemas, cada vez que el jugador hace click en un grupo, sus gemas cambian de posición moviéndose en el sentido de las agujas del reloj. Si el movimiento de las gemas produce algún emparejamiento se señala con un efecto visual. En la Figura 8.13 se muestra un ejemplo de un emparejamiento. El jugador ha clickado sobre el bloque de gemas superior izquierdo, sus gemas se han movido y al hacerlo se han emparejado dos gemas azules, se indica con una linea azul.

fig. 8.13

La zona izquierda de la interfaz muestran los atributos del jugador. En la parte superior estan las barras que indican su cantidad de vida, magia y de energía. En la parte inferior hay cinco iconos que se ajustan a las formas y colores de las gemas. Estos iconos indican el cálculo del valor de las gemas que el jugador conecte en la zona del tablero. Este cálculo depende los atributos del personaje y de las bonificaciones que pueda juntar en el tablero.

Finalmente, en la zona de la derecha están los atributos del enemigo en la parte superior, y en la inferior el botón “GO!“ para cambiar de turno. La información del enemigo puede que parezca escasa, pero es muy importante para el jugador, porque muestra la cantidad de vida restante del enemigo, y el ataque lanzado contra del jugador. El conocimiento de este valor es importante para que el jugador pueda decidir sus prioridades a la hora de conectar las gemas del tablero.

8.5.2.- Reglas juego puzzle

El jugador tiene un número limitado de veces en las que puede mover los grupos de gemas, ese límite es de tres movimientos. Para ayudar al jugador a planear su estrategia, en cada bloque se muestra el número de veces que ha movido las gemas de ese grupo. Cuando el número total de movimientos es igual o menor que tres, se activa el botón “GO!“. Cuando el jugador lo presiona acaba su turno, y empieza el turno del enemigo. Al acabar el turno, si hay gemas conectadas por el jugador, estas se destruyen y se generan unas nuevas en su lugar. En la Figura 8.14, se muestra el esquema de las conexiones posibles entre las gemas de los bloques. Una conexión se produce cuando se emparejan dos gemas idénticas. Al inicio del combate, el juego se asegura de que no exista ninguna conexión activa porque esto daría una ligera ventaja al jugador.

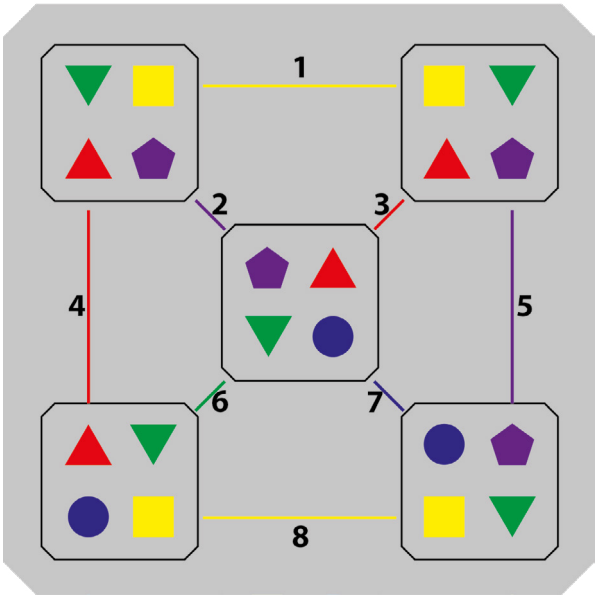


fig. 8.14

Existen 5 tipos diferentes de gemas:

-Gema roja: Ataque, se produce un ataque al enemigo.
-Gema amarilla: Defensa, el jugador puede defenderse total o parcialmente del ataque de enemigo.
-Gema verde: Curación, si el jugador esta herido recupera puntos de vida.
-Gema azul: Magia, el jugador llena parte de su barra de magia.
-Gema morada: Energía, el jugador llena parte de su barra de energía.

