Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en ingeniería informática

Proyecto de videojuego de estrategia táctica por turnos en Unity.

Turn-based tactical strategy video game project in Unity.

Sergio González Guerra

La Laguna, 12 de Junio 2022

D. **Jesús Miguel Torres Jorge**, con N.I.F. 43826207-Y profesor Contratado Doctor de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

C E R T I F I C A (N)

Que la presente memoria titulada:

"Proyecto de videojuego de estrategia táctica por turnos en Unity."

ha sido realizada bajo su dirección por D. **Sergio González Guerra**,

con N.I.F. 79093820-V.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 12 de Junio 2022

Agradecimientos

Quiero agradecer a todo el profesorado de la Universidad de La Laguna por su ayuda en mi formación durante la carrera.

A Jesús Miguel Torres Jorge por guiarme durante el período de desarrollo de este proyecto.

A los alumnos del CSMC Aarón Rodríguez Pérez, Jacob González Marrero y a su profesor Carlos Alberto González Herrera por su colaboración en el desarrollo de la música de este proyecto.

Y a toda mi familia en especial a mis padres por todo su apoyo.

Licencia

Resumen

El objetivo de este trabajo ha sido el desarrollo de un videojuego de estrategia RPG, con gráficos pixelart, en 2D, usando el motor de videojuegos Unity y con el lenguaje C#.

El videojuego está desarrollado en PC y consiste en un RPG de estrategia táctica donde una serie de personajes son controlados por el jugador con el fin de derrotar al enemigo en un escenario que se estructura por casillas parecido al de un tablero de ajedrez.

El enemigo por el contrario será controlado por una IA e intentará derrotar al jugador de forma que el jugador tendrá la posibilidad de ganar o perder en el juego dependiendo de cómo se desenvuelva en este.

Palabras clave: Unity, C#, Algoritmo de Dijkstra, IA, RPG, Estrategia, TRPG, Pixelart.

Abstract

The purpose of this work was the development of an RPG strategy video game, with pixelart graphics, in 2D, using the Unity video game engine and the C# language.

The video game is developed on PC and consists of a tactical strategy RPG where a series of characters are controlled by the player in order to defeat the enemy in a scenario that is structured by tiles similar to a chessboard.

The enemy on the other hand will be controlled by an AI and will try to defeat the player so that the player will have the possibility of winning or losing in the game depending on the way he performs in the game.

Keywords: Unity, C#, Dijkstra's algorithm, AI, RPG, Strategy, TRPG, Pixelart.

Índice general

Introducción **1**
Contextualización 1
Método seguido 1

Objetivos **2**
Objetivos principales 2
Objetivos extra agregados 2

Estado del arte **3**
Historia de los RPG 3
Historia de los RPG Tácticos o TRPG 4
Inspiraciones del proyecto 5

Fases y desarrollo del proyecto **7**
Mapa del juego 7
Movimiento con el algoritmo de Dijkstra 8
Estadísticas de los personajes aliados. 9
Sistema por turnos 10
Enemigos con Inteligencia Artificial 10
Interfaz del jugador 11
Cálculo de áreas de rangos de ataque o movimiento 13
Lectura de un fichero para cargar los datos del mapa 14
Movimiento de la cámara. 15
Sensación de juego. 15
Música adaptada a videojuegos. 16

Conclusiones y líneas futuras **17**
Líneas futuras 17
Conclusiones 17

Summary and Conclusions **18**
Summary 18
Conclusions 18

Presupuesto **19**
Presupuesto 19

Apéndice: Algoritmos **20**
Algoritmo de Dijkstra 20

Índice de figuras

Figura 3.1: Dragon Quest 3
Figura 3.2: Final Fantasy 3
Figura 3.3: The Dragon and Princess 4
Figura 3.4: Fire Emblem Shadow Dragon & the Blade of Light 4
Figura 3.5: Final Fantasy Tactics 5
Figura 3.6: Fire Emblem 5
Figura 3.7: Advance Wars 6
Figura 4.1: Mapa de juego 7
Figura 4.2: Camino generado por el algoritmo de Dijkstra 8
Figura 4.3: Estadísticas de uno de los personajes 9
Figura 4.4: Cambio al turno del enemigo 10
Figura 4.5: Enemigo acercándose al personaje más cercano 10
Figura 4.6: Enemigo acercándose al personaje con menos vida 11
Figura 4.7: Menú principal 11
Figura 4.8: Menú de selección de mapa 12
Figura 4.9: Menú de pausa 12
Figura 4.10: Botones para indicar acciones en el juego 12
Figura 4.11: Número instanciado por haber realizado daño al enemigo 13
Figura 4.12: Casillas de color azul, representando el área de movimiento 13
Figura 4.13: Casillas de color morado, representando el área de peligro 14
Figura 4.14: Fichero de datos del mapa 14
Figura 4.15: Mapa dentro del juego usando los datos 14
Figura 4.16: Animación de ataque del enemigo. 15
Figura 4.17: Animator con las animaciones de ataque del enemigo 16

Capítulo 1 Introducción

1.1 Contextualización

Los videojuegos forman parte de nuestra cultura, estos mezclan programación, arte, música, etc. y son una de las principales bases de la cultura digital. Los videojuegos pueden llegar a unir muchos de los aspectos que nos aportan el cine, la música o la literatura y su popularidad no ha hecho más que crecer en los últimos años.

A su vez también son parte de una industria, aquí en España su balance no para de crecer por la creación de empleo y la economía que esta genera.

El fin para este proyecto personal, es el de adquirir conocimientos y aprender algunas de las bases del desarrollo de videojuegos.

1.2 Método seguido

Se planteó una serie de tareas con objetivos necesarios para ir desarrollando el proyecto, cada una de estas tareas fueron marcadas con prioridades dependiendo de las metas necesarias para terminar el prototipo.

Durante el periodo de desarrollo de este proyecto se marcaron varios objetivos semanales siguiendo esta lista de tareas y añadiendo otras nuevas dependiendo de varias ideas que fueron surgiendo, también en varias ocasiones para arreglar los errores que se aparecían.

Capítulo 2 Objetivos

Para poder conseguir un prototipo funcional para este videojuego, planteamos una serie de objetivos que se describirán a continuación, aparte de los ya planteados, se han agregado otros nuevos según avanzaba el proyecto.

2.1 Objetivos principales

Estos son los objetivos planteados inicialmente con el fin de completar un prototipo del juego:

1. Un mapa dividido por casillas (llamados Tiles) donde los personajes se puedan mover y con propiedades para que el juego detecte si el personaje se puede mover o si hay un obstáculo / pared que le impida pasar.
2. Personajes aliados que controla el jugador y con diferentes acciones para moverse, atacar y cada uno con propiedades distintas.
3. Algoritmo de Dijkstra [11] para calcular el camino mínimo de un punto a otro y que el personaje recorra el mismo.
4. Sistema por turnos, cuando el jugador termina su turno, se pasa al turno del enemigo y viceversa.
5. Enemigos o rivales a los que el jugador tiene que derrotar para ganar la partida, estos tendrán su propia IA y se moverán por el escenario intentando derrotar a los personajes del jugador.
6. Condición para ganar (en caso de que el jugador derrote a todos los enemigos) o para perder la partida (en caso de que los enemigos venzan a todos los personajes aliados).
7. Una interfaz sencilla que permite al jugador realizar las diferentes acciones y visualizar las propiedades de los enemigos y los aliados. (Por ejemplo, la vida del enemigo).
8. La posibilidad de leer un fichero que cargue los datos del mapa automáticamente y así poder generar diferentes mapas del juego.
9. Menú de inicio para empezar el juego y la selección de 3 mapas del juego.

