Universitat Oberta de Catalunya

uoc.edu

Tactics War

José Luis Martín Almoguera
Máster universitario de Desarrollo de aplicaciones para dispositivos móviles
Desarrollo videojuego para plataforma Android

Francesc D´Assís Girald Queralt
Carles Garrigues Olivella

05/06/2020

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Tactics War*
Nombre del autor:	*José Luis Martín Almoguera*
Nombre del consultor/a:	*Francesc D´Assís Girald Queralt*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	06/2020
Titulación::	*Máster universitario de Desarrollo de aplicaciones para dispositivos móviles.*
Área del Trabajo Final:	*Trabajo final de máster*
Idioma del trabajo:	*Español*
Palabras clave	*Videojuego, smartphone, RPG táctico*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El principal objetivo de este trabajo es crear un videojuego pasando por todas las fases del desarrollo, recolección y definición de requisitos, diseño conceptual y DCU, implementación y pruebas, y por último la entrega del producto final.

Se han aplicados conceptos de metodologías ágiles para descomponer el proyecto en distintas tareas y objetivos, debidamente priorizadas, con el fin de realizar una mejor planificación y gestión de riesgo.

Para el desarrollo se ha utilizado el motor de videojuegos Unity con las librerías necesarias para exportar una versión de la aplicación a Android.

Abstract (in English, 250 words or less):

The main objective of this work is to create a video game by going through all the phases of development, gathering and defining requirements, conceptual design and UCD, implementation and testing, and finally the delivery of the final product.

Agile methodology concepts have been applied to decompose the project into different tasks and objectives, duly prioritized, in order to perform better planning and risk management.

For the development, the Unity game engine has been used with the necessary libraries to export a version of the application to Android.

Índice

1.- Introducción 1
1.1.- Contexto y justificación del Trabajo 1
1.2.- Objetivos del Trabajo 2
1.3.- Enfoque y método seguido 4
1.4.- Planificación del Trabajo 6
1.5.- Análisis de riesgos y plan de contingencia 9
1.6.- Breve sumario de productos obtenidos 9
1.7.- Breve descripción de los otros capítulos de la memoria 10
2.- Especificación de requisitos 11
2.1.- Funcionalidades 11
2.2.- Análisis de usuarios 13
2.2.1.- Fichas de personas 14
2.2.2.- Escenarios 15
2.2.2.- Conclusiones 15
2.3.- Benchmarking 16
2.3.1.- Conlusiones 17
2.4.- Diseño de la interfaz gráfica 18
2.4.1.- Árbol de navegación 18
2.4.2.- Moodboard 18
2.5.- Diseño inicial de la arquitectura 19
2.5.1.- Diagramas de secuencia 21
2.6.- Diseño final de la arquitectura 22
2.7.- Normas 24
3.- Implementación 25
3.1.- Breve resumen 25
3.2.- Aspectos mas relevantes 26
3.2.1.- Control del personaje 26
3.2.2.- Comportamiento de los NPC 26
3.2.3.- Escenario 27
3.3.- Pruebas 28
3.3.1.- Pruebas durante el desarrollo en vivo 28
3.3.2.- Logs 29
3.3.3.- Pruebas en móvil 29
3.4.- Ejemplo de código 29
3.5.- Ejemplo de animaciones 30
4.- Conclusiones del desarrollo 32
4.1.- Análisis del cumplimieto de la planificación 32
4.1.1.- Limitaciones y problemas 36
4.1.2.- Métricas 36
4.1.3.- Lecciones aprendidas 37
5.- Glosario 38
6.- Bibliografía 39
7.- Anexos 40
7.1.- Manual de usuario 40
7.1.1.- Introducción 40

7.1.2.- Elementos del escenario 40
7.1.3.- Aliados 41
7.1.4.- Enemigos 41
7.1.5.- Elementos 42
7.1.6.- Movimientos 42
7.1.7.- Objetivos 47
7.2.- Manual de compilación 48
7.3.- Manual de instalación 48

Lista de figuras

Ilustración 1: Zonas y rangos 3
Ilustración 2: Modelo por construcción de prototipos 5
Ilustración 3: Metodología de trabajo 6
Ilustración 4: Diagrama de Gantt 8
Ilustración 5: Árbol de Navegación 18
Ilustración 6: Moodboard 19
Ilustración 7: Diagrama inicial de clases 20
Ilustración 8: Diagrama de secuencia - Iniciar juego 21
Ilustración 9: Diagrama de secuencia - Movimiento 22
Ilustración 10: Diagrama final de clases 23
Ilustración 11: Trabajando con la herramienta Unity 25
Ilustración 12: Matriz escenario 27
Ilustración 13: Variables públicas en Unity 28
Ilustración 14: Variables públicas en Visual Studio 29
Ilustración 15: Ejemplo de código 30
Ilustración 16: Ejemplo árbol de animaciones 31

1.- Introducción

1.1.- Contexto y justificación del Trabajo

La aparición en el mercado de los dispositivos móviles (smarthphones, tablets, etc) nos ofrecen componentes cada vez más sofisticados para poder disfrutar de mejores rendimientos y experiencias audiovisuales, además de satisfacer las necesidades de comunicación y búsqueda de información.

Actualmente los videojuegos para móviles están en una constante evolución vertiginosa y no parece que vaya ha tener fin. Cada vez son más los videojuegos de las principales compañías (Nintendo, Sony, etc) que aparecen en los dispositivos móviles para ampliar su mercado, obtener más beneficios y alcanzar nuevas experiencias de usuarios.

Los dispositivos móviles deben superar una serie obstáculos para acercarse a la acción que ofrece una consola tradicional o un ordenador. El primero de ellos es el uso de batería, la experiencia de juego puede sufrir por miedo a que se acabe la batería, hay que tener en cuenta que los smarthphones se usan cotidianamente para múltiples acciones. Otro aspecto que debe mejorar es la temperatura, ya que el uso intensivo del teléfono puede hacer que se caliente más de lo debido, esto no es sólo un problema sensitivo, cuando un dispositivo opera con temperaturas elevadas el rendimiento de la GPU y la CPU se ve mermado para mantenerse fresco, por ello ya integran sistemas de refrigeración líquida. Por último, otro aspecto muy importante es el tema de los controles, aunque existen juegos exclusivamente diseñados para smartphones la mayoría de títulos son multiplataformas y están pensados para diferentes consolas y ordenadores. Hay distintas soluciones a este problema, como por ejemplo, usar controles táctiles que imiten los mandos físicos (son complicados de adaptarse), creación y optimización de accesorios (tendríamos que llevar los accesorios para poder jugar cómodamente) o fomentar el diseño de juegos exclusivos para dispositivos móviles.

En este último punto es donde surge la idea de aprovechar las características y usabilidad de los smartphones ya que aunque los juegos RPG tácticos vieron su nacimiento en las consolas tradicionales o PC se puede obtener un mejor rendimiento aprovechando la tecnología táctil y eliminado acciones que eran más incómodas para el usuario como manejar el puntero (ya sea con el ratón o con la cruceta).

El juego se basará en una saga clásica de RPG táctico como Fire Emblem [1] de la cuál no paran de salir secuelas, a grandes rasgos, consiste en un escenario donde hay diferentes tipos de zonas: tierra, agua, árboles, ... donde, según las características de los personajes podrán moverse por él o no, cada personaje dispone de habilidades distintas, por ejemplo, ataque a distancia, mayor rango de movimiento, mayor defensa, ... cada jugador (usuario y CPU) podrá realizar las diferentes acciones por turnos, por ejemplo, mover, atacar, usar objeto, ... Gana el usuario que logre derrotar a todos los

personajes, o realice la acción correspondiente de la misión, por ejemplo, encontrar un objeto, llegar a un punto, etc [2]

Mencionar que la realización de un videojuego largo, novedoso y con historias como Fire Emblem está fuera del alcance del proyecto, el objetivo es demostrar las habilidades obtenidas durante todas las fases de diseño e implementación.

1.2.- Objetivos del Trabajo

Los objetivos que se deben cubrir son los siguientes:

- Objetivos funcionales:
    - Crear un videojuego de RPG táctico al estilo Fire Emblem.
        - RF1: Creación de un escenario a modo de cuadrícula donde se colocarán los personajes.
        - RF2: Definición de las zonas de movimiento y ataque de cada tipo de personaje (Ver ilustración 1).
        - RF3: Implementación del control del jugador mediante gestos.
        - RF4: Implementación del modelo de comportamiento de la CPU.
        - RF5: Implementación de las animaciones de los personajes.
        - RF6: Implementación de los diversos escenarios.
        - RF7: Implementación de la música de ambiente.
        - RF8: Implementación de los efectos sonóros.
        - RF9: Implementación de las pantallas iniciales y resultados del juego.

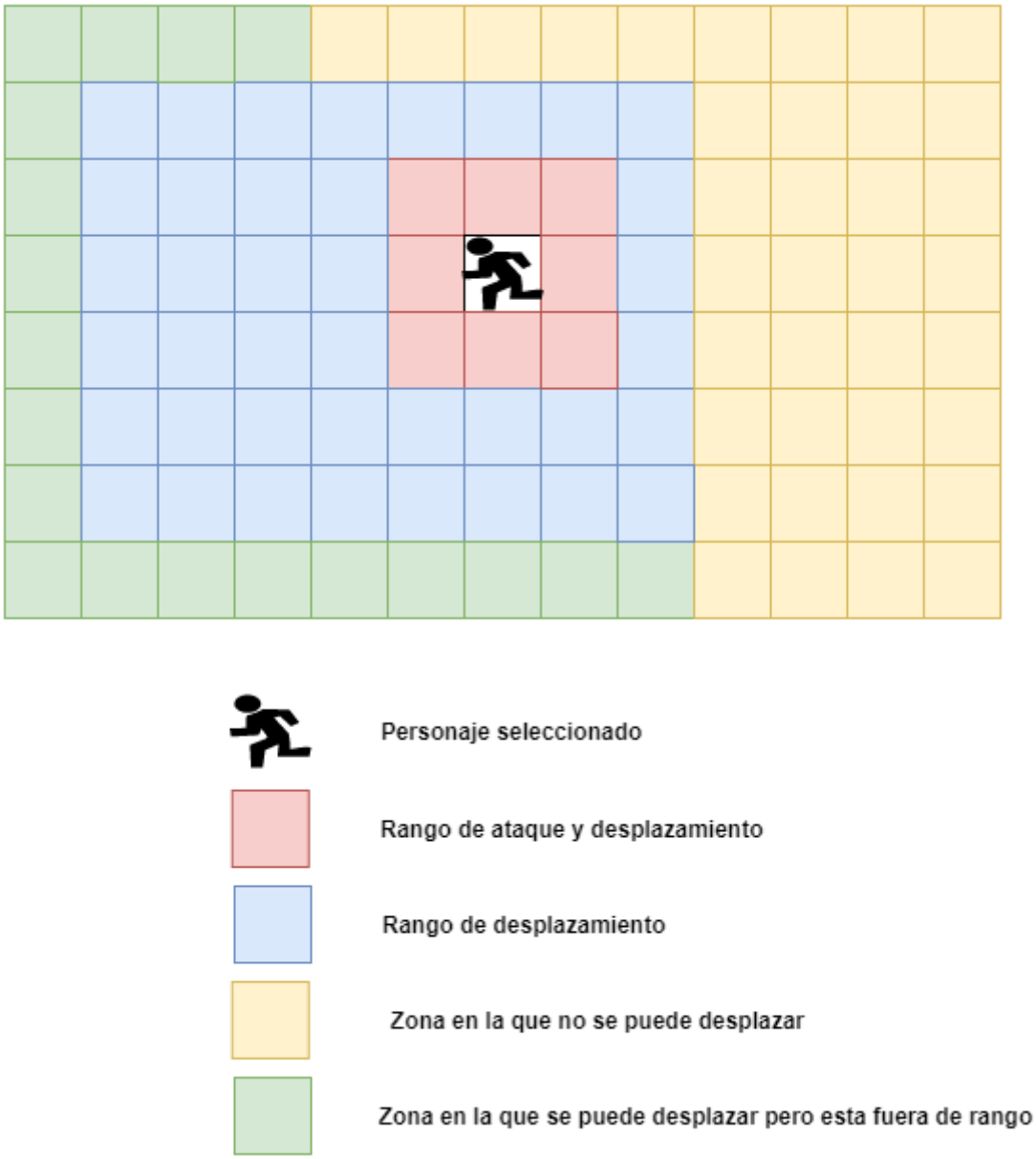


Ilustración 1: Zonas y rangos

- Objetivos no funcionales:
    - RNF1: El videojuego debe ser compatible con el mayor número posible de dispositivos Android de gama media.
    - RNF2: El videojuego debe ser compatible con el mayor número posible de pantallas.
    - RNF3: Crear documento de definición de requisitos de la interfaz de usuario.
    - RNF4: El videojuego se puede jugar sin conexión a Internet.

- RNF5: Definición de requisitos que establezcan las normas del juego.

- RNF6: Obtener un juego mantenible y estable.

1.3.- Enfoque y método seguido

Unos de los principales problemas para desarrollar un videojuego es su elevado coste de desarrollo. Los videojuegos actuales son el resultado de proyectos de gran tamaño y duración lo que conlleva que para obtener rentabilidad es necesario un gran número de ventas.