La potencia del ataque, defensa, la curación, o los puntos que gana de magia o energía depende de los puntos de atributo que el jugador tenga asignados en ese tipo de gema, o haciendo un combo. El combo se produce cuando el jugador realiza tres combinaciones contiguas. El valor de las gemas implicadas en el combo se duplica. Al ser un efecto tan potente, se activa en el jugador una predisposición a buscar este tipo de combinación aumentando así el punto estratégico que ya tiene el juego. En la Figura 8.15 se aprecia el ejemplo de un combo en la parte inferior del tablero, marcando su efecto con un símbolo “x2”.

fig. 8.15

Cuando acaba el turno del jugador, el enemigo no moverá gemas en el tablero, el tablero es un símbolo del control que tiene el brujo sobre el esqueleto. Aun así en su turno, el enemigo produce puntos de ataque, defensa, curación, energía y magia.

8.6.- Personaje

El personaje principal del juego es el esqueleto controlado por el jugador, mas bien dicho, por el brujo. La idea es que el brujo controla las acciones del esqueleto mediante las gemas del tablero, y así dar sentido a que cuando se active el juego de puzzle cuando se entra en combate. También se usa la imagen del esqueleto para intentar dar la impresión de que es algo prescindible, si muere se crea otro esqueleto y se intenta de nuevo, de este modo se da mas coherencia a la idea de que la muerte es permanente. Sin embargo, el tablero mágico solo tiene el poder de revivir al esqueleto en el primer nivel de la torre.

8.6.1.- Estadísticas personaje

En la Figura 8.16 se pueden apreciar las características del personaje, corresponden a:

-Barra de vida: La cantidad de vida del personaje, si llega a 0, muere.
-Barra de magia: Cantidad de poder mágico que puede gastar.
-Barra de energía: Cantidad de energía que puede usar.
-Bonus de ataque: Bonus al juntar gemas rojas.
-Bonus de defensa: Bonus al juntar gemas amarillas.
-Bonus de curación: Bonus al juntar gemas verdes.
-Bonus de magia: Bonus al juntar gemas azules.
-Bonus de energía: Bonus al juntar gemas moradas.
-Barra de experiencia: La cantidad de experiencia necesaria para subir de nivel.

fig 8.16

Los valores de los bonus, son lo que hacen que el jugador sea mas fuerte, por ejemplo, bonus de ataque hace que el enemigo reciba más daño, bonus de magia aumenta la cantidad de magia que se gana al combinar gemas azules, etc. Estos valores de bonus se pueden subir como el jugador elija cada vez que tenga puntos para gastar en ellos. Al principio de la partida, se le dan al jugador 4 puntos para que empiece a personalizar a su personaje. Para conseguir mas puntos, el jugador tiene que ganar experiencia para subir de nivel, y gana a razón de dos puntos por cada nivel. La cantidad de puntos que necesita el jugador para subir el nivel de bonus de una habilidad no es fijo, sino que cada vez va a necesitar invertir mas puntos para subir su nivel de bonus. En la Figura 8.17, se puede apreciar en la esquina superior derecha que el jugador tiene 2 puntos para gastar. También se puede ver que su bonus de ataque es de 2, y al lado del botón "+" se le indica que necesita invertir 3 puntos mas si quiere subir la habilidad de ese bonus.

fig 8.17

En el juego solo hay dos formas de ganar experiencia, ganando combates o encontrando altares de experiencia. Subir de nivel no solo te da puntos para gastar, sino que también aumenta tu barra de vida, magia y energía

8.6.2.- Habilidades del personaje

Durante el combate el jugador tiene acceso a dos habilidades especiales:

-El ataque mágico: Solo está disponible si en el tablero el jugador ha conectado un combo, y si tiene almacenado como mínimo 10 puntos en su barra de energía. Su poder hace que se duplique el poder de las gemas conectadas por el combo.

-El ataque de energía: Se activa al reunir 15 puntos de energía. Permite al jugador destruir un bloque entero de gemas. No se gana el valor de las gemas destruidas, pero si en ese mismo turno el jugador conecta gemas, estas ganan un bonus adicional.

8.7.- Enemigo.