2.2 Objetivos extra agregados

Además, se terminaron incorporando otros contenidos fuera de los que se habían planteado para el proyecto:

- Otro tipo de enemigo con una nueva Inteligencia Artificial (Intenta priorizar a las unidades con menos vida en vez de al más próximo).
- Un nuevo tipo de terreno que cura a los personajes al final del turno.
- Una cámara que sigue en cada momento a los enemigos que están actuando y la posibilidad de alejar la cámara para ver mejor el mapa.
- Sistema de niveles y experiencia para que los personajes mejoren sus estadísticas.
- Algunas animaciones para los personajes.

Estas últimas características han sido añadidas con la intención de conseguir un mejor acabado del juego y también para probar el funcionamiento de alguna mecánica jugable nueva.

También ha sido importante en mitad del desarrollo ir arreglando algunos errores que fueron surgiendo, como por ejemplo un mal funcionamiento de la IA, problemas con la interfaz, etc.

Capítulo 3 Estado del arte

En este capítulo hablaremos sobre la historia de los videojuegos centrándonos concretamente en los RPG y los RPG Tácticos o SRPG.

3.1 Historia de los RPG

Las siglas “RPG” hacen referencia a la expresión inglesa *role playing game* (“juego de rol”) [7], el cual trata de un juego que lleva al usuario a asumir el papel de uno o varios personajes. Para que exista una progresión en los personajes, este suele tener una serie de estadísticas (Ataque, Defensa, Vida, etc.) que le permite mejorar según avanza el juego.

El género surgió a mediados de los años 1970, con una inspiración en juegos de rol de mesa como “Dungeons & Dragons” aparte de otros videojuegos de aventura, estrategia o novelas de fantasía.

Algunos de los primeros éxitos fueron títulos como “Dragon Quest” [1] (Figura 3.1), “Final Fantasy” [2] (Figura 3.2) o “Megami Tensei”. A partir de aquí se dividió en dos estilos, juegos de rol japoneses (JRPG) y juegos de rol occidentales (WRPG).

Más adelante, en la década de 1990, se popularizaron los juegos multijugador como “Secret of Mana” y “Diablo” que fueron videojuegos de rol de acción.

Entre otras ramificaciones, se encuentran los juegos de rol táctico o TRPG.

Figura 3.1: Dragon Quest (1986)

Figura 3.2: Final Fantasy (1987)

3.2 Historia de los RPG Tácticos o TRPG

Los RPG Tácticos o TRPG [8] toman los mismos elementos que los juegos de rol y añade mecánicas de estrategia, donde el jugador controla una serie de personajes en un escenario estructurado por casillas.

Los primeros juegos de rol tácticos que surgieron fueron The Dragon and Princess [3] (1982) (Figura 3.3) y Ultima III: Exodus (1983).

Sin embargo, el género no se popularizó hasta la salida de “Fire Emblem” (Figura 3.4) el cual sirvió de plantilla a futuros juegos de rol de guerra tácticos, publicado por Nintendo y desarrollado por Intelligent Systems, introdujeron en el género el hecho de que los personajes no fueran simples peones, sino personajes únicos con características propias.

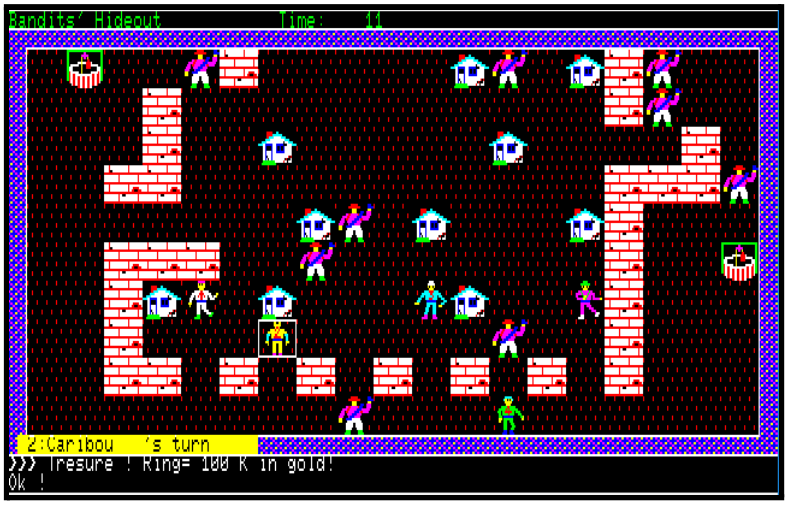

Figura 3.3: The Dragon and Princess (1982)

Figura 3.4: Fire Emblem Shadow Dragon & the Blade of Light (1990)

3.3 Inspiraciones del proyecto

Algunos de los juegos de este subgénero en los que está basado el proyecto son:

Final Fantasy Tactics: Un Spin Off de la saga “Final Fantasy”, es un RPG japonés pero con elementos estratégicos. Aquí gestionamos a un grupo de héroes a lo largo de varios mapas con vista isométrica. Final Fantasy Tactics [4] ha sido desarrollado por Square Co., Ltd. (Figura 3.5)

Figura 3.5: Final Fantasy Tactics

Fire Emblem: Fire Emblem [5] nació directamente como un RPG táctico, incluyendo también la posibilidad de gestionar a nuestros héroes para el progreso de la historia. Esta vez el mapa usa una “vista de pájaro” similar al que se usa en este proyecto. Fire Emblem ha sido desarrollado por Intelligent Systems. (Figura 3.6)

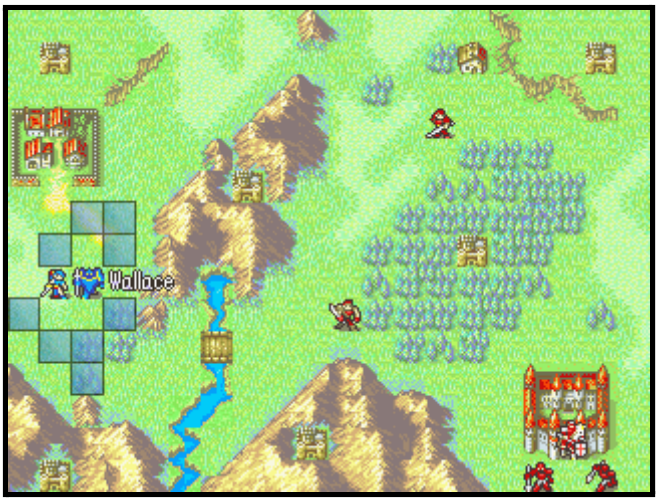

Figura 3.6: Fire Emblem

Advance Wars: De los mismos creadores de Fire Emblem, pero esta vez con una temática militar, aparece Advance Wars [6]. A diferencia del anterior, este no tiene personajes propios y definidos, sino unidades más genéricas como soldados, tanques, etc. Advance Wars ha sido desarrollado por Intelligent Systems. (Figura 3.7)

Figura 3.7: Advance Wars

Capítulo 4 Fases y desarrollo del proyecto

En este capítulo hablaremos no solo del funcionamiento de cada uno de los objetivos descritos anteriormente, sino también de cómo han sido desarrollados. Las fases del desarrollo se han dividido por objetivos, por lo que se describirán cada uno de los elementos de los que se compone el videojuego.