Desarrollar un videojuego desde cero, es decir, sin un motor de juego ya desarrollado resulta muy costoso y complejo, por ello se utilizará la herramienta Unity [3], la capacidad y tecnología que ofrece este motor se aleja mucho de los motores de juego de pago pero con esta herramienta es más que suficiente para cumplir con los requisitos citados anteriormente, además muchas compañias de videojuegos han sacando al mercado grandes exitos como por ejemplo, Cuphead [4] . Además el videojuego se exportará al sistema operativo Android ya que es más fácil obtener una licencia de desarrollo y aprovechar todas las ventansa que el Play Store nos ofrece, también mencionar que es el sistema operativo que más usuarios tiene [5].

Dotar al videojuego de gráficos potentes donde el movimiento de los personajes, la interacción entre ellos o los sonidos se asemejen lo máximo a la realidad es algo realmente díficil de conseguir y requiere un gran esfuerzo y trabajo, por todo esto se ha determinado que se va a realizar un videojuego usando el motor de Unity y siguiendo, no al pie de la letra, una estrategia de desarrollo por construcción de prototipos.

El modelo de desarrollo por construcción de prototipos se basa en realizar el sistema para satisfacer un subconjunto de los requisitos especificados y en posteriores versiones incrementar el programa con nuevas funcionalidades que satisfagan más requisitos o incluso mejorar las anteriores. De esta forma realizamos un ciclo de prueba y fallo donde podemos ir evaluando las funcionalidades, el diseño o la interacción con el usuario, además de corregirlas de forma directa, permitiendo también la modificación de requisitos durante el desarrollo.

El desarrollo se divide en seis fases, la primera sería Recolección de Requisitos, consiste en definir los objetos generales, requisitos y áreas pendientes de definición o desarrollo. En segundo lugar se encuentra Diseño Rápido, que se encarga de la representación de los aspectos visuales para el usuario. La tercera sería Construcción del prototipo, en esta fase se desarrolla el prototipo para el usuario. La cuarta fase es Evaluación del Prototipo, en ella el usuario evalúa el prototipo seguido de la fase Refinamiento del Prototipo donde el usuario utiliza el prototipo para refinar los requisitos. Por último

tenemos Producto de Ingeniería, en esta fase se realizan las pruebas, se verifica que se han cumplido los objetivos y se vuelve a la primera fase.

Pero no todo son ventajas, este modelo implica la toma de decisiones rápidas que pueden llegar a formar parte del producto final, los prototipos se realizan sin tener en cuenta la calidad o facilidad del desarrollo del prototipo, además requiere una participación activa con clientes y usuarios pero gracias a esto se reduce el riesgo de incertidumbre.

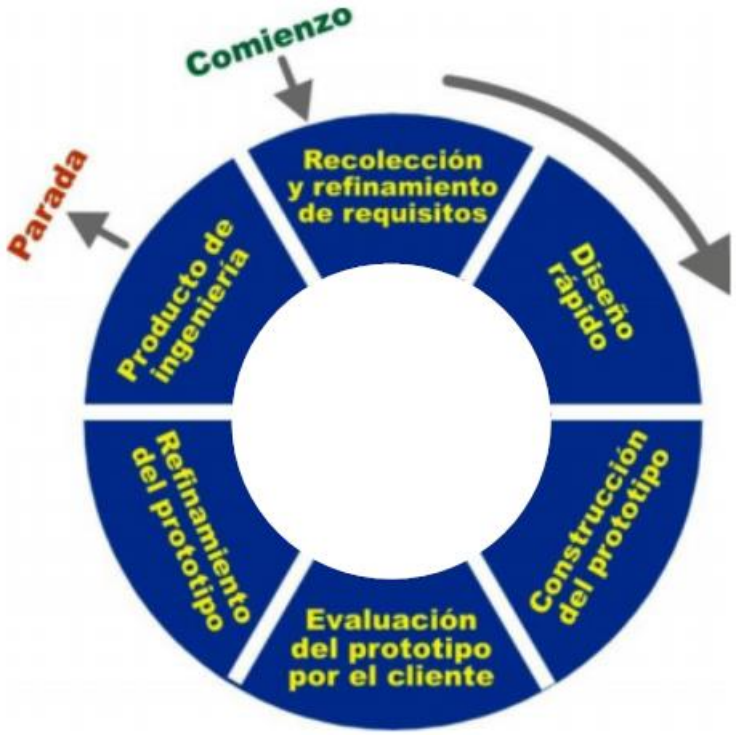


Ilustración 2: Modelo por construcción de prototipos

Se toma como base este modelo de desarrollo pero en este caso concreto los requisitos ya están previamente descritos a alto nivel y durante la etapa de desarrollo se entrará a mayor profundidad y se evaluará dicha funcionalidad pudiendo ser modificada. Además el diseño también estará previamente establecido y una vez empiece la fase de desarrollo también se evaluará en función de las dificultades que se puedan encontrar. Por lo tanto la metodología será evaluar la funcionalidad o requisito, además del diseño, implementar la funcionalidad, realizar pruebas unitarias y luego realizar pruebas de integración con el resto de funcionalidades para conseguir un producto estable. Hay que tener en cuenta el poco tiempo que se dispone para la realización del videojuego pero se intentará optimizar la estabilidad y las principales funcionalidades.

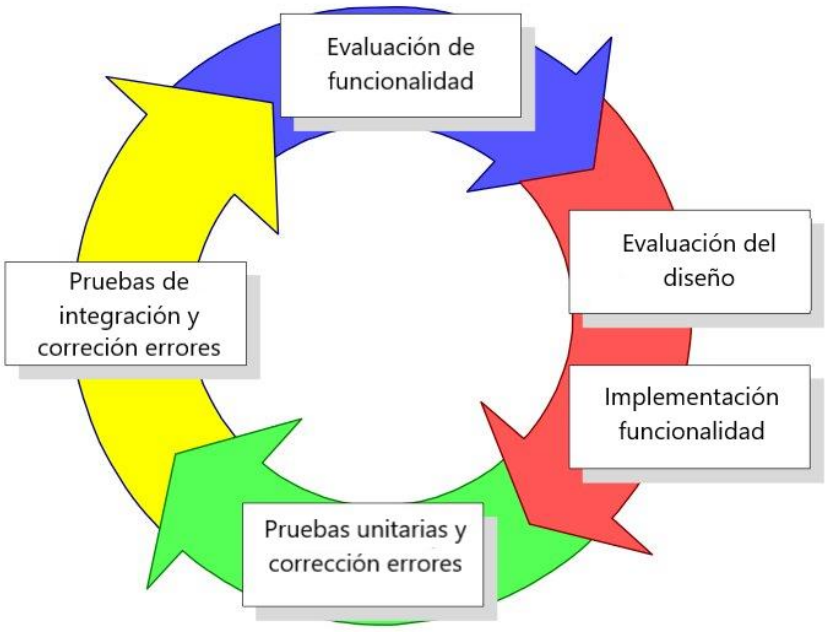


Ilustración 3: Metodología de trabajo

1.4.- Planificación del Trabajo

La planificación se tiene en cuenta desde el inicio de la PEC1 (19/02/2020) hasta la entrega final del producto y la presentación del video que se realiza en la PEC4 (14/05/2020), que aproximandamente son 15 semanas.

Algunos objetivos pueden sufrir cambios debido al tiempo muy ajustado que se dispone para la realización del producto, como por ejemplo, el número de escenarios que tenga el juego, que serán incluidas como mejoras futuras.

La siguiente tabla muestra el número de horas estimados que se le dedicará al desarrollo del proyecto.

TAREA	HORAS LABORABLES	HORAS FESTIVOS	TOTAL
PEC1: Planificación	14	29	33
PEC2: Diseño	30	42	72
PEC3: Implementación	58	98	156
PEC4: Entrega Final	12	56	68
			329

A continuación se muestra el diagrama de Gantt del proyecto donde se puede visualizar mejor la planificación. Todas las tareas de requisitos funcionales incluyen todos los pasos indicados en el apartado enfoque y diseño, es decir, contienen el analisis de requisito y diseño, además de las pruebas. A medida que se va avanzando, las pruebas de integración se realizan con más elementos por lo que al princpio puede que sean más ligeras, además se puede observar como en el tiempo dedicado al RF6 es mayor ya

que es ahí donde las pruebas se realizarán con todos los elementos que intervienen en el juego y problamente donde aparezcan más fallos. Por último indicar que a pesar de que durante la implementación se realizan diferentes pruebas se ha reservado una semana de pruebas al final para garantizar la calidad del producto y cumplir con el RNF6.

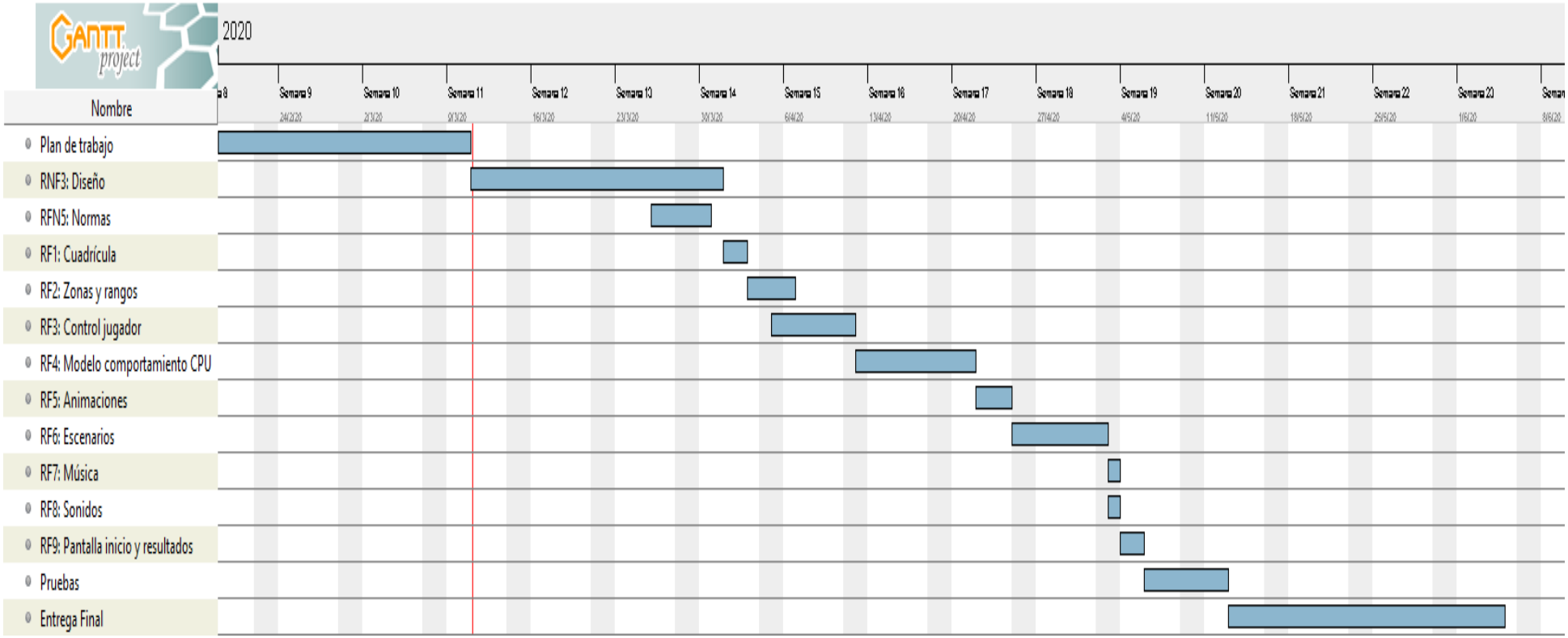


Ilustración 4: Diagrama de Gantt

1.5.- Análisis de riesgos y plan de contingencia

Uno de los principales problemas a la hora de iniciar el desarrollo del software es no conocer bien las necesidades del cliente, por ello en los capitulos posteriores se realiza un análisis de posibles usuarios para poder detectar necesidades y dificultades, además de un benchmarking para ver si podemos mejorar o añadir funcionalidades.

Otro de los grandes problemas son los plazos de entrega, en este caso el tiempo esta muy ajustado y cualquier retraso en cualquiera de las fases de desarrollo puede llevar a que no se cumplan los plazos de entrega, esto puede ser debido a varios factores:

- Plazos poco realistas.
- Cambios en la tecnología.
- Falta de compresión en los requisitos.
- Problemas con API´s.
- Otros: enferdades o situaciones personales que deriven en no realizar el trabajo con normalidad.

Para intentar paliar con estos posibles problemas se ha tomado la determinación de especificar, usando metodologías ágiles, con más detalle los requisitos del sistema, con esto se prentende evitar que haya dudas a la hora de saber que es lo que se tiene que realizar. Además, al estar dividido en diferentes secciones y tareas se puede intercambiar el orden de prioridad según las necesidades que vayan surgiendo, como por ejemplo dependecia entre funcionalidades u atascos a la hora del desarrollo, ya sea por falta de conocimiento o por fatiga mental.

Al ser los plazos establecidos muy cortos la estrategia a seguir es, una vez que se tenga implementada y probada una funcionalidad y cumpla con un mínimo de los requisitos establecidos se anotarán los detalles que quedan por afinar y se pasará a realizar otra funcionalidad con el objetivo de priorizar y cubrir al máximo los requisitos que luego en la fase final de pruebas se refinarán.

Por último indicar que se intentará, dentro de los posible, reutilizar componentes y funcionalidades para optimizar el crecimiento del desarrollo. Como se suele decir comunmente “no vamos a reinventar la rueda”.