El jugador no lo puede tener fácil para llegar hasta la princesa, por eso hace falta que en su camino se encuentre con enemigos. En cada nivel de la torre el jugador se puede encontrar con más cantidad de enemigos que de altares. Es totalmente necesario que esa cantidad sea mayor porque que el jugador necesita vencerlos para ganar experiencia y poder ganar puntos de atributos. La experiencia ganada al vencer por un enemigo depende del nivel de la torre en el que se encuentre el jugador, pero no todos los enemigos del mismo nivel otorgan la misma experiencia, hay un ligero factor aleatorio de mas o menos el 10% . Otro de los factores a tener en cuenta, es que si el nivel del personaje es superior al de los enemigos, estos otorgan menos experiencia, si el nivel de los enemigos es mayor, más experiencia. De esa manera se compensa la diferencia de dificultad que ofrece el acabar con un enemigo de diferente nivel.

La cantidad de experiencia necesaria para que el jugador suba de nivel se duplica cada vez, sin embargo la experiencia de los enemigos asciende de forma ligeramente exponencial. El porqué se hace de esa manera diferente entre los dos, es que el jugador subía de nivel demasiado rápido y se descompensaba el nivel de dificultad. De esta manera se consigue que el jugador se vea obligado a matar a mas cantidad de enemigos por cada nivel que quiera subir.

8.7.1.- Características del enemigo.

Todas las características del enemigo se crean en base al nivel en el que se encuentra y en un pequeño factor aleatorio para que los valores de cada enemigo sean diferentes. Ese pequeño factor aleatorio es muy importante porque se usa también en la inteligencia artificial. En la Figura 8.18 se muestra parte del código encargado de este proceso.

```
if(enemylevel<=5){enemyskill=Random.Range(1,6);}
else{enemyskill=Random.Range(4,10);}
enemymaxhealth=(enemylevel*10);
enemymaxhealth=enemymaxhealth+Mathf.Round(parseFloat(enemymaxhealth*enemyskill/10));
enemyhealth=enemymaxhealth;
for (var I:int=1; I<=5;I++){
    if (enemylevel<=I*2){enemymaxenergy=I*5;}
}
enemymagic=Random.Range(0,enemyskill);
enemyenergy=Random.Range(0,enemyskill);
enemymaxmagic=enemymaxenergy*1.5;
```

fig. 8.18

Las dos primeras lineas son las que se encargan de escoger el factor aleatorio, “enemyskill“ pero tiene en cuenta el nivel de la torre para no hacer enemigos de bajo nivel demasiado poderosos, o muy débiles de nivel alto. El enemigo posee barras de vida, energía y magia como las del personaje, porque también es capaz de realizar ataques especiales. En la creación se le da una ligera cantidad de magia y energía, se hace para que los enemigos se vuelvan mas impredecibles y ataquen mas fuerte al principio del combate, sobre todo los de nivel mas alto.

8.7.2.- IA enemigo.

La inteligencia artificial es uno de los puntos mas importantes de tener en cuenta a la hora de desarrollar un videojuego. Implica en como reaccionan tanto los enemigos como los aliados ante tus acciones. Si la inteligencia artificial de un juego no es buena puede implicar directamente en la dificultad de un juego haciéndolo demasiado fácil, o incluso puede llegar a hacerlo injugable. Los problemas en la IA crean una notoria frustración en el jugador, implicando que deje el juego sin terminar. De igual manera hay que dotar a la inteligencia artificial de cierta aleatoriedad, que sea capaz de cometer errores, porque si es demasiado “lista“, puede dar la impresión de que la máquina está haciendo trampas.