4.1 Mapa del juego

La generación del mapa del juego consta de 3 secciones, la generación de datos del mapa, la generación de un grafo que facilite la búsqueda de caminos con Dijkstra y la generación de una parte visual con la que el usuario podrá interactuar:

1. **Generación de datos del mapa:** En esta sección simplemente generamos una matriz con números que indiquen el tipo de casilla que le corresponde a una determinada coordenada. En este caso, el '0' representa casillas por las que los personajes se pueden mover y el '1' casillas por las que no se pueden mover. (Figura 4.1: En rojo se representan las casillas por las que no se pueden pasar y en verde las casillas por las que se puede pasar)

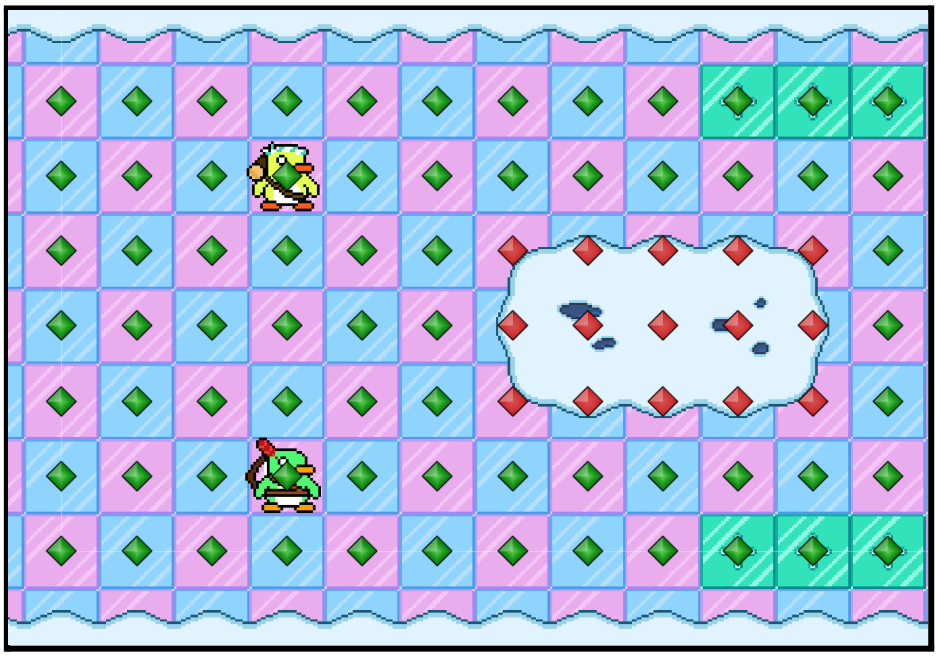

Figura 4.1: Mapa de juego

2. **Generación de un grafo para la búsqueda de caminos:** Para poder aplicar el algoritmo de Dijkstra necesitamos un grafo en el que todas las casillas del mapa están interconectadas. Conectaremos por cada casilla los vecinos que estén encima, debajo y a ambos lados, nuestro mapa se compone de casillas cuadriculadas por lo que tendremos 4 vecinos por cada casilla salvo que nos encontremos en un lateral o en una esquina.

3. **Generación de un mapa visual e interactivo:** Para que el usuario pueda interactuar con el mapa para mover a sus personajes o ver la información del enemigo, necesitamos instanciar un objeto especial llamado "Clickeable Tile" por cada casilla que conforme el mapa. Cada vez que el usuario haga clic sobre una de estas casillas, puede ocurrir una las siguientes acciones:
    a. **Hacer clic sobre una unidad:** Mostrará la información de la unidad y su rango de movimiento.
    b. **Hacer clic sobre un enemigo:** Mostrará la información del enemigo y su rango de peligro.
    c. **Hacer clic en modo de ataque:** Si el usuario ha decidido que quiere realizar un ataque y clica sobre un enemigo, se producirá el cálculo del daño que le hace el personaje seleccionado al enemigo.
    d. **Hacer clic en cualquier otra zona del mapa:** Si esa zona está dentro del rango de movimiento de la unidad, se generará un camino usando el algoritmo de dijkstra hasta la casilla que se seleccionó.

Estos Objetos especiales se generarán por todo el mapa fijándose en la posición en la que están dispuestos en el mapa de datos usando las coordenadas correspondientes.

4.2 Movimiento con el algoritmo de Dijkstra

Para poder generar un camino mínimo que los personajes del juego puedan recorrer, usaremos el algoritmo de Dijkstra, el cual intentará buscar el mejor camino posible con el menor coste posible. [15]

Existen dos tipos de casillas, una por la que los personajes pueden caminar con coste 1 y otro por donde los personajes no pueden pasar con coste 20, de esta forma y debido a que además queda restringido generar un camino mínimo en una casilla por la que no se pueda pasar, el camino mínimo siempre evitará pasar por una de estas. (Algoritmo 8.1)

Aparte, se ha interferido en el algoritmo para que detecte que cuando hay un aliado o un enemigo en la mitad del camino, este se cuente como una casilla de coste 20, por la que no se puede pasar y evitar uno de los errores que encontré donde dos personajes se podían encontrar en la misma casilla.

Este algoritmo se usará tanto para los personajes aliados en los que se activará cuando el usuario clique en una casilla dentro del rango de movimiento del personaje como en los enemigos en donde una IA indicará la coordenada final a la que este se quiere mover y se generará el camino correspondiente. (Figura 4.2: Camino generado por el algoritmo de Dijkstra desde el punto en el que se encuentra el personaje seleccionado al punto donde el jugador quiere moverse)

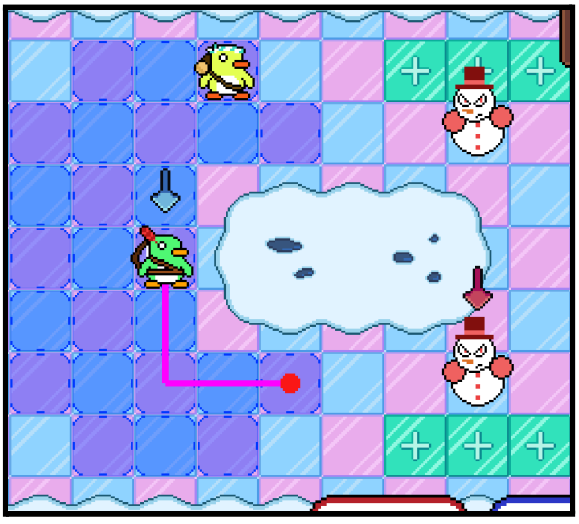

Figura 4.2: Camino generado por el algoritmo de Dijkstra.