1.6.- Breve sumario de productos obtenidos

- Memoria Proyecto
- Aplicación Android

- Código Fuente
- Video Presentación

1.7.- Breve descripción de los otros capítulos de la memoria

- Cápitulo 2. Especificación de requisitos.
    - Capítulo 2.1. Funcionalidades: Desglose de las funcionalidades en historias de usuario con más detalle usando las metodologías ágiles.
    - Capítulo 2.2. Análisis: Análisis de los jugadores objetivos, dificultades y necesidades.
    - Capitulo 2.3. Benchmarking: Análisis del mercado de juegos similares.
    - Capítulo 2.4. Diseño de la interfaz gráfica: Especificación y diseño de la interfaz gráfica para interactuar con el juego.
    - Capítulo 2.5. Arquitectura del sistema: Diagramas de clases (antiguo y actualizado) y secuencias con notación UML describiendo el sistema.
    - Capítulo 2.7. Normas: Especificación de las normas del juego que deben cumplirse en todo momento y su ciclo de vida.
- Cápitulo 3. Implementación.
    - Capítulo 3.1. Breve resumen: Breve resumen sobre los detalles del proceso de implementación.
    - Capítulo 3.2: Aspectos mas importantes: Descripción de los detalles más relevantes del proceso de la implementación, en concreto, control del personaje, comportamientos NPCs y escenario.
    - Capítulo 3.3: Pruebas: Descripción de los detalles más importantes durante las pruebas realizadas, tanto unitarias como de integración con otras funcionalidades y como detectar fallos.
    - Capítulo 3.4: Ejemplo de código: Se muestra un ejemplo de cómo está comentado el código.
    - Capítulo 3.5: Ejemplo de animaciones: Se muestra un ejemplo de como se ha realiado el árbol de animación.
- Cápitulo 4. Conclusiones del desarrollo.
    - Capítulo 4.1: Análisis del cumplimiento de la planificación: Se establece el detalle de cada historia de usuario y se anallizan el estado y el tiempo dedicado frente al estimado. Incluye un resumen con los principales problemas y limitaciones encontrados, además de las métricas y las lecciones aprendidas.

2.- Especificación de requisitos

2.1.- Funcionalidades

Para gestionar y listar las principales funcionalidades, declarados en el apartado 1.2, que debe implementar el producto software se han aplicado conceptos de metodologías ágiles para distribuir los requisitos en diferentes niveles jerárquicos que se explican brevemente a continuación:

- EPIC: Es el nivel de abstracción principal, en el se refleja el objetivo final que se pretende conseguir.

- FEATURE: En este nivel se establecen los requisitos funcionales a alto nivel.

- USER STORIES: Es el nivel más bajo, en él está reflejado con más detalle cual es la necesidad que se debe cubrir. Básicamente es donde se especifica que hay que implementar.

Con todo esto, a continuación se muestra la lista de funcionalidades que se deben implementar:

- [EPIC] Crear un videjuego RPG táctico al estilo Fire Emblem.
    - [FEATURE 1] Implementación del escenario donde interactua el jugador y los NPC.
        - [USER STORY 1] Creación de un escenario a modo de cuadrícula con distintas zonas (bosque, agua, ...) donde se colocarán los personajes (con personaje se refiere a los controlados por el jugador, PC, y los NPC).
        - [USER STORY 2] Implementación de las zonas de movimiento de los personajes.
        - [USER STORY 3] Implementación del rango de ataque de los personajes.
    - [FEATURE 2] Control del jugador mediante gestos.
        - [USER STORY 4] Al pulsar en un PC que pertenece al usuario se visualizará sus zonas de movimiento y ataque, si pulsa de nuevo sobre él no se visualizará. Si pulsa sobre otro personajes del jugador no se visualizará el primero y si el segundo.
        - [USER STORY 5] Una vez que el PC esta seleccionado si pulsa sobre una posición que está detro del rango y la zona lo permite el personajes se desplazará a esa posición.

    - [USER STORY 6] Si un NPC está en el rango de ataque el jugador puede atacarlo seleccionando al NPC.
    - [USER STORY 7] Si un PC está en rango de ataque el jugador puede curarlo seleccionadolo, si tiene el objeto necesario.

- [FEATURE 3] Modelo de comportamiento de los NPC.

    - [USER STORY 8] Se establecen distintos modos de comportamiento, la idea es que cada NPC adquiera uno o varios modelos de comportamiento aleatorio de los que se mencionan a continuación.
    - [USER STORY 9] Si un PC está en el rango del NPC su prioridad será atacar.
    - [USER STORY 10] Si un jugador esta en el rango del NPC su prioridad será buscar apollo.
    - [USER STORY 11] Si un NPC tiene daño acumulado y otro de ellos esta en el rango y dispone de un objeto curativo sanará al NPC dañado.
    - [USER STORY 12] Si el NPC dispone de varios PCs para atacar, atacará al que tenga mayor daño acumulado.
    - [USER STORY 13] Si el NPC dispone de varios PCs para atacar, atacará al que tenga menor daño acumulado.

- [FEATURE 4] Implementación de un modo de combate equilibrado.

    - [USER STORY 14] Establecer a los personajes un porcentaje de acierto en el ataque.
    - [USER STORY 15] Establecer una proporción adecuada y equilibrada entre ataque y defensa.
    - [USER STORY 16] Cuando un personaje es atacado y aún tiene vida este responde con otro ataque siempre que este en el rango de ataque.

- [FEATURE 5] Implementación de las animaciones.

    - [USER STORY 17] Cuando un PC o NPC realiza un ataque se debe visualizar la animación correspondiente.
    - [USER STORY 18] Cuando un PC o NPC realiza una acción de curación se debe visualizar la animación correspondiente.

- [FEATURE 6] Creación de diversos escenarios.

        - [USER STORY 19] Implementación del primer escenario.
        - [USER STORY 20] Implementación del segundo escenario.

    - [FEATURE 7] Implementación de la música de ambiente.

        - [USER STORY 21] Durante la partida sonará musica de fondo.

    - [FEATURE 8] Implementación de los efectos sonóros.

        - [USER STORY 22] Cuando se realice un ataque se debe escuchar el sonido correspondiente.
        - [USER STORY 23] Cuando se realice una curación se debe escuchar el sonido correspondiente.
        - [USER STORY 24] Cuando se derrote a un personaje o CPU se debe escuchar el sonido correspondiente.

    - [FEATURE 9] Implementación del ciclo del juego.

        - [USER STORY 25] Crear pantalla inicial con las opciones de iniciar el juego y activar/desactivar sónido y música.
        - [USER STORY 26] Cuando no quedan NPCs en la pantalla se pasa de nivel.
        - [USER STORY 27] Si no quedan PCs el jugador pierde la partida.
        - [USER STORY 28] Cuando se pasa el nivel 2 se da por terminada la partida.

    - [FEATURE 10] Implementación de diversos personajes, se deben poder diferenciar visualmente y por sus características.

        - [USER STORY 29] Crear personaje con un rango de movimientos y de ataque distintos del estándar.

2.2.- Análisis de usuarios

Una parte muy importante es analizar quienes pueden ser los posibles usuarios y en que condiciones o situaciones podrá disfrutar del producto. Durante el periódo de investigación se comenzó con una dinámica de grupo entre varios posibles usuarios ya que es algo más informal y más fácil romper el hielo, y para finalizar, se optó por entrevistas semiestructuradas y abiertas para favorecer la creatividad del entrevistado y obtener fichas de persona y casos de usos reales.

2.2.1.- Fichas de personas

FICHA DE PERSONA 1	DESCRIPCIÓN
Nombre	Álvaro
Sexo	Varón
Edad	30
Profesión	Contable
Tipo de usuario	Principal
Día cotidiano	Álvaro se levanta temprano, aproximadamente sobre las 7 de la mañana, se tomá su café correspondiente y coge el metro para ir a su trabajo. El trayecto al trabajo suele durar unos veinticinco minutos, a esa hora el metro suele estar concurrido pero siempre encuentra sitio para sentarse. Como es costumbre aprovecha ese tiempo para leer alguna noticia que le pueda interesar o jugar a algún juego. El metro está en permanente movimiento por lo que es incómo jugar a juegos complejos y que requieren mucha atención, además suele haber ruído. Con un descanso de una hora para almorzar suele hacer uso de nuevo del móvil hasta su vuelta al trabajo para salir a las seis y media y vuelve a realizar el mismo trayecto para volver a casa. Al llegar a casa suele realizar la compra si es necesario y comienza a preparar una cena ligera, y mientras espera la hora de cenar suele escuchar música o ver la tele mientras habla o juega con el móvil. Una vez termina la cena su objetivo es desconectar lo antes posible para afrontar otra dura jornada por lo que se pone a leer o ver alguna película.

FICHA DE PERSONA 2	DESCRIPCIÓN
Nombre	Carmen
Sexo	Mujer
Edad	26
Profesión	Estudiante
Tipo de usuario	Principal
Día cotidiano	Carmen estudió derecho y actualmente se está preparando para unas oposiciones que le tienen ocupada la mayor parte del tiempo.

	Se levanta temprano para desayunar tranquilamente y coger el autobús que le lleva hasta donde esta su preparador en unas oficinas junto a la biblioteca. El recorrido suele ser de unos cuarenta minutos y es bastante cómodo por lo que aprovecha para entretenerse con algún juego de su móvil. Suele almorzar en la cafetería junto a la biblioteca y luego sigue estudiando, a pequeños intervalos usa su móvil para chatear o echar un partida rápida para desconectar. Al volver a casa cena y suele ver una seríe y se acuesta a dormir.

2.2.2.- Escenarios

Escenario 1 - Persona implicada: Álvaro

Son aproximadamente las 8 de la mañana y como cada día Álvaro está en el metro sentado mirando las noticias en su móvil, le esperan unos veinticinco minutos para llegar a su parada y la cobertura es mala por lo que no tiene muy buena conexión a internet por lo tanto se dispone a jugar a un juego de RPG por turnos offline, lo cuál es perfecto ya que puede realizar sus movimientos con calma y estar pendiente de que no se le pase la parada ya que no tiene limite de tiempo para realizar sus movimientos y no necesita de excesiva precisión para realizar sus acciones.

Escenario 2 - Persona implicada: Carmen

Son las cinco y media de la tarde y Carmen, tras una mañana dura de estudios y haber almorzado con sus compañeros de preparación, continua estudiando comenzando a sentir la fatiga mental que esto implica por lo que echa un partida, al ser un juego de movimiento simples puede realizar sus movimiento y seguir estudiando. Durante la partida recibe una llamada de telefono de una de sus amigas, sale de la biblioteca contesta y vuelvea entrar, al iniciar de nuevo sus estudios recuerda que estaba en mitad de una de las acciones en su partida, pero no hay ningún problema el juego se mantiene en pausa y de un simple vistazo sabe que acciones son las que iba a realizar y continua de esa forma, alternando estudios y acciones del juego hasta las siete y media de la tarde que vuelve a casa.

2.2.2.- Conclusiones

Se han analizado los escenarios extraidos de la vida cotidiana de las personas y se han establecido las siguientes conclusiones:

- El juego se debe poder interrumpir sin problemas, además continuar debe ser sencillo para volver a coger el hilo de las acciones que se estaban realizando.

- Normalmente los jugadores son de tipo causal, suelen jugar para matar los entretiempos o desconectar de otras tareas.

- El juego tiene que ser sencillo e intuitivo que no sea muy complicado comenzar a jugar para atraer la atención del usuario.

- El entorno de uso de la aplicación puede ser medianamente hostil, es decir, puede a ver mucho ruido, vibraciones, etc, lo cuál implica que el control debe estar muy ajustado y el diseño debe adaptarse a estas necesidades.

2.3.- Benchmarking

Una buena estrategia para recolectar ideas y detectar nuevas funcionalidades o experiencias de usuario es analizar lo que nos ofrece el mercado, en este apartado realizamos un análisis de los principales juegos para dispositivos móviles.

Nombre	Descargas y valoración	Interfaz	Jugabilidad
Fire Emblem Héroes	Más de 5 millones de descargas. 4.3 de puntuación de 510.841 opiniones.	Fácil e intutiva, comienza con un menú y rápidamente ya estás iniciando la partida ya que puede omitir presentaciones e historia. Estás totalmente sumergido en el juego nada más comenzar. Música y sonidos espectaculares. Diseño de personajes, animación y ambientación muy detallado y preciso.	Todas las acciones se realizan seleccionando el personaje deseado y se arrastra a la posición deseada tanto para atacar como para moverse. Al principio muestra un tutorial. Es bastante intuitivo a pesar de no mostrar que acción se está realizando.
EGGLIA: Legend of the Redcap Offline	Mas de 100.000 de descargas. 3.6 de puntuación de 324 opiniones.	Es algo más compleja, tarda bastante en comenzar el juego y no se pueden omitir las	En este caso incorpora el lanzamiento de un dado para poder desplazarse por

		escenas. Música y sonidos muy agradables y acordes con la ambientación Diseño de personajes, animación y ambientación acordes al estilo que representa.	las distintas casillas y puedes interactuar con elementos del escenario. Las acciones se realizan tocando sobre el personaje, antes de comenzar aparece una descripción de como jugar. No es tan intuitivo pero tampoco es complicado de jugar. No aparecen que acciones está realizando.
Tactics Age: Turn based strategy	Más de 50.000 descargas 4.6 de puntuación de 1.000 opiniones	La interfaz es fácil, de hecho no hay ni pantalla de inicio directamente estás en la batalla. La música es adecuada y el sónido también Diseño de personajes un poco extraño ya que la diferencia de tamaño entre unos y otros se hace evidente ocupando el mismo espacio. Animaciones sencillas	Se parece bastante a lo que se busca en este proyecto pero no queda muy claro que acciones puedes realizar. Todas las acciones se realizan tocando sobre el jugador.

