Producción de un Videojuego Multijugador en Unity Combinando los Géneros MOBA y RTS

Ingeniería del Software y Diseño del Videojuego

Daniel Cuesta Boluda

Guillermo Cuesta Boluda

Javier Rodríguez-Osorio Jiménez

Facultad de Informática
UNIVERSIDAD COMPLUTENSE DE MADRID

Proyecto de Sistemas Informáticos

Madrid, Junio de 2014

Director: Prof. Dr. D. Fernando Rubio Diez

Co-director: Prof. Dr. D. Federico Peinado Gil

Autorización

Los abajo firmantes, matriculados en la asignatura Sistemas Informáticos de la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes de los autores, la documentación y/o el prototipo desarrollado durante el curso académico 2013-2014 bajo la dirección del Dr. Fernando Rubio Diez y la codirección del Dr. Federico Peinado Gil, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Daniel Cuesta Boluda

Guillermo Cuesta Boluda

Javier Rodríguez-Osorio Jiménez

Dedicatoria

Queremos dedicar este proyecto a todos aquellos ingenieros e ingenieras en informática que quisieron dedicarse a la producción y desarrollo de videojuegos y que actualmente, por causas de fuerza mayor, están trabajando en algo completamente diferente.

Resumen

A lo largo de los últimos años, la industria de los videojuegos ha experimentado un impresionante crecimiento, llegando su facturación a superar en 2008 a la de la industria del cine y la música juntas. España es además el cuarto país europeo con más consumo de este tipo de entretenimiento y el sexto a nivel mundial.

Existe un género del videojuego que ha tenido mucho éxito desde principios de la década de los 90. Se conoce como RTS (siglas del inglés *Real Time Strategy*) o juego de estrategia en tiempo real. De este género, nació a su vez otro que ha cobrado popularidad en los últimos años. Se denomina MOBA (siglas del inglés *Multiplayer Online Battle Arena*) o campo de batalla multijugador en línea.

El éxito del MOBA es fruto de un sistema de juego competitivo, muy directo y de acción continua, donde se controla un héroe que evoluciona en niveles y habilidades. Por el contrario el más tradicional RTS ha gozado de popularidad debido a la capacidad de gestionar, administrar y dirigir un ejército con numerosos tipos de unidades y edificios, dejando a voluntad del jugador la forma de hacerlo evolucionar.

El proyecto que se presenta aquí fusiona estos dos modos de juego en una misma obra, haciendo uso de una jugabilidad asimétrica, esto es, combinando las jugabilidades de ambos géneros, sin modificar sus mecánicas. Se trata de un proyecto muy ambicioso, para el cual se ha tenido que realizar un estudio previo del estado del arte para confirmar su viabilidad.

En cuanto a la organización, se ha dividido en distintas tareas clasificadas mediante una EDT (Estructura de Desarrollo de Trabajo) y para llevarlas a cabo se ha usado la metodología *scrum*, debido a que es la más utilizada a la hora de desarrollar videojuegos. Además, para estimar el tiempo de cada tarea se ha usado una planificación a largo plazo, así como un diagrama *Burndown Chart*.

Otro de los aspectos importantes ha sido la especificación y diseño del juego, de modo que la experiencia del juego sea lo más divertida posible. Esto implica cuidar mucho el diseño del mapa, así como el diseño del software para que sea lo más modular posible y admita todas las mecánicas en las que queremos que se involucre el jugador.

El equipo de trabajo ha estado formado por tres alumnos del Grado en Desarrollo de Videojuegos de ESNE, encargados de apartado artístico del videojuego, y seis alumnos de Ingeniería Informática de la Universidad Complutense de Madrid, responsables de llevar la producción, el diseño y la programación del mismo. Estos últimos están a su vez divididos en dos grupos de tres personas que se han dedicado a documentar las diferentes facetas del proyecto en dos memorias separadas. Esta memoria corresponde a los apartados de Ingeniería del Software y Diseño de Juego, mientras que para la otra memoria complementaria se dejan los aspectos tecnológicos y de implementación (memoria del mismo título con subtítulo *Tecnología e Implementación* (Gálvez Ruiz, Miranda Esteban, & Monasterio Martín)).

Palabras clave: Scrum, EDT, Burndown Chart, GDD, juego, online, UML.

Abstract

Over the last years, the video game industry has experienced a great growth, exceeding in 2008 the turnover over the sum of the cinema and music industry. Spain is the fourth country in consumption of electronic entertainment.

There is a genre of videogame that has been very successful since the beginning of decade of 90. It is known as RTS (Real Time Strategy), and this genre resulted in another genre that has gained much popularity in last years. It is called MOBA (Multuplayer Online Battle Arena).

The success of MOBA is the result of a competitive game, very direct and with continuous action, where the player controls a hero that evolves in levels and skills. The RTS genre has so succeed due to the ability of handling, managing and commanding an army with different kinds of units and buildings, letting the player advance in the match as he wants.

The project which presents here joins both game modes in a same videogame using an asymmetric playing. It combines, but does not mix both genres. It is an ambicious project, therefore a preliminary market study was necessary to make sure it is viable.

Refering to organization, the project split in different tasks classified in a WBS (Work Breakdown Structure) and to do that it has been necessary to use the scrum metholodogy (wich is very used in videogames development). Besides, to estimate the length of each task, it has been used a long planification and a *Burndown Chart*.

Other important features has been the specification and game design, because the game experience has to be as fun as possible. It has been important the map design and the software design to make a modular structure.

The work team has been integrated by three students of Videogame Design and Development at ESNE, who have worked as graphic designers and six students of Computer Science Faculty at Complutense University of Madrid, who have worked in the production, the design and the development of this project. This team of developers has been divided in two groups both integrated by three people, who have documented different parts of the project in two different handbooks, one of them is this one; another one is titled as *Tecnología e Implementación* (Gálvez Ruiz, Miranda Esteban, & Monasterio Martín).

Keywords: Scrum, WBS, Burndown Chart, GDD, game, online, UML.

Agradecimientos

Esta es una gran oportunidad para dar las gracias a todas aquellas personas que nos han apoyado y ayudado, dándonos todo el cariño necesario para poder realizar la carrera y el proyecto.

Primeramente, queremos dar las gracias a Federico Peinado Gil y a Fernando Rubio Diez, nuestros directores del proyecto, por habernos guiado este año. A Federico, que también ha sido nuestro profesor en *Laboratorio de Programación de Sistemas* y en *Ingeniería del Software* y ha sido tanto un profesor simpático y cercano como exigente y bueno para prepararnos lo máximo para la vida laboral. A Fernando, primero por haber accedido a un proyecto que no tenía pensado hacer y segundo por su enorme esfuerzo y dedicación en el proyecto y por habernos dado la oportunidad de aprender de él, ya que si como director de proyecto es impresionante, como persona es aún mejor.

Una ayuda que ha sido imprescindible para poder desarrollar este videojuego es la de Jacobo Estévez Arias, David Vortrefflich Pérez y Fernando Marco Figueroa, estudiantes del Grado en Diseño y Desarrollo de Videojuegos de ESNE, ya que desde el principio han trabajado muy duramente en la parte artística de este proyecto. También, queremos agradecer su ayuda al final del proyecto a Luis Álvarez Castañón, Luis Suárez Álvarez y Santiago Rico Bayón, estudiantes del Grado en Diseño y Desarrollo de Videojuegos de ESNE.

También, por supuesto, queremos dar las gracias a nuestros padres, que nos han ayudado desde pequeñitos a formarnos como personas y nos han motivado y, sobre todo, dado la oportunidad de cursar esta carrera. También por haber sido nuestro apoyo en los momentos más difíciles y estresantes, ya no sólo de la vida, sino de la carrera: gracias a ellos hemos podido finalizar una carrera tan exigente como es Ingeniería en Informática.

Además, damos las gracias a nuestras parejas por habernos ayudado en los momentos más complicados de la carrera y habernos soportado en momentos tan decisivos y desquiciantes como son los exámenes de esta complicada carrera universitaria.

Para finalizar, queremos dar las gracias a la Facultad de Informática de la Universidad Complutense de Madrid, a nuestros profesores y compañeros de carrera por habernos acompañado curso tras curso; y a los trabajadores de la facultad con los que hemos compartido lugar de trabajo durante estos años, especialmente a los trabajadores de la biblioteca.

Índice General

Índice de Figuras

1. Introducción

a. Motivaciones

Como se ha comentado en el resumen de la presente memoria, en los últimos años ha habido un gran crecimiento en el número de videojuegos desarrollados de forma independiente, es decir, sin la financiación de grandes editores o *publishers*.

Un buen ejemplo de este interés creciente, se vive en la Facultad de Informática de la Universidad Complutense de Madrid, donde cada vez más estudiantes se aventuran a realizar videojuegos para las asignaturas de prácticas de programación, o de *Ingeniería del Software* y, cómo no, como proyecto de *Sistemas Informáticos* suponiendo un colofón al final de una carrera tan exigente como es Ingeniería en Informática.

Un ejemplo de este tipo de estudiantes somos los que hemos realizado el presente proyecto, cuyo germen se gestó durante el pasado año gracias al haber participado todos juntos en el mismo grupo en la asignatura de *Ingeniería del Software*, desarrollando, con la ayuda de los profesores Dr. Federico Peinado Gil y Dr. Samer Hassan, un videojuego matamarcianos de nombre *Project_Chandra*[1] con la tecnología XNA de Microsoft. Según avanzaba el proyecto de esta asignatura, teníamos la intención cada vez más fuerte de realizar un videojuego en 3D como proyecto fin de carrera. Esta intención fue rápidamente bien recibida por D. Federico Peinado, sin embargo, debido a la ambición con la que se quería dotar a este proyecto, se pensaba que un grupo de tres personas se antojaba escaso, por lo que se intentó durante todo el verano conseguir el apoyo de un segundo director de proyecto, y así D. Fernando Rubio Diez ofreció su visto bueno y comenzó el desarrollo de este proyecto combinado por dos grupos de tres personas.

Aunque la mayoría de las industrias, incluida la del videojuego, se ha visto afectada por una situación de recesión, esta última, a la que nos acercamos, muestra un crecimiento significativo y la mayoría de las compañías siguen mostrando beneficios y buenos datos sobre ventas en el mercado. Esto es, en parte, gracias al aumento de nuevas oportunidades de negocio debido a la irrupción de modelos de negocio como el *freeToPlay*, en el que se ofrece el juego sin coste para el jugador y se consigue recaudar gran cantidad de dinero gracias, tanto a la venta de complementos para el propio juego, como a la gran aceptación de público que acaban consiguiendo. Dichos datos son un aliciente para nuestra decisión de afrontar el reto de desarrollar un videojuego con una futurible visión comercial, ya que tomando como ejemplo un juego de este tipo, *League of Legends*, las pretensiones y motivaciones son bastante ilusionantes.

El juego mencionado en el párrafo anterior se trata de un MOBA, y sirve como ejemplo perfecto de que un videojuego con monetización tipo *free-to-play* puede tener un gran éxito comercial logrando atraer una comunidad muy numerosa, tanto de jugadores habituales como otros más ocasionales, gracias a una jugabilidad que destaca por su sencillez inicial y por su exigencia para alcanzar niveles superiores de dificultad. Por supuesto, gracias al haber formado

[1] http://projectchandra.co.nf/

una comunidad tan amplia, permite obtener grandes beneficios de forma distinta a la forma clásica de venta de juegos.

Más allá de este hecho, por afición, gusto o atracción a nivel personal o grupal, esta industria del videojuego nos aporta suficiente motivación para pretender sumergirnos en ella.

La decisión del modo de juego derivó de nuestros propios gustos y del conocimiento actual de esta industria. A nivel personal y grupal la idea de mezclar dos modos de juego tan atractivos, ya no sólo para nosotros, sino para la comunidad *gamer* en general, fue una de las mayores motivaciones, ya que a día de hoy la unión de jugadores desempeñando las labores de cada modo es algo inexistente, calificando este proyecto de novedoso y único debido a la idea original que materializa.

b. Interés

La forma de plantear y visualizar el proyecto no tiene un punto final en el momento de finalización del curso. Desde el principio se estableció un sistema de *Ingeniería del Software* con el fin de poder continuar con la evolución del proyecto tras la finalización del curso. Por esto se desarrolló una *Gestión de Proyecto* eficiente y que va más allá de la idea de proyecto de fin de carrera, centrándose en potenciar un desarrollo eficaz y competitivo en la industria española para tener planificado un rumbo y un progreso que permita crecer dentro de esta industria.

Esta idea ha ejercido de influencia a la hora de tomar decisiones de configuración de la estructura de trabajo y ha producido un aumento en el interés de adentrarse en la industria del videojuego. Además ha resultado de impulso el hecho de que el Ministerio de Industria, Energía y Turismo publicó recientemente el lanzamiento de ayudas al sector del videojuego y adelantaron que se abriría la convocatoria de ayudas. Este es el resultado del cumplimiento de uno de los compromisos recogidos en el Plan de Impulso de la Economía Digital y los Contenidos Digitales integrado dentro de la Agenda Digital para España aprobada por el Consejo de Ministros en febrero de 2013. Así mismo, se espera que el resultado final satisfaga las necesidades del sector y conseguir el mayor impacto de las ayudas.

Gracias a todo esto y al proyecto innovador en el que nos encontramos, vemos la posibilidad de buscar el hueco dentro de la industria que nos permita cumplir con los objetivos establecidos.

c. Objetivos

Decidido el estilo del videojuego a realizar durante el proyecto, el siguiente paso fue fijar los objetivos a cumplir durante el proyecto. Principalmente se fijó tener a final de curso un nivel jugable entre cuatro jugadores, divididos en dos equipos, y comprobar el resultado de la unión innovadora entre los modos MOBA y RTS.

Además, no sólo se planteó como objetivo poder combatir entre los jugadores, sino también, y gracias a la disponibilidad inusual de contar con gente de distintos ámbitos, dar a los jugadores variedad en la elección de juego: héroes, ejércitos, etc.

Pero los objetivos no acaban ahí, el hecho de ser un proyecto ambicioso, especial y con una idea novedosa que lo convierte en un videojuego único al mezclar los modos MOBA y RTS, nos marcan el objetivo de evolucionar el juego de forma correcta para convertirlo, en lo posible, en un videojuego con una gran jugabilidad y con el don de enganchar a jugadores con facilidad.

d. Composición de Grupos de Trabajo

Para poder realizar un videojuego de semejantes características hace falta un grupo de personas muy numeroso y entre las cuales tiene que haber gente dedicada al apartado artístico; por esto, durante el proceso de desarrollo han trabajado dos grupos de dos universidades diferentes, uno del Grado en Desarrollo de Videojuegos de ESNE encargado del Arte del videojuego, y otro de Ingeniería en Informática de la Facultad de Informática de la Universidad Complutense de Madrid, encargado de la implementación del videojuego. La idea nació de los componentes de la Universidad Complutense de Madrid y ambos grupos han tenido tareas comunes como la producción y el diseño, habiendo una persona, Maximiliano Miranda Esteban, que ha sido el nexo de unión de estos grupos ya que pertenece a ambas universidades.

Finalmente, el grupo de desarrollo de la Universidad Complutense ha estado integrado por seis alumnos de Sistemas Informáticos. Sin embargo, a la hora de documentar el proyecto se estableció desde el principio una división en dos grupos de tres personas. Por un lado, un grupo ha estado documentando la presente memoria y el otro grupo una memoria sobre *Tecnología e Implementación* (Gálvez Ruiz, Miranda Esteban, & Monasterio Martín)**.**

2. Estado del Arte

Al comienzo del presente proyecto de Sistemas Informáticos se discutió intensamente el tipo de videojuego que se deseaba desarrollar, y se barajaron múltiples géneros y alternativas. Al final se decidió realizar un videojuego *RTS*, sin embargo, se entendía muy interesante la posibilidad de mezclar este género, muy longevo y con multitud de títulos lanzados, con un videojuego *MOBA*, género mucho más joven con una gran proyección en los últimos años. Fruto de esta decisión, se realizó un estudio del estado del arte sobre los videojuegos de ambos géneros, y el resultado se expone a continuación.

a. Los videojuegos RTS y MOBA

RTS

Los videojuegos de estrategia en tiempo real o *RTS* (siglas en inglés de *Real-Time Strategy*) son videojuegos de estrategia en los que todos los jugadores se desenvuelven de forma continua y no por turnos, siendo así uno de los tipos de videojuegos de estrategia más dinámicos.

Los *RTS* se centran en dos pilares fundamentales: la recolección y gestión de recursos y la acción militar. Así, las principales características de este subgénero son la variedad y utilidades de los recursos y la funcionalidad y características de las unidades y construcciones. Se desarrollan multitud de batallas con un gran número de unidades en las que dicha recolección se convierte en algo fundamental. Todo esto hace del *RTS* uno de los subgéneros más populares y extendidos gracias a su dinamismo.

Se suele trabajar mucho el apartado gráfico debido a que el tamaño de los terrenos de juego se ve disminuido con respecto a otros subgéneros, haciéndose muy importante el diseño y moldeado del terreno y la cámara utilizada con sus distintas perspectivas (algunos juegos permiten un mayor nivel de zoom, e incluso una rotación y movimientos más libres).

Otra característica de los juegos *RTS* son los distintos modos de juego, entre ellos los más comunes son campaña, multijugador y mapa aleatorio, aunque dependiendo del juego en cuestión pueden variar bastante.

MOBA

Los *MOBA* o *Multiplayer Online Battle Arena* nacieron como subgénero de los *RTS*, tomando algunos elementos de estos (como el control y la cámara), simplificando conceptos (como la gestión de recursos y de ejércitos) y potenciando otros (como elementos de evolución de personajes o uso de habilidades y magias). Sin embargo, llegaron a tomar entidad como género independiente debido principalmente a la rápida evolución que han ido sufriendo potenciada por su fuerte popularidad desde sus inicios.

En este género, como suele ser habitual en las partidas multijugador de los *RTS*, dos equipos de jugadores compiten entre sí en un escenario. Sin embargo, aquí aparece el

principal cambio introducido en los juegos *RTS*: el manejo de una única unidad y la carencia de construcción de unidades y edificios. Esta unidad posee varias habilidades convirtiéndose en una unidad de un nivel muy por encima de las generales de los juegos *RTS*. Con la unión de estas unidades en un equipo hace que este subgénero se centre en el juego cooperativo.

El objetivo también difiere de los juegos *RTS* convencionales y consiste en destruir la estructura principal de los oponentes con la ayuda de unidades controladas por una I.A. generadas periódicamente que marchan hacia la estructura principal del enemigo a través de calles distribuidas en el mapa.

Estudio de mercado

Hay una gran cantidad de juegos que han ido caracterizando los géneros *MOBA* y *RTS*, así como otros han ido adaptándose a estos cambios. Entre todos estos cabe la pena destacar los siguientes, ya sea por su popularidad o por su originalidad e influencia en otros juegos:

- League Of Legends
- Prime world
- Bloodline champions
- Airmech
- Super MNC (Monday Night Combat)
- Smite
- Smashmuck Champions
- Dota 2
- Starcraft 2
- Warcraft III
- Saga Age of Empires

A continuación se exponen los *gameplays* (o *jugabilidad*) de dichos juegos, y veremos la representación de los subgéneros *RTS* y *MOBA* en cada una de las características de estos juegos.

- **League Of Legends** [2]

  Juego al más puro estilo *MOBA* donde los personajes se controlan en modo *RTS 5 vs 5*, en el que los jugadores están divididos en dos equipos y luchan por destruir la base enemiga. Aparecen todas las características de dicho modo de juego: recolección de dinero, gama de cuatro habilidades (específicas para cada campeón), subidas de nivel, torretas defensivas y oleadas de *minions*. Existe una tienda con objetos que potencian las características de los personajes y los campeones cumplen distintos roles (apoyo, combate, distancia, mago, tanque).

  Los modos de juego vienen especificados con su propio escenario. Un primer modo Clásico de calles principales con torres para defender y *spawn* de *minions* que las recorren y consistiría en escenarios de *3 vs 3* (dos calles) o *5 vs 5* (tres calles). Un segundo modo denominado *Dominion* de puntos de control con una torreta defendiendo cada uno de los cinco puntos a capturar y *spawn* de *minions* que recorren el terreno comprendido entre los puntos capturados y el siguiente punto adyacente a

[2] http://leagueoflegends.com/

capturar. Y por último, *Aram*, en el cual la selección de los campeones es aleatoria, hay una única calle principal y *spawn* de *minions* que la recorren.

- **Prime World** [3]

Juego al más puro estilo *MOBA*, los personajes se controlan en modo *RTS 5 vs 5*, y se puede potenciar las características de los personajes comprando objetos en la tienda. Los campeones cumplen distintos roles (apoyo, combate, distancia, mago). Tiene 3 calles principales con torres para defender y oleadas o *spawn* de *minions* que las recorren.

Los personajes pueden jugar a un minijuego en su base que les da pergaminos con poderes a los aliados en combate. Este minijuego tiene un *gameplay* del estilo puzle *Zuma*, cuanto más difícil es el nivel mayor es la mejora que reciben los compañeros por el pergamino.

- **Bloodline Champions** [4]

Es un juego desarrollado en *XNA*, tipo *MOBA* basado en Arenas en el que no hay *minions*, el *gameplay* se basa en la lucha entre jugadores y tiene la más amplia gama de habilidades, hasta 7, sean habilidades físicas o mágicas. En los *MOBA* convencionales suele haber 4, y sea cual sea la amplitud de la gama, cada campeón o héroe tiene su propia gama de habilidades. El juego no hace tanto hincapié en la compra y la evolución de los personajes sino en la acción de la lucha entre ellos.

En este juego, los jugadores están divididos en dos equipos (Caliente y Frío) de hasta cinco jugadores por equipo. El objetivo es diferente dependiendo del modo de juego, desde *el último que quede en pie* hasta *capturar la bandera* o *controlar el punto del mapa*.

- **Airmech** [5]

Combina el elemento de creación del ejército del que carecen los *MOBA*. El jugador controla un héroe o campeón con dos posibles formas: aérea o terrestre. Desde ambas se puede disparar, pero solo a otras unidades en la misma posición, es decir, en posición terrestre solo se puede disparar a unidades enemigas terrestres.

Para la creación del ejército, se escoge qué unidades se van creando alimentándose en una cola, creándose en una fábrica de una en una, y hay que recogerlas y llevarlas al lugar donde se quieran descargar, y esto solo puede hacerse en forma aérea, ya que las unidades creadas aparecen en la azotea de la fábrica. A partir de ahí el movimiento de las unidades es automático (IA) y lineal por el mapa en dirección a la base enemiga. Para finalizar el juego es necesario controlar todas las fábricas o eliminar al enemigo por completo.