4.3 Estadísticas de los personajes aliados.

Por su parte de RPG, el juego necesita que los personajes tengan una serie de estadísticas que midan el poder de estos y para poder progresar en el juego también se han añadido niveles y experiencia, un concepto muy usado en los videojuegos que ayudan a mejorar a los personajes según se avanza. (Figura 4.3: Estadísticas del personaje indicando su ataque, vida, defensa, etc.)

En este caso, se han usado estadísticas como las siguientes:

- **Ataque:** Necesario para calcular el daño cuando se va a realizar un ataque.
- **Defensa:** Se utiliza para contrarrestar parte del daño producido por un ataque.
- **Vida:** Se divide en vida máxima y vida actual, de modo que la vida actual del personaje no pueda superar a la vida máxima. Cuando a un personaje se le acabe la vida, este es derrotado.
- **Suerte:** Lo usamos como una probabilidad en un intervalo de 0 a 1, de modo que aleatoriamente y según el porcentaje de suerte que tenga el personaje, podemos llegar a realizar un daño más fuerte de lo normal (el daño normal multiplicado por 1.5).
- **Variabilidad:** Esta estadística está oculta al usuario, para que el juego no pueda llegar a ser muy predecible, se emplea esta variabilidad para que el daño cambie aleatoriamente entre un 20% menos o un 20% más al calculado inicialmente. (Si el daño original es 10, este puede variar entre 8 y 12).
- **Número de pasos:** Indica el número máximo de pasos por el que puede moverse un personaje.
- **Rango de ataque:** Indica el rango máximo al que un personaje puede atacar desde el punto en el que se encuentre.

Para el cálculo del daño utilizamos una fórmula sencilla (Daño = 4*ataque Aliado - 2*defensa Rival) que es pasada posteriormente por otros factores como la variabilidad o la suerte.

Otro factor importante de los personajes son los niveles y su experiencia, cada vez que un personaje suba de nivel, este mejorará sus estadísticas. Todos los personajes tendrán unas estadísticas base que serán con las que empiezan al nivel 1 e irán mejorando dependiendo de sus estadísticas de crecimiento, estas se usan para saber que ataque debe tener según el nivel actual del personaje.

La experiencia se usará para saber cuánto le queda a la unidad para subir al próximo nivel y este número irá creciendo a medida que el nivel sea mayor. Se conseguirá experiencia al derrotar a un enemigo y cada enemigo dará más o menos experiencia según su nivel.

Figura 4.3: Estadísticas de uno de los personajes.

4.4 Sistema por turnos

Otro elemento importante del juego es el sistema de turnos, cuando el jugador termina de realizar las acciones de sus personajes, este pasa el turno a la IA enemiga que a su vez actuará por su cuenta y le devolverá el turno al jugador hasta que llegue el punto en el que uno de los dos gane o pierda.

Para ello, usaremos un mediador encargado de pasar el turno de forma correcta, el *GameManager*. Este se encargará, entre otras muchas cosas, de quitarle el control al jugador para que sea la Inteligencia Artificial la que actúe en cada momento, haciendo uso de las corrutinas de Unity [9] que nos permite establecer el tiempo necesario para que cada enemigo actúe sin problemas.

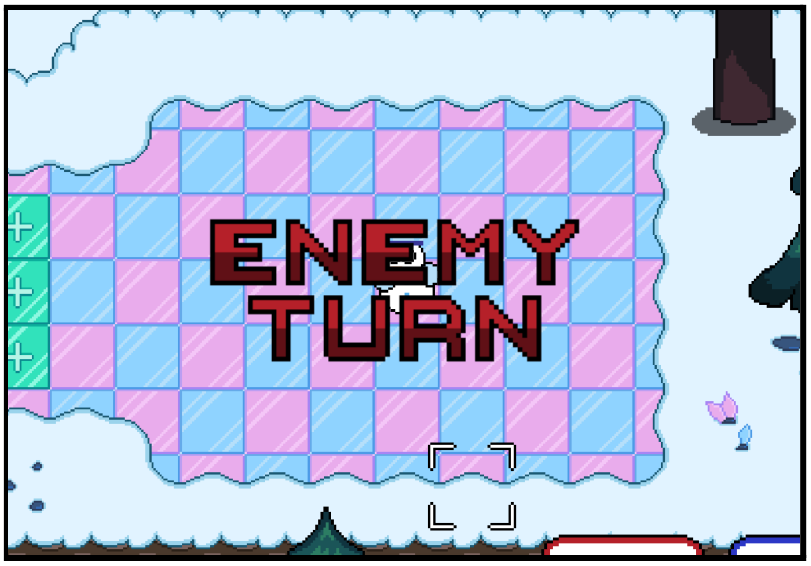

Figura 4.4 Cambio al turno del enemigo.

4.5 Enemigos con Inteligencia Artificial

Para poder conseguir que los enemigos sepan los objetivos a los que tienen que atacar en su turno, se ha utilizado una Inteligencia Artificial sencilla.

Gracias al uso de la herencia [17], hemos podido realizar dos tipos de comportamientos para esta IA, el primero avanza al personaje aliado más cercano desde su posición y el segundo hace lo mismo, con la diferencia de que intentará priorizar a los aliados con menos vida dentro de su rango de peligro (rango máximo que este alcanza para poder atacar).

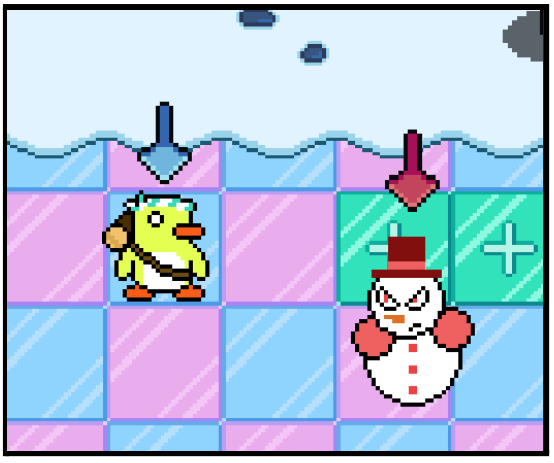

Figura 4.5 Enemigo acercándose al personaje más cercano. (Representado con color rojo)