2.3.1.- Conlusiones

En ninguno de los tres juegos analizados se muestra un menú con las acciones que podemos realizar. Son todos bastante intuitivos a la hora de jugar ya que sólo con tocar los jugadores se realizan las acciones.

2.4.- Diseño de la interfaz gráfica

En este apartado se mostrará el diseño de la interfaz gráfica, es decir, el diseño de las pantallas, la interacción y la navegación entre ellas, esta tarea tiene mucha más importancia a la hora de realizar una aplicación convencional ya que consta de multiples pantallas. En este caso particular tiene más importancia la implantación de las normas del juego y el diseño de los niveles top-down.

2.4.1.- Árbol de navegación

El árbol de navegación es bastante sencillo ya que apenas existe navegación entre las pantallas con el objetivo de que el usuario comience a jugar lo antes posible. El juego consta de la pantalla principal donde se podrá iniciar la partida y modificar las opciones de sonido y música.

Una vez se haya iniciado la partida el usuario debe superar los diferentes niveles (estos no se consideran como una navegación) y si fracasa volverá a la pantalla principal.

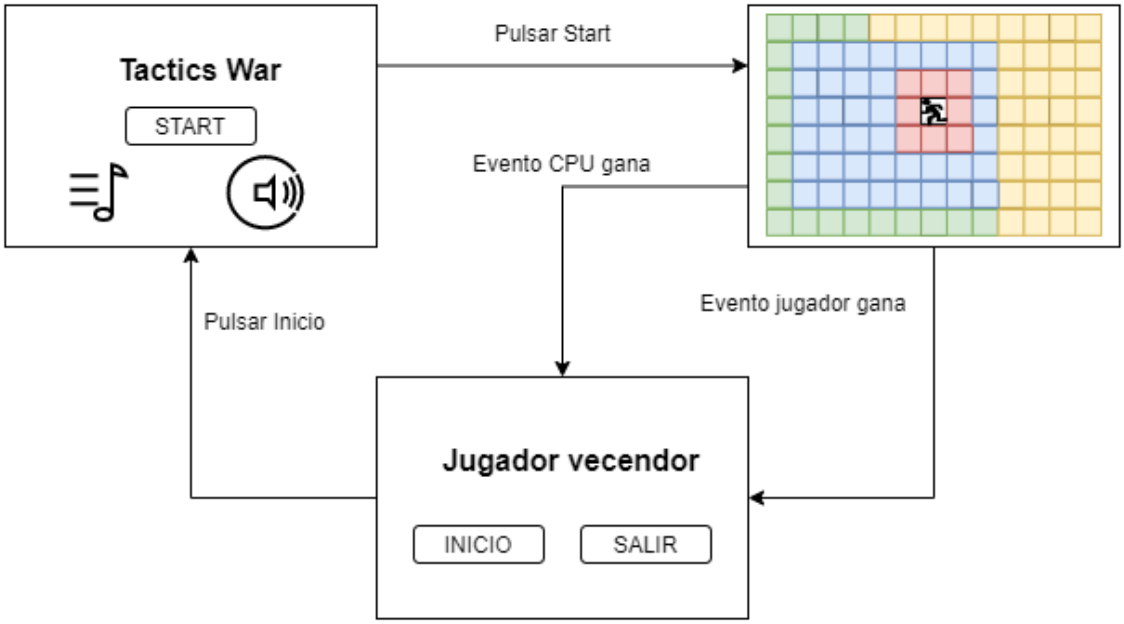


Ilustración 5: Árbol de Navegación

2.4.2.- Moodboard

Para los amantes de los RPG tácticos nunca podremos olvidar como fueron sus inicios en consolas de 8 y 16 bits con sus gráficos pixelados y en honor a esos juegos que han inspirado a realizar este proyecto se ha elegido una temática aventurera al estilo clásico [6].

Ilustración 6: Moodboard

2.5.- Diseño inicial de la arquitectura

Como se ha mencionado en apartados anteriores el motor gráfico elegido es Unity que posee una estructura muy amplia que no se va a mostrar en el siguiente diagrama de clases. Este caso concreto se centra en las partes esenciales de Unity que son necesarias y las clases que se generarán para definir el comportamiento del videojuego.

Mencionar que cómo la estrategia de desarrollo elegida esta basada en prototipos la estructura del modelo de sistema mostrada puede sufrir algunos cambios por problemas derivados de la implementación, por supuesto se intentará evitar cambios y si no es posible estos cambios sean mínimos y afecten lo mínimo para no influir en la planificación establecida.

Indicar que todas las clases que heredan de MonoBehaviour formarán el conjunto de scripts de comportamiento de los diferentes elementos que intervienen.

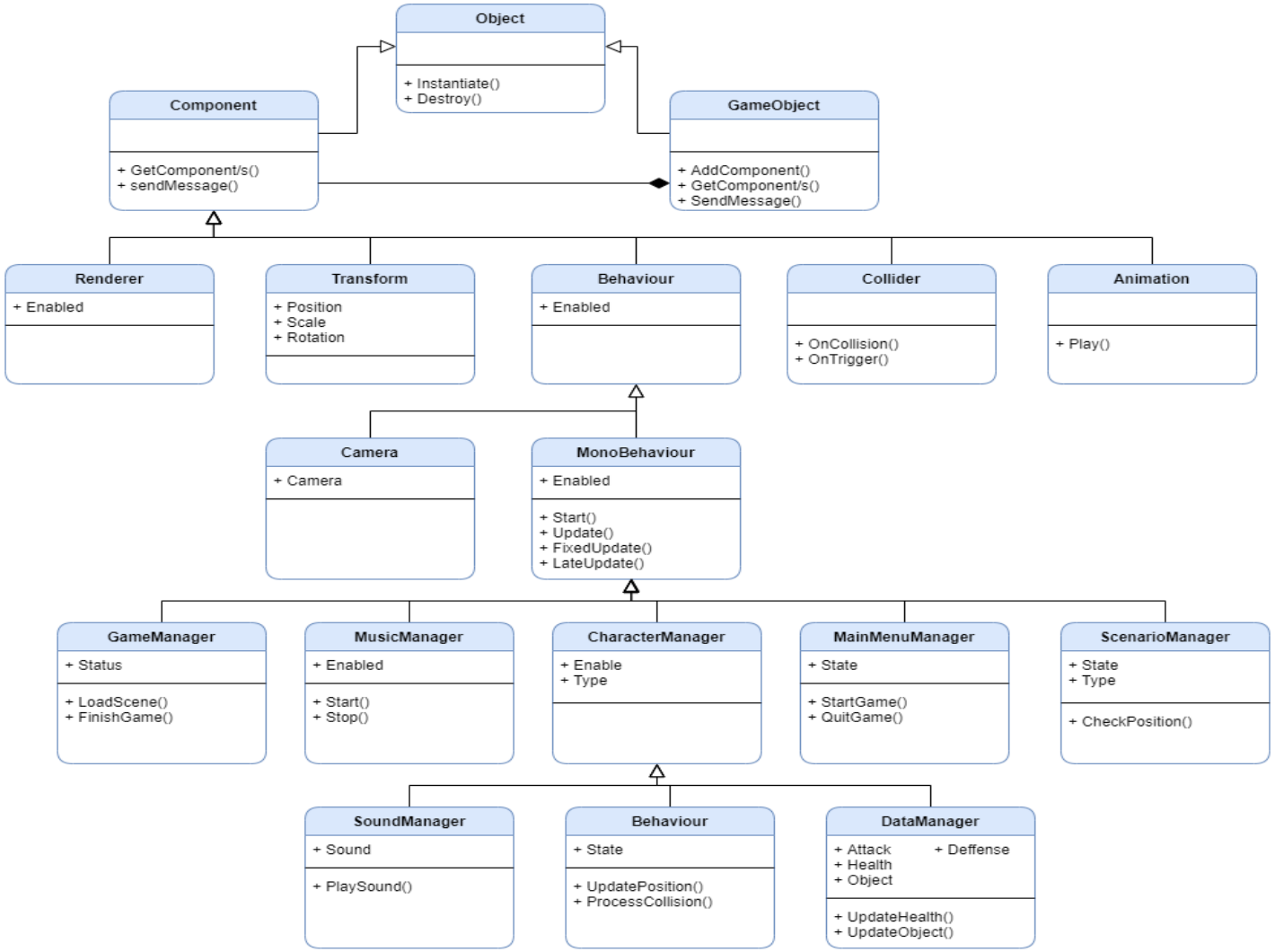


Ilustración 7: Diagrama inicial de clases

2.5.1.- Diagramas de secuencia

En esta sección se muestran los principales diagramas de secuencias y los elementos que intervienen, hay que tener encuenta que el objetivo de estos diagramas es mostrar una visión general de los elementos que intervienen y no entrar el el detalle concreto.