Las unidades cuestan cierto dinero, que va consiguiéndose progresivamente durante el juego, pero a más fábricas bajo control, mayor será el recurso que vaya obteniéndose.

[3] http://en.playpw.com/
[4] http://www.bloodlinechampions.com/
[5] https://carbongames.com/

- **Super MNC (Monday Night Combat)** [6]

  Es un juego de disparos en tercera persona, con controles *WASD*, con distintas clases en las que el objetivo es destruir la base enemiga. Sin embargo, el juego tiene más rasgos de juegos de estrategia en tiempo real, como los *MOBA*, que de los juegos *shooter*.

  En *Super Monday Night Combat*, dos equipos de 5 jugadores luchan por el objetivo del juego, que es destruir la base enemiga. Al igual que en otros juegos *MOBA*, cada equipo tiene oleadas de *bots* aliados que avanzan destruyendo, poco a poco, las torretas enemigas y, llegado el caso, el objetivo final de la base enemiga. Matar a los *bots* y jugadores enemigos proporciona dinero, el cual puede gastarse comprando potenciadores o *bots* adicionales.

- **Smite** [7]

  Juego de mecánica *MOBA* donde los personajes se controlan en tercera persona, tienen tienda con objetos que potencian las características de los personajes y los campeones cumplen distintos roles (apoyo, combate, distancia, mago, tanque).

  Los controles para moverse son *WASD*. Los modos de juego se dividen en dos tipos: el primero, una Arena donde se trata de conseguir que cierto número de *minions* lleguen a la base contraria antes que el otro equipo, y un segundo modo, al estilo de calles con torretas como en otros *MOBA* donde cada equipo tiene oleadas de *bots* aliados que avanzan destruyendo, poco a poco, las torretas enemigas y, llegado el caso, el objetivo final de la base enemiga. Los escenarios pueden tener una, dos o tres calles. Matar a los *bots* y jugadores enemigos proporciona dinero, el cual puede gastarse comprando objetos.

- **Smashmuck Champions** [8]

  El juego consiste en batallas en las que dos bandos luchan por la supremacía en cinco modos de juego: *Gauntlet* (modo cooperativo), *Plunder Ball* (un modo parecido al típico captura la bandera), *Siege*, (un modo de asalto/*Tower defense*), *Conquest* (modo en el que se deben capturar puntos estratégicos del mapa) o *Destroyer* (un modo de recolección en el que hay que enfrentarse a un enorme robot gigante).

  Los controles para moverse son *WASD* y se poseen habilidades que se seleccionan con las teclas numéricas y se activan con el botón secundario del ratón, teniendo un ataque simple con el botón primario del ratón. Este manejo y características del personaje es representativo de los *MOBA*, notándose diferencia en los sistemas de juego y los terrenos.

- **Dota 2** [9]

  Un juego *MOBA* puro, con muchas similitudes con el *League of Legends* a nivel característico de personajes, escenarios y sistema de juego. Los personajes se controlan

[6] http://www.uberent.com/smnc/
[7] http://www.hirezstudios.com/smitegame/
[8] https://www.smashmuck.com/
[9] http://es.dota2.com/

en modo *RTS 5 vs 5*, tienen tienda con objetos que potencian las características de los personajes, los campeones cumplen distintos roles (apoyo, combate, distancia, mago). El mapa tiene 3 calles principales con torres para defender y *spawn* de *minions* que las recorren. Matar a los *minions* y jugadores enemigos proporciona dinero, el cual puede gastarse comprando objetos.

Un detalle curioso son los tutoriales, que se desarrollan en un mapa tipo campaña en el que deben alcanzarse unos objetivos mientras se va explicando el manejo y la utilidad de los objetos fundamentales. Este mapa solo puede ser jugado en el tutorial.

- **StarCraft 2** [10]

Es un juego estilo *RTS* que se caracteriza por tener tres razas totalmente diferentes y equilibradas. Hay dos tipos de recursos diferentes que se pueden recolectar, que son los minerales y el gas vespeno. Con el primero se pueden construir unidades sencillas, y con el segundo se tiene acceso a unidades avanzadas y a evoluciones de las unidades.

Otra de las características más llamativas de este juego es que con la unidad más cara y fuerte no siempre se gana la partida, ya que siempre hay unidades más baratas que las contrarrestan de forma efectiva.

Por último cabe destacar que, cuando se juega por equipos, los recursos se pueden compartir en el modo multijugador, así como dejar el manejo de las unidades al resto de aliados.

- **Warcraft III** [11]

Es un juego estilo *RTS* que se caracteriza por tener cuatro razas totalmente diferentes y equilibradas. Hay dos tipos de recursos diferentes que se pueden recolectar, que son el oro y la madera. Con ambos recursos se pueden crear cualquier edificio o unidad, en algunos habrá una mayor cantidad requerida de un recurso, y en otros, al revés. También existen unidades que sólo necesitan de un recurso.

Otra de las características más llamativas de este juego es que existe una unidad (héroe) que es capaz de subir niveles y que se complemente con un ejército de unidades más baratas. Esta unidad especial no es capaz de ganar la partida ella sola.

Por último cabe destacar que existen tesoros esparcidos por el mapa que otorgan mayor capacidad militar a la unidad especial o héroe.

- **Saga Age Of Empires** [12]

Es un juego fiel al estilo *RTS*, caracterizado por la variedad de civilizaciones a elegir y sus modos de juego. Además de los modos comunes (modo campaña, mapa aleatorio y multijugador), en ciertos juegos de la saga se han podido ver otros modos: combate total, partidas por tiempo o por obtención de algún objetivo.

[10] http://eu.battle.net/sc2/es/
[11] http://us.blizzard.com/es-mx/games/war3/?locale=es-es
[12] http://www.ageofempires.com/

Refiriéndonos a las características fundamentales de los juegos *RTS*, respecto a la recolección de recursos cabe destacar el número de materias primas distintas, de hasta cuatro, y la diferencia de unidades o edificios a construir dependiendo de la civilización en cuestión.

Siendo un juego por equipos, en el juego *Age of Mithology* se incorporó el compartir el límite de población, así como la posibilidad de elegir una mejora u otra al mejorar el nivel de la civilización.

En un juego posterior, *Age of Empires 3*, cabe destacar la aparición de una civilización neutral manejada de forma automática por el juego (IA) con la que poder interactuar comerciando o batallando.

b. Otros videojuegos similares

De entre todos los juegos mencionados y estudiados en el apartado anerior, los juegos con un estilo más cercano al juego que se desarrollará en este proyecto son: a nivel *RTS* *WarCraft III* y *StarCraft 2*, y a nivel *MOBA League of Legends* y *Smite*.

Similitudes RTS

A nivel *RTS* las similitudes son notables: el número de recursos es el mismo, dos tanto en nuestro proyecto (minerales y economía) como en el *WarCraft III* y *StarCraft 2*, cada uno con su influencia en el progreso, dependiendo de la dinámica del juego.

En los tres, existe una unidad única (héroe) que sube de nivel y gana en potencia militar, la diferencia es que esa unidad tanto en el *WarCraft III* como en el *StarCraft 2* es controlada por el mismo jugador que controla el ejército, mientras que en nuestro proyecto es controlado por otro jugador. Así mismo la capacidad militar del héroe es mucho mayor debido a habilidades especiales a su disposición.

El objetivo es el mismo, hay que destruir la base del equipo rival. En nuestro proyecto es la única construcción donde se pueden crear nuevas unidades (de cualquier tipo) para el ejército, pero en *WarCraft III* y *StarCraft 2* las unidades se crean en multitud de edificios, aunque las unidades recolectoras solo se pueden crear en la base principal.

Similitudes MOBA

En referencia a la parte *MOBA* del proyecto, la mayor similitud corresponde al uso de hechizos a disposición del jugador, siendo cuatro hechizos en *League of legends* y *Smite*, y tres en nuestro proyecto, aunque la dinámica de su uso es similar, ya que incluso los tipos de hechizo y manejo son compartidos. Este manejo es compartido sobre todo con *Smite*, ya que poseen una cámara similar (en *League of Legends* la cámara es mucho más perpendicular sin llegar a la horizontalidad, pero acercándose más a una vista propia del *RTS*) y la forma de acceder a los hechizos mediante número es la misma.

En nuestro proyecto el héroe se podrá desenvolver por el mapa al principio de la partida para conquistar torretas neutrales junto a ingenieros, o para buscar tesoros. En *League of Legends* y *Smite* se pueden conseguir ciertos potenciales temporales luchando contra

monstruos neutrales controlados por una I.A., que aunque la temática y funcionalidad varía, en ambos casos, profiere una mejora al héroe.

Por último, otros juegos estudiados con los que posee ciertas similitudes son *Dota 2*, muy parecido a *League of Legends* en cuanto a cámara y mecánicas de juego y la saga de juegos *Age of Empires*, que son juegos completamente estilo *RTS*, sin unidades especiales como son los héroes, pero con el mismo objetivo.

Conclusiones

Como se puede apreciar, el parecido con estos juegos se limita a tan solo una parte de lo que se propone con este proyecto (similitudes en la parte *RTS* y similitudes en la parte *MOBA*). Sin embargo, no existe ningún otro juego en el que se combinen los dos géneros de una manera similar al que lo hace el de este proyecto. Esto es, combinando los dos géneros pero sin fusionarlos en una única jugabilidad, sino manteniendo las características de cada una y permitiendo elegir qué jugabilidad se acerca más a los gustos del jugador.

Por esto se considera importante remarcar, de cara a la originalidad, la propuesta ofrecida de juntar en un juego dos jugabilidades diferentes en un ambiente competitivo y cooperativo multijugador, de una manera nunca antes vista en un videojuego.

c. Los motores de videojuegos en la actualidad

Al inicio del desarrollo del proyecto se exploraron múltiples alternativas para desarrollar el videojuego. Fruto de esta exploración, se estudiaron los motores de videojuegos más utilizados en la actualidad. Así, en este apartado se mencionan algunas de las herramientas más utilizadas para el desarrollo de videojuegos, herramientas que en la actualidad están a la vanguardia de la creación y desarrollo de videojuegos.

UDK4 (Unreal Engine 4)

Unreal Engine[13] es un motor de videojuegos creado por la compañía *Epic Games* para el videojuego *Unreal* (tipo *shooter* en primera persona). En principio, fue desarrollado para videojuegos de este género, sin embargo ha sido utilizado en videojuegos que cubren una mayor variedad de géneros. En la actualidad, y durante los últimos 8-10 años, ha sido uno de los motores más utilizados en la industria del ocio electrónico, gracias a su versatilidad a la hora de portar a múltiples plataformas, y a su potente motor gráfico. Algunos videojuegos o sagas que han utilizado este motor son: *Unreal Tournament*, *Turok*, *Gears of War*, *BioShock*, *Mass Effect*, o la saga *Batman Arkham*.

Está escrito en C++ y es compatible con la mayoría de las plataformas como PC (*Microsoft Windows, GNU/Linux*), *Apple Macintosh* (*Mac OS, Mac OS X*), para la mayoría de consolas que se han desarrollado (en orden cronológico de aparición: *Dreamcast, PlayStation 2, Nintendo Gamecube, Xbox, Wii, Xbox 360, PlayStation 3, WiiU, Xbox ONE, PlayStation 4*), e incluso para plataformas portátiles (*PlayStation Vita*) y dispositivos móviles (*Android* e *iOS*).

[13] http://www.unrealengine.com/

El lenguaje de programación principal de *Unreal Engine* (hasta la versión 4) es el *Unreal script*, similar a *Java*, orientado a objetos y con un fuerte acoplamiento de herencia. Aunque también se permite bajar a un menor nivel y programar dentro del propio motor en C++.

Unreal Engine ofrece múltiples herramientas adicionales de gran ayuda para diseñadores y artistas como un profundo editor de materiales, un completo y potente editor de escena y un motor de *scripting* visual por medio de cajas y flechas llamado *Kismet* (hasta la versión 4).

La última versión (*UDK4*), supone un fuerte cambio respecto a la línea anterior de versiones, desechando por completo *Unreal Script* y *Kismet* y potenciando mucho la capacidad técnica del motor gráfico. Ofrece reflejos de escena completa, multitud de luces dinámicas por escena, uso de sombras con física, mejora en la eficacia de la simulación de partículas incluyendo un sistema de colisiones y la posibilidad de que las partículas puedan recibir y emitir luz dentro de una escena. Así mismo, también ofrece distintas herramientas como *Blueprint scripting visual* (que vendría a sustituir a *Kismet*), que permite crear prototipos y contenido jugable sin tocar una línea de código, y visualizar de forma interactiva el desarrollo del juego e inspeccionar los valores de las propiedades, mientras se realizan pruebas del juego.

Otra característica muy interesante de *UDK4* es *Hot Reload*, que permite hacer cambios en la jugabilidad en tiempo de ejecución, es decir, editar el código C++ y ver esos cambios reflejados inmediatamente en el juego sin tener que parar y recompilar el código del juego. Por último ofrece personajes controlados por la IA, de mayor consciencia espacial del mundo que les rodea y les permite realizar movimientos inteligentes, así como una malla de navegación dinámica que se genera en tiempo real a medida que un objeto está en movimiento por la escena.

Uno de los principales cambios introducidos por *Epic* en *Unreal Engine 4* es su modelo de pago. Anteriormente se ofrecían diferentes modelos con un alto porcentaje por royalties más un alto mínimo inicial, quedando fuera del alcance de desarrolladores pequeños o *Indies*. Ahora se ofrece bajo un modelo de subscripción por 19$/€ al mes más un 5% de royalties por ventas.

CryENGINE

CryENGINE[14] es un motor de videojuegos creado por *Crytek*, una empresa alemana desarrolladora de software. Su primera aparición fue en la implementación del juego *FarCry*, un *shooter* en primera persona desarrollado por esta misma empresa. La versión que ha gozado de mayor popularidad ha sido *CryEngine3*, y después de esta (desde agosto de 2013) se desarrolló la cuarta generación del motor, a partir de la cual se pierde la nomenclatura con el número de versión.

Es multiplataforma y ofrece soporte para *Microsoft Windows*, *Linux*, *PlayStation 3*, *Xbox 360, WiiU, PlayStation 4, Xbox ONE* y dispositivos móviles (*Android* e *iOS*). Algunos videojuegos desarrollados con *CRYENGINE* son: la saga *Crysis* (de la propia *Crytek*), *Evolve* o *Ryse: Son of Rome*.

[14] http://cryengine.com/

La programación en *CRYENGINE* se realiza de 3 maneras alternativas y complementarias, por un lado, a nivel de *scripting* de la lógica de la jugabilidad, el editor del motor cuenta con una herramienta similar al *Kismet* de *UDK* llamada *Flow Graph editor*, y también se ofrece la programación de *scripting* con *LUA*. Por otro lado, el motor está escrito en C++ y se ofrece la posibilidad de programar heredando las clases del propio motor.

CRYENGINE hace alarde de una imponente capacidad técnica y de un potente editor. Visualmente, presume de ser el *high-end renderer* (motor de renderizado de alto acabado, de no tiempo real) más rápido del mundo. Cuenta con una gestión de personajes (animaciones faciales, IA, etc.) muy avanzada, un motor físico con soporte multi-núcleo integrado de gran versatilidad y potencia, iluminación dinámica en tiempo real, suavizado de sombras, niebla volumétrica, etc. y todo esto con un gran rendimiento.

Su editor ofrece un sistema de partículas en tiempo real, herramientas para la creación de ríos y terreno con múltiples opciones (conocida como *Sandbox*), sistema de animación de personajes y editor de animación facial, así como un editor de IA, mallas de navegación dinámicas, gestión de entornos destructibles, físicas deformables, herramientas de análisis de rendimiento, sonidos y música dinámicos, audio envolvente y renderización de texturas de alta velocidad.

Como novedad en la última iteración del motor, se ofrece desde el año 2014 un sistema de distribución *as a service* de 9,90€/$ al mes, mostrándose así mucho más al alcance de pequeños desarrolladores *Indie*.

Unity

Unity[15] es la herramienta de creación de videojuegos más utilizada en la actualidad. Cuenta con un excelente editor, con multitud de opciones pero muy amigable, lo que facilita mucho la entrada a la creación de videojuegos con este motor a gente sin experiencia previa. Es totalmente multiplataforma (*PC, MAC, Linux,* navegadores *Web, iOS, Android, Blackberry, Windows Phone, Xbox 360, PlayStation 3, Wii, Xbox ONE, PlayStation 4*, *PlayStation Vita* y *WiiU*), permitiendo exportar un proyecto para cualquiera de estas plataformas de manera sencilla y directa (e incluso seleccionando un botón).

La programación en Unity funciona bajo el *framework* .NET de Microsoft y permite programar tanto en JavaScript como en C# o Boo, de forma independiente (los *scripts* del juego pueden estar en cualquiera de los 3 lenguajes). Además utiliza un modelo de programación por componentes, lo que simplifica mucho el paso de mensajes entre objetos de distintas clases y ofrece mucha versatilidad a la hora de crear objetos diferentes pero que compartan algunos scripts en el juego.

En el plano técnico, Unity está un peldaño por detrás de sus más directos competidores (*Unreal Engine* y *CryENGINE*), pero eso no significa que no sea potente o que contenga muchas carencias. De hecho, Unity cuenta, entre otros, con un buen motor de iluminación global, un editor de terrenos (no tan potente como el de *CryENGINE*), el motor de físicas *PhysX* de *NVIDIA* integrado, un motor de audio *FMOD* o un sistema de partículas avanzado llamado *Shuriken Particle System*. Por último, Unity ofrece acceso directo al *Unity Asset Store*, una tienda de

[15] http://unity3d.com/

assets que cuenta con gran cantidad de material para usar en proyectos propios, algunos de pago pero también muchos de forma gratuita, como modelos 3D, sonidos, partículas, sprites para menús, etc. e incluso *plugins* que extienden el motor.

Otro de los puntos fuertes de Unity es que cuenta con una excelente y extensa documentación oficial[16], de fácil acceso y entendimiento y en constante actualización. Además, Unity cuenta con una de las mayores comunidades de usuarios del mundo, lo que facilita la búsqueda de ejemplos, tutoriales y la solución a casi cualquier problema que se plantee a lo largo del desarrollo del videojuego.

Conclusiones

Tras explorar todas estas alternativas, se toma la decisión de desarrollar el proyecto con Unity, por múltiples razones: el equipo conocía el lenguaje de programación C# y la plataforma *.NET* tras la experiencia con un proyecto anterior para la asignatura de Ingeniería del Software. Además, otra ventaja fundamental era la documentación oficial de la aplicación y la fuerte presencia de la comunidad de desarrolladores del motor en internet, haciendo muy accesible la búsqueda y resolución de problemas que pudieran surgir a lo largo del desarrollo. Por último también se valoró muy positivamente la fácil aproximación que ofrece el editor de Unity a desarrolladores sin experiencia previa en este tipo de aplicaciones.

Como consecuencia de esta elección, se ha elaborado una sección en la que se describe con mayor profundidad y detalle el motor y las herramientas que ofrece, en el apartado 4.a de la memoria **Producción de un Videojuego Multijugador en Unity Combinando los Géneros MOBA y RTS: Tecnología e Implemetación** (Gálvez Ruiz, Miranda Esteban, & Monasterio Martín).

[16] http://docs.unity3d.com/ScriptReference/index.html

3. Especificación y Plan de Proyecto

a. Objetivos y material

El objetivo principal del proyecto ha consistido en desarrollar un videojuego con las principales mecánicas diseñadas jugables en modo online por 4 personas de forma simultánea, en el plazo previsto. Este objetivo tiene su importancia debido al interés propio de cada uno de los miembros del proyecto, ya que una de las causas de la decisión de emprender un proyecto así es la afición por el mundo del videojuego.

Así mismo, tras la experiencia de haber llevado a cabo un juego en la asignatura *Ingeniería del Software*, se convirtió en un reto realizar un juego de mayor calibre, quedando claro como objetivo probarnos a nivel tanto personal como colectivo.

En cuanto al material a utilizar, además de disponer de Unity como herramienta principal de desarrollo, hemos utilizado diversas herramientas para comunicación, como son *Whatsapp*, *Hangout, Gmail* y *Google Groups*, y herramientas de contenido y planificación, como *Dropbox*, *Trello* y *Google Drive*.

Cada uno de los miembros del grupo ha desempeñado los roles de diseñador, desarrollador y programador. En total, además de los seis miembros de la facultad de informática, hemos tenido la colaboración de cinco diseñadores gráficos con los que hemos ido avanzando de forma paralela según las necesidades del proyecto.

En lo referente al método de desarrollo, se eligió el método Scrum, un modelo ágil que permite adaptarse a cambios según las necesidades puntuales. Este método se ha basado en la planificación de reuniones continuas para la asignación de tareas, pudiendo ser asignaciones individuales o por parejas dependiendo de la propia tarea.

En relación a estas tareas, se estructuró el trabajo con una *EDT* (Esquema de División de Trabajo), siendo un esquema muy útil para ver las partes del proyecto más amplias, primordiales o que más dedicación urgente necesitan.

El siguiente paso para tener un control en el avance de dichas tareas es llevar a cabo un seguimiento mediante Trello y Burndown Chart. Este último consiste en una actualización continua del tiempo dedicado a cada tarea tras una estimación de las mismas, contabilizando también el tiempo de cada una y el tiempo total de plazo de entrega del proyecto y de esta forma se puede comprobar si el nivel de dedicación puntual al proyecto permitirá llevar a cabo la planificación prevista.

Trello es un sistema de control de tareas que nos permite clasificarlas en estados a partir del momento en que hayan sido asignadas, es decir, incluirlas en secciones que indican si la tarea está en disposición de hacerse, haciéndose o terminada. Así mismo, se puede incluir un autor para la tarea y una fecha estimada de realización.

b. Organización

La organización es una de las claves fundamentales para el progreso de un proyecto. Las herramientas y materiales mencionados en el apartado anterior y el método de desarrollo forman la estructura básica para una buena organización durante el curso.

A continuación se definen y explican cada una de las partes que permiten un seguimiento progresivo y un control para garantizar un buen progreso del proyecto.

i. Scrum

Como se mencionó en el apartado anterior, la característica principal para la elección de la metodología de desarrollo Scrum era la fácil adaptación a cambios o necesidades, pero realmente posee otras características que se adaptan con gran exactitud a la forma de planificar el desarrollo de un videojuego.