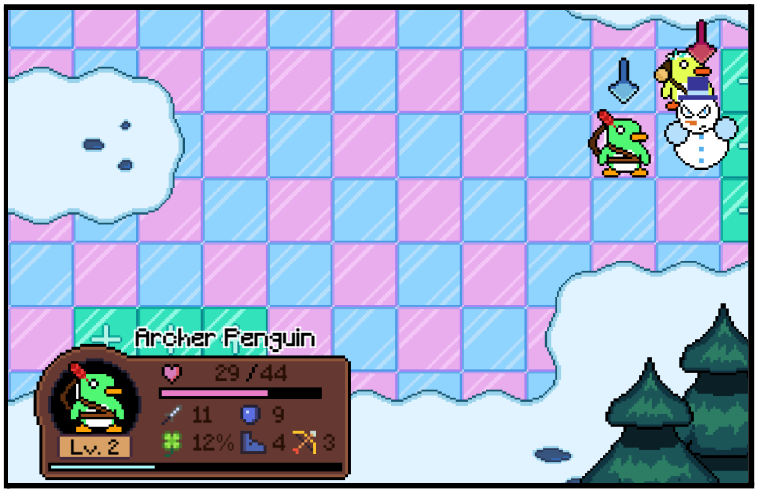

Figura 4.6 Enemigo acercándose al personaje con menos vida. (Representado con color azul)

La idea en ambos casos es la de encontrar un objetivo al cual el enemigo ha de moverse, siguiendo una serie de reglas y aprovechando el algoritmo de dijkstra ya utilizado para que encuentre un camino mínimo hasta ese punto.

4.6 Interfaz del jugador

Se ha escogido una interfaz sencilla con el uso de Canvas [10] en Unity para poder iniciar el juego o salir de la aplicación y seleccionar entre 3 mapas diferentes. (En el menú principal solo funcionan los botones de Start y Exit).

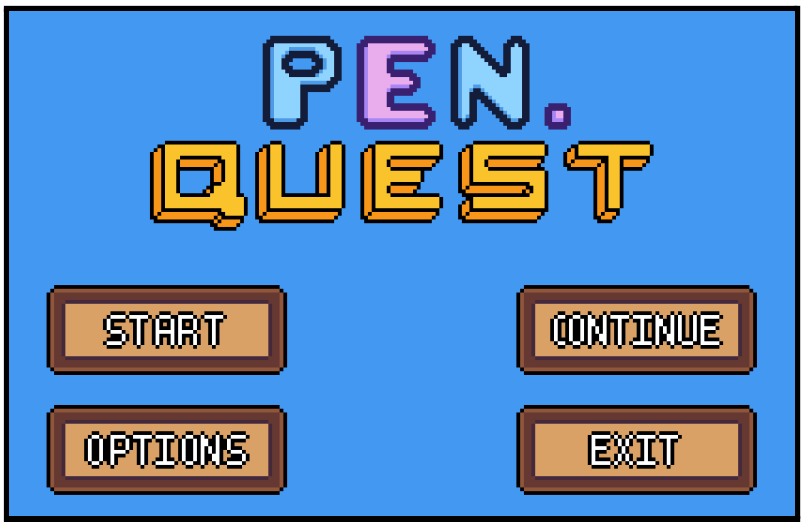

Figura 4.7 Menú principal

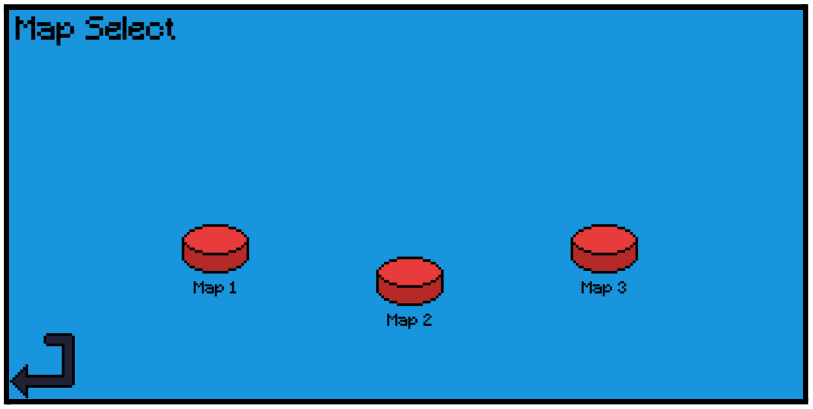

Figura 4.8 Menú de selección de mapa (3 mapas a elegir)

Una vez dentro del juego, tendremos un menú de pausa que nos permitirá parar el juego en cualquier momento, una vez dentro, podremos reanudar la partida o salir de esta.

Figura 4.9 Menú de pausa

Para dar algunas órdenes básicas a los personajes que el jugador controla, tendremos los botones para atacar, moverse y otro que nos permitirá acabar el turno.

Al usar cada uno de estos botones se activan varias flags para evitar por ejemplo que el jugador se mueva en el turno del enemigo o que 2 personajes se puedan mover a la vez. Algo necesario para que no sucedan errores en el juego.

Figura 4.10 Botones para indicar acciones en el juego.

Existen también algunos elementos de la interfaz, que aportan información al jugador, para esta función podemos encontrar un indicador que sigue al ratón del jugador cuando pasa sobre una casilla del mapa con el objetivo de que el usuario sepa las casillas que está seleccionando.

También se indica con una flecha a los aliados y a los enemigos que el usuario tiene seleccionado en ese momento, al tener a una unidad seleccionada aparece una tarjeta con todas las estadísticas de los personajes como el ataque, una barra de vida, niveles, etc. (Figura 4.3)

Por último cuando cualquiera de los personajes realice algún daño, aparecerá un número en pantalla indicando el daño realizado [16], este cambiará visualmente en caso de que el daño sea crítico.

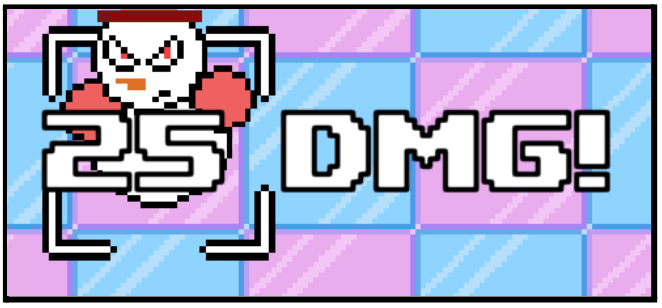

Figura 4.11 Número instanciado por haber realizado daño al enemigo.

4.7 Cálculo de áreas de rangos de ataque o movimiento

Con la intención de limitar el rango de ataque o de movimiento de los personajes en el escenario, se ha realizado un algoritmo que guarda las coordenadas de los puntos que abarca una determinada área como pueden ser el rango de ataque o de movimiento.

Esta área nos será de utilidad no solo para limitar el rango que el jugador puede seleccionar dentro del mapa, sino también para marcarlo visualmente y que el usuario tenga esta información. De forma visual existe el área de movimiento representado con casillas de color azul (Figura 4.12), el área de ataque representado con el color rojo y el rango de peligro del enemigo, que abarca la suma de su rango de ataque y su rango de movimiento, alertando al jugador en que zonas del mapa el enemigo puede atacarle (Figura 4.13).