En este juego la inteligencia artificial del enemigo se basa en prioridades (Figura 8.19). Cada vez que llega su turno analiza si va a recibir daño y si esa cantidad de daño es peligroso para la cantidad de vida restante que le queda. En base a esa información reacciona, si va a recibir un daño importante, aumenta la prioridad de que se defienda ante ese daño, si no, aumenta la prioridad de ataque. De la misma manera si esta falto de vida aumenta la de curarse. Sin embargo la prioridad de ganar magia o energía es mas aleatoria. El factor “enemyskill“ vuelve a jugar un papel importante (Figura 8.20), porque dependiendo de su valor, aumenta en mayor o menor cantidad todas las prioridades. Esto se hace para que enemigos con un “enemyskill“ alto se comporten de manera menos aleatoria y resulten mas difíciles de vencer. Aun así, una prioridad nunca va a llegar a ser del 100%, de ese modo se evita que los enemigos de alto nivel tengan una mínima probabilidad de equivocarse. El resultado final es que la prioridad toma un valor decimal entre “0” y “0.8“. La prioridad de curarse se ve mas reducida que las demás para evitar que el enemigo resulte excesivamente difícil.

```
var atthealth:float=0;
atthealth=parseFloat(playrattack)/parseFloat(enemyhealth);
var maxhhealth:float=0.00;
maxhhealth=parseFloat(enemymaxhealth)/parseFloat(enemyhealth);
for (var I:float=0.0; I<=5.0;I++){
    if (atthealth>I/10){attackpriority=priority[parseInt(I)];defencepriority=priority[5-parseInt(I)];}
}
if (atthealth==0){attackpriority=priority[0];defencepriority=priority[5];}
for (var J:float=5.0; J<=0.0;J--){
    if (enemymaxhealth/enemyhealth>=J*0.2){healingpriority=priority[parseInt(J)];}
}
if (enemyenergy>=10){energypriority=true;}else{energypriority=false;}
if (enemymagic>=15){magicpriority=true;}else{magicpriority=false;}
```

fig. 8.19

```
attackprob=(parseFloat(enemyskill)+parseFloat(attackpriority))/25;
defenceprob=(parseFloat(enemyskill)+parseFloat(defencepriority))/25;
healingprob=(parseFloat(enemyskill)+parseFloat(healingpriority))/45;
enemyhealing=enemyhealing+CalSuccess(healingprob,2);
enemyattack=enemyattack+CalSuccess(attackprob,2);
enemydefence=enemydefence+CalSuccess(defenceprob,2);
```

fig. 8.20

El siguiente proceso es calcular el éxito o no de la acción basándose en las prioridades calculadas anteriormente (Figura 8.21). Este proceso se hace escogiendo un número decimal aleatorio entre "0" y "1", si el número escogido se encuentra dentro del rango de la prioridad, la acción sera un éxito. Se repite una segunda vez con una prioridad mucho menor para dar mas variedad al resultado y suman los éxitos. El único proceso que no es aleatorio es el de los ataques especiales, que se realizan inmediatamente cuando el enemigo reúna la magia o energía necesaria. Estas habilidades especiales simplemente lanzan un ataque sin pasar por el proceso de cálculo de las prioridades. El ataque de energía es un poco mas poderoso que el de magia porque necesita más puntos para activarse.

```
for (var I:int=0;I<repeat;I++){
    if (Random.Range(0.0,1.0)<=pvalue){
        success++;
        pvalue=pvalue/2.0f;
        gemsconnected++;
    }
}
```

fig. 8.21

Dependiendo de las prioridades que supere, se determina que la inteligencia artificial del enemigo a escogido realizar una acción de ataque, defensa, curación, etc... . Lo siguiente es cuantificar el valor de la potencia de esa acción. Para ese proceso también se escoge un número aleatorio, pero para calcular el rango en el que se encuentre este número, se usan tanto el nivel del enemigo como el valor "enemyskill", cuanto mayor sean ambos mas fuerte será el ataque, defensa o cualquiera de las acciones que hayan superado.

8.8.- Estilo y diseño

Una vez decidida la idea principal del juego, es hora de moverse al apartado gráfico. El estilo del juego está basado en la idea tradicional de un mundo de fantasía, princesas, torres, brujos, caballeros, etc... así que hay que buscar un enfoque medieval en cuanto al escenario. Como el juego transcurre en un torreón, las paredes de las habitaciones han de ser de piedra. Este proceso empezó con la creación de las paredes de las habitaciones usando el programa ZBrush. (Figura 8.22).