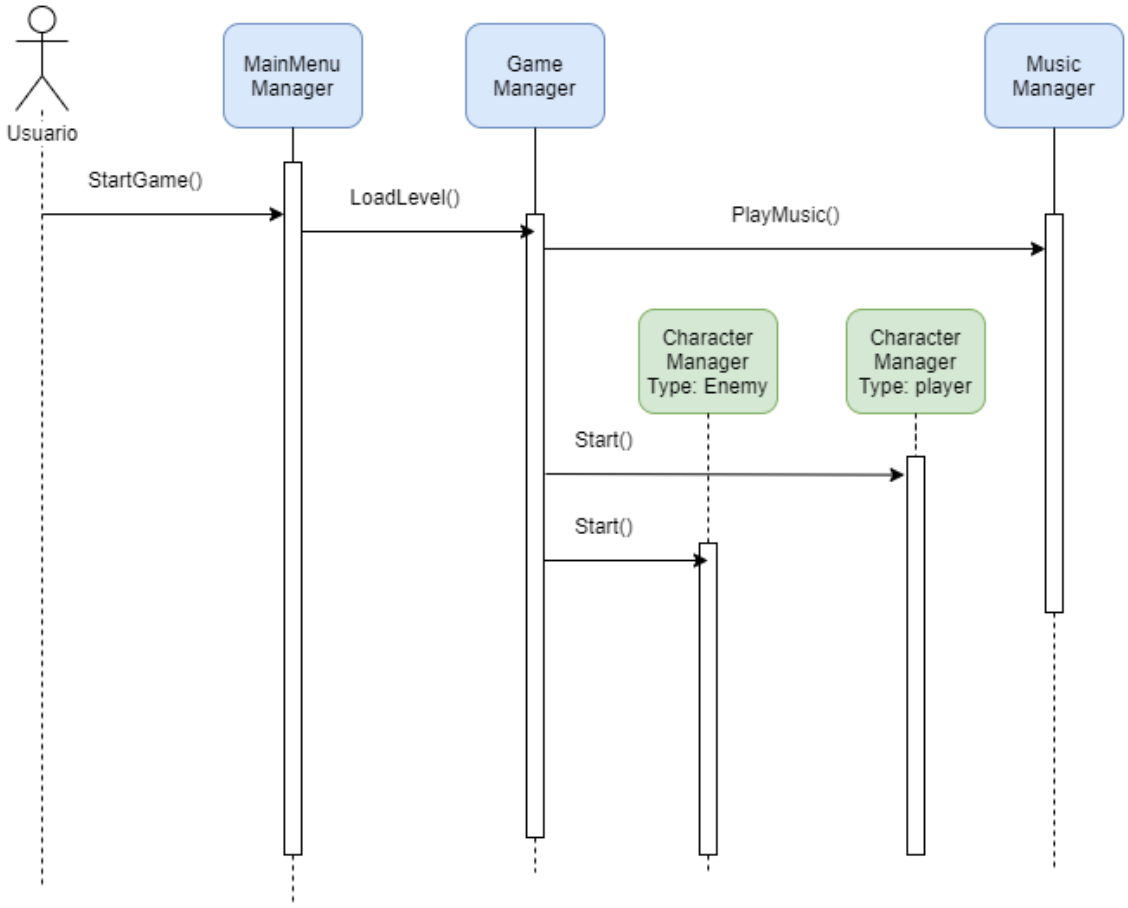


Ilustración 8: Diagrama de secuencia - Iniciar juego

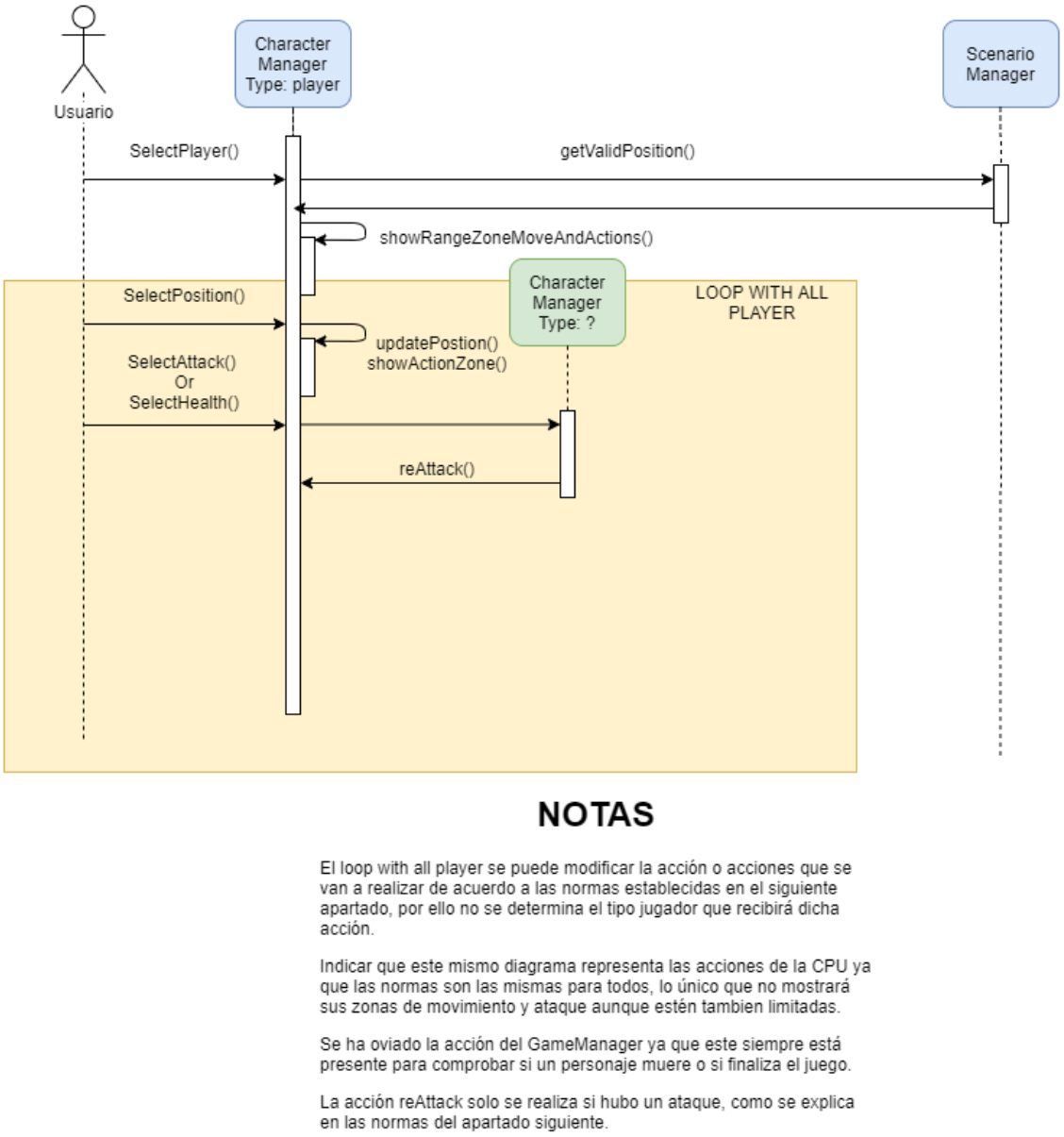


Ilustración 9: Diagrama de secuencia - Movimiento

2.6.- Diseño final de la arquitectura

Durante el desarrollo la arquitectura inicial expuesta ha sufrido cambios con el fin facilitar su desarrollo y mejorar su funcionalidad y estructura, en concreto se ha eliminado toda la herencia que partia de la clase CharacterManager, por lo tanto las funcionalidades de SoundManager quedan todas centralizadas en la clase MusicManager, las funcionalidades de Behaviour quedan recogidas en CharacterManager, además esta clase solo la usan los NPCs, la clase CharacterDataManager queda al mismo nivel jerárquico que CharacterManager y se añade la clase CharacterController que gestiona las animaciones. Con estos cambio se ha facilitado la implementación de las funcionalidades especificadas ya que en este caso la herencia de datos no aportaba ninguna mejora y era más complicado establecer las relaciones entre las distintas clases que conforman la aplicación.

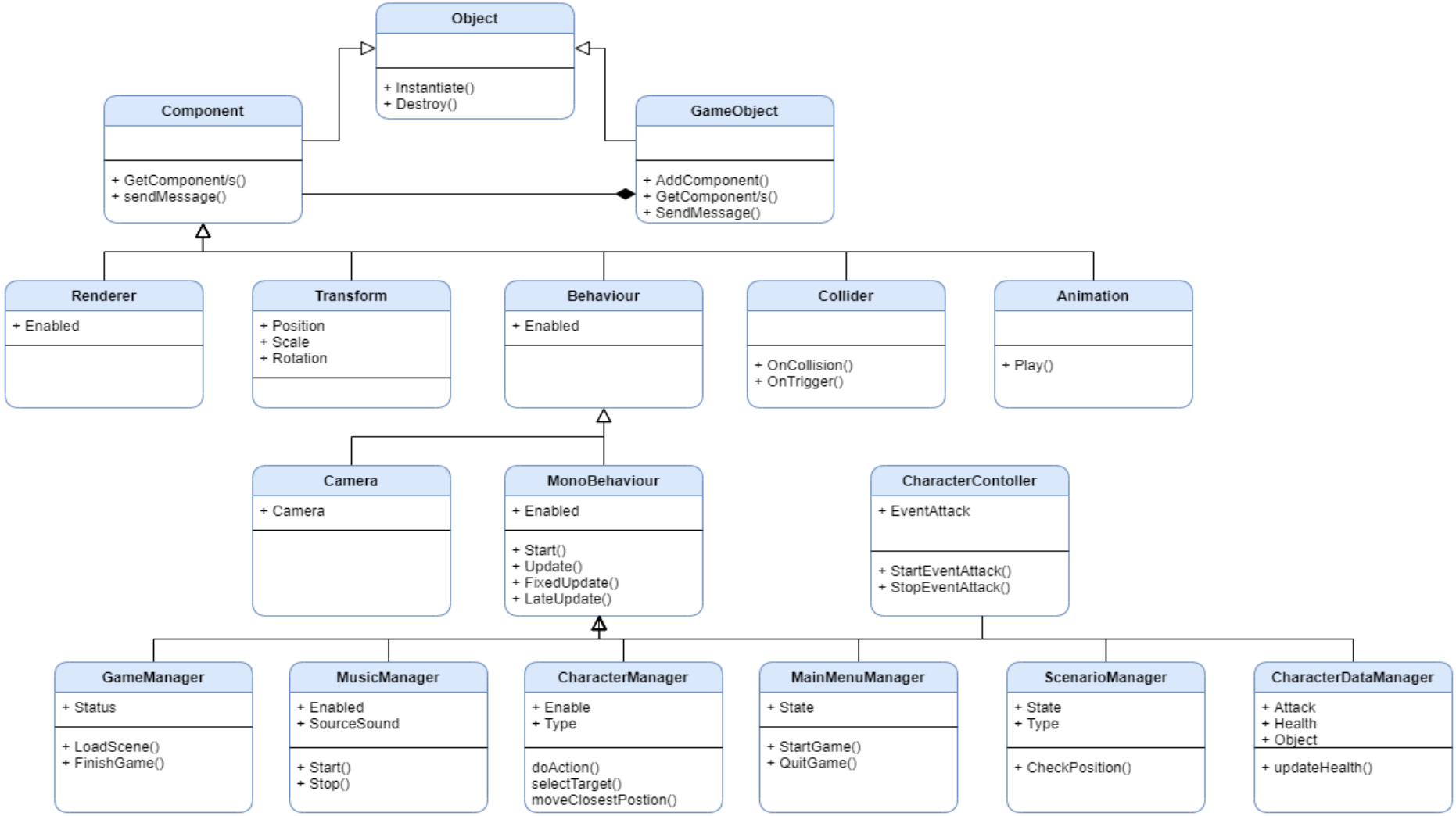


Ilustración 10: Diagrama final de clases

2.7.- Normas

A continuación, se definen las reglas del juego que se deben estar cumpliendo en todo momento.

1. Los personajes sólo se podrán desplazar dentro del rango de movimiento definido y por la zonas establecidas.

2. Gana el que derrote a todos los personajes del otro equipo.

3. El usuario es el que tiene el primer turno.

4. Tanto el usuario como la CPU sólo podrán realizar un movimiento por cada personaje.

5. Se considera movimiento a cualquiera de las opciones que se listan a continuación y no se puede modificar el orden establecido en cada una de ellas:

    a. Desplazar un personaje a una zona dentro del rango.
    b. Desplazar un personaje a una zona dentro del rango y realizar una acción (atacar o curar a otro personaje).
    c. Realizar una acción (atacar o curar a otro personaje).

6. La CPU se moverá según los criterios definidos en la historia de usuario [FEATURE 3] Modelo de comportamiento de la CPU del apartado 2.1.

7. Sólo se podrá atacar si el personaje esta dentro del rango establecido para el ataque.

8. Si un personaje es atacado y aún le queda vitalidad respondrá con otro ataque siempre que este dentro de su rango de ataque.

9. Sólo se puede seleccionar un personaje como objetivo enemigo si es para el ataque o un usuario aliado si es para cura, no se puede seleccionar a si mismo.

10. Si un personaje se queda sin puntos de salud es derrotado.

3.- Implementación

3.1.- Breve resumen

En primer lugar se instaló correctamente la herramienta Unity en su última versión, al inicio del proceso de implementación la última versión era 2019.3.9.f1 y se realizó la busqueda de los tileset y sprites que definen nuestro escenario y personajes.

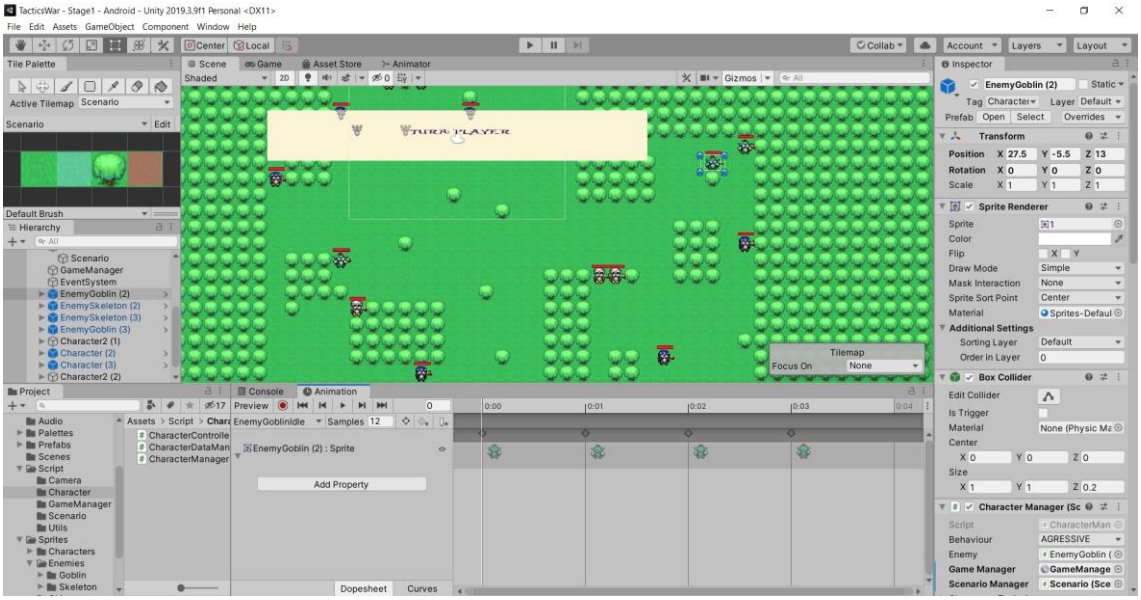


Ilustración 11: Trabajando con la herramienta Unity

Gracias al trabajo previo de separar en historias de usuarios las distintas funcionalidades y de priorizarlas desde el primer momento fue sencillo saber que había que hacer y como se iba respecto a la planificación. Los primeros contratiempos llegaron a la hora de probar las acciones que se realizan según que parte del escenario se seleccione, estos problemas se fueron solventando a medida que poco a poco se iba avanzando en otras historias de usuarios, que también hacian que surgieran más problemas debido a que interactúan mas elementos, como por ejemplo el comportamiento de los NPC.

Para los gráficos, como se menciona en el apartado 2.4.2, se compraron un paquete de fantasía estilo pixelado top-down. Esto desenbocó en otro problema ya que el paquete adquirido, a pesar de que encajaba perfectamente con la temática, los Sprites y tiles no estaban divididos, es decir, casi todos formaban parte de la misma imagen y con distintos tamaños, por lo que se le dedicó tiempo en la búsqueda una herramienta para solucionar dicho problema y generar y ajustar las imágenes individuales, además de adjuntar el arma correspondiente y en la dirección adecuada, otro detalle que no se tuvo en cuenta a la hora de la planificación ya que el estilo top-down es el que más animaciones generan.

En la fase final, se decició no acomenter alguna historia de usuario ya que se priorizó el apartado de pruebas y resolución de problemas. Esta decisión fue muy acertada ya que durante todo ese periódo se resolvieron muchos errores, algunos se quedaron sin resolver, y ajustes que hacen que el usuario pueda tener una buena exprecian con la aplicación y cumpliento, casi en su totaliadad, el RNF6 de obtener una aplicación mantenible y estable.

3.2.- Aspectos mas relevantes

Todos los aspectos a la hora de desarrollar un vieojuego son importantes, debido al tiempo limitado se ha tenido especial detalle en el desarrollo del control del personaje y del control de los NPCs ya que son los aspectos a los que el usuario se enfrenta durante toda su experiencia de juego y con el fin de evitar acciones no deseadas.