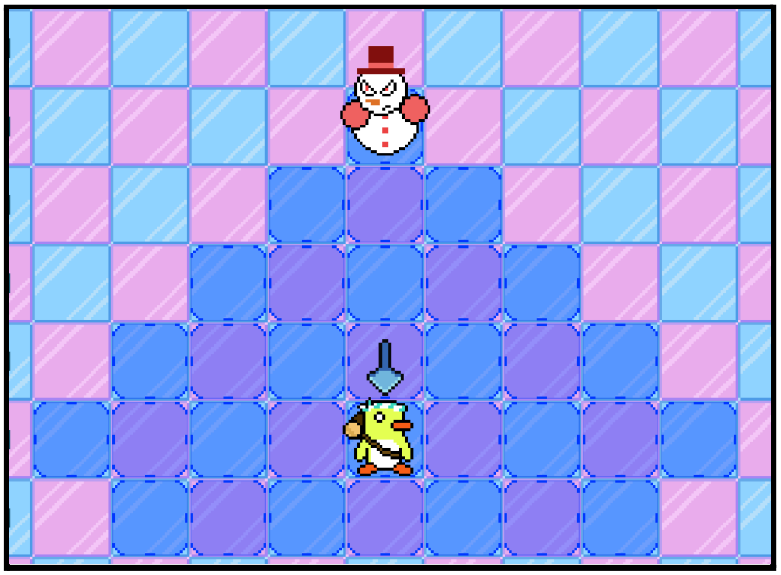

Figura 4.12 Casillas de color azul, representando el área de movimiento.

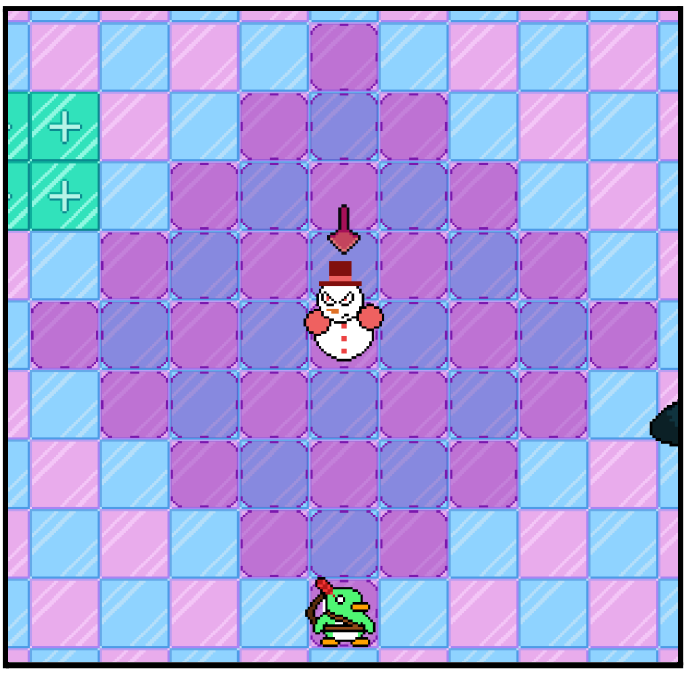

Figura 4.13 Casillas de color morado, representando el área de peligro del enemigo.

4.8 Lectura de un fichero para cargar los datos del mapa

Con el fin de poder hacer distintos mapas de juego, se ha decidido cargar un fichero .txt con los datos necesarios del mapa, en ellos se representan con un ‘0’ las casillas a las que los personajes pueden pasar, con un ‘1’ por las que no pueden pasar y una de las últimas mecánicas que se añadieron al proyecto que son las casillas de terreno de curación, representados por un ‘2’ y por las cuales los personajes se curarán un 20% de su vida máxima al final del turno. (Figura 4.14, 4.15)

Estos ficheros se pueden cargar después de la *build* gracias a la carpeta especial en Unity llamada “StreamingAssets”. [14]

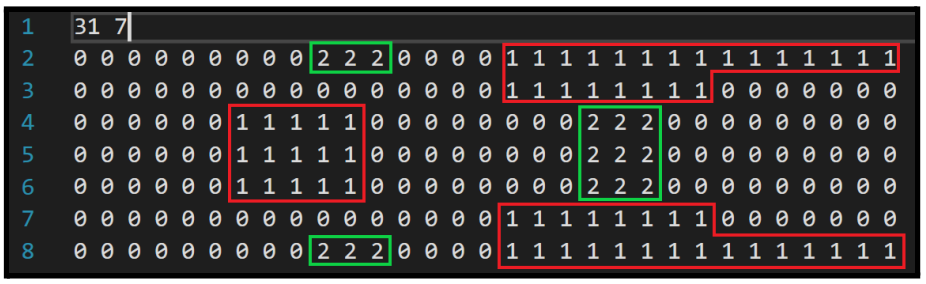

Figura 4.14 Fichero de datos del mapa. (En rojo se representan las casillas donde el usuario no puede pasar y en verde el terreno de curación del personaje)

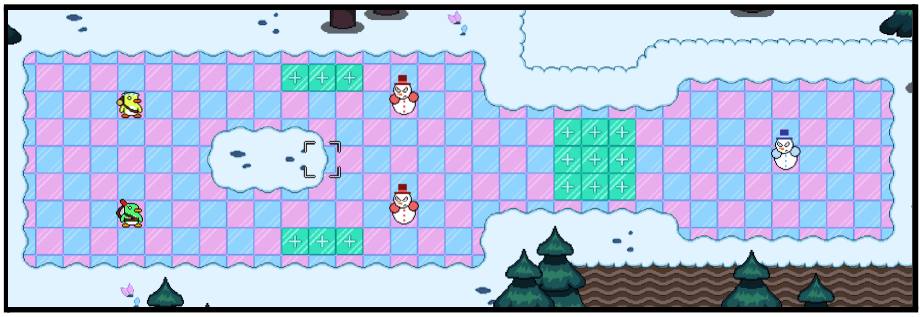

Figura 4.15 Mapa dentro del juego usando los datos de la figura 4.14.

4.9 Movimiento de la cámara.

Para poder hacer un seguimiento de las acciones de la IA enemiga y para que el jugador pueda observar todo el escenario. Se ha empleado una cámara que pasará el control al jugador cuando sea su turno, de forma que este pueda moverse por el escenario usando las teclas A, S, D y W. Cuando sea el turno del enemigo, la cámara dejará de estar bajo el control del usuario y se moverá automáticamente hacia cualquier acción que ocurra en el juego, como por ejemplo en el momento en el que el enemigo vaya a actuar o cuando el terreno de curación, cure a un personaje.

Para el caso de las acciones que ocurren en el turno del enemigo, se usa un Gameobject vacío llamado “End Marker” que se mueve al punto en el que va a ocurrir una determinada acción y luego la cámara seguirá al End Marker. La cámara seguirá constantemente a dicho objeto dentro de un Update de Unity hasta que termine el turno del enemigo y se pase el control de nuevo al usuario.