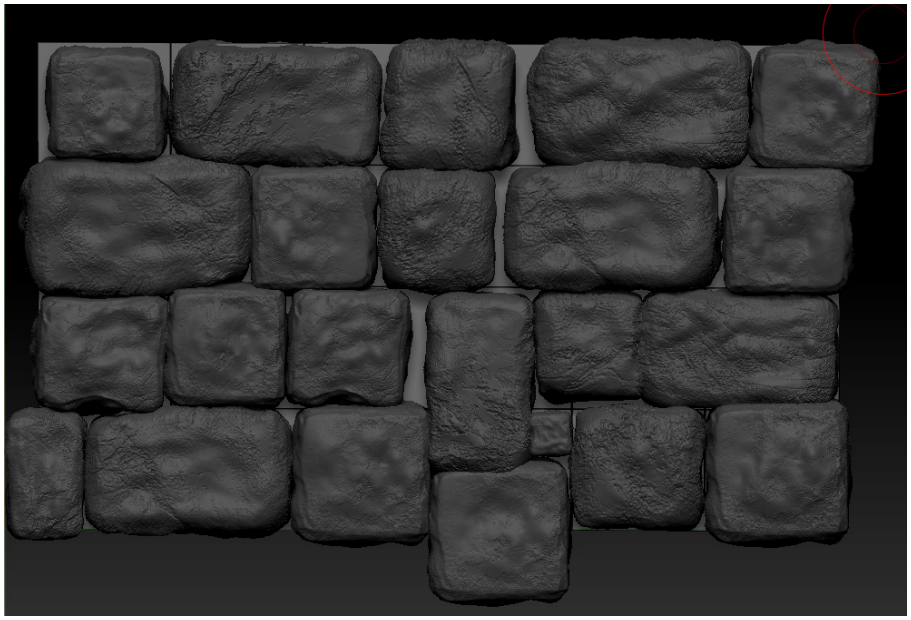


fig. 8.22

ZBrush es un software 3D que usa técnicas similares a las de la escultura para modelar. Es un programa muy versátil, sobre todo para hacer formas orgánicas y se usa mucho en el cine y los videojuegos. El resultado era bastante bueno, pero al importar el modelo en Unity, el hecho de tener la cámara alejada del muro hacía que se perdieran todos los detalles. También el hecho de usar la versión gratuita de Unity lastraba la calidad de las sombras. Así que se buscó un estilo totalmente diferente para representar los muros de las habitaciones, pasar a un estilo mas bidimensional, de estilo dibujado. En la Figura 8.23 se muestra un ejemplo del estilo a seguir.

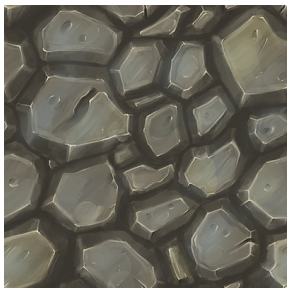


fig. 8.23

Para conseguir un mejor efecto en este tipo de texturas, se desactiva en el motor unity las luces direccionales y las sombras, y se ilumina la escena con una luz ambiente. Así se consigue una escena iluminada de forma homogénea y sin sombras. Hacer este estilo de texturas implica mas trabajo, porque hay que pintar a mano las sombras, medios tonos, reflejos, etc.. siempre teniendo en cuenta en donde se ubicaría la fuente de luz para que todas las sombras sigan la misma dirección. El resultado final era bastante satisfactorio (Figura 8.24), con las lineas bien marcadas que le daban el aspecto caricaturesco que se buscaba. Gracias al buen resultado, este diseño sera el realizado en todo el proyecto.

fig. 8.24

El segundo cambio de estilo importante surgió en medio del desarrollo de proyecto. El problema de hacer texturas por separado sin ver como queda todo en conjunto hasta que el desarrollo del proyecto es avanzado. Siguiendo el diseño principal, todos los objetos tenían las aristas marcadas en negro. El problema estaba sobre todo en el tablero, las lineas negras daban aspecto de ser un diseño demasiado limpio y no quedaban del todo bien (Figura 8.25).