3.2.1.- Control del personaje

Este apartado es uno de los que mas tiempo se le ha dedicado debido a que es el modo que el usuario tiene para interactuar con nuestra aplicación, y si no cuidamos los detalles puede llevar a que la aplicación se comporte de forma errónea, ya sea por una mala acción del usuario o por no tener en cuenta los distintos casos de uso.

En mitad del desarrollo se dio a probar la aplicación a varios usuarios para analizar las acciones que realizaban y que consecuencias tenian, algunas se corrigieron y otras quedan recogidas como mejoras, ya que no se dispone de mucho tiempo y no se consideraron críticas.

Al principio las pruebas y correción de errores se realizaron a través de la herramienta Unity debido a la comodidad, y en la fase final de pruebas se realizaron sobre el terminal móvil, donde se tuvieron que realizar varios ajustes ya que el control táctil es mucho más sensible que el movimiento del ratón.

3.2.2.- Comportamiento de los NPC

El comportamiento de los NPC también ha sido una tarea costosa, debido a las multiples pruebas y correcciones.

En primer lugar había que tener especial cuidado en que el NPC siempre tomará una decisión, ya que en muchos casos, debido a la distancia de los personajes no realizaba ninguna acción. También es necesario ordenar tales acciones, en principio todos los NPC realizaban las acciones al mismo tiempo llegando a obstacullizarse entre ellos y ocupando las mismas posiciones, algo que no se podía permitir que ocurriera, por lo tanto se estableció un orden de actuación, de esta forma el NPC puede tomar la decisión más acertada y además quedá mejor visualmente para el usuario que puede ver que acciones esta realizando su rival.

Uno de los requisitos es que los NPC tuvieran distintos comportamientos, se han desarrollado dos tipos de comportamiento , uno

agresivo, prioriza el ataque a los enemigos y otro conservador, que prioriza curar a sus aliados, que se asignan de forma aleatoria para dar diversidad al juego.

El principal problema de este apartado del desarrollo es el consumo de tiempo que se lleva en las pruebas, ya que es complicado crear situaciones en las que se produzcan los fallos. A pesar de disponer de un sistema de logs a veces los fallos se producen tras haber realizado varias acciones y es dificil de volver a reproducirlo para verificar que se ha solucionado el comportamiento erróneo.

3.2.3.- Escenario

Este es elemento del que, prácticamente, todos los elementos de la aplicacición dependen. En el se han establecido todas las funcionalidades del control del usuario, con la excepción de las pantallas de inicio y final, debido a su relación directa ya que el usuario está constamente tocando cada aspecto del escenario, además se han incluido que sea a través de esta clase (ScenarioManager) el control de las acciones de ataque y movimiento ya que estas se realizan en función de las coordenadas de Grid que forma el escenario.

También en él, se almacenan los distintos Diccionarios que controlan los gameObjects que están en pantalla, actualizando su información constantemente y actualizando la información de la matriz que forma el escenario, es posible evitar el uso de la matriz pero la idea inicial para desarrollar el comportamiento fue usar algoritmos de caminos mínimos como los de Dijkstra [7] o Floyd-Warshall [8] que se rechazó por falta de tiempo, pero se decidió establecer la matriz para mantener el estado del escenario, evitar errores de solapamiento de posiciones y como actualización para el futuro.

```
case "Stage1":
{
    scenario = new int[rows, columns] {
    {1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
    {1, 1, 1, 1, 0, 0, 0, 2, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
    {1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
    {1, 1, 1, 1, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
    {0, 1, 1, 1, 0, 3, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
    {0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3},
    {0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 3, 0, 0},
    {2, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 2},
    {0, 1, 1, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0},
    {0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 3, 3, 1, 0, 0, 1, 1, 1, 0, 0},
    {0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 0, 0, 3, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},
    {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 2, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0}
    };

    break;
}
```

Ilustración 12: Matriz escenario

3.3.- Pruebas

En todos los proyectos se insiste en la importacia de las pruebas y siempre se le intenta dedicar mucho tiempo y el resultado siempre suele ser el mismo, falta tiempo para probar todo,a pesar de disponer de una planificación bien estructurada y definida y de un plan de riesgos.

Debido al tiempo limitado es díficil encajar todo a la perfección en la planificación, muchas veces debido a contratiempos y otras muchas debido a que se quiere afinar tanto algunos detalles que se acaba alejando del objetivo principal.

3.3.1.- Pruebas durante el desarrollo en vivo

Como se indico en el apartado 1.3, a medida que se desarrollaban funcionalidades e historias de usuario se iban realizando las pertinentes pruebas unitarias y de integración con el resto de elementos. Un aspecto fundamental para agilizar este proceso ha sido la declaración de variables públicas en Unity permitiendo cambiar los valores durante la propia ejecución de la aplicación sin necesidad de modificarlos mediante scripts.

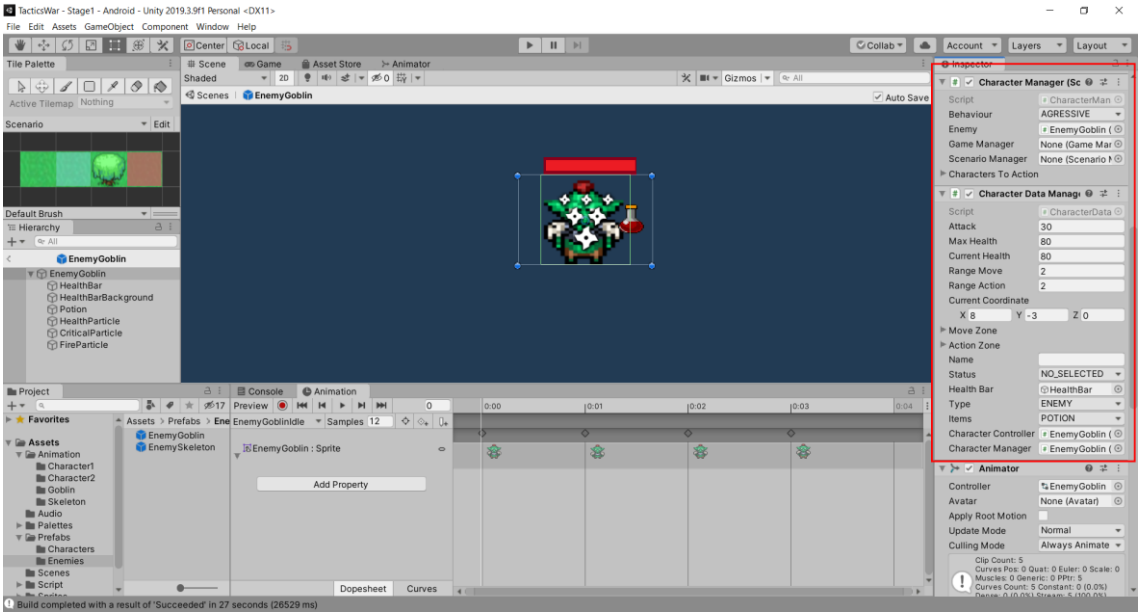


Ilustración 13: Variables públicas en Unity

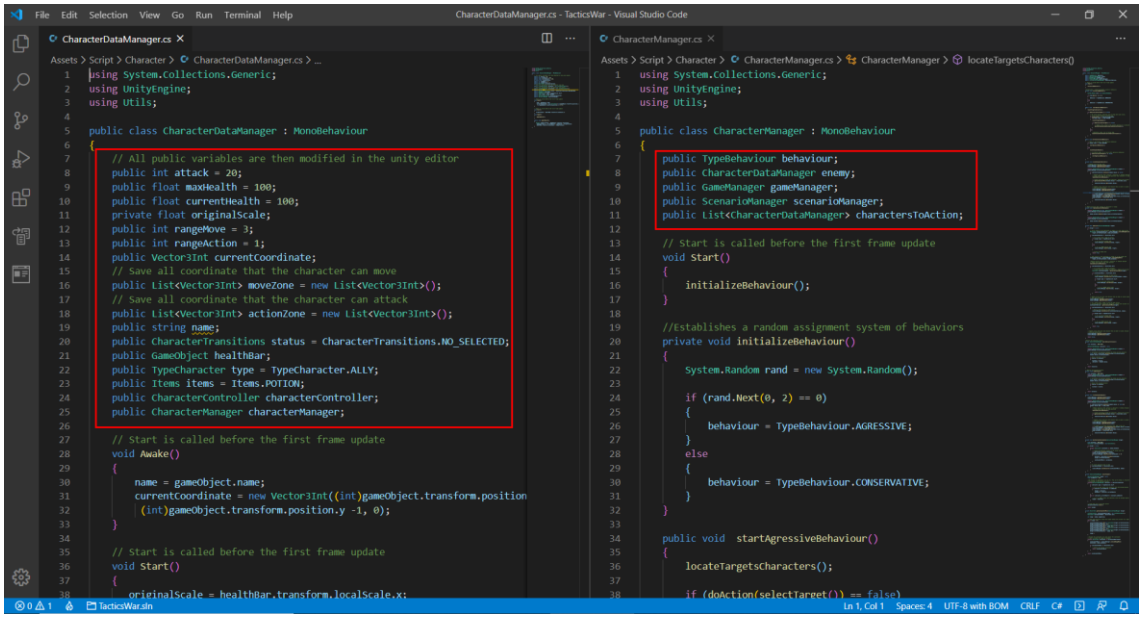


Ilustración 14: Variables públicas en Visual Studio

3.3.2.- Logs

Como en casi todas las pruebas, la piedra angular ha sido el sistema de logs a través de la consola de Unity, este sistema es la manera más secilla de trazar por donde va la ejecución y visualizar si los datos son los correctos.

3.3.3.- Pruebas en móvil

Compilar y transladar el ejecutable al móvil provoca una considerable súbida del tiempo dedicado a la hora de realizar pruebas, por ello, sólo se realizaron pruebas en la última semana con el terminal para facilitar el desarrollo.

3.4.- Ejemplo de código

En este apartado se muestra un fragmento de código. En el podemos ver como el código está perfectamente comentado para facilitar su compresión, en esta última versión del código se han eliminado los logs para facilitar su lectura, ya que para realizar un tracking correcto de ejecución hacen falta muchos logs.

```
// Perform the attack action
    public void attack(CharacterDataManager source,
    CharacterDataManager target)
    {
        // Check it's an accurate attack
        if (attackProbability() == true)
        {
            // If the attack is from character2, it activates the fire particles
            if (source.rangeAction == 2)
            {
                FindObjects.FindObjectInChilds(target,
```

```
                "FireParticle").GetComponent<ParticleSystem>().Play();
            }

            // Calculate the damage caused
            calculateDamage(source, target);
        }
        else // Check it is not an accurate attack to reproduce the fail sound
        {
            // Check if the mute option is select
            if (GameObject.Find("MusicManager").GetComponent<MusicManager>().backgroundMus
ic.mute == false)
            {
                GameObject.Find("MusicManager").GetComponent<MusicManager>().attackFailed.
Play();
            }
        }

        // Check the health of the target gameobject to destroy
        if (target.currentHealth <= 0)
        {
            // Destroy the gameobject
            destroyCharacterOrEnemy(target);
        }
        else // If still alive
        {
            // Check if it's in the action range
            if (isInsideRangeAction(source.currentCoordinate,
             target.actionZone) != Vector3Int.zero)
            {
                // Start the reattack process
                target.characterController.isReattack = true;
                startAttack(target, source);
            }
        }

        if (source != null)
        {
            //Finish the action
            source.status = CharacterTransitions.ACTION_DONE;

            // Upate the ranges
            updateActionZone(source);
            updateMoveZone(source);
        }

        if (target != null)
        {
            // Upate the ranges
            updateActionZone(target);
            updateMoveZone(target);
        }
    }
```

Ilustración 15: Ejemplo de código

3.5.- Ejemplo de animaciones

Las animaciones son un aspecto muy importante para captar la atención del usuario, se ha hecho especial hincapié en ellas intentando cubrir al

máximo los distintos tipos de animaciones para el ataque, al tratarse de un juego estilo top-down debemos tener en cuenta las animaciones en todas las direcciones, para ello se han creado eventos para controlar cuando acaba una animación y cuando debe comenzar la siguiente, de esta forma evitamos que se solapen y no entren en ciclos infinitos por la reutilización de funciones.