Estas características son la planificación mediante *sprints*, reuniones que en nuestro caso se han planificado de forma quincenal para los seis miembros, pero que para los colaboradores que han aportado el diseño gráfico se han establecido de forma mensual. Cabe mencionar que uno de los desarrolladores del grupo siempre ha estado presente también en las reuniones de los colaboradores gráficos, facilitando así la coordinación entre ambos grupos para llavar de forma paralela los mismos aspectos y que la evolución se pudiera visualizar de forma conjunta.

Otra ventaja que aporta Scrum, es la división del trabajo en equipos mediante la asignación de distintas tareas y, dependiendo de ellas, establecer la numerosidad del grupo. Todo esto unido a las reuniones quincenales, ha permitido una fácil y constante comunicación verbal entre todos los miembros y un mayor control de las tareas asignadas.

Al comienzo de este apartado se ha comentado la fácil adaptación a cambios de esta metodología de desarrollo, que definiéndolo de una forma un poco más específica nos aporta una mejor adaptación a lo que se quiere o necesita, y que los problemas se puedan enfrentar con mayor predicción y planificación, pudiendo así responder a requisitos emergentes. Además, permite llevar una gestión regular que puede evitar riesgos, aumentar la productividad y calidad, y una gran colaboración entre el grupo o grupos de trabajo.

Todas estas características han permitido una colaboración entre estudiantes de dos universidades (*ESNE y UCM)* adecuada a los requerimientos puntuales del proyecto que han ido emergiendo durante el curso. Estos *sprints* quincenales o mensuales han permitido que lo que se ha ido desarrolando e implementando lo haya hecho de forma paralela al diseño artístico, y ha permitido visualizar y configurar una coherente adhesión a medida que el proyecto ha avanzado.

Con todo esto, se perciben de forma clara dos equipos, cada uno con su propia planificación, que gracias a la metodología de trabajo SCRUM han podido progresar a un ritmo similar durante el curso.

La planificación de ambos equipos, basada en sprints, se ha plasmado mediante actas de reuniones, en las que se ha anotado el día, asistentes, motivo, tareas realizadas, a realizar y asignadas. Esto ha permitido ver junto a otras herramientas como el Burndown Chart la

duración de una tarea junto a su estimación. Se pueden ver todas las actas en el apartado 9.f de esta memoria. Se han seguido pautas marcadas en el libro *Essential Scrum: A Practical Guide to the Most Popular Agile Process* (S. Rubin, 2012).

ii. EDT

Nuestro Esquema de División de Trabajo (*EDT*) presenta todas las tareas que componen el proyecto diseñado a comienzos de curso. Entre todas ellas se definen cinco grupos de tareas: General, MOBA, RTS, Producción y Menús/GUI,

Entre estos cinco grupos mencionados, MOBA, RTS y General se componen de tres subgrupos (arte, diseño e implementación). Por su parte Menús/GUI se compone de tres subgrupos haciendo referencia a los grupos General, MOBA y RTS. Por último, Producción no se divide en subcategorías:

1. Producción: contiene las tareas externas al proyecto, es decir, las que no se basan ni en implementación, arte o diseño, pero sí las que especifican la planificación, organización y visión por parte externa al proyecto.
2. General: incluye las tareas comunes a los modos de juego RTS y MOBA, basadas en el escenario, características específicas del videojuego, audio y conexión por red, fundamentales en este proyecto.
3. RTS: muestra las tareas que definen el comportamiento y diseño cada uno de los componentes del ejército RTS y de las construcciones, así como su manejo a nivel colectivo e individual. Además, incluye todas las tareas que se necesitan para especificar cada característica o funcionalidad de cada componente del ejército.
4. MOBA: al igual que en el grupo RTS, todas las características y funcionalidades vienen definidas para las unidades, en este caso, para los héroes. También incluye las tareas respectivas al diseño basadas en la evolución y las habilidades de combate de estas unidades MOBA héroes.
5. Menús/GUI: contiene las tareas referentes a los menús de selección de los dos modos de juego, y su menú previo General para acceder a dichos modos, así como los menús de selección de características *outgame* del videojuego.

Tras la asignación de las tareas y como se ha comentado en el apartado anterior, se procede a actualizarlas en Trello, y atendiendo al acta de reuniones a dicha tarea se le asigna un autor y una fecha que indica el plazo para realizarla.

En la Figura 3.1 se puede ver que las tareas asociadas pertenecen a tres subgrupos claramente diferenciados (diseño, arte y programación), y, a su vez, estos pertenecen al grupo MOBA.

La tabla completa de nuestro EDT se encuentra en el apéndice 9.b.

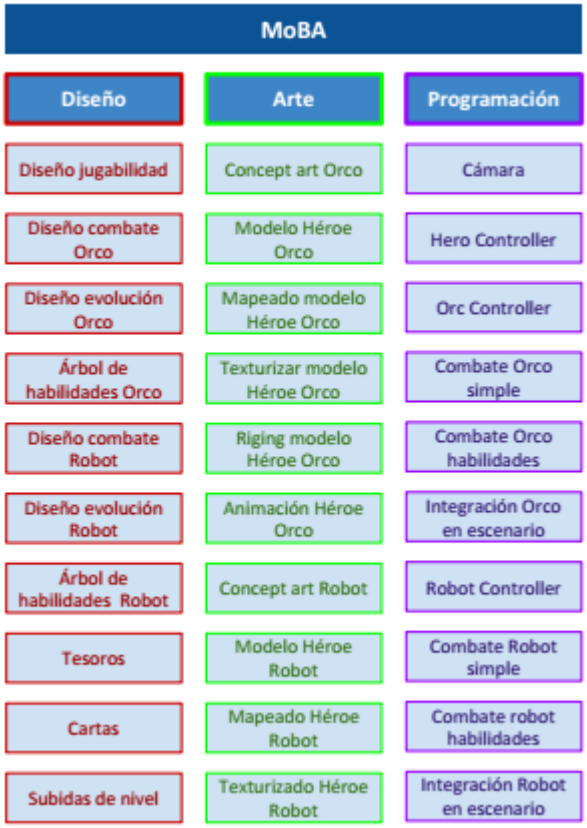


Figura 3.1: Ejemplo de tareas del EDT.

iii. Burndown Chart

En el apartado a de esta sección se comentó la importancia y funcionalidad de este diagrama: representar gráficamente el trabajo por hacer en un proyecto contabilizando un tiempo estimado para la elaboración de todas las tareas del proyecto. De este modo se puede valorar el progreso llevado a cabo comparando el tiempo estimado y el tiempo real gastado en una línea temporal hasta el final del plazo en cuestión, en este caso, la duración del curso.

Este diagrama se produce en consecuencia en decisión al modelo de desarrollo elegido, *scrum*, que al ser un modelo de desarrollo ágil de software tiene unas características que se adecúan de forma perfecta para poder llevar a cabo un diagrama como este.

El *Burndown Chart* permite tener una predicción de cuándo se completará todo el trabajo y así actuar en consecuencia, ya que la estimación a medida que se avanza el curso podrá valorarse en referencia al progreso.

El documento *Burndown Chart* de este proyecto contiene en el eje vertical todo el listado de tareas con un tiempo de dedicación estimado para cada una, y un tiempo dedicado que se va actualizando a medida que se dedique tiempo a la tarea en concreto, sin importar el número de sprints en los que se elabore la tarea.

Como se puede ver en la Figura 3.2 las tareas tienen una columna en la que se anota el número de horas estimadas y otra en las que se apuntan el total de horas invertidas, de modo que el objetivo es llegar a las horas estimadas. En las siguientes columnas se muestra la fecha del sprint y, según la fila, el número de horas invertidas en dicho sprint.

Task(2)	Task(3)	Task(4)	(estimated)	(spent)	12/10/2013	26/10/2013	9/11/2013	23/11/2013
MoBA	Diseño	Diseño jugabilidad	24	24	12	12		
		Diseño combate Orco	13	13				12
		Diseño evolución Orco	8	8				
		Árbol de habilidades Orco	6	6				
		Diseño combate Robot	12	12				
		Diseño evolución Robot	12	6				
		Árbol de habilidades Robot	12	0				
		Tesoros	12	0				
		Cartas	12	0				
		Subidas de nivel	12	1				
	Arte	Concept Art Orco	6	6				6
		Modelo Héroe Orco	12	12				12
		Mapeado Modelo Héroe Orco	6	6				6
		Texturizar modelo Héroe Orco	6	6				
		Riging modelo Héroe Orco	6	6				
		Animación Héroe Orco	24	24				
		Concept Art Robot	6	6				6
		Modelo Héroe Robot	12	12				
		Mapeado Héroe Robot	6	6				
		Texturizado Héroe Robot	8	8				
		Rigging Héroe Robot	6	6				

Figura 3.2: Resumen de tareas en el *Burdown Chart*.

Así mismo, para plasmar la línea temporal hasta la finalización del curso y el progreso durante el mismo se estima en el eje horizontal un tiempo a dedicar en cada sprint, que se compara con la suma del tiempo dedicada a las tareas del respectivo sprint.

También se elabora una comparación desde el comienzo del curso entre las horas totales a dedicar restando en cada sprint las horas estimadas a dedicar, y esas mismas horas totales a dedicar, pero en este caso restando las horas dedicadas. Finalmente la diferencia entre estas dos cifras puede indicar varios detalles, como pueden ser una buena o mala estimación, un notable o deficiente progreso del trabajo realizado, etc… Estos detalles que definen la valoración del progreso permiten, junto al modelo SCRUM tomado, tomar rápidas decisiones cimentadas en datos empíricos.

Como se puede ver en la Figura 3.3 hay 4 filas. En la primera se muestra el total de horas de cada columna, en la segunda se muestra las horas planificadas en cada sprint, en la tercera el total de horas necesarias para finalizar el proyecto teniendo en cuenta las planificadas en cada sprint, y en la última es lo mismo que en la tercera, pero teniendo en cuenta las horas invertidas.

TOTAL	2650	2222	37	54	50
Daily burnout			72	72	72
Total time left (from estimate)			2578	2506	2434
Total time left (from spent)			2613	2559	2509

Figura 3.3: *Burndown chart*, total de horas acumuladas y estimadas.

Este documento también presenta un gráfico que permite apreciar rápidamente el avance del proyecto en base a las horas estimadas totales (*Total time left from estimate*) con el consumo real de estas horas (*Total time left from spent*). El resultado se presenta en dos líneas que corresponden a estos dos parámetros, y como se mencionó anteriormente, la diferencia de altura entre ambos representa la desviación de la estimación con el tiempo real consumido.

En la Figura 3.4, producida como resultado al final del presente proyecto, se aprecia como en el mes de febrero hubo un desfase entre las horas estimadas y las realmente consumidas debido a la época de examen, también se aprecian momentos de mayor o menor dedicación al desarrollo del proyecto, concluyendo de manera satisfactoria a la finalización.

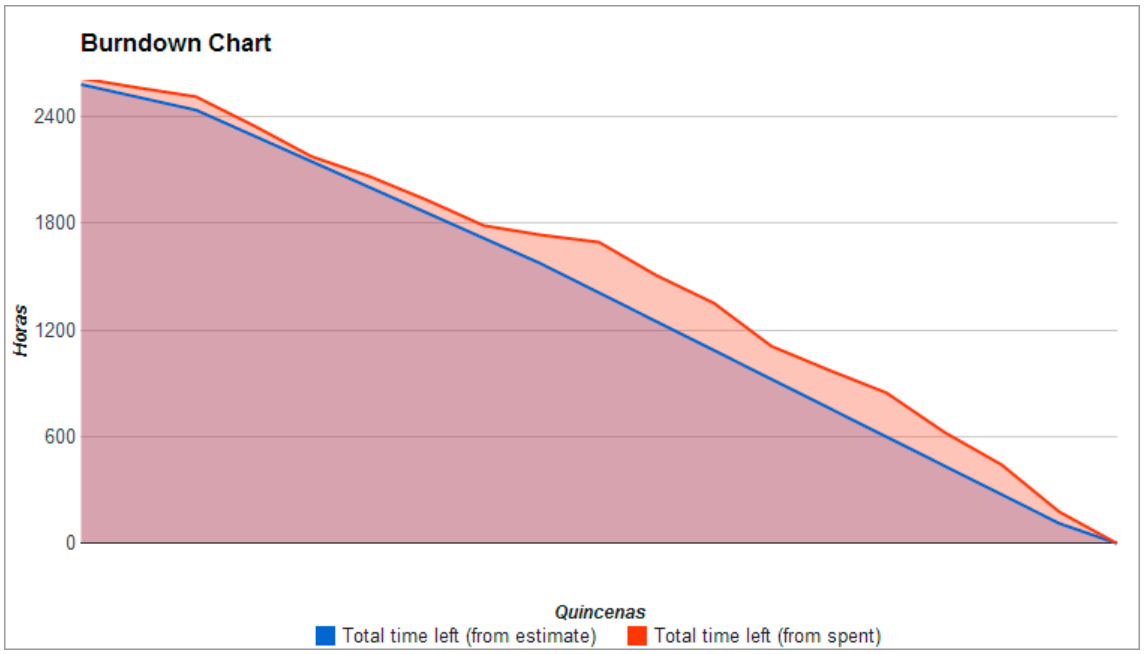


Figura 3.4: Aspecto final de la gráfica del *Burndown Chart* de este proyecto.

En este proyecto, la valoración de este diagrama ha tenido gran importancia ya que un pequeño número de tareas referentes a menús y funcionalidades *outgame* se han tenido que descartar, aunque se hayan diseñado y sigan contempladas en el *EDT*. Gracias al diagrama, dicho descarte pudo realizarse con tiempo suficiente para no afectar al núcleo del proyecto.

La conclusión final al utilizar *Burndown Chart* no es sólo que proporcione una predicción sobre cuándo se finaliza el trabajo, que refleje el progreso del proyecto o que ayude a planificar posteriores sprints basados en el modelo *scrum*, sino también potenciar el análisis de las tareas y su valoración cuando se estima, se elabora y se ha elaborado, aportando un aprendizaje en este ámbito analítico.

El *Burndown Chart* completo realizado en el proyecto se presenta junto con su gráfica se presenta en el apéndice 9.c de esta memoria.

iv. Planificación a largo plazo

En el momento de construir el proyecto a comienzos de curso se necesitó realizar una estimación mediante hitos, que han marcado y fijado los objetivos a conseguir en ciertas fechas a lo largo del curso.

Antes de definir cada hito, se tuvo que especificar el diseño y estructura básica del proyecto mediante reuniones, y así conocer la inmensidad de las pretensiones y saber la capacidad exigida a los miembros del grupo y la carga de trabajo que todo suponía evitando en su justa medida que las pretensiones fuesen demasiado altas.

Con todo esto se estimó una planificación a largo plazo adecuada al diseño y estructura escogido, teniendo en cuenta por supuesto dicha capacidad de trabajo y así intentar elaborar una planificación adecuada al trabajo que los miembros del grupo eran capaces de llevar a cabo.

Esta planificación a largo plazo se marcó mediante los hitos mencionados, que contienen las tareas agrupadas a grandes rasgos sin especificar funcionalidades o características de los componentes visibles del videojuego, así como los requisitos marcados en cada uno de ellos.

El control sobre este documento se ha llevado a cabo mediante un código de colores que representaba el estado de cada tarea, pudiendo apreciarse con facilidad si estaba por comenzar, en proceso o acabada. Además, también ha incluido las tareas que se han podido desechar o descartar porque la carga de trabajo que se estimó fue demasiado alta, o bien, porque la estimación por si misma no se adecuaba al contenido y dedicación de cada tarea, y en conjunto, no han permitido que el hito fuese completamente viable.

En verde se van coloreando las tareas que sí que se han completado para el hito, en rojo las que no se han realizado, en ámbar las que no se han completado por necesitar tareas previas realizadas, éstas se pasan al siguiente hito en color azul.

A continuación se muestra como ejemplo un hito de este documento:

- 26 de Febrero de 2014:
    - Montaje de assets creados hasta la fecha en el escenario del Proyecto Unity.
    - Lógica Unidades artillería pesada del RTS (tanques). Lógica + modelo y animaciones 3D.
    - Skybox del escenario.
    - Modelo 3D de las minas de recursos del ejército goblin.
    - Modelo 3D de las torretas neutrales del escenario (está hecho el concept art, falta modelo 3d).
    - Primera versión de la GUI (se pospone para más adelante).
    - Primera integración de audio (se pospone para más adelante).
    - Preparar presentación ESNE, PP + video (queda algo muy regular).
    - Lógica Unidades bélicas del ejército RTS (falta terminarlo).

4. Documento de Diseño de Juego

Hasta la fecha se habían realizado multitud de videojuegos RTS y MOBA, pero por separado; no se había visto nunca un videojuego que una los dos géneros. Por esta razón, una de las etapas más importantes es la de diseño, ya que al ser la primera vez que se implementa un proyecto de estas características hay que dejar muy bien definido el comportamiento del videojuego.

Una parte fundamental al comienzo del proyecto ha sido este diseño del propio videojuego, establecido a lo largo de múltiples reuniones y plasmado en diferente documentación que más adelante se detalla, comenzando por documentos de pre-producción que ayudaron a definir las líneas básicas del diseño del videojuego como el documento de diseño del videojuego.

El documento de diseño del videojuego (normalmente conocido por las siglas GDD de su nombre en inglés: *Game Design Document*) es una pieza fundamental a la hora de preparar la producción de un videojuego mínimamente complejo, ya que en él se describe hasta el más mínimo detalle cualquier cosa concerniente al diseño del videojuego.

En lo referente a ingeniería del software, el GDD, también es un documento muy importante, ya que permite organizar los esfuerzos en un equipo de desarrollo, además de resultar imprescindible para planificar en el tiempo el desarrollo.

a. Pre-producción

Durante el periodo de pre-producción se generaron dos documentos principales: un documento de concepto del videojuego y una ficha resumen con los puntos básicos de lo que sería el videojuego. El primero acabó derivando en el GDD posterior, mientras que el segundo sintetiza a grandes rasgos las ideas principales del diseño y se detalla a continuación:

1. *Ficha del Juego*

Título:	Project NewDetroit (nombre provisional)
Género:	Estrategia, RTS (*estrategia en tiempo real*) + MOBA (*multijugador online en arena de batalla*).
Plataformas:	PC y MAC
Modos:	100% online, 1vs1, 2vs2
Público:	Jugadores habituales, mayores de 12 años

2. *Descripción*

Juego de estrategia que mezcla mecánicas de RTS (*estrategia en tiempo real*) y de MOBA (*arena de batalla multijugador online*) enfocado al multijugador en red, mezclando

componentes cooperativos y competitivos, de manera que un equipo lo forman dos jugadores, uno controla un ejército y el otro un héroe, y luchan en una arena contra otros equipos.

El ejército se controlará como un RTS clásico tipo StarCraft (selección de unidades y desplazamiento), con una vista aérea. El héroe, sin embargo, con una cámara en tercera persona (control directo sobre el avatar).

La popularidad del género MOBA ha crecido muchísimo en los últimos años y ha tenido mucha proliferación en el número de juegos. Se pretende dar una vuelta de tuerca al género uniendo una parte RTS que también ha gozado de mucha popularidad en el *gaming* competitivo a lo largo de muchos años.

3. Ambientación

Al haber varios tipos de escenarios (con distintas temáticas) y distintos tipos de héroes, la ambientación del juego dependerá directamente del escenario en el que se juegue. En principio, el primer escenario en el que se trabajará tendrá una estética futurista distópica como la vista en películas de los 80-90 como *1997: rescate en Nueva York (Escape from New York, John Carpenter, 1981)*.

4. Mecánicas Centrales

El objetivo principal de los jugadores será el de complementarse bien entre los miembros del equipo (ejército + héroe) y destruir la base del equipo contrario. Para ello el ejército asistirá al héroe, consiguiendo recursos para poder construir más unidades y evolucionar algunos atributos del héroe. El héroe, a su vez, explorará el mapa para conseguir tesoros que más tarde podrá utilizar en combate y subirá de nivel, subida que también afectará al ejército.

Por el mapa, aparte de tesoros, también existirán torres de defensa neutrales (desactivadas) que podrán ser tomadas por los equipos.

Los héroes tendrán un árbol de habilidades para desbloquear características y objetos entre partidas.

El ejército contará con varios tipos de unidades: recolectores (unidad individual que se encarga de recolectar los recursos y reparar posibles daños en la base y pueden curar al héroe), artillería básica (soldados rasos que atacan a distancia), exploradores (soldados muy rápidos que atacan cuerpo a cuerpo), artillería pesada (unidad individual con gran poder de ataque a distancia, de movimientos lentos y poco resistente a ataques a corta distancia) e ingenieros (soldados que se encargan de tomar las torres de defensa que se encuentran desperdigadas por el escenario, y de construir algunas torretas especiales, además de los almacenes de recursos).

5. Referentes

- MOBAs: League Of Legends, El Señor De Los Anillos: Guardianes de la Tierra Media, DOTA, Smite.
- RTS: StarCraft, Warcraft.
- Otros: World of Warcraft (combate), Overlord.

6. *Riesgos*

Problemas derivados debido a la poca experiencia en desarrollo de videojuegos con *Unity*.

Al tratarse de un juego totalmente multijugador en red, pueden existir problemas derivados de este tipo de juegos: *lag*, lógica servidor-cliente, saturación, etc.

b. GDD

Se trata de un juego de estrategia que combina mecánicas de RTS (estrategia en tiempo real) y de MOBA (arena de batalla multijugador online) enfocado al multijugador en red, mezclando componentes cooperativos y competitivos, de manera que un equipo lo forman dos jugadores, uno controla un ejército y el otro un héroe, y luchan en una arena contra otros equipos, haciéndose uso de una jugabilidad asimétrica (cada jugador controla el juego de una manera determinada, no se mezclan los controles y jugabilidades).

El ejército se controla como un RTS clásico tipo *StarCraft* (selección de unidades a través de clic izquierdo del ratón sobre estas y ejecución de órdenes de forma automática al hacer clic con el botón derecho), con una vista aérea. El héroe, sin embargo, funciona con un control directo sobre el avatar, y tiene una vista en tercera persona (como, por ejemplo, *World of Warcraft*).

En los siguientes apartados se definen y explican cada una de las características de ambas mecánicas de juego, adentrándose tanto en las especificaciones de las unidades como en las funcionalidades de las mecánicas en sí.

i. Jugabilidad y Mecánicas MOBA

Como se mencionó en el apartado introductorio, ambas mecánicas de juego no comparten controles ni jugabilidades, y por este motivo, a continuación se exponen estas características distintivas del modo MOBA del proyecto y las características concretas de las unidades.