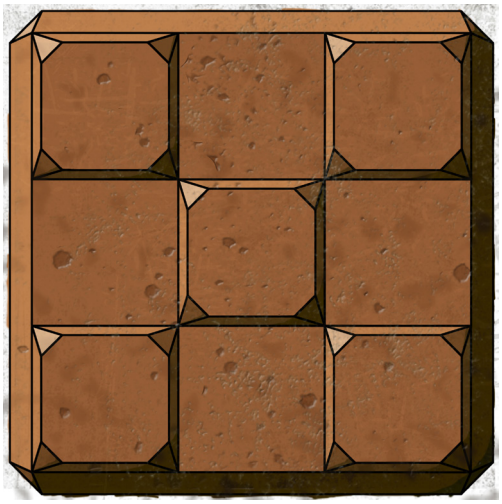


fig. 8.25

La solución fue la de cambiar las aristas negras por otras mas claras, hacerlas más finas y hacerlas menos definidas (Figura 8.26). Con estos cambios el tablero tiene mejor aspecto y tiene apariencia de estar gastado. Lo mismo se hace en todos los elementos de la interfaz, los marcos y los botones, pero las lineas negras se mantienen en el resto de elementos del juego.

fig. 8.26

8.9.- Recursos externos

Prácticamente todo el material usado en el juego ha sido creado especialmente para este proyecto, pero como el proceso de la creación de un juego es tan extenso, se han tenido que usar recursos externos. Concretamente en el modelado del personaje y del enemigo, sus animaciones y en los efectos sonoros y música del juego. Estos recursos se usan con las pertinentes autorizaciones por parte de sus autores. El resto de modelos, habitaciones, altares, puertas, etc... se han creado con el programa de modelado y animación Cinema 4D

8.9.1.- Sonido

Desde un principio se ha tenido en cuenta que para conseguir efectos sonoros bien grabados y sin ruidos había que recurrir a internet. Todos se han sacado de la web SoundBible.com *[19]*. La música sin embargo se ha obtenido de la web incompetech.com *[20]*. En un principio la idea era la de crear la música del juego con el software Fruit Loops, pero se descartó la idea debido a que el aprendizaje del software no era sencillo.

8.9.2.- Modelado y animación

Tanto el modelo de personaje principal como el del enemigo , se han adquirido desde la tienda de recursos de Unity *[21] ,[22]*. Ambos modelos incluían animaciones, solo las útiles para el proyecto se han utilizado. En el caso del esqueleto, se han cambiado algunas de las proporciones Como el modelo original era un simple esqueleto, hacia falta darle algo más de personalidad. Mediante el programa Cinema4D, se modelan partes de armadura y se añaden al modelo original. En la Figura 8.27 se pueden observar los cambios realizados. La textura original del esqueleto (Figura 8.28), se ha tenido que rehacer para dar un resultado visual acorde con el resto del proyecto (Figura 8.29). El enemigo también se ha retexturizado.

fig. 8.27

fig 8.28

fig 8.29

En las siguientes figuras, se aprecian las diferencias entre el texturizado nuevo del enemigo, (Figura 8.30), y la del texturizado original (Figura 8.31). El texturizado original es demasiado detallado, y tiene un estilo totalmente diferente al del juego.

fig 8.30

fig 8.31

Las animaciones importadas de la tienda de recursos son las de correr, morir, y la de “no hacer nada“. La animación de “no hacer nada”, se activa cuando el personaje está quieto. Es un conjunto de movimientos sutiles como por ejemplo el movimiento del pecho al respirar, movimientos de cabeza o de hombros, que se usan para que el personaje no se quede “congelado“ al estar quieto y dé aspecto de estar vivo.

8.10.- Problemas de programación y resolución de los mismos.