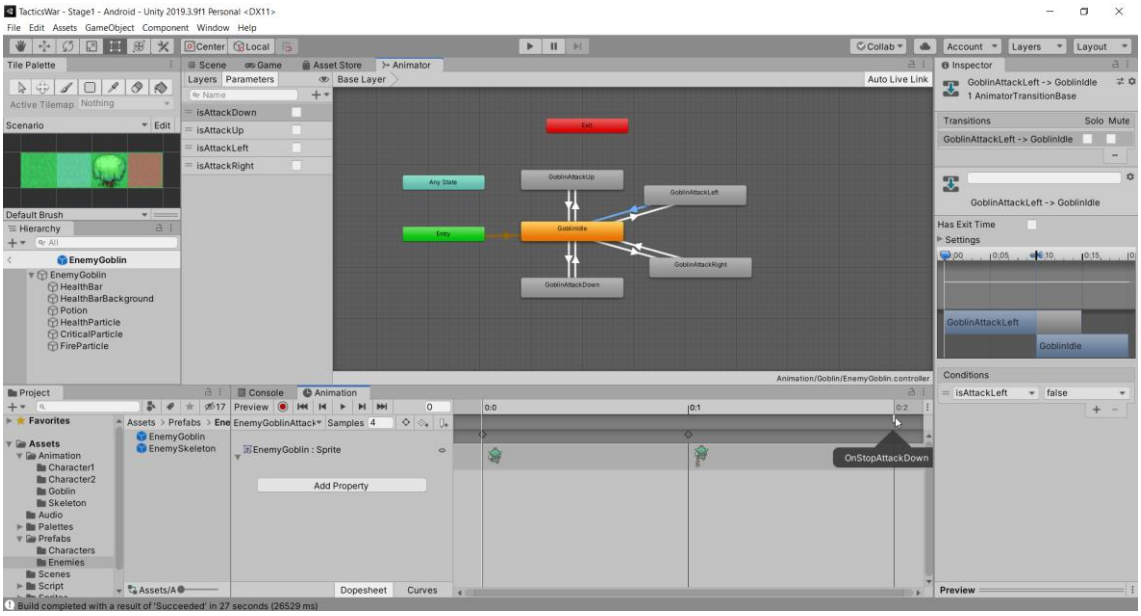


Ilustración 16: Ejemplo árbol de animaciones

4.- Conclusiones del desarrollo

4.1.- Análisis del cumplimieto de la planificación

Como se ha indicado en capítulos anteriores, el proyecto se ha realizado siguiendo el orden de las historias de usuarios especificadas. Aunque en general el resultado ha sido positivo vamos a analizar cada una de ellas para ver las desviaciones que se han producido.

User Story	Esfuerzo Estimado	Esfuerzo Realizado	Completado	Detalles
User Story 1: Creación escenarioa modo cuadricula	5 horas	10 horas	100%	Los tiles obtenidos no estaban separados debidamente y se realizaron ajustes de tamaño para que encjará en la cuadrícula.
User Story 2: Zonas de movimiento	20 horas	30 horas	100%	Ajuste de tiles. Configurable mediante una variable (no era un requisito pero se realizo pensando en futuros desarrollos). No se tuvieron en cuenta los distintos elementos que influyen para determinar si una posición es válida. No se tuvo en cuenta que los NPC usarían esta función para detectar PC en sus zonas de ataque, por lo que se incluyen y se excluyen posiciones.
User Story 3: Zonas de ataque	20 horas	10 horas	100%	Reutilización de funciones de la User Story 2
User Story 4: Seleccionar personaje	10 horas	30 horas	95%	No se tuvieron en cuenta las múltiples acciones erróneas que puede realizar el usuario y las consecuencias que trae. Se da casi por completada ya que cuando un personaje es eliminado debemos pulsar dos veces para realizar la siguiente selección. Se

				recogió el problema y se decidió continuar
User Story 5: Movimiento PC	10 horas	12 horas	100%	Problemas con la selección de posiciones erróneas y las posteriores acciones que realiza, por ejemplo no mostrar sus zonas y no perder el turno.
User Story 6: Ataque a NPC	10 horas	15 horas	100%	Problemas con la selección de objetivos y mantener quien era el que realizaba la acción. Problemas con las selecciones erróneas y el contraataque.
User Story 7: Sanar a PC	5 horas	3 horas	100%	Los problemas encontrados en las anteriores User Stories facilitaron el trabajo.
User Story 8: Asignación modelo comportamiento NPC	2 horas	1 hora	100%	Se asignan sin problemas de forma aleatoria uno de los dos modos de comportamiento. Se descarta la idea de que se pudieran asignar varios con el fin de probar estos comportamientos adecuadamente.
User Story 9: Comportamiento NPC prioriza atacar	10 horas	15 horas	100%	Se establece con el modo de comportamiento agresivo. Problemas a la hora de seleccionar el objetivo del ataque cuando no esta dentro de su rango de movimiento y acción.
User Story 10: Comportamiento NPC prioriza apoyo	5 horas	0 horas	0%	Se descarta para priorizar pruebas de los dos modelos de comportamiento establecidos.
User Story 11: NPC sanar	10 horas	10 horas	100%	Reutilizando todas las funciones de la User Story 9 se incluyen en el modo de comportamiento conservador
User Story 12: NPC Selección de objetivo mayor daño	3 horas	5 horas	100%	Al incluir los nuevos personajes con menor vitalidad

acumulado				se tuvieron que realizar ajustes
User Story 13: NPC Selección de objetivo menor daño acumulado.	2 horas	0 horas	0%	Se descarta para priorizar pruebas de los dos modelos de comportamiento establecidos, además este requisito facilitaría mucho las cosas al usuario ya que la opción lógica es atacar al que tenga menor vida
User Story 14: Porcentaje de acierto en el ataque.	5 horas	1 hora	100%	Se tardó más en realizar el ajuste. Se decidió que todos los personajes tuvieran el mismo porcentaje de acierto para dedicar tiempo a cosas más prioritarias.
User Story 15: Equilibrio entre ataque y defensa	5 horas	1 hora	100 %	Gracias al trabajo realizado en la anterior User Story se pudo incluir un porcentaje de ataque crítico para favorecer la experiencia de usuario y dedicar tiempo a tareas prioritarias.
User Story 16: Contraataque	5 horas	2 horas	100%	Anteriores User Stories facilitaron el trabajo
User Story 17: Animación ataque	5 horas	10 horas	100%	Ajuste de los Sprite obtenidos. Se reutilizan las funciones de ataque lo que provocó ciclos infinitos entre ataque y contraataques (usan las mismas animaciones) por lo que se realizo la gestión por eventos para coordinarlas. Se incluye un sistema de particulas si el ataque es critico con el fin de distinguirlo visualmente.
User Story 18: Animación sanar	5 horas	1 hora	100%	Se crea un sistema de particulas para evitar crear más animaciones y dedicar tiempo a tareas prioritarias.
User Story 19: Primer	2 horas	2 horas	100%	Se realizan

escenario				pequeños ajustes tras jugarlo.
User Story 20: Segundo escenario	2 horas	2 horas	100%	Se realizan pequeños ajustes tras jugarlo.
User Story 21: Música de fondo.	2 horas	1 hora	100%	Esta estimación es un poco aleatoria ya que depende de si la música con licencia libre que encontramos es adecuada. La implementación es sencilla y la prueba también.
User Story 22: Sonido atacar.	2 horas	2 horas	100%	Los personajes del mismo tipo tienen los mismos sonidos se distinguen entre sonido de ataque normal y crítico.
User Story 23: Sonido sanar.	1 hora	1 hora	100%	Es el mismo sonido para todos, lo que facilitó mucho la tarea.
User Story 24: Sonido muerte personaje	1 hora	1 hora	100%	Es el mismo sonido para todos, lo que facilitó mucho la tarea.
User Story 25: Pantalla inicial	2 horas	4 horas	100%	La mala decisión de no usar plantillas provocó el aumento de tiempo. Se hacen pequeños retoques cuando se exportó a móvil.
User Story 26: Paso de nivel User Story 27: Pierde partida User Story 28: Terminar partida	2 horas	2 horas	100%	Estas User Stories se realizaron al mismo tiempo. Se reutilizaron elementos de la pantalla inicial
User Story 29: Distintos personajes.	10 horas	10 horas	100%	Aprovechando las funciones y eventos de animaciones de anteriores Users Stories se determinó priorizar esta tarea con el fin de dotar de mayor variedad a la aplicación
RNF6: Obtener un juego mantenible y estable	20 horas	20 horas	85%	El juego es estable y funciona bien en una smartphone de gama media, falta refactorizar el código para mejorar su mantenibilidad. En las pruebas finales se detecto un fallo ,un NPC ocupó la misma posición

		que un PC,(indicar que durante toda la semana de pruebas no ocurrió este error, por lo que su probabilidad es baja) que no se consiguió volver a reproducir y ante la falta de tiempo se tomo nota sobre ello para corregirlo en versiones posteriores

4.1.1.- Limitaciones y problemas

Como se indica en la tabla anterior hay algunas historias de usuario que no se han completado o tienen alguna limitación. Concretamente:

- [USER STORY 10] y [USER STORY 13]: Se descartan los comportamientos de buscar apollo y atacar al oponente que tenga más vitalidad para priorizar las pruebas y comprobar el buen fucionamiento del resto de comportamientos. Indicar que las historias de usuario referentes al modelo de comportamiento de los NPC se recogieron casi todas en dos modos de comportamiento, agresivo y conservador.

- [RNF6]: Este requisito no funcional es de especial importancia por lo tanto será tarea prioritaria para futuros desarrollos.

Los problemas encontrados en las distintas historias de usuario quedan recogidos para su posterior resolución:

- [ISSUE 1]: Cuando un personaje es eliminado en batalla, la posterior selección de PC se debe realizar pulsando dos veces.

- [ISSUE 2]: Durante una batalla un NPC se coloco en la misma posición que un PC, esto incumple las normas del juego, además de no permitir correctamente la selección del personaje. El problema es bastante complicado de reproducir (durante la semana de pruebas ocurrió solo una vez) por lo que se prioriza establecer los logs necesarios para que durante posteriores pruebas detectar por que está ocurriendo esto, probablemente se actualizaron a destiempo las zonas de acción del NPC.

4.1.2.- Métricas

A pesar de los contratiempo, el corto plazo de entrega y los problemas encontrados se han completado casi la totalidad de los requistos propuestos.

Gracias a que las historias de usuario estaban debidamente priorizadas, y al descarte de algunas de ellas, se han realizado las mas importantes.

Esfuerzo estimado	Esfuerzo realizado	Desviación	Requisitos completados
156 horas	201 horas	+1.77%	92.6%

4.1.3.- Lecciones aprendidas

- El uso de plantillas facilita la creación de menús y son elementos repetitivos, no se tuvo en cuenta a la hora de desarrollar lo cual implica unas pantalla con un peor diseño y una mayor inversión de tiempo.

- Hay que tener en cuenta el tiempo que se invierta en la fase de adaptación al móvil. Las mayoría de pruebas se realizan en Pc por la comodidad a la hora de manipular variables y aplicar la estrategia prueba y error.

- Hay que darle mayor importancia a las pruebas, a pesar de que se reservo una semana para realizar pruebas y que con la realización de cada historia de usuario se realizan pruebas unitarias y de integración con el resto de funcionalidades no han sido suficiente para obtener al 100% un producto estable y mantenible (aunque se ha alcanzado un gran grado de satisfacción)

- Establecer un plan de pruebas y priorizar para evitar que algunas pruebas se queden fuera por cualquier tipo de motivo.

- A la hora de adquirir Tiles y Sprites asegurarse que viene correctamente divididos.

- Las aplicaciones estilo top-down son las que más animaciones implican, tenerlo en cuenta para futuras estimaciones en aplicaciones de este estilo.

5.- Glosario

- GPU: Graphics Processing Unit o Unidad de procesamiento gráfico, es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central

- CPU: Central Processing Unit o Unidad Central de procesos, cumple la tarea de procesamiento de todas las funciones así como también de almacenamiento de la información. En nuestro caso también se usa para indicar al rival del jugador.

- RPG: Role Playing Game, Se trata de un juego que lleva a los participantes a asumir un rol o papel, interpretando a un personaje.

- RPG Táctico: Cuyas siglas son TRPG o SRPG se refiere ha juegos RPG donde tienen un enfoque similar los juegos tradicionales como el ajerez y juegos de guerra en la que cada ficha tiene un rol distinto.

- RF: Se refiere a un requisito funcional.

- RNF: Se refiere a un requisito no funcional.

- NPC: Non Player Character. En el mundo de los videojuegos, y en el de los juegos de rol del que proviene el término, son todos los personajes no controlados directamente por el jugador.

- PC: Player Character. Personaje controlado directamente por el jugador.

- Top-down: Vista de videojuegos en la que los personajes se ven desde arriba

6.- Bibliografía

[1] **Wikipedia (2020).** Fire Emblem
https://es.wikipedia.org/wiki/Fire_Emblem

[2] **Youtube (2020).** Gameplay FIRE Emble GBA
https://www.youtube.com/watch?v=ArGxu00WTdo

[5] **Unity Technologies** (2020). Unity 3D.
https://unity.com/

[4] **Unity Technologies** (2020). Cuphead.
https://unity.com/madewith/cuphead

[5] **PcWorld 2020**. Iphone vs Android: couta de mercado.
https://www.pcworld.es/articulos/smartphones/iphone-vs-android-cuota-de-mercado-3692825/

[6] **Graphicriver 2020.** Compra de Sprites y tiles.
https://graphicriver.net/

[7] **Wikibooks 2020**. Algoritmo Dijkstra
https://es.wikibooks.org/wiki/Algoritmia/Algoritmo_de_Dijkstra

[8] **Wikipedia 2020.** Algoritmo de Floyd-Warshall
https://es.wikipedia.org/wiki/Algoritmo_de_Floyd-Warshall

7.- Anexos

7.1.- Manual de usuario

7.1.1.- Introducción

Nkhalango, un precioso reino lleno de tierras y bosques verdes y fértiles se prepara para librar sus batallas más importantes contra hordas de malvados goblins y esqueletos sedientos de venganza.