1. Unidad

El jugador controla a un **Héroe** (ver apartado 4.b.i.9 héroes) que posee los siguientes atriibutos: nivel (se empieza desde el nivel 1 en cada partida), vida, experiencia, ataque y defensa física, ataque y defensa mágica, alcance de ataque físico y mágico, maná (que se consume al usar ataques mágicos, se rellena de forma lenta automáticamente, o se potencia con acciones del propio héroe o del ejército), adrenalina (que se consume al usar ataques físicos, cuando se termina la *barra* el héroe ataca a una velocidad considerablemente menor y recibe más daño físico) y velocidad de ataque.

2. Cámara

La cámara utilizada es en tercera persona y sigue al personaje a cierta distancia, pero no está 100% anclada a la espalda: cuando el personaje rota, esta se ajusta a su espalda de forma

suave y progresiva. Existe la posibilidad de colocar la cámara detrás del personaje de forma inmediata pulsando una vez el botón derecho del ratón.

Referencias:

- Super Mario 64.
- Ratchet & Clank[17].

3. Movimiento

Movimiento libre como en los juegos de plataformas 3d pero sin salto.

- W: desplazamiento hacia adelante.
- S: desplazamiento hacia atrás.
- A: giro y desplazamiento hacia la izquierda.
- D: giro y desplazamiento hacia la derecha.

Referencias: las mismas que la cámara.

4. Atributos (stats)

Se han creado diferentes atributos para definir los héroes, además de para poder interactuar correctamente con el juego.

- Vida: puntos de vida del héroe.
- Ataque físico y mágico: fuerza con la que ataca el héroe.
- Velocidad de ataque: indica la frecuencia con la que el héroe es capaz de lanzar ataques.
- Defensa física y mágica: capacidad de resistir el ataque de un enemigo.
- Maná: los ataques mágicos consumen maná, si se termina este, no se pueden lanzar ataques mágicos hasta que se recupere. Se rellena con el paso del tiempo.
- Adrenalina: los ataques físicos consumen adrenalina, si se termina esta, no se pueden lanzar ataques físicos hasta recuperarla. Se rellena con el paso del tiempo.

5. Ataque

Se han creado distintos ataque y habilidades, diferentes en cada héroe.

- Básico:
    - Clic izquierdo del ratón: lanza su ataque básico en dirección a la normal del personaje en línea recta, que puede ser a distancia o no. Si es a distancia aparece una mirilla que indica dónde va a caer el ataque.
    - Clic derecho del ratón: lanza su ataque secundario (de manera similar al primario).
    - Ruleta - botón central del ratón (y números del teclado): selección del tipo de ataque secundario que se utiliza con el clic derecho.
- Habilidades (ataques secundarios):

[17] http://www.youtube.com/watch?v=vbvZhSghtkA

- Tecla 1: habilidad 1.
- Tecla 2: habilidad 2.
- Tecla 3: habilidad 3.
- Tecla 4: habilidad 4.
- Tecla 5: habilidad 5.

6. Evolución (dinámica, dentro de partidas)

La evolución dentro de las partidas depende de unos puntos de experiencia que se ganan al matar enemigos y recogiendo tesoros. Cada vez que se alcanza una cierta experiencia se produce una subida de nivel. Características de los niveles:

- Se podrá subir hasta el nivel 4.
- Cada vez que se sube de nivel se desbloquea una habilidad nueva.
- La habilidad más potente solo se puede seleccionar al subir al último nivel.
- La subida de niveles afecta directamente al ejército RTS, modificando su aspecto y algunos atributos en función de la habilidad seleccionada.

7. Búsqueda de Tesoros y Recompensas

Por el mapa se encuentran repartidos diversos tesoros que pueden ser recogidos por el héroe. Algunos de estos tesoros están protegidos por NPCs (personajes que se controlan mediante inteligencia artificial) cuya resistencia depende de lo valioso que sea el tesoro que defiende:

- **Recompensas**:
  - Al matar a cualquier enemigo o edificio se consiguen puntos de experiencia.
  - Al matar enemigos salvajes se consiguen recompensas, que son objetos utilizables.
- **Tesoros**: habrá tesoros equipables, que se guardan en el inventario y se pueden activar cuando el jugador lo precise. Al hacerlo, activarán una mejora durante un determinado tiempo y transcurrido este tiempo desaparecerá la mejora y el objeto del inventario.

Al cabo de un cierto tiempo, los tesoros con los NPCs podrán volver a aparecer en el escenario de manera automática para aportar jugabilidad y labores secundarias más allá del combate contra los otros jugadores.

Tipos de tesoros (y qué atributos afectan):

- Cáliz Rojo: aumenta el daño físico del héroe.
- Elixir Rojo: aumenta la defensa física del héroe.
- Cáliz Azul: aumenta el daño mágico del héroe.
- Elixir Azul: aumenta la defensa mágica del héroe.
- Pócima Roja: recupera cansancio.
- Pócima Azul: recupera maná.
- Pócima Verde: recupera vida.
- Botas: aumenta la velocidad del héroe.

- Guanteletes: aumenta la velocidad de ataque físico del héroe.
- Capa: aumenta la velocidad de ataque mágico del héroe.
- Anillo: aumenta el cansancio máximo del héroe.
- Pendientes: aumenta el maná máximo del héroe.

8. Condición de Muerte

Cuando la vida llega a 0 el héroe muere. Al morir se reaparece en la base principal después de un cierto tiempo, pero desactivado. Durante un pequeño periodo de tiempo, el héroe tiene que ser activado por unidades del ejército. Este proceso se muestra como una barra que aparece encima del héroe *dormido*, el ejército deberá asignar recolectores para curar al héroe: cuanto mayor sea el número de recolectores que estén designados, antes despertará.

Cuanto más larga es la partida, el tiempo que tarda en activarse el héroe tras reaparecer en la base es mayor (la barra que se muestra encima es más larga).

También puede depender de la diferencia de nivel con el equipo contrario, de manera que si el héroe muerto tenía un nivel mayor que el del equipo contrario, el tiempo que tarda en activarse para volver es mayor.

9. Héroes

Se han diseñado e implementado dos tipos de héroes, que tienen comportamientos y ataques diferentes.

a. Rob Render

Jefe del ejército orco (pandilla de los *Red Lobster*), cuadrilla callejera de Nueva Detroit.

- Ataques / habilidades:
  - Ataque básico: combo encadenable de 3 movimientos con los puños.
  - Habilidad 1: escupe mocos que ralentizan al contrario.
  - Habilidad 2: eaca un radiocasete grande y se pone a hacer gestos de cuernos. Deja a los minios que tenga alrededor X segundos paralizados y X-Y segundos paralizado al otro héroe.
  - Habilidad 3: carga como un toro y avanza unos metros llevándose todo por delante.
  - Habilidad 4: golpe al suelo (a lo Donkey Kong), aprovechando lo grande que son sus brazos. Lanza a los enemigos que estén en el radio de alcance hacia atrás.
  - Habilidad 5: eructa y se pone un mechero en la boca, sale una llamarada de fuego como si estuviera escupiendo fuego.
- Animaciones:
  - Idle: para cuando está en reposo.
  - Idle-espera: animación de espera para romper el ciclo en idle, cuando lleva varios segundos sin recibir órdenes realiza esta acción.
  - Andar: para cuando el personaje se desplaza de una manera no demasiado rápida.

- Correr: el personaje se desplaza rápidamente.
- Muerte: cae al suelo.
- 3 ataques físicos encadenados.
- Escupir.
- Eructar y colocarse el mechero en la boca.
- Golpe al suelo
- Cargar como un toro.

- Modelo 3D:

Figura 4.1: Aspecto del modelo 3D de Rob Render.

b. Skelterbot

Jefe del ejército robot (pantilla de los *Skelters*), una raza de gusanos alienígenas que controlan a robots para realizar tareas y desplazarse por Nueva Detroit.

- Ataques / habilidades:
  - Ataque básico: combo encadenable de 3 movimientos con una espada de mano, lanza la espada hacia un lado, luego hacia otro, y por último lanza una estocada.
  - Habilidad 1: ataque especial con espada: alza la espada al aire durante un segundo y hace un movimiento circular con la espada sobre su propio cuerpo.
  - Habilidad 2: disparos de dos tipos. El personaje tiene un arma en el brazo derecho.
    - Un disparo ralentiza al enemigo.
    - Otro disparo reduce la cantidad de maná y adrenalina del enemigo.
- Animaciones:
  - Idle: para cuando está en reposo.
  - Idle-espera: animación de espera para romper el ciclo en idle.

- Andar: para cuando el personaje se desplaza de una manera no demasiado rápida.
- Correr: el personaje se desplaza rápidamente.
- Muerte: cae al suelo.
- 3 ataques con la mano-espada encadenados.
- Ataque circular con la espada.
- Ataque de disparo, se pone en postura y dispara.

- Modelo 3D:

Figura 4.2: Aspecto del modelo 3D de Skelterbot.

ii. Jugabilidad y Mecánicas RTS

De la misma manera que en la sección MOBA anterior, a continuación se describen las distintas funcionalidades que pertenecen al RTS, a nivel de controles del jugador, a nivel gráfico y visual, y a nivel funcional de cada unidad.

1. Interfaz gráfica

El jugador dispondrá de:

- Minimapa: situado en la parte inferior derecha de la pantalla, donde se podrá ver todo el escenario en diminuto para poder acceder a la visión de cualquier parte del escenario y tropas propias o enemigas, siempre y cuando lo permita la niebla de batalla.
- Recuadro de tropas: las tropas seleccionadas se mostrarán por unidades para así poder visualizar específicamente qué unidades forman dicha tropa. Está situado en la parte inferior izquierda de la pantalla. Dependiendo de la cantidad y tipo de unidades seleccionadas se agruparán por tipo de unidades, cada agrupación será de un máximo de 20 unidades y con el tabulador podrá escogerse entre las distintas agrupaciones seleccionadas.
- Recuadro individual: en este recuadro se visualiza el tipo de unidad escogido de forma individual entre las agrupaciones disponibles. Se mostrarán todas las estadísticas de dicha unidad. Se sitúa a la derecha, a continuación del recuadro de tropas.

- Recuadro de acciones: se sitúa entre el minimapa y el recuadro individual. Muestra las acciones de la tropa o unidad seleccionada.
- Botón de menú: situado en la parte superior izquierda de la pantalla, permitirá acceder a las diferentes opciones del juego: reanudar, salir, configuración, etc.
- Mostrador de recursos: situado en la parte superior derecha de la pantalla, consistirá en un número que serán los recursos actuales disponibles.

2. Cámara

La cámara estará situada sobre el mapa, con una vista *a tres cuartos* para poder ver una parte considerable de terreno y tropas.

Referencia de la cámara del videojuego StarCraft[18].

El jugador, para moverse a lo largo y ancho del escenario podrá:

1. Situar el puntero del ratón en los límites de la pantalla para mover la cámara en la dirección que corresponde.
2. Utilizar las teclas direccionales para mover la cámara en la dirección correspondiente.

Además, el jugador dispone de las siguientes funcionalidades:

- Hacer clic en una parte del minimapa para visualizar la zona correspondiente.
- Presionar la tecla de espacio para moverse de forma instantánea a la localización del héroe de tu equipo.
- Realizar zoom para acercar la vista o alejarla a su gusto con la rueda del ratón.

3. Creación de las Tropas

Habrá distintas tropas, que tendrán diferentes características, roles y cometidos, de manera que ayudarán cada una al jugador a su manera:

- Artillería básica.
- Recolectores.
- Ingenieros.
- Artillería pesada.
- Exploradores.

En cuanto a la aparición de tropas, hay que tener un mínimo de recursos necesarios. En el momento que se decide crear una tropa se selecciona y se espera un tiempo determinado. Cuando este tiempo pasa aparecerán las nuevas tropas de forma que la vida será compartida por el grupo nuevo y en el momento de atacar, atacarán todas juntas y en el momento de morir lo harán también.

4. Selección de las Tropas

El jugador podrá seleccionar, con el botón izquierdo, una o varias unidades. Para seleccionar unidades se puede hacer de las siguientes formas:

[18] http://youtu.be/vB5ja3XKl7g

1. Selección simple: haciendo clic encima de una tropa se selecciona esa única unidad.
2. Selección múltiple absoluta: se dibuja un recuadro en pantalla haciendo clic en una esquina y, sin soltar, se desplaza el ratón. Cuando se suelta el botón del ratón, las unidades del ejército que estén dentro de este recuadro quedarán seleccionadas.
3. Selección múltiple por tipo de unidad: este modo consiste en seleccionar todas las unidades de un tipo en concreto (por ejemplo, todas las unidades recolectoras) que estén en ese momento en pantalla. Este modo se puede realizar de dos maneras distintas:
    a. Haciendo doble clic sobre una unidad, se seleccionarán todas las unidades de ese tipo.
    b. Haciendo clic simple mientras se presiona la tecla de Control izquierdo.
4. Selección múltiple selectiva: con la tecla *shift* izquierda presionada se van seleccionando unidades (sin deseleccionar las previamente seleccionadas). De esta manera se van acumulando las unidades seleccionadas de manera selectiva.

Estos modos de selección, siempre que sea posible, son acumulativos. De esta manera, por ejemplo, se pueden seleccionar todas las unidades de un tipo (haciendo doble clic) y luego añadir otras unidades concretas (individuales) mientras se pulsa la tecla *shift*. O seleccionar dos tipos de unidades (artillería y exploradores) haciendo clic sobre un artillero y sobre un explorador mientras se mantiene presionada la tecla de control izquierdo.

5. Movimiento de las Tropas

Para poder mover tropas, lo primero que hay que hacer es seleccionarlas de la forma que se ha explicado en el punto anterior. Después, pulsando con el botón derecho sobre una zona del mapa, se especificará dónde se quiere que vayan las tropas seleccionadas, sorteando, gracias al *path finding*, los obstáculos que se encuentren.

6. Ataque de las Tropas

Hay dos tipos de ataque:

1. Automático:
    a. Cuando una tropa está parada, estará alerta a las tropas enemigas, de esta forma si una tropa enemiga aparece en un radio determinado, por defecto irá a atacarla (mientras siga dentro de este radio), aunque se podrá pasar de activa (que es la que se acaba de comentar) a pasiva, que, en este caso la tropa se quedará en su sitio sin moverse.
    b. Cuando una tropa está parada y una enemiga le ataca, la tropa se defiende atacando también a la enemiga. Si no puede llegar a esta tropa enemiga porque está atacando desde fuera del radio, esta huirá o se quedará según esté activa o pasiva respectivamente.
2. Manual: para poder atacar con tropas, lo primero que hay que hacer es seleccionarlas de la forma que se ha explicado anteriormente. Después,

pulsando con el botón derecho sobre el enemigo, las tropas irán sorteando también los obstáculos a atacar a las contrarias.

7. Recursos

Una de las labores principales del ejército será la recolección de recursos, que estarán repartidos por el mapeado en forma de minas, para poder generar más unidades y asistir al héroe (curarle, evolucionar algunos atributos, etc.).

Estas minas de recursos tendrán una cantidad inicial, que se irá descontando a medida que las unidades recogen los recursos. Cuando esta cantidad llegue a 0, la mina quedará en un estado especial de *agotamiento*, cambiará su aspecto y ya no podrá ser utilizada para recoger más recursos.

Inicialmente, habrá dos unidades recolectoras para poder empezar a recoger recursos.

8. Comportamiento de las Unidades

Las unidades del ejército poseen los siguientes atributos: nivel (se empieza desde el nivel 1 en cada partida), puntos de vida, ataque, alcance de ataque, defensa, velocidad de movimiento, velocidad de construcción, velocidad de ataque.

Las unidades pueden tener dos comportamientos:

- Defensivo: la unidad o unidades correspondientes no se mueven aunque un enemigo entre en el rango de visión, pero sí pueden atacarla si está dentro de su rango de ataque.
- Ofensivo: la unidad o unidades correspondientes se mueven, persiguen y atacan a las unidades que entren en su rango de visión. La persecución termina cuando la unidad enemiga es eliminada o ha salido del rango de visión. En ambos casos, la unidad se para en la última posición en la que la ve.

Acciones comunes a todos los tipos de unidad:

- Atacar: la unidad o unidades seleccionadas tomarán el camino más corto hacia la unidad o edificio a atacar tras seleccionar esta acción.
- Posición ofensiva: la unidad o unidades seleccionadas toman comportamiento ofensivo.
- Posición defensiva: la unidad o unidades seleccionadas toman comportamiento defensivo.

a. Características de los recolectores:

Es la única unidad capaz de recolectar recursos en el escenario. Su velocidad de movimiento y su vida serán estándar. El rango de ataque será cuerpo a cuerpo.

Acciones específicas:

- Recolectar: podrán recolectar los recursos que se obtendrán en los lugares correspondientes.
- Curar al héroe y a otras unidades: estas unidades pueden ser asignadas al héroe para curarle.

- Construir almacén de recursos: podrán construir un almacén que haga las funciones de la base y acortar las distancias para transportar recursos.

b. Características de los Ingenieros

Es la única unidad capaz de conquistar y construir torres. Su velocidad de movimiento y su vida serán estándar. El rango de ataque será a distancia, pero dicha distancia será corta.

Acciones específicas:

- Conquistar: la unidad o unidades seleccionadas conquistarán una torreta aportada por el escenario. Dependiendo del número de ingenieros, el tiempo en conquistarla se verá reducido.
- Reparar: la unidad o unidades seleccionadas repararán una torreta seleccionada. Dependiendo del número de ingenieros, el tiempo en reparación se verá reducido.
- Construir torreta: la unidad o unidades seleccionadas construirán la torreta en la posición seleccionada por el jugador. Tras hacer clic en esta acción decidirá con el botón izquierdo del ratón dónde se situará la torreta. Esta construcción tendrá un tiempo de construcción, que se verá reducido dependiendo del número de ingenieros construyendo.
- Construir almacén de recursos: podrán construir un almacén que haga las funciones de la base y acortar las distancias para transportar recursos.

c. Características de los Exploradores

Tienen más velocidad de movimiento que el resto de unidades pero menos vida que el resto de tipos de unidad. El rango de ataque será cuerpo a cuerpo.

Su acción específica consiste en la funcionalidad de patrullar: la unidad o unidades seleccionadas patrullarán el camino más corto entre varios puntos que el jugador seleccionará con el botón izquierdo del ratón tras seleccionar esta acción (tecla P). La acción comienza cuando, después de marcar varios puntos, se suelta la tecla.

d. Características de la Artillería Básica

Su velocidad de movimiento y cantidad de vida serán estándar. Su rango de ataque será a distancia.

Su acción específica consiste en la funcionalidad del modo de ataque cuerpo a cuerpo: el rango de ataque de la artillería pasa a ser cuerpo a cuerpo, aumentando el daño de ataque en distancias cortas. Para revertir este modo basta con volver a seleccionar dicha acción.

e. Características de la Artillería Pesada

Su velocidad de movimiento se verá reducida con respecto a la artillería básica pero la cantidad de vida y rango de ataque será mayor.

Penalización: no podrá atacar a distancias cortas.

Su acción específica consiste en la funcionalidad del modo de despliegue: las unidades de artillería pesada podrán transformarse en una unidad estática pasado un cierto tiempo tras

la selección de dicha acción, pasando a tener mayor alcance a distancia y mayor poder de ataque. Para revertir este modo, se deberá volver a seleccionar esta acción, lo cual tardará de nuevo ese cierto tiempo. Para consultar el comportamiento de una unidad de artillería pesada puede verse en la referencia de tanques *Terran* del StarCraft2[19].

f. Tabla Resumen

En la siguiente tabla se muestran las características de las unidades del modo RTS.

Unidad	Velocidad de movimiento	Vida	Distancia de ataque	Potencia de ataque	Mov. especial
Recolectores	Normal	Baja	Corta	Baja	
Art. Básica	Normal	Media	Corta / media	Media	
Art. Pesada	Lenta	Alta	Larga	Alta	Despliegue
Ingenieros	Normal	Media	Corta / media	Baja	
Exploradores	Rápida	Baja	Corta	Baja-media	Saltos

9. Modelos 3D

Se han usado diferentes modelos 3D aportados por los colaboradores del Grado de Videojuegos de la Universidad de ESNE. A continuación se muestran las características de cada uno, con los Dummies que tienen cada uno.

a. Ejército Goblin

Goblin01: este modelo se utiliza tanto para las unidades recolectoras como para los exploradores (junto con un cortacésped a modo de montura)

- Altura: 1.50 m
- Dummies:
    - En la cabeza para poder colocar cascos, conos, etc.
    - En las manos para poder colocar la pica de recolección.
    - En las manos para poder colocar los recursos cuando vuelven de la mina.
    - Entre los ojos para poder colocar gafas.
    - En la espalda para poder colocar mochilas.
    - En el pecho para poder colocar una armadura.

[19] http://www.youtube.com/watch?v=KWBpWPFej0Y min 5:30

Figura 4.3: Aspecto del modelo 3D del Goblin01.

Goblin02: este modelo se utiliza para la artillería ligera y para los ingenieros.

- Altura: 1.70 m (con pelo).
- Dummies:
    - En ambas manos para poder colocar metralletas.
    - En la espalda para poder colocar mochilas a los ingenieros.
    - En los hombros para poder colocar hombreras.
    - En frente para poder sacar un portátil y que teclee en él.

Figura 4.4: Aspecto del modelo 3D del Goblin02.

Goblin03: este modelo se utiliza para la artillería pesada.

- Altura: 1.85 m
- Dummies:
    - En las manos para poder colocar una *gaunlet*.
    - En la espalda para poder colocar el mortero.
    - En los hombros para poder colocar hombreras de protección.

Figura 4.5: Aspecto del modelo 3D del Goblin03.

Goblin01 + cortacésped: este modelo es una combinación del modelo del Goblin01 y de una máquina cortacésped. Se utiliza para las unidades exploradoras.

- Dummies:
    - En el parachoques, centrado, para poder colocar mejoras.
    - En los laterales.
    - En el manillar para poder colocar banderas.

b. Ejército Robot

Skelterbot01: este modelo se utiliza tanto para las unidades recolectoras como para los ingenieros

- Altura: 1.50 m
- Dummies:
    - En la cabeza para poder colocar cascos vikingos, cascos, etc.

- En las manos para poder colocar los recursos cuando vuelven de la mina.
- En el pecho para poder colocar un collar con el símbolo del dolar.

Figura 4.6: Aspecto del modelo 3D del Skelterbot01.

Skelterbot02: este modelo se utiliza para la artillería ligera y para los exploradores (junto con un cortacésped a modo de montura).