Por último, el usuario podrá reducir o aumentar la cámara del juego [12] para poder ver mejor el escenario usando la rueda del ratón, el juego detectará 2 puntos diferentes, uno para la cámara alejada y otro para una cámara más cercana.

4.10 Sensación de juego.

La sensación de juego o *Game feel* trata sobre la percepción que recibe el jugador cuando interactúa en un videojuego, para este proyecto hemos intentado abarcar dos aspectos importantes para mejorar este aspecto.

El primero consiste en dar como respuesta a cada acción del usuario un sonido, gracias a esto cada vez que el jugador pulse sobre un botón, seleccione un personaje o realice un ataque, se produce un sonido acorde.

El segundo es otorgar algunas animaciones a los personajes del juego, en este caso solo hemos añadido dos animaciones a los enemigos, uno para cuando este recibe daño y otro para cuando este ataca a nuestras unidades. (Figura 4.16)

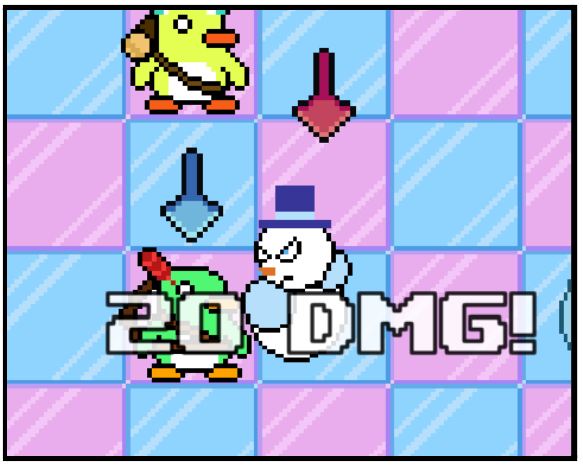

Figura 4.16 Animación de ataque del enemigo al realizar daño a las unidades aliadas.

Las animaciones se han realizado gracias al *animator* [18] una herramienta de Unity específicamente diseñada para configurarlas, desde aquí he podido seleccionar que *sprites* aparecen en cada sección del tiempo y un movimiento que da la sensación del ataque del enemigo. Aparte con esta herramienta, también podemos establecer un punto de la animación en la que se ejecuta una determinada función, en este caso se usa para escuchar el sonido de ataque en el momento adecuado. (Figura 4.17)

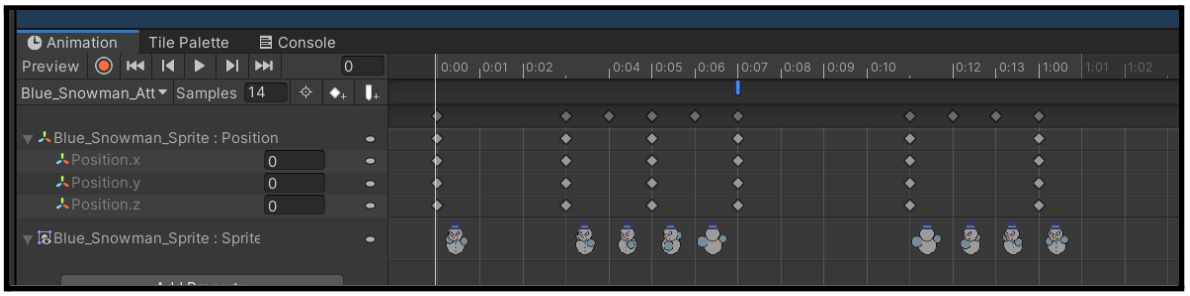

Figura 4.17 *Animator* con las animaciones de ataque del enemigo, en el punto azul se ejecuta el sonido de ataque del enemigo.

4.11 Música adaptada a videojuegos.

En colaboración con el Conservatorio Superior de Música de Canarias (CSMC) hemos podido añadir una música de fondo para el juego haciendo uso de FMOD [19] una herramienta que nos permite elaborar y editar música en tiempo real.

La idea principal para implementar la música en este proyecto ha sido la de una música que se adapta según lo que ocurra en el juego, el cambio se produce en función de si es el turno del enemigo o el turno del jugador.

También existe la posibilidad de aumentar la intensidad de la música la cual dará más énfasis en un momento del juego en el que exista más tensión como por ejemplo cuando queden pocos enemigos.

Capítulo 5 Conclusiones y líneas futuras

5.1 Líneas futuras

El trabajo total realizado en este proyecto es el de un prototipo funcional de un RPG táctico con los elementos más básicos de este género, durante el desarrollo del mismo han surgido muchas ideas sobre las características que se necesita para completarlo, incluyendo mecánicas jugables únicas que lo diferencien de los demás.

Entre otras muchas de las ideas que se podrían añadir, se destacan las siguientes:

- Un inventario con objetos utilizables que por ejemplo curen a los personajes o consigan otros efectos durante el combate.
- Un equipamiento para mejorar las estadísticas de los personajes.
- Un lobby o lugar principal al que los personajes acuden para comprar objetos, equipamiento, entre otras actividades.
- Una historia sencilla y desenfadada donde los personajes tengan un objetivo a seguir.
- Mejoras en la Inteligencia Artificial.
- Y más contenido con nuevos mapas, personajes y mecánicas únicas.

5.2 Conclusiones

Como conclusión vamos a hablar de todo lo logrado en este TFG desde los contenidos totales del proyecto hasta la experiencia y conceptos aprendidos.

En general, se ha logrado desarrollar un prototipo de videojuego TRPG con muchas de las características que estos suelen traer, como un mapa de juego dividido por casillas, personajes con características únicas, enemigos con una IA propia y con dos comportamientos diferentes, el uso del algoritmo de Dijkstra para encontrar un camino mínimo que evite los obstáculos del mapa y una interfaz que aporte suficiente información al jugador intentando que sea lo más intuitiva posible.

El proyecto ha servido para aprender a usar Unity junto con muchas de las características que esta nos ofrece para el desarrollo de videojuegos, entre otros, se destacan por ejemplo el animator que nos sirve para animar los objetos haciendo uso de *sprites*, el uso de *Singletons* [13] para acceder fácilmente a objetos que solo aparecen una vez como es el caso del *GameManager* y el uso de corrutinas con el fin de controlar a los personajes de la IA enemiga correctamente.

Para acabar quería comentar que he disfrutado mucho realizando este proyecto, ya que mi pasión siempre ha sido el desarrollo de videojuegos, he aprendido mucho y he conseguido bastantes más objetivos de los que me había propuesto a desarrollar. Me ha gustado mucho la experiencia y me gustaría continuar formándome y trabajar algún día en la industria de los videojuegos.

Capítulo 6 Summary and Conclusions

6.1 Summary

The total work done in this project is that of a working prototype of a tactical RPG with the most basic elements of this genre, during the development of this project many ideas have emerged about the features needed to complete it, including unique gameplay mechanics that differentiate it from others.