Los principales problemas ocurrían en el campo de la programación, ya que es la parte mas complicada del proyecto. El primero de los problemas que aparecieron en el proyecto, era que muchas veces se generaban laberintos sin salida. Para evitarlo, había que pensar en una manera de que siempre hubiese una salida, dándole unas pautas al código generador del laberinto. Esas pautas incluían el que en cada habitación tenia que haber como mínimo dos puertas, y que cada fila de habitaciones tuviese como mínimo una puerta que conectase con la fila inferior. El resultado fue desastroso porque al haber esas pautas, los laberintos se generaban casi idénticos, y la ruta hacia la salida era casi siempre la misma.

La solución acertada fue la de crear un código que recorría las habitaciones pasando por las puertas, y marcaba las habitaciones conectadas (Figura *8.32*). Primero se borraron las pautas de la solución anterior, de tal manera que el laberinto se volvía a generar de manera totalmente aleatoria. Solo se añadió una pauta, que la habitación superior izquierda siempre estuviese conectada a otra, es necesario para que el código que analiza el laberinto tenga siempre que empezar por la primera habitación. Luego de generar la habitación, el analizador recorre el laberinto y analiza si hay salida. Como aun no están colocados la entrada y la salida, se determina que el laberinto tiene salida si hay conexión entre la primera y la última fila. Para evitar que el nivel tenga pocas habitaciones, el analizador solo dará por valido un laberinto si tiene salida, y si como mínimo tiene 12 de las 16 habitaciones conectadas. Si no se cumple, se genera otro laberinto y se vuelve a comprobar.

```
for(var Z: int = 0; Z < 16; Z++){
    for(var X : int = 0; X < doorcheck.GetLength(0); X++){
        for(var Y: int = 0; Y < doorcheck.GetLength(1); Y++){
            if(doorcheck[X,Y]==1){
                if(X<3&&doorSetup[X,Y,2]==1){doorcheck[X+1,Y]=1;}
                if(Y<3&&doorSetup[X,Y,1]==1){doorcheck[X,Y+1]=1;}
                if(X>0&&doorSetup[X,Y,0]==1){doorcheck[X-1,Y]=1;}
                if(Y>0&&doorSetup[X,Y,3]==1){doorcheck[X,Y-1]=1;}
            }
        }
    }
}

notempty=0;
tempr=0;
for(var J : int = 0; J < doorcheck.GetLength(0); J++){
    for(var K: int = 0; K < doorcheck.GetLength(1); K++){
        tempr=doorcheck[J,K];
        notempty=notempty+tempr;
    }
}

if(notempty<12){Init();}
```

fig. 8.33

El segundo gran problema, apareció en la creación del modo puzzle. Al generarse las gemas de manera aleatoria, causaba que tan pronto empezaba el combate, habia la posibilidad de que algunas gemas ya estaban conectadas. El mismo problema aparecía cuando se regeneraban las gemas después del turno, ambos casos daban una gran ventaja al jugador. La solución fue la de guardar la información de cada bloque de gemas en la memoria. Cada vez que se regenera una gema, se busca la información de las gemas del bloque contiguo, y regenera una gema diferente si hay una conexión. Aquí surgía otro problema, existía la posibilidad de que dos gemas en conexiones de bloques contiguos se regenerasen a la vez, y esto producía que la una no sabía que era la otra. Por eso se cambio el proceso para que se realizase en orden, bloque por bloque en el sentido de las agujas del reloj, y el central de último.

Estos no fueron los únicos problemas en el transcurso del proyecto, pero si los que mas tiempo se tardaron en resolver. La web de Unity *[23]* posee una amplia documentación tanto sobre código (UnityScript, C#, y Boo) como sobre el uso del programa que ayuda mucho a la hora de comprender cómo funciona el motor.