- Altura: 1.70 m.
- Dummies:
  - En una mano para poder colocar una escopeta.

Figura 4.7: Aspecto del modelo 3D del Skelterbot02.

Skelterbot03: este modelo se utiliza para la artillería pesada.

- Altura: 1.85 m
- Dummies:
    - En el brazo derecho un arma pesada.

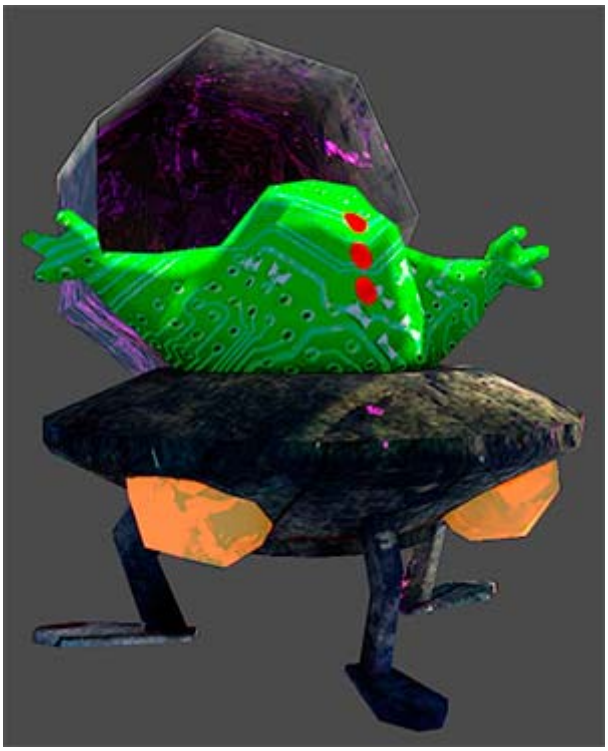


Figura 4.8: Aspecto del modelo 3D del Skelterbot03.

Skelterbot02 + cohete: este modelo es una combinación del modelo del Skelterbot01 y de un cohete. Se utiliza para las unidades exploradoras.

- Dummies:
    - En la punta del cohete para colocar mejoras.

10. Animaciones de los modelos

Cada, modelo para poder expresarse y realizar acciones de manera más vistosa, necesita animaciones, que son movimientos que realizan una acción determinada.

a. Animaciones necesarias para todas las unidades

Todas las unidades del modo RTS tienen unas animaciones para acciones comunes. A pesar de ser comunes estas acciones, cada modelo tendrá una propia para cada una:

- Idle: cuando la unidad está estática en un sitio determinado (respirando).
- Idle-wait: cada cierto tiempo en Idle se desata una animación de *espera*. Por ejemplo, se seca el sudor de la frente, o se mira la suela del zapato si la unidad es bélica, repasa su arma, tropieza, etc, a elección del artista.
- Andando-desplazando: movimiento, cuando la unidad se dirige a un punto.

- Ataque: animación de ataque, exclusiva de cada tipo de unidad (especificada en los siguientes puntos).
- Muerte: cuando la vida de la unidad llega a 0.

b. Animaciones exclusivas de los recolectores:

Estas unidades tienen animaciones para las acciones de recolectar, curar y el resto de acciones entre estas:

- Recolectar: cuando las unidades están picando en la mina.
- Iddle cargando recursos: cuando la unidad tiene recursos recogidos pero está esperando, tendrá que ser parecida al *Iddle*, pero, por ejemplo, con los brazos estirados como si cargara algo, o doblado hacia adelante, como si tuviera los recursos sobre su espalda.
- Andando cargado de recursos: análogo al punto anterior.
- Ataque: el ataque de los recolectores es muy débil, solo golpean con los puños.
- Curar: los recolectores pueden curar al héroe del otro jugador, esta curación se hará de manera estática (sin moverse del sitio).
- Construir almacenes de recursos para acortar distancias en el transporte de recursos.

c. Animaciones exclusivas de la artillería básica:

Estas unidades tienen las animaciones características de los modos de ataque:

- Ataque a distancia: ataque básico de la unidad, atacan con ráfagas de disparos con uzis.
- Ataque cuerpo-a-cuerpo: las unidades pueden pasar a un modo de ataque cuerpo a cuerpo. En este modo, la artillería atacará con bates de baseball.

d. Animaciones exclusivas de la artillería pesada:

Estas unidades tienen las animaciones características de los modos de ataque, desplegados y sin desplegar:

- Ataque 1 (sin desplegar): ataque básico de la unidad, lanza un cañonazo (preparación, disparo, retroceso).
- Despliegue: la unidad se despliega para atacar a una mayor distancia y para producir un mayor daño.
- Ataque 2 (desplegada): similar al ataque 1 pero con la unidad desplegada.
- Des-despliegue: la unidad se levanta y vuelve a la posición normal.

e. Animaciones exclusivas de los ingenieros:

Estas unidades tienen las animaciones características del modo de ataque y de las acciones propias que puede realizar:

- Ataque: los ingenieros pueden atacar a distancia lanzando bombas fétidas. Este ataque se desarrolla de la siguiente manera: la unidad se agacha y comienza a

construir algo con sus manos en el suelo durante 1 o 2 segundos, después se levanta y lo lanza como si fuera una granada.

- Conquistar, reparar y construir torreta: animación de construcción.

f. Animaciones exclusivas de los exploradores:

Estas unidades tienen las animaciones características del ataque y movimiento:

- Ataque: el ataque de los exploradores es de cuerpo-a-cuerpo, con navajas.
- Movimiento: los exploradores se desplazan a una velocidad mayor por el escenario que el resto de unidades.

11. Torretas

El juego presenta dos tipos de torretas diferentes: por un lado están las neutrales que ya se encuentran presentes por el escenario al comienzo de una partida, y por otro las construibles propias de los ejércitos.

a. Torretas neutrales

Por el escenario se encuentran desperdigadas torretas ya construidas pero desactivadas. En un principio permanecerán en un estado neutral y no atacarán a nadie hasta que un ejército las tome. Estas torres serán más potentes (en cantidad de vida y valor de ataque) que las que puede construir un ingeniero por sí mismo.

La conquista de estas torretas ha de ser realizada por unidades de ingenieros de los ejércitos. Esta conquista requiere de un cierto tiempo hasta poder ser realizada. Además cuanto mayor sea el número de ingenieros que estén tomando la torre, este tiempo será menor según una escala logarítmica (por ejemplo, si el tiempo para conquistar una torreta es 1 unidad de tiempo, un solo ingeniero tardará 1 unidad de tiempo, si hay 2 ingenieros no se tarda la mitad de tiempo, sino $(1 - \frac{log2}{2})$; si hay 3 será $(1 - \frac{log3}{2})$.

Si llegado el caso, hay dos grupos de ingenieros de dos ejércitos enemigos compitiendo por la conquista de la torre, la torre tendrá dos *cuentas atrás* diferentes, una por cada grupo de ingenieros, y la primera que llegue a 0 será la que determine qué ejército se queda con la torre.

Una vez tomada, las torretas neutrales no son controlables (pero sí seleccionables, para, por ejemplo, ver la cantidad de vida que les queda), simplemente atacan al primer enemigo que vea, a excepción de si tiene al héroe del equipo rival a su alcance. En este caso, siempre atacará al héroe.

Si en un momento dado, la vida de las torretas es menor a su vida inicial, los ingenieros pueden repararlas.

Si el equipo rival consigue destruir esta torre, no desaparecerá del mapa, sino que pasará a un estado semi-derruida, en el cual volverá a ser neutral, es decir, no atacará a nadie y podrá ser recapturada por uno de los ejércitos.

b. Torretas de los ejércitos

Los ingenieros de los ejércitos, además de poder conquistar torretas neutrales y repararlas, tienen la habilidad de construir torretas en cualquier lugar del escenario donde no haya obstáculos en el terreno.

El proceso de construcción de estas torretas se detalla a continuación:

- Se selecciona un ingeniero (o varios), y se selecciona el botón de construir torreta (o la tecla *atajo*).
- En el escenario aparecerá una torreta semitransparente en la posición del ratón. Cuando se coloca sobre una posición donde se puede construir, el material aparece en un tono verdoso; si el ratón está sobre una posición donde no se puede construir, lo hará en un tono rojizo.
- Una vez escogido el lugar donde construir la torreta, se hará clic con el botón derecho del ratón y los ingenieros seleccionados comenzarán el proceso de construcción de la torreta.
- Cuanto mayor sea el número de ingenieros trabajando en la construcción de la torre, más rápido será este proceso (de forma similar al proceso de conquista de las torretas neutrales comentado anteriormente).

Estas torretas sí que podrán ser seleccionables. Por defecto, funcionarán de manera similar a las neutrales (atacan al primer enemigo que ven), pero el jugador podrá determinar el enemigo al que tienen que atacar (siempre que estén en el área de visión de la torreta).

12. Almacenes de recursos

Los ingenieros de los ejércitos también pueden construir otro tipo de edificaciones llamados almacenes de recursos. Estos edificios sirven para facilitar la recolección de recursos por parte de las unidades recolectoras, de manera que los recolectores después de recolectar una cierta cantidad de recursos irán a descargarlos a la base del ejército (si no hubiera almacenes de recursos construidos) o al almacén de recursos más cercano a la mina donde se está trabajando.

iii. Funcionalidades comunes a ambos modos de juego

1. Evolución conjunta (dinámica, dentro de partidas)

Hay varios tipos de héroes y solo una facción (una sola *raza*) de ejército. Sin embargo, el héroe seleccionado es determinante en la forma en la que evoluciona el ejército del jugador que tiene como compañero de equipo. Por ejemplo, si el héroe es un orco, las unidades del ejército, al subir el héroe de nivel, cambiarán levemente su aspecto y parecerán pequeños orcos.

La evolución del ejército depende de dos factores: del nivel del héroe (que evolucionará los atributos generales del ejército) y de los poderes desbloqueados en la base (por ejemplo más velocidad para un tipo de unidad, o mayor distancia de ataque para otra).

Un cierto número de unidades pueden ser asignadas al héroe (por ejemplo 2 unidades, y el jugador que controla el ejército le asigna al héroe una unidad de artillería básica y un recolector para que le cure) que lo seguirán por el mapa.

2. Misiones comunes

Hay un tipo de misiones que son comunes a RTS y MOBA y que servirán para mejorar las características del equipo que las consiga.

Búsqueda de tesoros: Al comienzo de una partida por el mapa se colocarán unos tesoros repartidos aleatoriamente (de forma equitativa) que puede recoger solamente el héroe. Estos son potenciadores de atributos o usables temporalmente. Por ejemplo, un potenciador de adrenalina: el héroe lo guarda y puede usarlo cuando quiera, tendrá un efecto temporal, ej. durante 30 segundos atacará más rápidamente. Además estarán protegidos por criaturillas NPCs de dificultad variable, que son personajes que se controlan sólos mediante inteligencia artificial.

Boss: Un NPC aleatorio que sale al principio (o más tarde) de la partida que da una ventaja al equipo que lo derrote. Tiene un porcentaje bajo de soltar una carta usable en la evolución estática del héroe, lo que lo hace especialmente deseable de eliminar por ambos héroes.

Captura de torres (*Tower Defense*): Están colocadas estratégicamente por todo el mapeado y son capturables por las unidades de ingenieros del ejército. Estas torres serán más potentes que las que pueden construir un ingeniero por sí mismo. Al principio de la partida son neutrales, y los ejércitos y el héroe tienen que luchar por mantener estas construcciones estratégicas.

Condición de victoria: la condición primaria es eliminar la base del equipo contrario. Para ello será necesario llegar hasta la base y superar las propias defensas del ejército además de soportar los ataques del héroe (si sigue activo). También existirá la opción de jugar a partidas de menos de 45 minutos: si transcurrido ese tiempo no se ha eliminado la base enemiga, gana el equipo que más recursos haya recolectado sin gastar.

Otra posible forma de terminar pasados los 45 minutos es recolocar a los héroes en arenas vacías y que luchen hasta la muerte súbita.

3. Sistema de mejora de habilidades outgame (estática, entre partidas)

Cada perfil de jugador dispondrá de dos árboles de habilidades, para MOBA y RTS respectivamente. En cada uno podrá mejorar, en base a la experiencia ganada por partidas jugadas, distintas habilidades ofensivas y defensivas en el MOBA, y características de las unidades y de utilidad en el RTS.

La experiencia conseguida en cada tipo de juego está separada. A medida que un jugador vaya acumulando experiencia subirá niveles que le permitirán desbloquear habilidades de mayor nivel, hasta un total de 4 niveles; dando en cada nivel cuatro puntos de habilidad a colocar en las mejoras a elegir.

Experiencia

La experiencia obtenida por partida ganada será de 75 puntos, y por partida perdida 25 puntos en ambos modos de juego.

Cada mejora de los niveles 1, 2 y 3 consistirá en una mejora de dos puntos de habilidad que desbloqueará una mejora de un punto de habilidad en ese mismo nivel en forma de porcentaje. En nivel 4 solo habrá una mejora con un desbloqueo similar al de los otros niveles. Para optar a colocar puntos de experiencia en nivel 2, 3 y 4 se necesitarán 4, 7 y 9 puntos de experiencia en esa misma rama de habilidades respectivamente.

Habilidades MOBA (Figura 4.9)

Habilidades ofensivas:

- Nivel 1: se podrá mejorar el ataque físico y mágico en 2/4 puntos y en un aumento del 5% como mejora desbloqueada.
- Nivel 2: se podrá mejorar el maná y la adrenalina en 20/40 puntos y en un aumento del 5% como mejora desbloqueada.
- Nivel 3: se podrá mejorar el ataque físico y mágico como una habilidad conjunta a mejorar en 1/2 puntos y en un aumento del 5% como mejora desbloqueada y la velocidad de ataque en otra habilidad conjunta en 3%/6% y un aumento del 10% como mejora desbloqueada.
- Nivel 4: se podrá reducir el cooldown de los ataques especiales en un 3%/6% y una reducción del 10% como mejora desbloqueada.

Habilidades defensivas:

- Nivel 1: se podrá mejorar la defensa física y mágica en 2/4 puntos y en un aumento del 5% como mejora desbloqueada.
- Nivel 2: se podrá mejorar la vida en 25/50 puntos y en un aumento del 5% como mejora desbloqueada, y la velocidad de movimiento en 3%/6% y un aumento del 10% como mejora desbloqueada.
- Nivel 3: se podrá mejorar la defensa física y mágica como una habilidad conjunta a mejorar en 1/2 puntos y en un aumento del 5% como mejora desbloqueada, y la vida y la velocidad de movimiento en otra habilidad conjunta en 1%/2% y un aumento del 5% como mejora desbloqueada.
- Nivel 4: se podrá reducir el cooldown por muerte en un 3%/6% y una reducción del 10% como mejora desbloqueada.

Habilidades RTS (Figura 4.10)

Características de las unidades:

- Nivel 1: se podrá mejorar la vida en 10/20 puntos y en un aumento del 5% como habilidad mejorada y el ataque en 1/2 puntos y en un aumento del 5% como mejora desbloqueada.
- Nivel 2: se podrá mejorar la defensa física y mágica en 2/4 puntos y en un aumento del 5% como mejora desbloqueada en cada una.

- Nivel 3: se podrá mejorar la velocidad de ataque y de movimiento en 1%/2% y un aumento del 5% como mejora desbloqueada en cada una.
- Nivel 4: se podrá mejorar la vida en 10/20 puntos y en un aumento del 5% como habilidad mejorada y el ataque en 1/2 puntos y en un aumento del 5% como mejora desbloqueada.

Utilidad:

- Nivel 1: se podrá mejorar la velocidad de movimiento del explorador en 3%/6% y un aumento del 10% como mejora desbloqueada, y la velocidad de recolección del recolector en 1%/2% y un aumento del 5% como mejora desbloqueada.
- Nivel 2: se podrá mejorar la velocidad de construcción y reparación del ingeniero como habilidad conjunta y la velocidad de curación del recolector en 1%/2% y un aumento del 5% como mejora desbloqueada en cada una.
- Nivel 3: se podrá mejorar la velocidad de conquista del ingeniero y la velocidad de curación y recolección del recolector como habilidad conjunta en 1%/2% y un aumento del 5% como mejora desbloqueada en cada una.
- Nivel 4: se podrá reducir el tiempo de creación de las unidades en base en un 3%/6% y una reducción del 10% como mejora desbloqueada.

4. Items

Hay diferentes objetos que pueden instanciarse en los diferentes modelos que dan una visión más creativa de estos.

Vehículos: es posible que entre las personalizaciones de algunos héroes pueda estar la capacidad de tener una montura.

Items propios del héroe:

- Espada: aumenta el daño físico del héroe.
- Bastón: aumenta el daño mágico del héroe.
- Botas: aumenta la velocidad del héroe.
- Guanteletes: aumenta la velocidad de ataque del héroe.
- Escudo: aumenta la defensa física del héroe.
- Capa: aumenta la defensa mágica del héroe.
- Cáliz rojo: aumenta la vida máxima del héroe.
- Cáliz azul: aumenta el maná máximo del héroe.
- Anillo: aumenta el cansancio máximo del héroe.

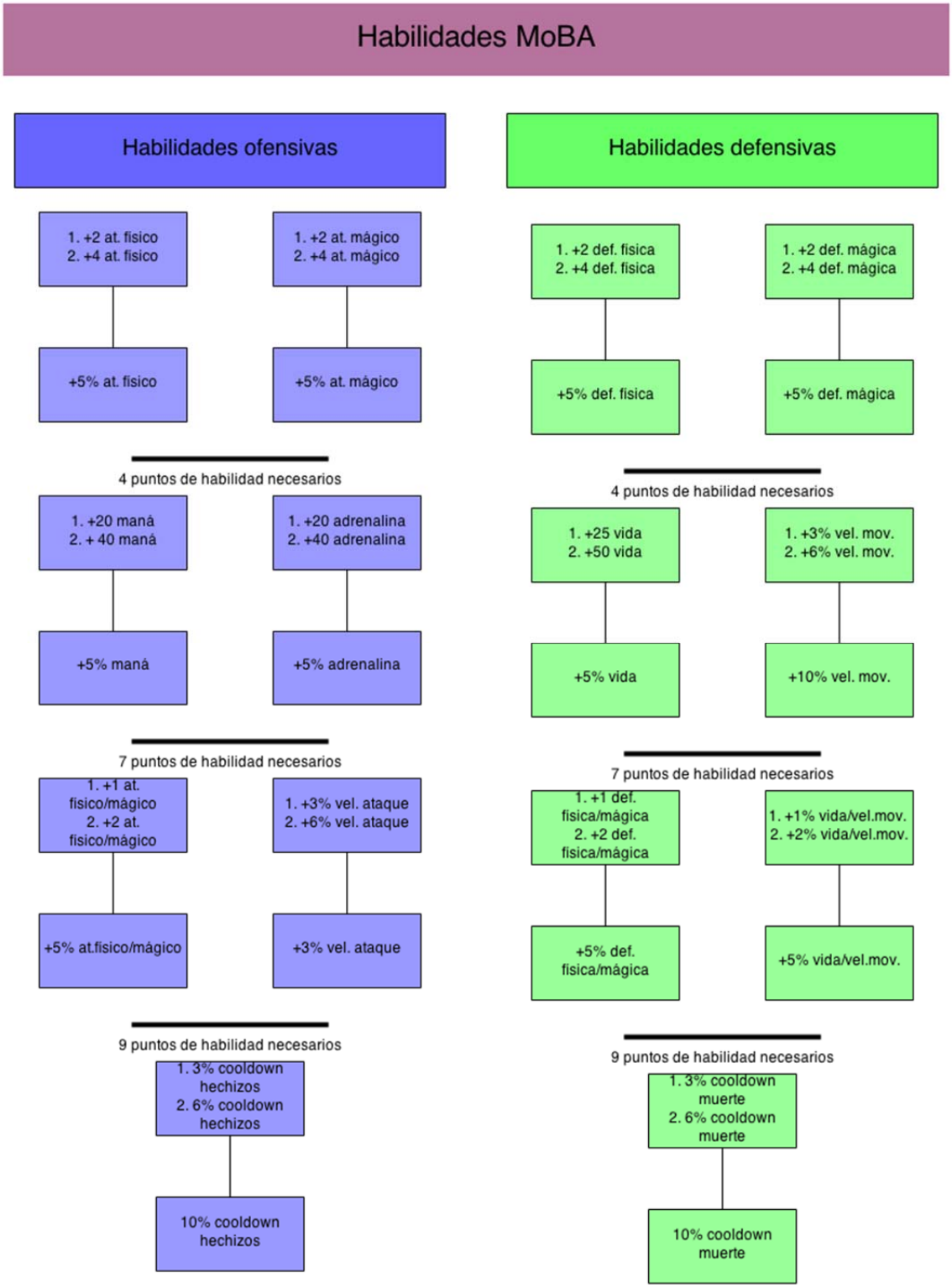


Figura 4.9: Esquema de evolución de las habilidades MOBA.