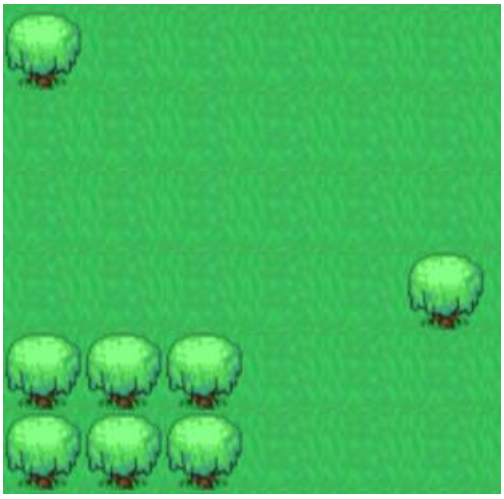
Dirige los ejércitos de este precioso reino en las múltiples batallas para que sus habitantes puedan vivir en paz y evitar la devastación del último reino libre de esta tierra mágica.

7.1.2.- Elementos del escenario

Nkhalango es un reino lleno de fertilidad y de gran vegetación, allí nos podemos encontrar dos tipos de zonas:

La hierba es el principal elemento que nos encotramos, donde se pueden ubicar los distintos personajes.

Los árboles impiden el movimiento de los personajes y no podemos ubicarnos en su misma posición.



7.1.3.- Aliados

El reino cuenta con dos tipos de guerreros:

Los caballeros cuya habilidad principal es que pueden recorrer mayor distancia y atacar cuerpo a cuerpo con su espada.

- Salud : 100
- Ataque: 20
- Rango Movimiento: 3
- Rango Acción: 1

Las magas, tienen mucho más poder de ataque con su lanza de fuego y pueden atacar cuerpo a cuerpo y a distancia.

- Salud : 80
- Ataque: 30
- Rango Movimiento: 2
- Rango Acción: 2

7.1.4.- Enemigos

Los enemigos del reino Nkhalango son de dos tipos:

Skeletons, usan la dureza de sus puños de huesos para atacar cuerpo a cuerpo, pueden recorrer mayores distancias.

- Salud : 100
- Ataque: 20
- Rango Movimiento: 3
- Rango Acción: 1

Goblins, gracias a su cetro usan mágia de fuego con gran poder a cortas y largas distancia pero su recorrido es menor.

- Salud : 80
- Ataque: 30
- Rango Movimiento: 2
- Rango Acción: 2

7.1.5.- Elementos

Cada personaje cuenta con un objeto de apollo que no puede usar sobre si mismo, es decir, sólo se puede usar sobre un personaje del mismo bando.

Poción: Se usa para sanar a un aliado, no se puede usar sobre uno mismo. Junto a cada personaje debe aparecer este icono para indicar que posee una de ellas.

Sobre cada personaje se puede visualizar la barra de salud, todas tienen el mismo tamaño pero no todos los personajes tienen el mismo valor de salud.

7.1.6.- Movimientos

Cada personaje puede realizar un movimiento, de forma obligatoria, por turno. Existen tres posibles movimientos:

<u>Movimiento simple</u>

Consiste únicamente en desplazar el personaje a una zona determinada dentro de su rango de movimiento.

1. Pulsar sobre el personaje deseado: Aparecerá su zona de movimiento (azúl) y su zona de acción (rojo), indicar también que se puede desplazar por la zona de acción. Si se pulsa sobre cualquier elemento que no este dentro las zonas establecidas se cancela la acción.

2. Pulsar sobre la zona deseada (marcada en amarillo en la imagen) para desplazar el personaje (recuerda que la zona de acción también es zona de movimiento siempre que no haya otro personaje en ella)

3. El personaje se desplazará a la zona indicada y mostrará la zona de acción. Llegados a este punto ya no se puede cancelar ninguna acción.

4. Pulsar sobre cualquier zona (incluida la zona de acción si no hay ningún personaje en ella) y el movimiento acaba dejando de mostrar la zona de acción.

Solo acción

Consiste en realizar una acción sin ningún movimiento.

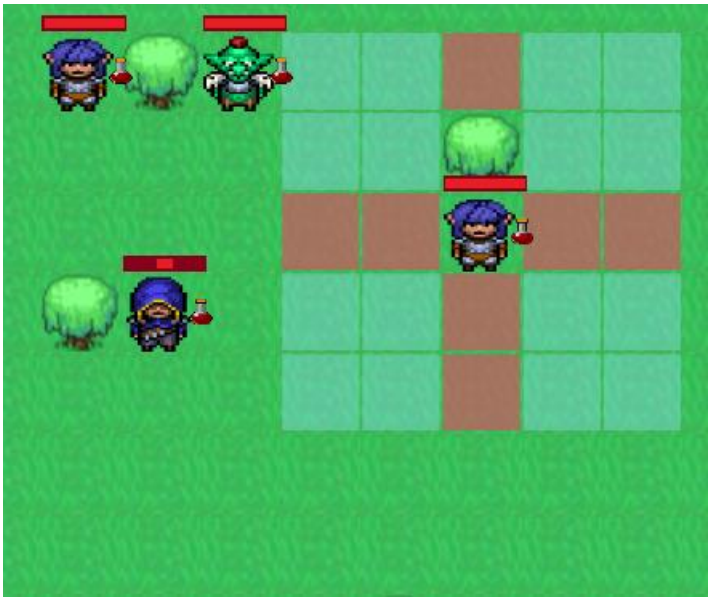
1. Pulsar sobre el personaje deseado: Aparecerá su zona de movimiento (azúl) y su zona de acción (rojo), indicar también que se puede desplazar por la zona de acción. Si se pulsa sobre cualquier elemento que no este dentro las zonas establecidas se cancela la acción.

2. Pulsar sobre un enemigo dentro de la zona de ataque, entonces se realizará un ataque y si el oponente aún le queda salud responderá con un ataque (siempre que este en su rango de ataque) y se dará el movimiento por acabado. Si en lugar de un enemigo, se selecciona un aliado en la zona de acción se realizará una curación (siempre que disponga de poción, si no es así se terminará el movimiento) y se dará por terminado el movimiento.

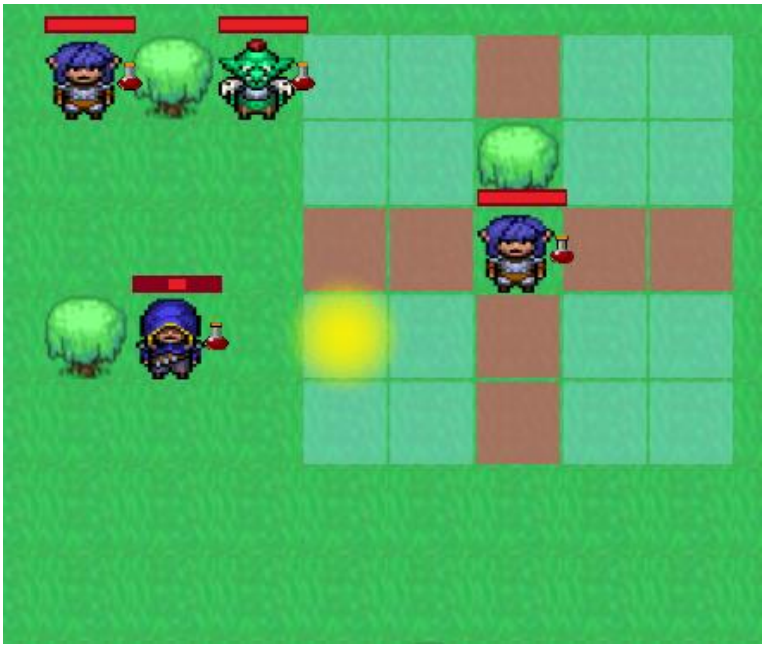
Movimiento y acción

Consiste en desplazarse a una zona concreta y realizar una acción, ya sea curar a un aliado o atacar a un enemigo

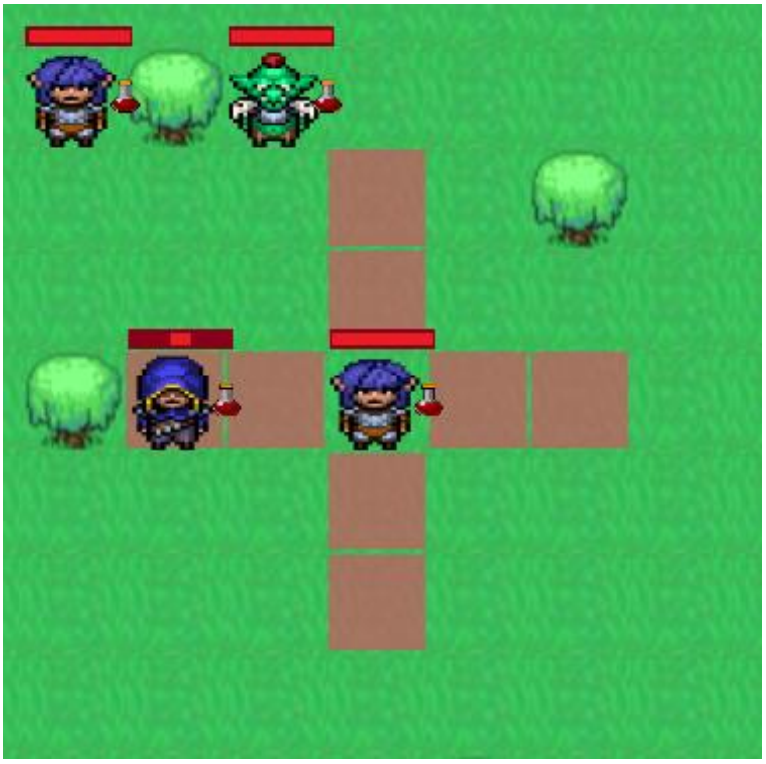
1. Pulsar sobre el personaje deseado: Aparecerá su zona de movimiento (azúl) y su zona de acción (rojo), indicar también que se puede desplazar por la zona de acción. Si se pulsa sobre cualquier elemento que no este dentro las zonas establecidas se cancela la acción.



2. Pulsar sobre la zona deseada (marcada en amarillo en la imagen) para desplazar el personaje (recuerda que la zona de acción también es zona de movimiento siempre que no haya otro personaje en ella)

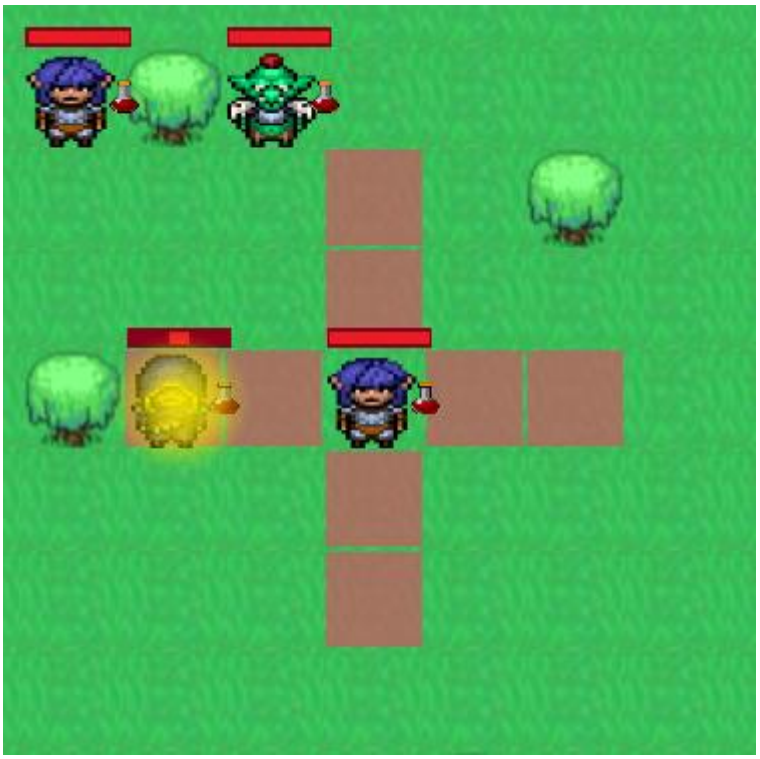


3. El personaje se desplazará a la zona indicada y mostrará la zona de acción. Llegados a este punto ya no se puede cancelar ninguna acción.



4. Pulsar sobre un enemigo dentro de la zona de ataque, entonces se realizará un ataque y si el oponente aún le queda salud responderá con un ataque (siempre que este en su rango de ataque) y se dará el movimiento por acabado. Si en lugar de un enemigo, se selecciona un aliado en la zona de acción se realizará una curación (siempre

que disponga de poción, si no es así se terminará el movimiento) y se dará por terminado el movimiento.



7.1.7.- Objetivos

El objetivo es sencillo, derrotar a todos los enemigos para pasar de nivel y poner a salvo el reino de Nkhalango superando todos los niveles.

7.2.- Manual de compilación

El proyecto se ha realizado con la herramienta Unity, concretamente la versión del editor es 2019.3.9.f1, la última versión disponible cuando se comenzó la implementación.

Para su compilación se debe descomprimir el fichero Tactics War.zip a continuación se accede al directorio ../Assets/Scenes y se abre la MainScreen, para desplazarse por las distintas escenas sólo debe seleccionarla a través Unity en la ruta mencionada. Para probar con el editor se debe pulsar el play (Se debe comenzar por la MainScreen para evitar fallos de de inicialización de otras escenas).

Si desea compilar el proyecto para Android solo se debe seleccionar File>BuildSetting, se selecciona Android y se pulsa sobre build con las opciones que vienen por defecto.

7.3.- Manual de instalación

Se debe copiar el fichero .apk en algún directorio de su terminal móvil, luego se accede a la ruta indicada y se pulsa sobre el icono con nombre TacticsWar.apk se debe indicar que se acepta instalar aplicaciones de desconocidos, una vez instalado tan solo se debe pulsar sobre el icono correspondiente.