9.- Conclusiones

Hacer un juego desde cero es una tarea difícil, mas si lo hace una sola persona y tiene poca experiencia en la tarea. Como el abanico de tareas es tan grande, animación, programación, modelado, diseño gráfico, etc... es muy difícil para una persona cubrir todos los campos satisfactoriamente sin la necesidad de ayuda. Implica tener que familiarizarse con software nuevo o tener que hacer tareas en las que no estas habituado en trabajar. Aunque en el desarrollo del proyecto solo se necesitase echar mano de un par de modelos y del sonido, el resultado ha sido bastante satisfactorio. Lo ideal para la creación de un videojuego es tener como mínimo una persona especializada en cada tarea, el proceso seria mucha más rápido y de mayor calidad.

El desarrollo del juego no tiene porque estar acabado, siempre hay trabajos que se pueden hacer mejor, pulir fallos o añadir contenido. Como objetivo principal, el rehacer los modelos de la tienda de contenido por unos propios y mejorar las animaciones. Entre las opciones que se pueden añadir al juego, entran la idea de hacer un sistema de inventario y añadir tesoros en la mazmorra. En los juegos Roguelike es importante el contenido, cuanto más opciones tenga el jugador de jugar su aventura, mejor, porque pueden hacer una experiencia de juego aún más aleatoria y divertida. Otra de las opciones a tener en cuenta es la de añadir Jefes Finales cada cierto número de niveles para complicar la vida aun mas al jugador. Después de añadir estos puntos, se puede decir que el juego podría estar listo para el mercado.

10.- Agradecimientos

-Agnés Borrâs Angosto. Por la gran guía, ayuda y comprensión en el desarrollo del proyecto.

-Rebeca Rodriguez Martinez. Por apoyarme desde el principio y de ayudarme a comprender de lo que soy capaz.

-Kevin MacLeod (incompetech.com). Por la gran labor que hace ofreciendo su música sin pedir nada a cambio.

-Comunidad Unity, Por la gran cantidad de tutoriales y ayuda que ofrece para usuarios no experimentados.

11.- BIBLIOGRAFÍA

[1] Roguelike: http://en.wikipedia.org/wiki/Roguelike

[2] Berlin Interpretation: http://www.roguebasin.com/index.php?title=Berlin_Interpretation

[3] Rogue: http://en.wikipedia.org/wiki/Rogue_(video_game)

[4] Rogue imagen 1: http://upload.wikimedia.org/wikipedia/en/1/17/Rogue_Screen_Shot_CAR.PNG

[5] Rogue imagen 2: http://upload.wikimedia.org/wikipedia/en/a/af/Rogue_Unix_Screenshot_CAR.PNG

[6] Strange Adventures: http://en.wikipedia.org/wiki/Weird_Worlds:_Return_to_Infinite_Space

[7] Chocobo's dungeon: http://www.gamesetwatch.com/2010/05/column_play_chocobos_dungeon_f.php

[8] FTL: http://www.ftlgame.com/

[9] Tetris: http://www.gamerzona.com/wp-content/uploads/2014/06/tetris.jpg

[10] El profesor Layton Layton: https://www.nintendo.es/Juegos/Nintendo-3DS/El-profesor-Layton-y-la-mascara-de-los-prodigios-527690.html

[11] Bejeweled: http://bejeweled.popcap.com/html5/0.9.12.9490/html5/Bejeweled.html

[12] Binding of Isaac: http://bindingofisaac.com/

[13] Puzzle Quest: http://imagenes.es.sftcdn.net/es/scrn/61000/61144/puzzle-quest-32.jpg

[14] Unity: http://news.dice.com/2013/06/03/how-unity3d-become-a-game-development-beast/

[15] Unity 2: https://store.unity3d.com/products

[16] Unreal Development Kit: https://www.unrealengine.com/products/udk/

[17] Unreal Engine 4: https://www.unrealengine.com/what-is-unreal-engine-4

[18] Arrayprefs2: http://wiki.unity3d.com/index.php/ArrayPrefs2

[19] Soundbible: http://soundbible.com/

[20] Incompetech: http://incompetech.com/

[21] Modelo esqueleto: http://u3d.as/content/bshgame/skeletons-pack/2fA

[22] Modelo enemigo: https://www.assetstore.unity3d.com/en/#!/content/15209

[23] Documentación Unity: http://docs.unity3d.com/Manual/