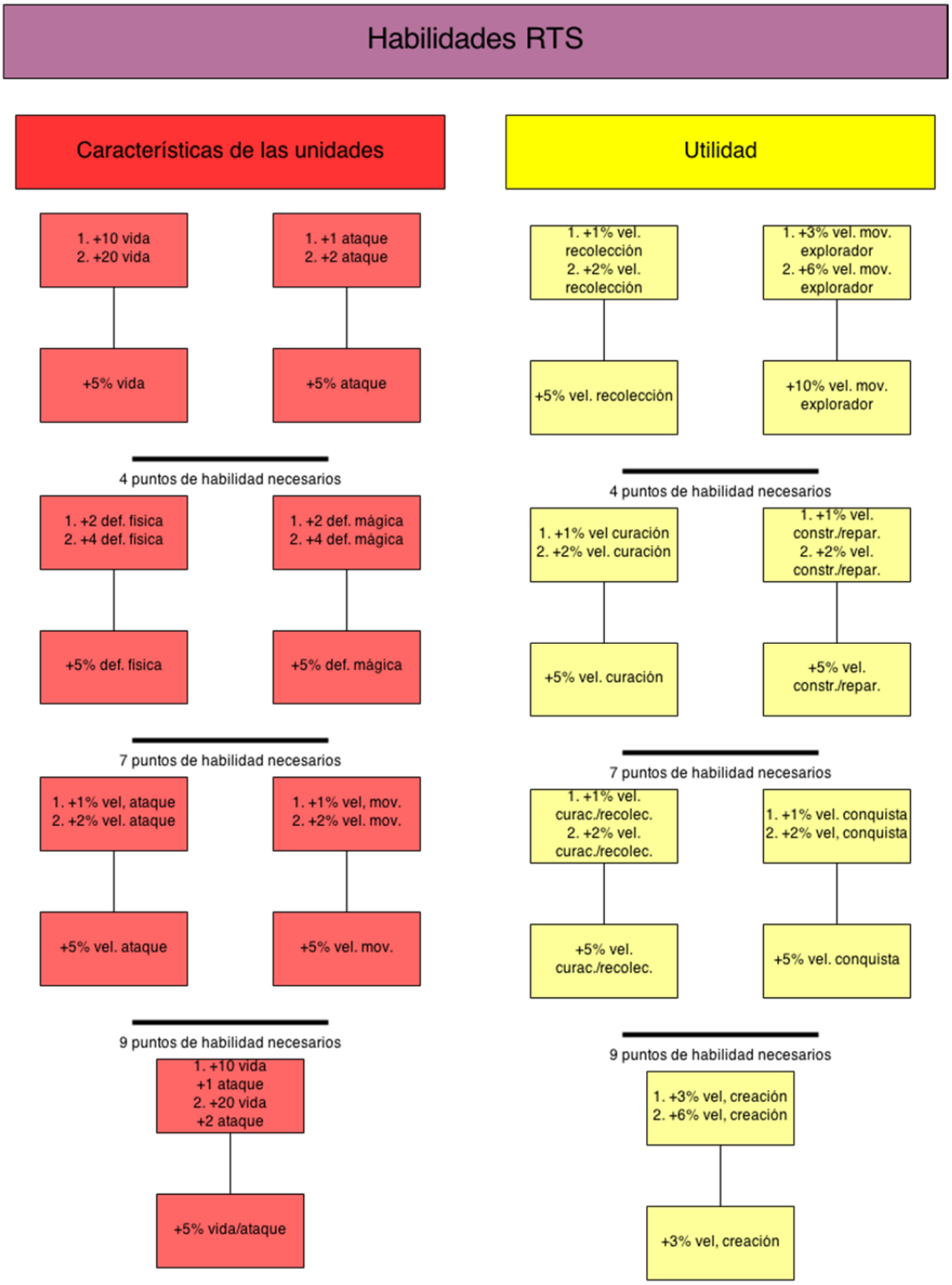


Figura 4.10: Esquema de evolución de las habilidades RTS.

5. Opciones

Tipos de partida:

Partida 2 vs 2 MOBA + RTS: un jugador de cada equipo controla a un héroe y el otro controla el ejército.

Partida 1 vs 1 MOBA: los jugadores eligen un héroe y deberán derrotar al enemigo para ganar la partida. Ganará el equipo que más muertes acumule en un lapso de tiempo. En caso de empate se procederá a una muerte súbita.

Modo de partida:

Partida aleatoria: se selecciona una partida con oponentes que estén buscando partidas en la red.

Partida predefinida: se selecciona una partida en la que decides quién entra a jugar.

6. Otros fundamentos

Hay otras funcionalidades del juego comunes a RTS y MOBA que son:

Comunicación chat y mensajes: al ser un juego online con un alto componente cooperativo es importante disponer de una comunicación en tiempo real fluida, en un principio desarrollaremos un sistema de mensajes rápido y un chat de texto, y más adelante chat por voz.

IA: en principio, al ser un juego enfocado puramente al online, no es necesario programar una IA que simule el comportamiento de jugadores amigos o enemigos. Sin embargo, no dejamos la puerta cerrada a realizarla más adelante (de hecho creemos que puede ser muy interesante de cara al proyecto). Sí que será necesario programar una IA simple para los NPCs que acompañarán a algunos de los tesoros que se encontrarán dispersos por el mapa, además de un algoritmo de *pathfinding* para las unidades del ejército.

Introducir un tutorial: tenemos en cuenta la intención de crear un tutorial cuando el proyecto esté muy avanzado.

7. Planificación

Se ha realizado una planificación a largo plazo del proyecto teniendo que haber una sincronización correcta y eficiente entre los grupos de ESNE y UCM, para poder programar en base a unos modelos fabricados.

Aspectos fundamentales

A la hora de distribuir las tareas a desarrollar se comenzará con el documento de diseño del videojuego y con el desarrollo de la parte MOBA mientras se construye la infraestructura online. El primer objetivo a alcanzar sería que dos jugadores en distintas máquinas pudieran enfrentarse en una arena con dos héroes (cada jugador controlando un héroe).

Después, mientras se sigue profundizando en el juego en general y en la jugabilidad de esta parte MOBA (árbol de habilidades de los diferentes héroes, sistema de cartas, etc), se comenzarán los aspectos RTS del juego: *pathfinding*, creación y control de tropas, etc.

Por último, llegará el momento de juntar y de equilibrar las dos partes del juego. Además se creará el menú del juego, se incluirán las distintas opciones de las partidas y otros aspectos generales del juego.

Aspectos opcionales

Se contempló la posibilidad de realizar características adicionales y se han retrasado para después de la entrega del proyecto

- IA cooperativa y competitiva: realización de una inteligencia artificial que sea capaz de jugar como un jugador humano.
- Generación aleatoria de terrenos.
- Tutorial.
- Muchos tipos de héroes.
- Más unidades del ejército.
- Partidas simplificadas: solo 2 héroes, o solo 2 ejércitos.

c. Atributos

En este apartado se exponen las distintas valores o stats de todas las unidades que forman parte del proyecto, pertenecientes a las dos mecánicas de juego, MOBA y RTS.

1. Valores de las unidades RTS

Los datos numéricos correspondientes a las unidades del ejército RTS son los siguientes:

Unidad	Vida	Ataque	Defensa física	Defensa mágica	Velocidad de movimimiento
Recolector	150	5	20	20	50
Ingeniero	150	10	25	25	45
Explorador	100	10	20	20	65
Artillería básica	150	20-30	30	30	55
Artillería pesada	250	40-60	35	35	35

Unidad	Velocidad de ataque	Velocidad de construcción	Velocidad de curación	Tipo de ataque	Alcance
Recolector	0.8	-	1	Cuerpo a cuerpo	-
Ingeniero	0.8	1	-	A distancia	40
Explorador	0.9	-	-	Cuerpo a cuerpo	-
Artillería básica	1	-	-	A distancia y cuerpo a cuerpo	60 / -
Artillería pesada	1	-	-	A distancia	75 / 100

Todos los campos que tienen como valor - es porque no tienen ese atributo, ya que no realizan esa acción. Por ejemplo, el recolector tiene como velocidad de construcción - ya que no son capaces de construir.

2. Valores de los Héroes

a. Experiencia para subir de nivel

Como se mencionó en el apartado anterior, la evolución de los héroes se produce mediante la ganancia de experiencia en combate, cuyos valores son los siguientes:

- Nivel 1-2: 200 puntos de experiencia.
- Nivel 2-3: 600 puntos de experiencia.
- Nivel 3-4: 1000 puntos de experiencia.

b. Experiencia ganada por el héroe al matar unidades

En conjunto a los datos expuestos en la tabla anterior, la experiencia obtenida en combate depende de la unidad en cuestión. La experiencia que se obtiene al eliminar a los distintos tipos de unidad son:

- Héroe: 100
- Recolector: 10
- Ingeniero: 10
- Explorador: 20
- Artillería básica: 20
- Artillería pesada: 40

c. Cooldown al morir

En lo referente al tiempo de espera de renacimiento del héroe y como se comentó en el apartado predecesor, se puede disminuir mediante la curación del héroe en la base por parte de los recolectores. Aún así, existe un tiempo máximo que tras concluir, el héroe renacerá completamente.

- Nivel 1: 20 segundos
- Nivel 2: 30 segundos
- Nivel 3: 40 segundos
- Nivel 4: 50 segundos

d. Valores héroe Orco

En cuanto a los valores concretos predefinidos para el héroe orco, aquí exponemos todos los *stats* en cada nivel, actualizados a medida que avanza una partida.

Héroe Orco	Nivel 1	Nivel 2	Nivel 3	Nivel 4
Vida	375	550	725	900
Ataque físico	30	45	60	75
Ataque mágico	25	35	45	55
Velocidad de ataque	0.9	1	1.1	1.2
Defensa física	25	30	35	40
Defensa mágica	20	25	30	35
Maná	175	250	350	500
Adrenalina	150	250	350	450
Velocidad de movimiento	50	55	60	65

Y a continuación, esta tabla muestra el coste de maná o adrenalina de cada habilidad.

Ataque especial	Adrenalina	Daño físico	Maná	Daño mágico	Cooldown (segundos)
1	-	-	50	50	5
2	75	75	-	-	10
3	150	150	-	-	20

e. Valores héroe Robot

En cuanto a los valores concretos del héroe robot, aquí exponemos todos los stats en cada nivel, actualizados a medida que avanza una partida.

Héroe Robot	Nivel 1	Nivel 2	Nivel 3	Nivel 4
Vida	400	600	800	1000
Ataque físico	25	35	45	55
Ataque mágico	30	45	60	75
Velocidad de ataque	0.8	0.9	1	1.1
Defensa física	20	25	30	35
Defensa mágica	25	30	35	40
Maná	200	300	450	600
Adrenalina	125	200	275	350
Velocidad de movimiento	45	50	55	60

Y por último, esta tabla muestra el coste de maná o adrenalina de cada habilidad.

Ataque especial	Adrenalina	Daño físico	Maná	Daño mágico	Cooldown (segundos)
1	-	-	50	50	5
2	75	75	-	-	10
3	150	150	-	-	20

3. Valores de las torretas

Cuando explicamos la distinción entre los dos tipos de torretas, la intención era dar más importancia a la parte estratégica del modo RTS, de este modo, las torretas neutrales para conquistar dispuestas en el mapa tienen distintos stats:

Torreta	Vida	Ataque	Vel. de ataque	Defensa	Alcance
Torreta conquistada	450	75	1.1	40	80
Torreta construida	300	50	0.9	30	70

4. Objetos, tesoros y/o recompensas

Los objetos expuestos, de forma aleatoria, al comienzo de una partida serán alguno de los definidos a continuación:

Objeto	Descripción
Cáliz Rojo	Aumenta el daño físico del héroe un 10%
Elixir Rojo	Aumenta la defensa física del héroe un 15%
Cáliz Azul	Aumenta el daño mágico del héroe un 10%
Elixir Azul	Aumenta la defensa mágica del héroe un 15%
Pócima Roja	Recupera un 25% de la adrenalina máxima
Pócima Azul	Recupera un 25% del maná máximo
Pócima Verde	Recupera un 25% de la vida máxima
Botas	Aumenta la velocidad del héroe un 10%
Guanteletes	Aumenta la velocidad de ataque físico del héroe un 20%
Capa	Aumenta la velocidad de ataque mágico del héroe un 20%
Anillo	Aumenta el cansancio máximo del héroe un 10%
Pendientes	Aumenta el maná máximo del héroe un 10%

d. El diseño de los mapas

El mapa se ha diseñado para que sea lo más simétrico posible, de modo que ambos jugadores empiecen en las mismas condiciones. Como se puede ver en la imagen (Figura 4.11) el mapa está dividido por un río, el cual solo puede ser saltado por las unidades exploradores, mientras que el resto de unidades tendrán que usar los dos puentes para cruzar de un lado al otro, provocando así que los puentes sean la principal zona de batalla. Para este punto, ha resultado de gran ayuda el libro *The Art of Game Design: A book of lenses* (Schell, 2008) que cuenta con un gran capítulo dedicado al diseño de mapas.

Figura 4.11: Diseño del mapa.

En referencia a las posiciones del comienzo de los equipos, las bases están situadas en esquinas opuestas, de modo que el comienzo y final del río quedan en las otras dos esquinas.

En cada una de las partes simétricas del mapa se encuentran el mismo número de torretas para conquistar y minas de recursos. Además se han diferenciado calles mediante carreteras y una serie de obstáculos por todo el territorio, como por ejemplo árboles, coches, etc, y en referencia a la estética pintoresca se puede destacar la existencia de canchas de baloncesto, lagos, neumáticos, vallas de obra, etc.

e. Diseño Software

En esta sección se trata el diseño de las unidades de los modos RTS y MOBA, y de los edificios, y se definen y exponen cada uno de sus componentes. Para realizar los diagramas de estados y los UML se ha utilizado las referencias mostradas en la página web *Allen Holub's UML Quick Reference* (Holub, 2011).

Además se recomienda consultar la documentación sobre Implementación tratada en la otra memoria desarrollada de este proyecto *Tecnología e Implementación* (Gálvez Ruiz, Miranda Esteban, & Monasterio Martín) para obtener una mejor visión sobre la implementación de cada tipo de unidad y de edificios, y sus características dentro del videojuego.

i. Resumen

Al comienzo del desarrollo del proyecto se han desarrollado dos prototipos por separado y en paralelo, correspondientes a ambos géneros de juego. En ambos prototipos se ha desarrollado poco a poco el diseño y la implementación.

El proyecto ha necesitado unos criterios de organización de archivos y carpetas, evitando una navegación complicada dentro del proyecto. Esta organización se ha adaptado a la estructura aportada por Unity, en la que, de forma automática, se crea una carpeta llamada *Assets* que en nuestro proyecto contiene todos los archivos del proyecto divididos en tres subcarpetas:

1. *Community Assets: assets* usados de la comunidad de Unity, como pueden ser los *assets* de Photon.
2. *Standard Assets: assets* propios que Unity permite importar, como los assets de los *skybox* o componentes del editor de terrenos.
3. *NewDetroit: assets* propios creados a lo largo del desarrollo, inlcluye todos los componentes pertenecientes a *prefabs*, *scenes*, *scripts*, etc…

Tras establecerse una jerarquía de carpetas, se ha necesitado crear una regla para nombrar los propios *assets*, evitando complejidad en las búsquedas dentro de la propia herramienta ya que estos objetos forman parte del videojuego y aparecen en la jerarquía de Unity:

```
[NombreSeparadoPorMayúsculas]_[fecha-version]_[TipoDeAsset]_[DatosAuxiliares]
```

En la parte RTS del videojuego se han diseñado características y funcionalidades de la mecánica del juego. En lo referente a funcionalidades se han diseñado la selección de unidades y edificios, y su control; el algoritmo de bandada de las unidades, su *pathfinding* y la herramienta para la medición de distancias; la cámara del modo RTS.

En la parte MOBA se ha diseñado la cámara en tercera persona y todo lo correspondiente a los héroes: habilidades, atributos, etc.

A pesar de haberse desarrollado de manera independiente, comparten algunos elementos en común pues, a pesar de las diferencias, las unidades RTS, los héroes y los edificios comparten vida, resistencia, sistema multijugador, *shaders*, etc.

ii. Diseño común

Todos los objetos del juego poseen componentes en común, y estos son:

- *Transform:* este componente se encarga de almacenar la posición, la rotación y la escala.
- *Collider:* este componente representa el colisionador, y puede ser tanto un cubo como una esfera (*BoxCollider*, *SphereCollider*).
- *Controller:* este componente se encarga de manejar a la unidad, es decir, de reaccionar frente a eventos del teclado o del ratón (moverse, atacar, etc), y de establecer la animación correspondiente.
- *CLife:* este componente se encarga de almacenar la vida.
- *ScriptNetwork:* este componente se encarga de enviar o recibir información a través de Photon. Envía información si es la instancia local y la recibe si es la instancia remota.
- *PhotonView:* este componente es necesario para que la unidad funcione a través de Photon.
- *CTeam:* este componente almacena el equipo al que pertenece el objeto.
- *FogOfWarUnit:* este componente se encarga de aclarar el plano de la niebla de guerra donde se sitúa el objeto.

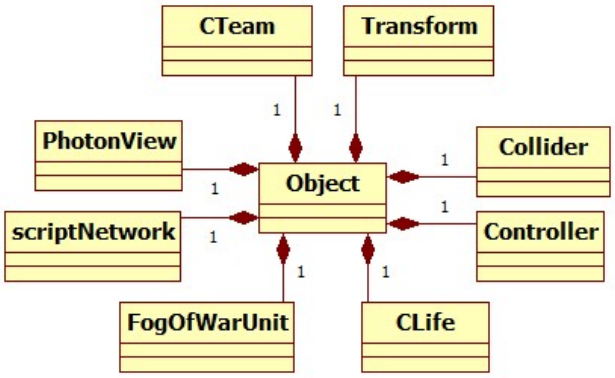


Figura 4.12: Componentes comunes de todos los objetos

Se considera importante mencionar que *scriptNetwork* sólo tiene efecto en el propio objeto, ya sea remoto o local. Por lo tanto, este script solamente envía información a sus propias instancias remotas, o recibe información de su instancia local.

Las unidades del juego poseen componentes extra en común con respecto a los explicados anteriormente, que son:

- *Animation:* este componente es necesario para las animaciones.
- *CStateUnit:* este componente se encarga de ejecutar la animación correspondiente.

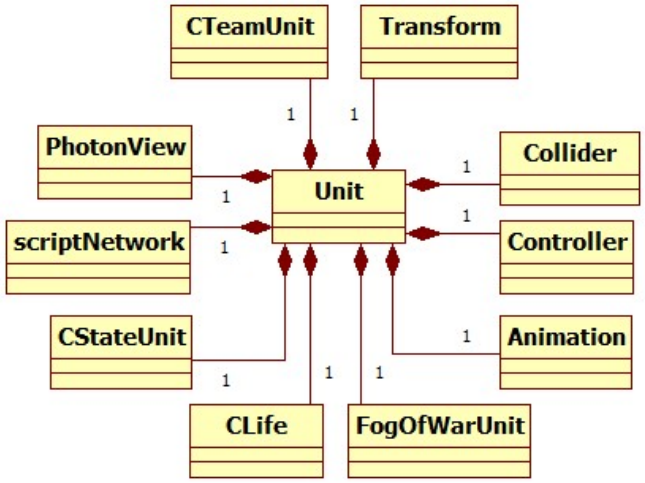


Figura 4.13: Componentes comunes de las unidades RTS y de los héroes

iii. Diseño RTS

Los objetos RTS del juego poseen un componente extra en común con respecto a los explicados al principio, que es:

- *CSelectable:* este componente es necesario para poder seleccionar los objetos.

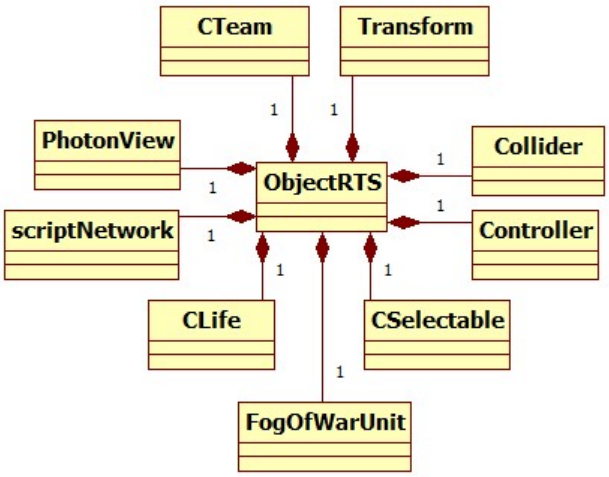


Figura 4.14: Componentes comunes de los objetos RTS

1. Unidades

Las unidades del modo de juego RTS poseen un elemento adicional con respecto al explicado anteriormente, que es:

- *NavMeshAgent:* este componente permite a la unidad desplazarse por la malla de navegación teniendo en cuenta los obstáculos que pueda haber (objetos móviles y objetos estáticos).

El script *Controller* del que heredan todas las unidades se llama ***UnitController.cs***, y contiene, entre otras cosas, el código necesario para que las unidades se muevan por el escenario del juego, paren cuando llegan a su destino, ataquen a enemigos, reciban daño, reciban curación, etc.

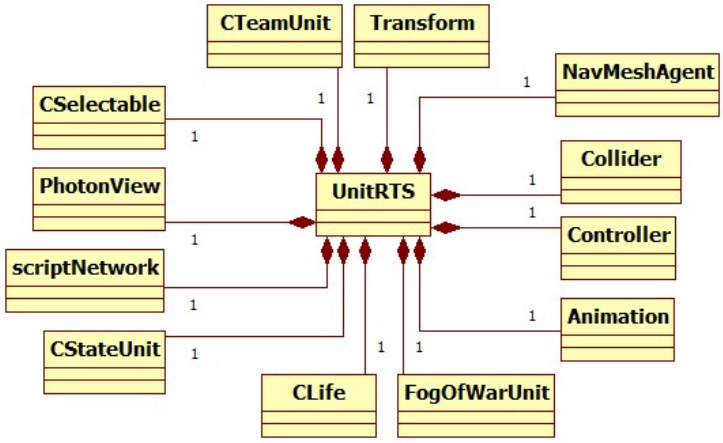


Figura 4.15: Componentes de las unidades RTS

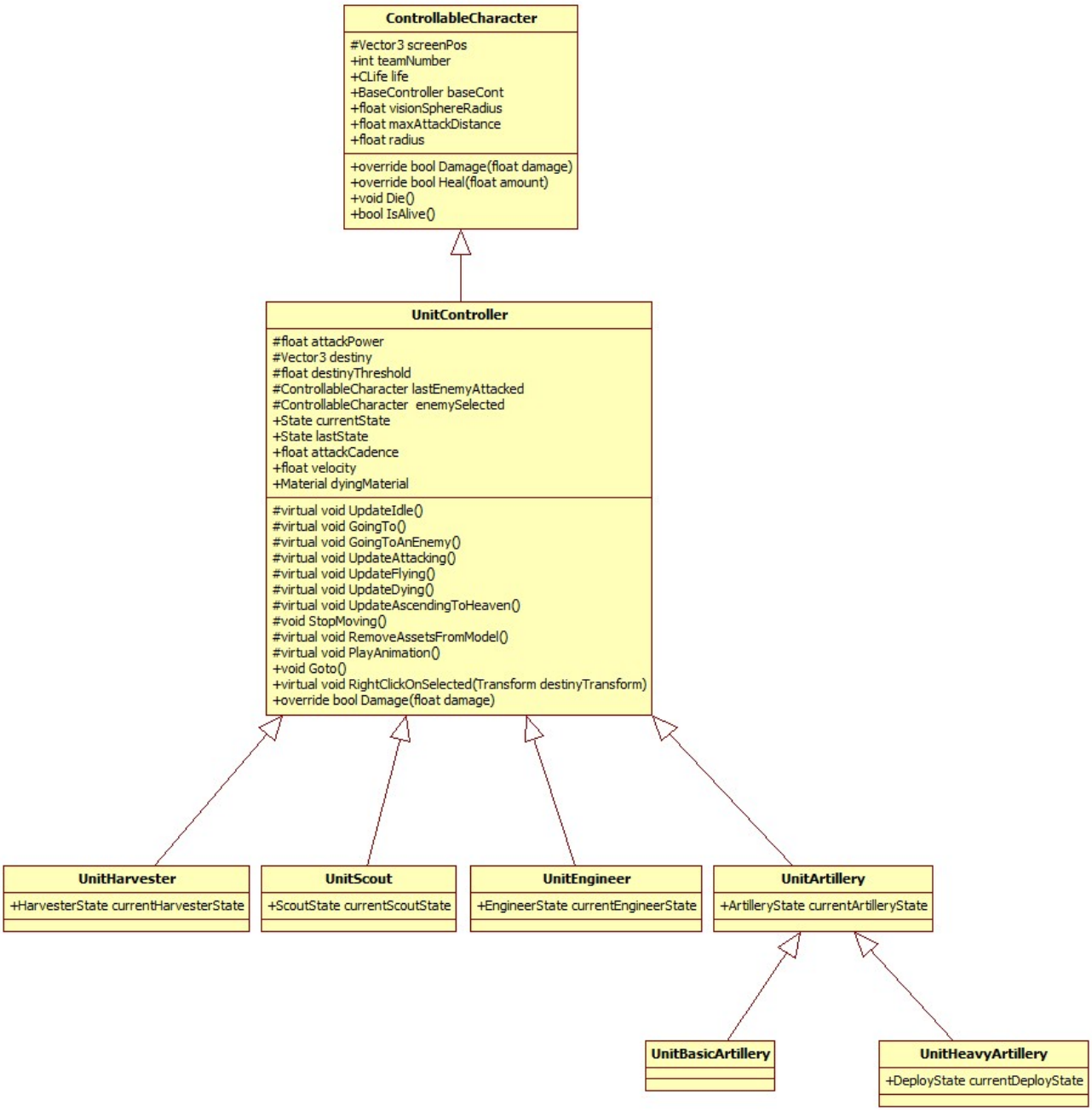


Figura 4.16: Diagrama de clases simplificado de las unidades del RTS

Tanto la clase padre de las unidades (*UnitController*) de los ejércitos como los héroes (*HeroController*) del juego heredan de una clase padre llamada *ControllableCharacter*, que contiene los atributos y métodos propios a todos los tipos de unidades controlables por el jugador del juego (como por ejemplo las referencias a los componentes *CLife* y *CTeam*, o métodos como el *Damage* para recibir daño).