Among many other ideas that I could add, I would like to point out the following:

- An inventory with usable items that for example heal characters or get other effects during combat.
- An equipment to improve the stats of the characters.
- A lobby or main place where the characters go to buy items, equipment and other activities.
- A simple and lighthearted story where the characters have a goal to follow.
- Improvements in Artificial Intelligence.
- And more content with new maps, characters and unique mechanics.

6.2 Conclusions

In conclusion we will talk about everything achieved in this TFG from the total contents of the project to the experience and concepts learned.

In general, we have managed to develop a prototype TRPG video game with many of the features that these usually bring, such as a game map divided by squares, characters with unique characteristics, enemies with their own AI and with two different behaviors, the use of Dijkstra's algorithm to find a minimum path to avoid the obstacles of the map and an interface that provides enough information to the player trying to make it as intuitive as possible.

The project has served to learn to use Unity along with many of the features that this offers us for the development of video games, among others, stand out for example the animator that serves to animate objects using sprites, the use of Singletons to easily access objects that only appear once as is the case of the GameManager and the use of corrutinas in order to control the enemy AI characters correctly.

To finish I wanted to comment that I have enjoyed doing this project, since my passion has always been the development of video games, I have learned a lot and I have achieved many more goals than I had proposed to develop. I really liked the experience and I would like to continue training and work someday in the video game industry.

Capítulo 7 Presupuesto

7.1 Presupuesto

Un presupuesto aproximado para este proyecto suponiendo que el coste de cada hora invertida sea de 20€ /hora.

Para esto se ha dividido el trabajo invertido en:

- **Programación:** Tiempo invertido en escribir el código de los Scripts en C#
- **Arte:** Tiempo empleado en crear los *sprites* del juego con un estilo pixelart.
- **Diseño:** Tiempo gastado en idear las mecánicas jugables y todos los elementos necesarios para conseguir realizar un prototipo.
- **Producción:** Tiempo que se ha necesitado para pensar en cómo estructurar y desarrollar el prototipo.

Tipos	Tiempo / Dinero invertido
Programación	256 horas => 5120€
Arte	64 horas => 1280€
Diseño	32 horas => 640€
Producción	32 => 640€

Tabla 7.1**:** Resumen de tipos y su relación con el tiempo / dinero invertido.

Total de horas: 384 horas.

Total de presupuesto: 7680 €

Capítulo 8 Apéndice: Algoritmos

8.1 Algoritmo de Dijkstra

Fichero: TileMap.cs

Descripción: Algoritmo de Dijkstra adaptado para la búsqueda del camino mínimo en el prototipo, se evita que por ejemplo al encontrar otro personaje en un punto, dicho punto cueste lo mismo que una casilla por la que no se puede pasar, de forma que se evita atravesar a los personajes.

```
GENERATE PATH (Grafo G, puntoX X, puntoY Y)

   si comprobarUnidadNoPuedePasar(X, Y)
      return null

   array distancia[Nodo] = new
   array previo [Nodo] = new
   array noVisitado [Nodo] = new
   Nodo nodoFuente = G[personajePuntoX, personajePuntoY]
   Nodo objetivo = G[X, Y]
   distancia[nodoFuente] = 0
   previo[nodoFuente] null;

   // Inicializar todo para que tenga distancia INFINITA
   para v ∈ G
      si v != nodoFuente
         distancia[v] = INFINITO
         previo[v] = null
      añadir(noVisitado, v)

   mientras noVisitado.Size > 0
      Nodo u = null
      para posibleU ∈ noVisitado
         si u == null || distancia[posibleU] < distancia[u]
            u = posibleU
      si u == objetivo
         break
      Eliminar(noVisitado, u)
```

```
    para v ∈ u.vecinos
        coste = 0
        si existePersonaje(v.x, v.y)
            coste = distancia[u] + 20
        else
            coste = distancia[u] + CosteParaEntrar(v.x, v.y)
        si coste < distancia[v]
            distancia[v] = coste
            previo[v] = u

// Si se llega a esta línea, ya encontramos la ruta más corta
// o no existe una ruta a nuestro objetivo.
si previo[objetivo] == null
    return null

caminoActual = new
Nodo actual = objetivo
// Ir avanzando por el array "previo"
// e irlo añadiendo a nuestro camino.
mientras actual != null
    Añadir(caminoActual, actual)
    actual = previo[actual]

// El camino está invertido, hay que invertirlo de nuevo
Revertir(caminoActual)
return caminoActual
```

Bibliografía

- [1] Chunsoft. Dragon Quest. Wikipedia. https://es.wikipedia.org/wiki/Dragon_Quest_(videojuego)
- [2] Squaresoft. Final Fantasy. Wikipedia. https://es.wikipedia.org/wiki/Final_Fantasy_(videojuego)
- [3] KOEI. The Dragon and Princess. Fandom Wiki. https://j-rpgs.fandom.com/wiki/The_Dragon_and_Princess
- [4] Square Co., Ltd. Final Fantasy Tactics. Wikipedia. https://es.wikipedia.org/wiki/Final_Fantasy_Tactics
- [5] Intelligent Systems. Fire Emblem. Wikipedia. https://es.wikipedia.org/wiki/Fire_Emblem
- [6] Intelligent Systems. Advance Wars. Wikipedia. https://es.wikipedia.org/wiki/Advance_Wars
- [7] Videojuego de rol. Wikipedia. https://es.wikipedia.org/wiki/Videojuego_de_rol
- [8] Juego de rol táctico. Hmong. https://hmong.es/wiki/Tactical_role-playing
- [9] Unity Technologies. Coroutines. Unity Manual. https://docs.unity3d.com/es/2018.4/Manual/Coroutines.html
- [10] Unity Technologies. Canvas. Unity Manual. https://docs.unity3d.com/es/2019.4/Manual/UICanvas.html
- [11] Algoritmo de Dijkstra. Wikipedia. https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra
- [12] Zoom a camera in Unity. Gamedevbeginner. https://gamedevbeginner.com/how-to-zoom-a-camera-in-unity-3-methods-with-examples/
- [13] Singletons. Gamedevbeginner. https://gamedevbeginner.com/singletons-in-unity-the-right-way/
- [14] Unity Technologies. Persistent data path. Unity Manual. https://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html
- [15] quill18creates. Movimiento por casillas y búsqueda de caminos. Youtube. https://www.youtube.com/watch?v=kYeTW2Zr8NA&list=PLbghT7MmckI55gwJLrDz0UtNfo9oC0K1Q&ab_channel=quill18creates
- [16] Textos de daño. Wintermute Digital. https://wintermutedigital.com/post/damage-text-unity/
- [17] Unity Technologies. Herencia. Unity Learn. https://learn.unity.com/tutorial/herencia#5e419554edbc2a0a62170fb6
- [18] Unity Technologies. Animator. Unity Manual. https://docs.unity3d.com/ScriptReference/Animator.html
- [19] Firelight Technologies Pty Ltd. Música adaptable para juegos. FMOD. https://www.fmod.com/