De la clase *UnitController* heredan las clases que definen el comportamiento específico de cada tipo de unidad, sobreescribiendo los métodos esenciales para sus acciones concretas. Además estas clases contienen una nueva máquina de estados que extiende de la máquina de la clase padre.

Por último, de cada una de las clases concretas, heredan otras clases más por cada ejército diferente en el juego. Esto es necesario porque aunque dos unidades de un tipo tengan las mismas cualidades y funcionalidad, su modelado y armas son diferentes, y las referencias a los *dummys* y *GameObjects* que representan sus armas y accesorios son exclusivas de ese tipo de unidad de ese ejército en concreto.

2. La Clase UnitController

Esta clase contiene el comportamiento que comparten todos los tipos de unidades, sea cual sea su clase, e implementa la máquina de estados general de las unidades.

a. Máquina de estados

Debido a las diferentes características y comportamientos (alguno de bastante complejidad) de las unidades se opta por diseñar sus comportamientos en base a máquinas de estado en diferentes niveles. De esta manera, la clase padre (*UnitController*) contiene el nivel más básico de la máquina de estados de cada tipo de unidad con los estados que pueden alcanzar todas las unidades independientemente de su tipo (recolectora, artillería, etc.), y, después, en las clases hijas (las que representan el comportamiento de un tipo de unidad en concreto) una máquina de estados con los estados propios de cada clase.

Descripción de los estados:

- *Idle*: la unidad se encuentra en reposo esperando a recibir órdenes del jugador o a otros eventos.
- *GoingTo*: se dirige al destino marcado por el atributo *destiny*.
- *GoingToAnEnemy*: se dirige a un enemigo.
- *Attacking*: ataca a un enemigo.
- *Flying*: la unidad que ha recibido un ataque con explosión ha sido lanzada por los aires.
- *Dying*: este estado indica que la unidad está muriendo (su vida ha llegado a 0) y se está ejecutando la animación de muerte.
- *AscendingToHeaven*: estado previo a la desaparición de la unidad del juego, donde asciende mientras desaparece.

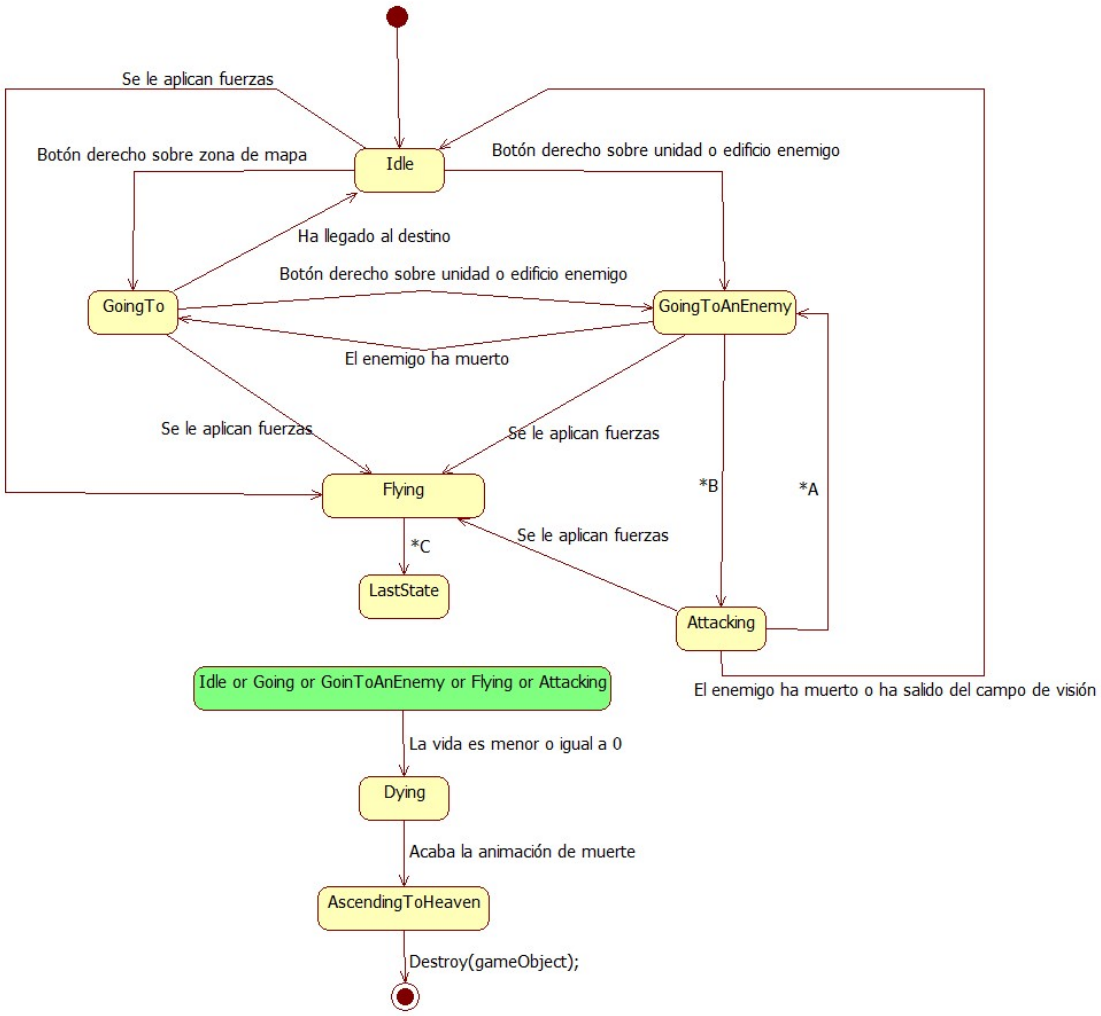


Figura 4.17: Diagrama de estados de la clase UnitController

3. Unidades recolectoras

El principal cometido de estas unidades es la recolección de recursos por el escenario para el ejército. Además, estas unidades también son capaces de realizar tareas de curación, tanto a otras unidades del ejército, como al héroe del equipo.

a. La clase UnitHarvester

Esta clase hereda de la clase *UnitController* y contiene una nueva máquina de estados más específica, extendiendo a la de la clase padre.

Máquina de estados

Para que se puedan especificar bien las acciones de recolección de estas unidades, se ha creado una máquina de estados propia, usando también la de la clase padre (*UnitController*), para poder efectuar estas acciones de manera efectiva. Estos estados tienen que recoger las acciones de recolección y de curación de unidades aliadas.

Descripción de los estados:

- None: en reposo, sin ninguna acción.

- GoingToMine: la unidad se dirige a una mina para comenzar a recolectar minerales.
- Waiting: la unidad se encuentra esperando a que la mina tenga lugares de extracción libres.
- GoingToChopPosition: la mina tiene un lugar de extracción libre y la unidad se dirige a este.
- Choping: se están extrayendo minerales.
- ReturningToBase: la unidad tiene recursos cargados y se dirige a la base (o a un almacén de recursos) para descargarlos.
- GoingToHealUnit: se ha seleccionado a un aliado para su curación y la unidad se dirige hacia su posición para sanarlo.
- Healing: sana a un aliado.

Transiciones:

- *None -> GoingToMine*: la unidad se encuentra en reposo y se ha seleccionado una mina para recolectar. Además, la cantidad de recursos con la que carga la unidad es menor a la cantidad máxima.
- *None -> ReturningToBase*: la unidad se encuentra en reposo y se ha seleccionado una mina para su recolección, pero la cantidad de recursos con la que carga es igual a la cantidad máxima con la que puede cargar. Vuelve a la base (o al almacén de recursos más cercano a la mina seleccionada) para dejarlos y, a continuación, ir a la mina seleccionada para comenzar otra extracción.

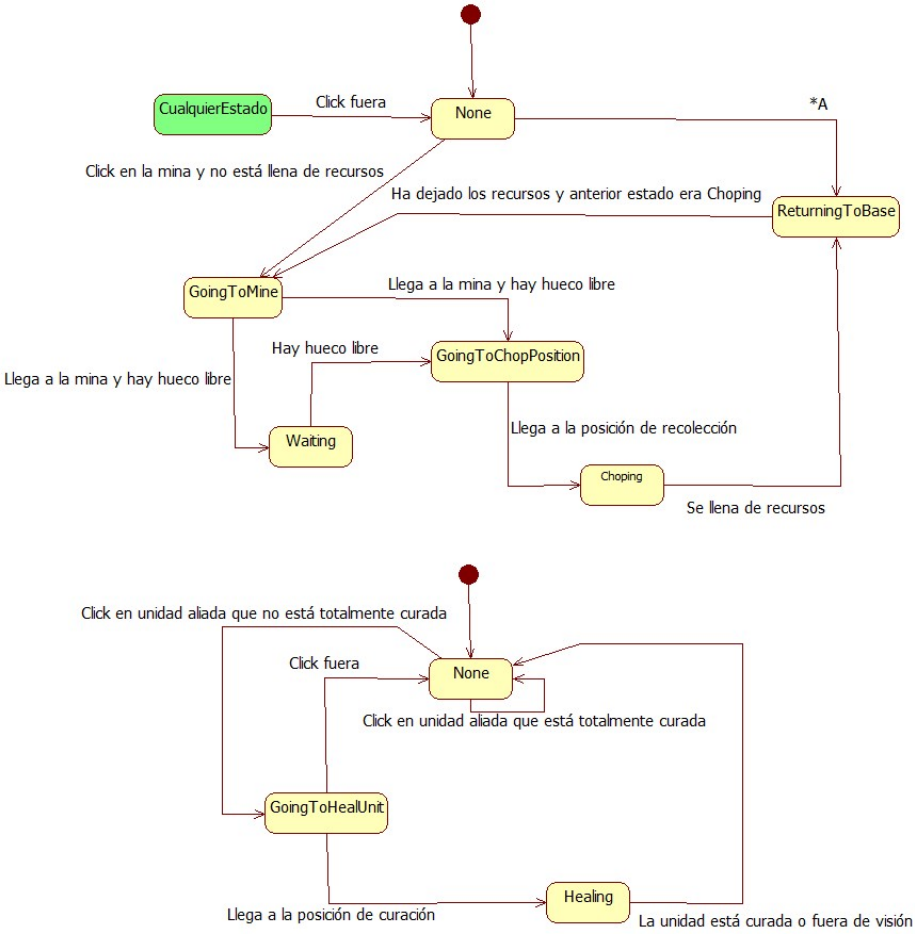


Figura 4.18: Diagrama de estados de las unidades recolectoras del RTS

4. Unidades ingenieras

Las funciones principales de estas unidades son las de construcción de torres y almacenes de recursos de su raza, conquista de torres neutrales y reparación de cualquier tipo de edificio. Además tiene un tipo de ataque característico, lanzando una bola de fuego que actúa como una granada, explota al pasar un tiempo y daña a las unidades enemigas en un radio determinado.

a. La clase UnitEngineer

Esta clase hereda de la clase *UnitController* y contiene una nueva máquina de estados más específica, extendiendo a la de la clase padre.

Máquina de estados

Estas unidades son las que tienen el comportamiento más complejo de todas, ya que pueden realizar muchas acciones (conquista, reparación, construcción y ataque) y se especifican como una máquina de estados para facilitar su implementación. Si no tuvieran esa máquina de estados la clase sería mucho más compleja, además de ser mucho más proclive a errores.

Descripción de los estados:

- *None*: sin ninguna acción.
- *GoingToRepairItem*: la unidad se dirige a un edificio para repararlo.
- *GoingToConstructItem*: la unidad se dirige a un zona/edificio para construir.
- *GoingToConquerableItem*: la unidad se dirige a un edificio para conquistarlo.
- *Waiting*: la unidad se encuentra esperando a que el edificio al que va a realizar alguna de las tres acciones posibles (reparar, construir o conquistar) tenga lugares libres.
- *GoingToConquestPosition*: el edificio tiene un lugar de conquista libre y la unidad se dirige a este.
- *GoingToRepairPosition*: el edificio tiene un lugar de reparación libre y la unidad se dirige a este.
- *GoingToConstructPosition*: el edificio tiene un lugar de construcción libre y la unidad se dirige a este.
- *Repairing*: la unidad se encuentra reparando un edificio.
- *Conquering*: la unidad se encuentra conquistando un edificio.
- *Constructing:* la unidad se encuentra construyendo un edificio.

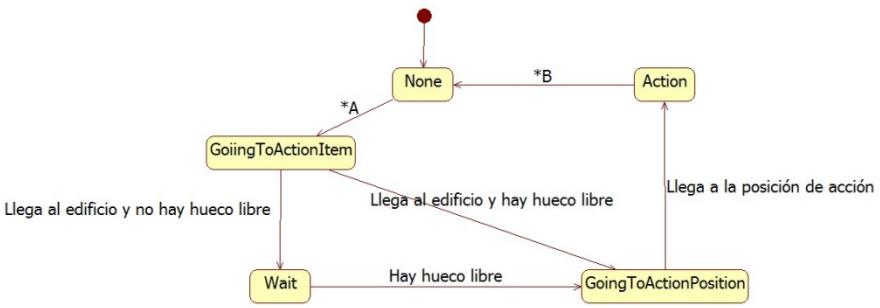


Figura 4.19: Diagrama de estados de las unidades ingenieras del RTS

5. Unidades artilleras

La función principal de estas unidades es atacar a todos los enemigos. Poseen dos tipos de ataque: las unidades básicas tienen ataque a distancia y ataque más rápido a corta distancia. La unidad pesada tiene ataque a distancia y ataque de artillería con una bomba que explota al pasar un tiempo y daña a las unidades enemigas en un radio determinado.

a. La clase UnitArtillery

Esta clase hereda de la clase *UnitController* y contiene una nueva máquina de estados más específica, extendiendo a la de la clase padre. Además, es un script donde está implementada la clase con su comportamiento y representa el principal componente. De esta clase heredan las que implementan las unidades básica y pesada: *UnitBasicArtillery* y *UnitHeavyArtillery*.

- *UnitBasicArtillery:* este componente define el comportamiento de la artillería básica.
- *UnitHeavyArtillery:* este componente define el comportamiento de la artillería pesada.

Máquina de estados

La clase *UnitBasicArtillery* y *UnitHeavyArtillery* tienen la misma máquina de estados, ya que las dos son las únicas unidades que pueden atacar de forma automática. Aparte la unidad de artillería pesada tiene una máquina de estados independiente para el tipo de ataque de despliegue.

Descripción de los estados:

- *None*: en reposo, sin ninguna acción.
- Alert: cada cierto tiempo (valor guardado en el atributo *alertHitTimer*) la unidad busca enemigos dentro de su radio de visión.

El estado *Attacking* está en verde porque es de la clase padre *UnitController*.

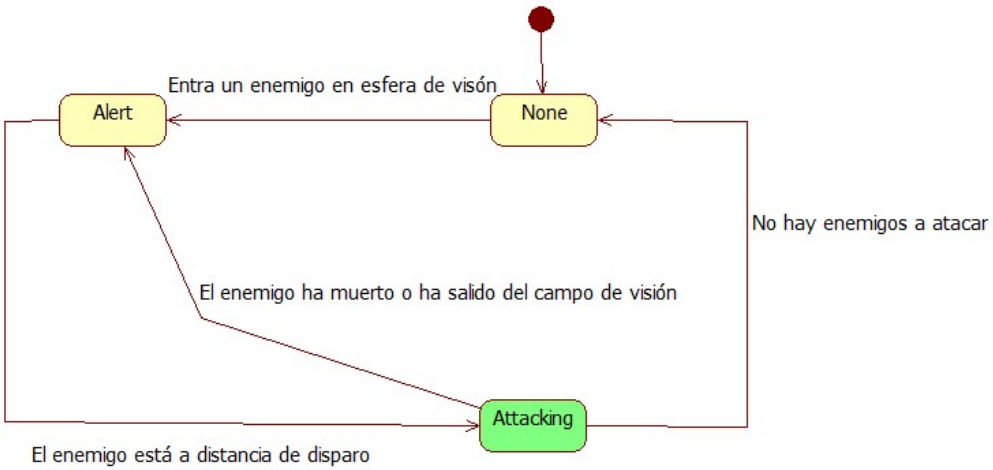


Figura 4.20: Diagrama de estados de las unidades artilleras.

La máquina de estados del modo despliegue de la unidad de artillería pesada tiene cuatro estados y empieza en modo *sin desplegar*.

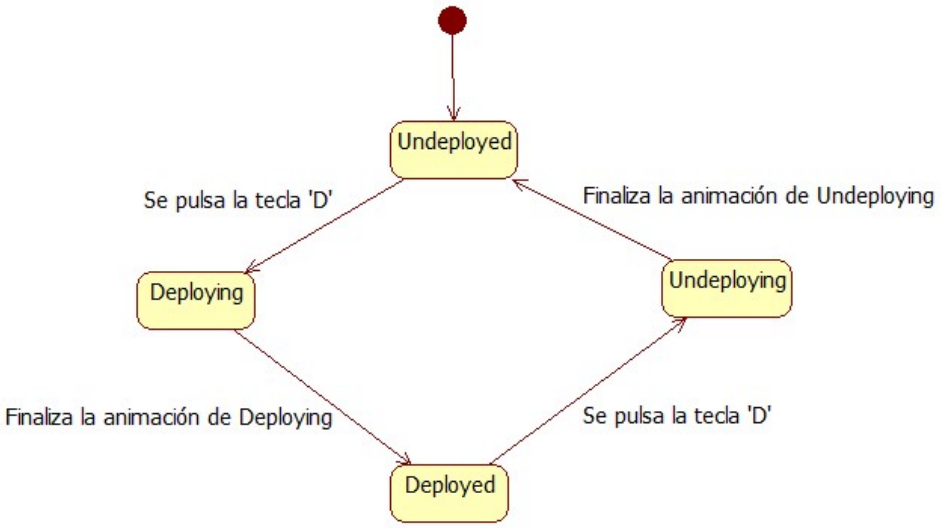


Figura 4.21: Diagrama de estados de las unidades artilleras pesadas del modo despliegue.

6. Unidades exploradoras

La labor principal de estas unidades es el descubrimiento del mapa gracias a su mejorada capacidad para desplazarse por el escenario. Son las únicas unidades capaces de atravesar ciertos puntos del mapa mediante un salto y se mueven más con una mayor agilidad. Además tienen la posibilidad de patrullar una zona de terreno.

a. La clase *UnitScout*

Esta clase hereda de la clase *UnitController* y contiene una nueva máquina de estados más específica, extendiendo a la de la clase padre.

b. Máquina de estados

El conjunto de estados que define el comportamiento de las unidades exploradoras es el siguiente:

Descripción de los estados:

- *None*: sin ninguna acción.
- *Patrolling*: patrulla entre una serie de puntos elegidos.

Transiciones:

- None -> Patrolling: la unidad se encuentra en reposo y se ha seleccionado una secuencia de puntos a patrullar.

7. Edificios

Los componentes de los edificios ya se han explicado anteriormente.

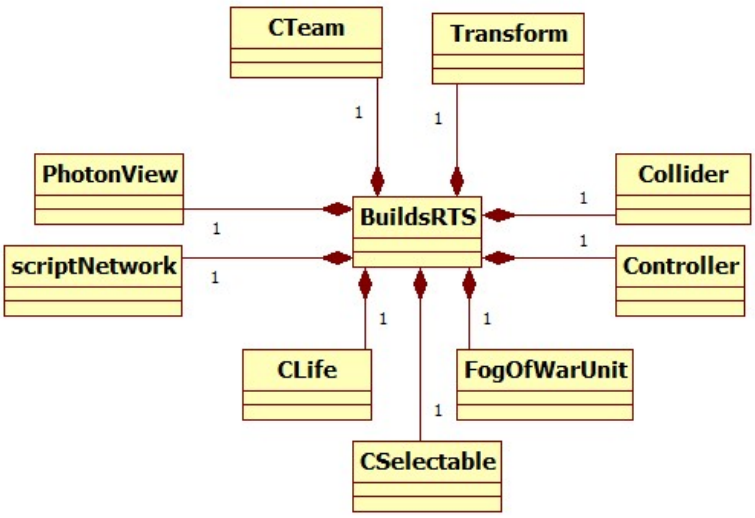


Figura 4.22: Componentes de los edificio.

Existen diversos edificios y los más importantes son: la base del ejército (única en toda la partida), las torretas de defensa y los almacenes de recursos. Todos pueden ser destruidos por las unidades enemigas, o el héroe del equipo enemigo, y pueden ser reparadas por las unidades ingenieras aliadas, además unos pueden ser construidos y otros pueden ser conquistados (como las torretas neutrales).

Figura 4.23: Aspecto gráfico de los 3 tipos de edificios de cada ejército (almacén de recursos, torretas y base).

Figura 4.24: Aspecto gráfico de una torreta neutral conquistable (izquierda), y del proceso de conquista por unidades ingenieras (derecha).

Hay una clase padre llamada *BuildingController* que a su vez hereda de *Photon.MonoBehaviour*. Las clases hijas de *BuildingController* son *Tower*, de la que heredan todas las clases de las torres, y *CResourceBuilding*, de la que heredan todas las clases que manejan recursos.

Los scripts *TowerNeutral* y *TowerArmy* heredan de *Tower* y serán los que se añadan en Unity a las torres neutrales y torres goblin, y torres robot respectivamente. Por su parte, *BaseController* y *Warehouse* heredan de *CResourceBuilding* y serán los scripts que se añadan a las bases y almacenes de recursos, respectivamente.

El diagrama de clases se ha dividido en dos para mejorar su visibilidad: los edificios que manejan recursos y las torres.

La clase *CResourceBuilding* es la que se encarga de incrementar y decrementar recursos, además mediante *numEngineerPositions*, *engineerPositions* y *engineerPosTaken* gestiona las colas para construir, reparar y conquistar, y también guarda una referencia a *ArmyController*. De ahí se ha dicho que heredan *Warehouse* y *BaseController:* la primera es para los almacenes que pueden ser construidos por las unidades ingenieras; la segunda es para las bases y son las que van a añadir nuevas unidades al jugador cuando se posean los suficientes recursos para ello.

La clase *Tower* es la que se encarga de todos los ataques de la torre que va a instanciar, además tiene un atributo que dice si la torre puede o no ser conquistada. De ahí se ha dicho que heredan *TowerNeutral* y *TowerArmy*: la primera es para las torres neutrales, que debe ser conquistada por un ejército u otro y tiene un contador por cada equipo que la está conquistando hasta que pase a la posesión del primer equipo que llegue a *finalCont*; la segunda se usa para las torres goblin y la torres robot, que pueden ser construidas por las unidades ingenieras.

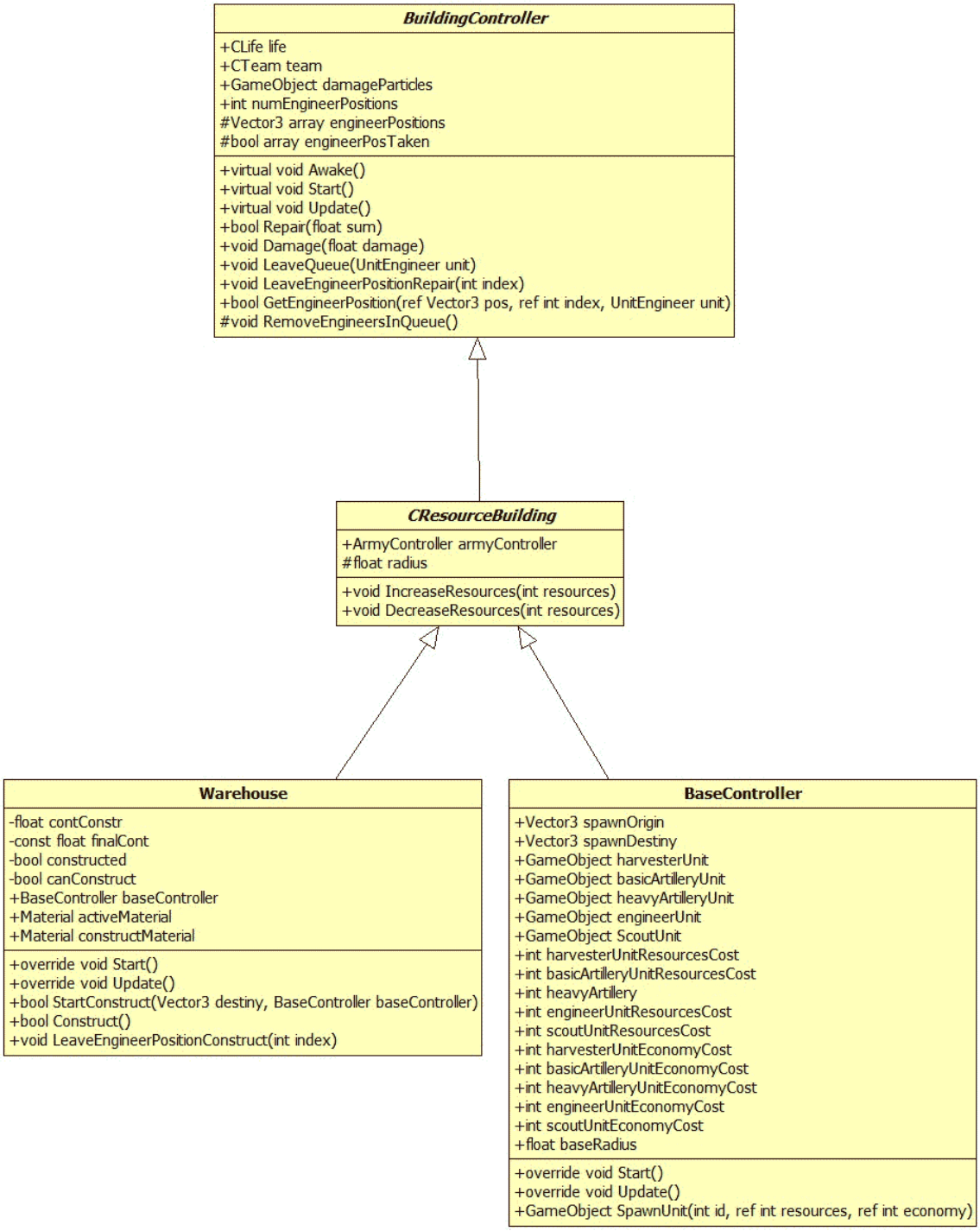


Figura 4.25: Diagrama de clases simplificado de los edificios de recursos del RTS.

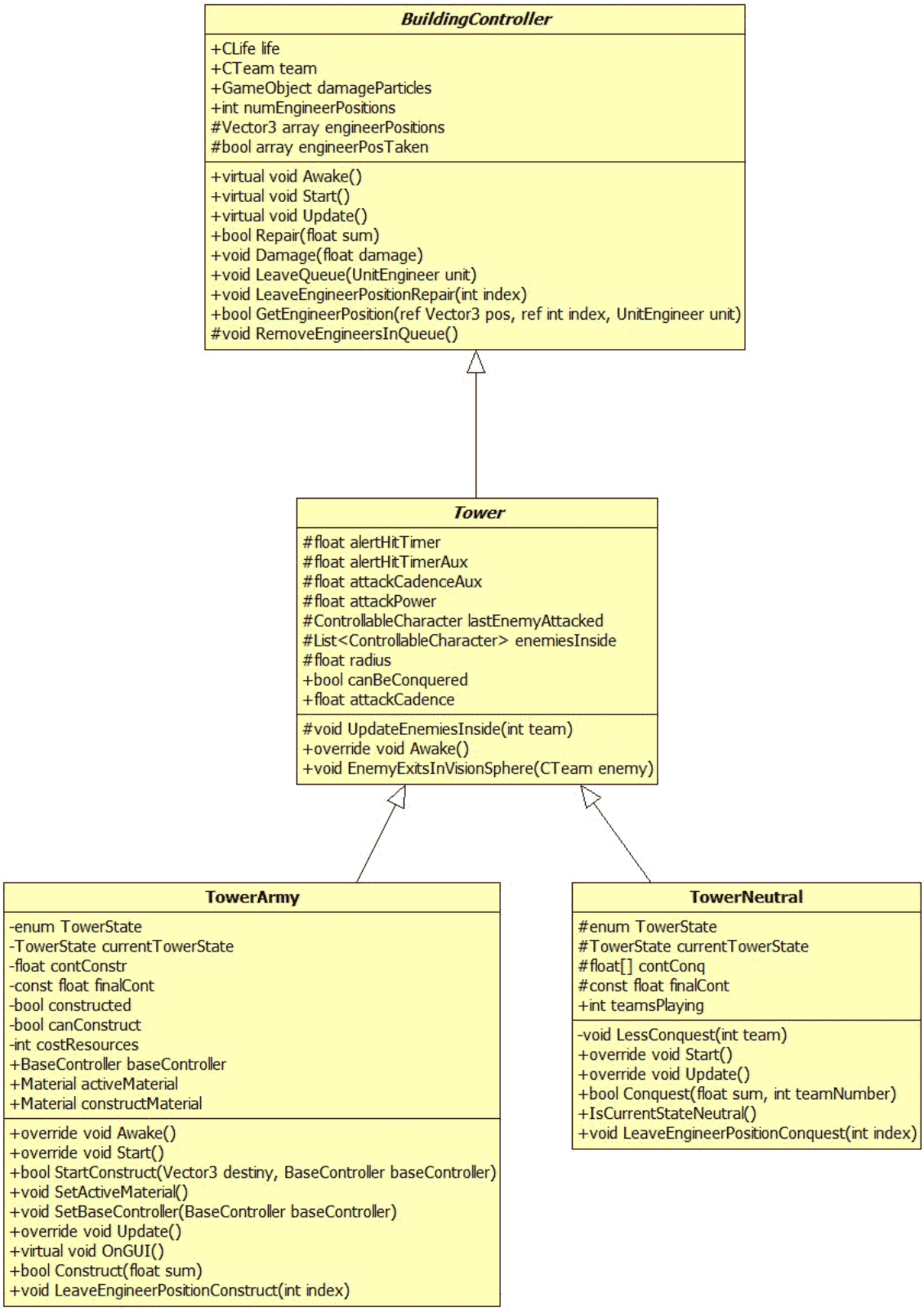


Figura 4.26: Diagrama de clases simplificado de las torres.

a. Máquina de estados de las torres

Las torres se comportan de una manera diferente al resto de edificios y es necesaria una máquina de estados para implementarla correctamente ya que pueden atacar. Las torres neutrales se pueden conquistar y las torres goblin y robot se pueden construir.

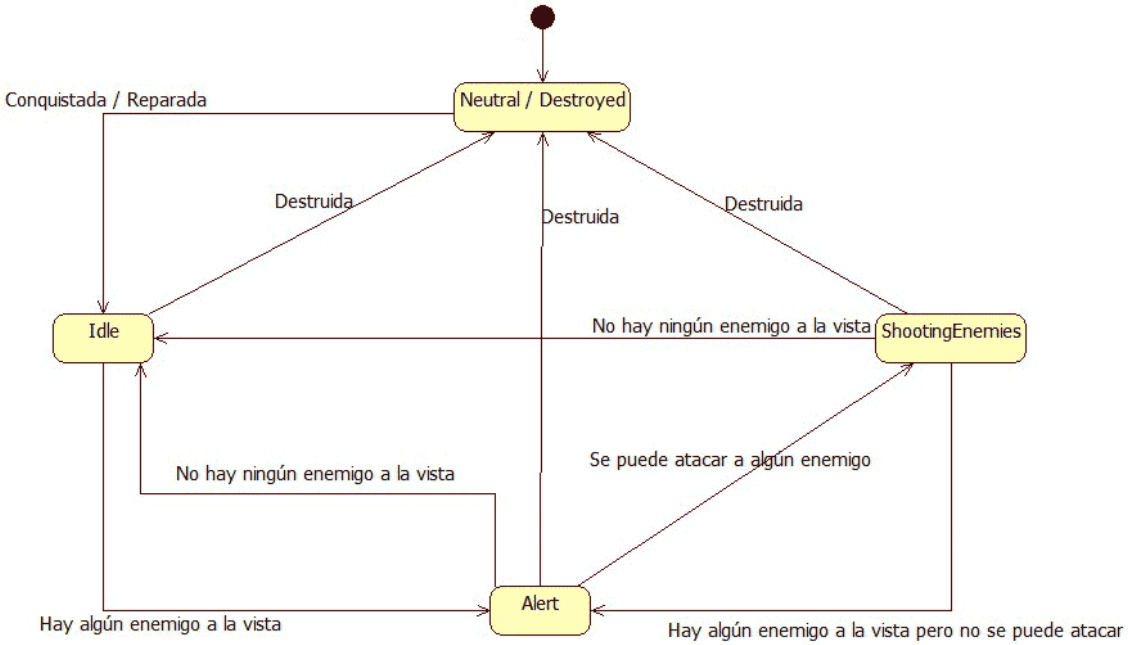


Figura 4.27: Diagrama de estados de las Torres.

b. ArmyController

El script *ArmyController*.cs, es uno de los scripts más importantes del juego, ya que se encarga de manejar todo lo relacionado a un equipo y, por lo tanto, habrá uno por cada equipo. También tiene como objetivo la gestión de las minas, los almacenes de recursos y la elección del mejor camino de retorno en la recogida de recursos. Por último tiene como misión fundamental la gestión de los eventos del teclado y del ratón sobre las unidades y edificios del equipo.

iv. Diseño MOBA

Todos los héroes poseen los siguientes componentes

- *Transform:* este componente se encarga de la posición, rotación y escalación del héroe.
- *CStateUnit:* este componente se encarga de ejecutar un cambio de animación, o una animación en la que ya esté.
- *ThirdPersonCamera:* este componente se encarga del comportamiento de la cámara, que básicamente sigue al héroe como una cámara en tercera persona.
- *CharacterController*: este componente es necesario para el manejo del héroe en tercera persona y proporciona su colisionador.
- *HeroeController:* este componente es común a todos los héroes y se encarga de almacenar y manejar sus atributos activos (ataque físico, ataque mágico, velocidad de ataque…), y también de actualizar sus estados (caminar, correr, atacar…). Se tiene

en cuenta que cada héroe posee un componente de control específico que es una clase que hereda de ésta (*OrcController*, *RobotController*).

- *CTeam:* este componente se encarga de relacionar al héroe con alguno de los bandos existentes en el juego (también es usado por cualquiera de las unidades del juego).
- *CBasicAttributesHero:* este componente se encarga de almacenar los valores de maná, adrenalina, nivel actual, defensa mágica y física del héroe, y además de pintar su GUI.
- *NavMeshObstacle:* este componente se encarga de considerar al héroe como un obstáculo a la hora de calcular el recorrido en las unidades RTS.
- *CLife:* este componente se encarga de manejar la vida del héroe (también es usado por cualquiera de las unidades del juego).
- *HeroNetwork:* este componente se encarga de enviar la información de nuestro héroe a través de la red.
- *FogOfWarUnit:* este componente se encarga de manejar la niebla de guerra para el héroe.
- *PhotonView:* este componente es necesario para la conexión a través de Photon.
- *Animation:* este componente contiene las animaciones del héroe.

El diagrama de componentes se muestra en la Figura 4.28.

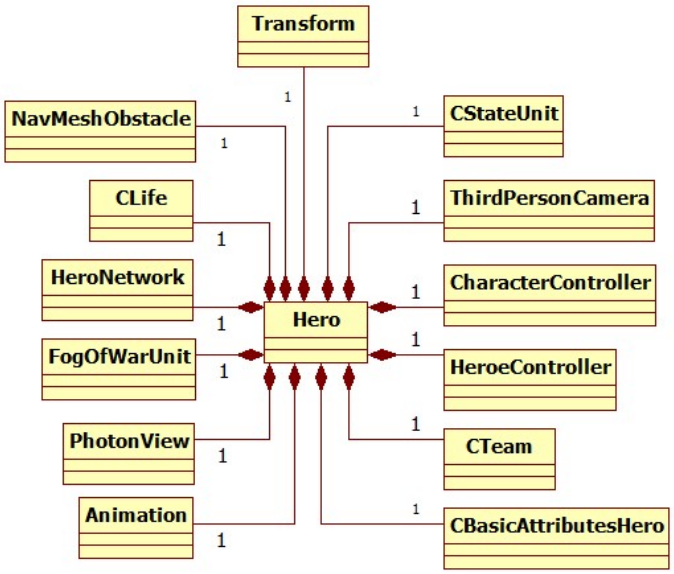


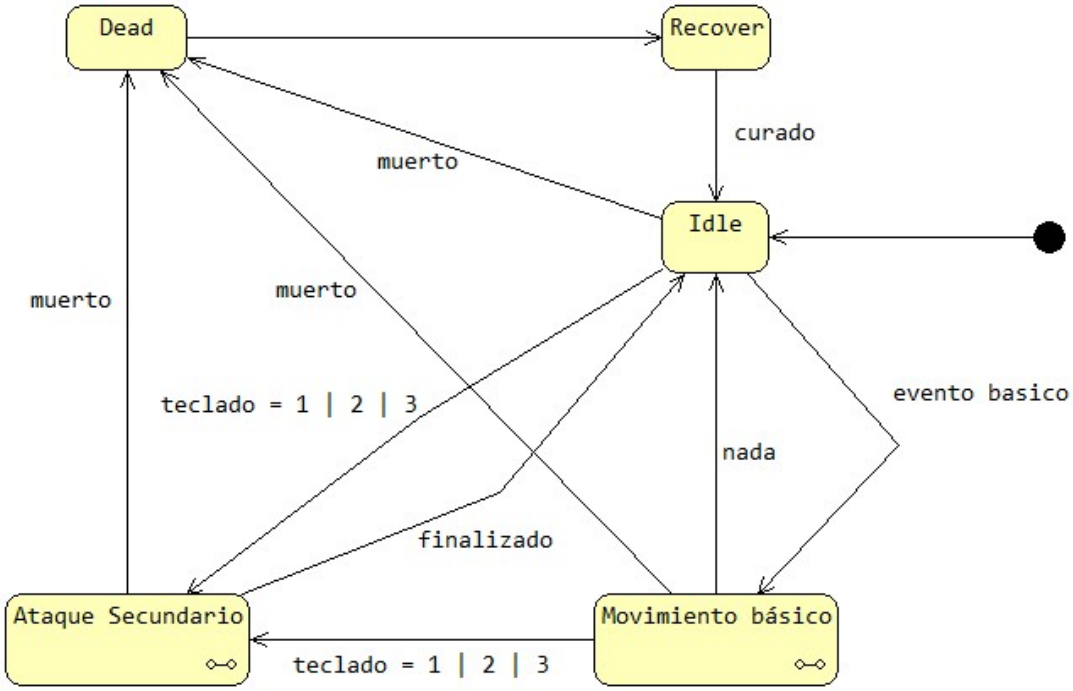
Figura 4.28: Diagrama de componentes del héroe.

1. HeroeController

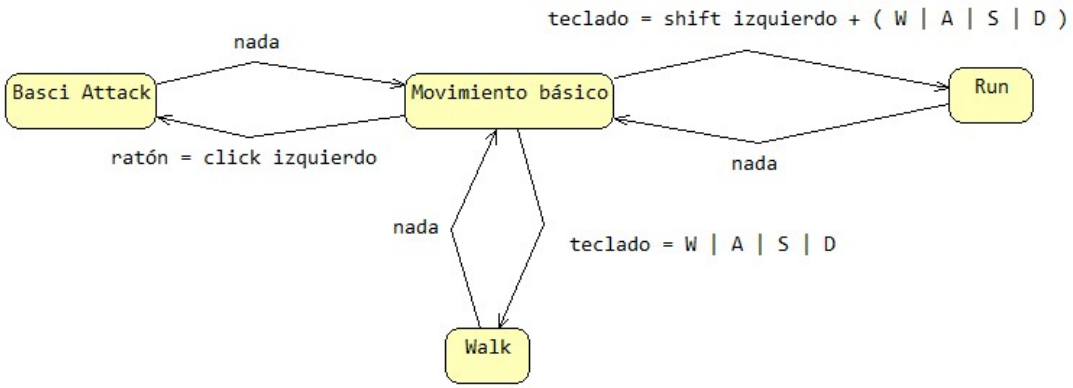
Este componente es una clase que hereda de la clase *ControllableCharacter.cs*. Se encarga de almecenar algunos atributos del héroe (ataque físico y mágico, velocidad de ataque, etc), y de reaccionar frente a los posibles eventos de teclado y ratón.

Una de sus partes importantes es el control que lleva sobre el estado de un héroe para poder animarlo después, haciendo uso del componente *CStateUnit.cs*. Todos los héroes del

juego poseen una máquina de estados común en función de la acción que el jugador quiera realizar. Las posibles acciones son la de reposo (*idle*), la de caminar (*walk*), la de correr (*run*), la de realizar un ataque básico (*basic attack*), la de realizar uno de los ataques secundarios (*secondary attack*), la de muerte (*dead*), y la de revivir (*recover*). A continuación se muestra el diagrama de como funciona la máquina de estados:



evento básico:
teclado = (shift izquierdo + (W | A | S | D)) | (W | A | S | D)
ratón = click izquierdo



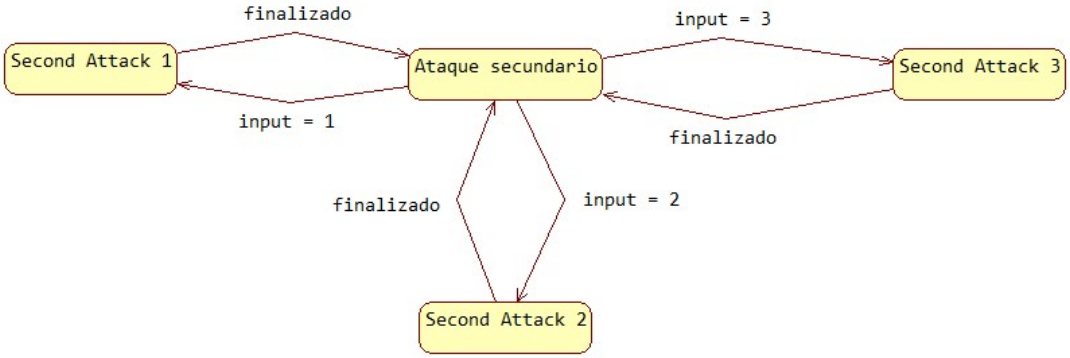


Figura 4.29: Máquina de estados del héroe.

El diagrama UML de esta clase se muestra en la Figura 4.30.

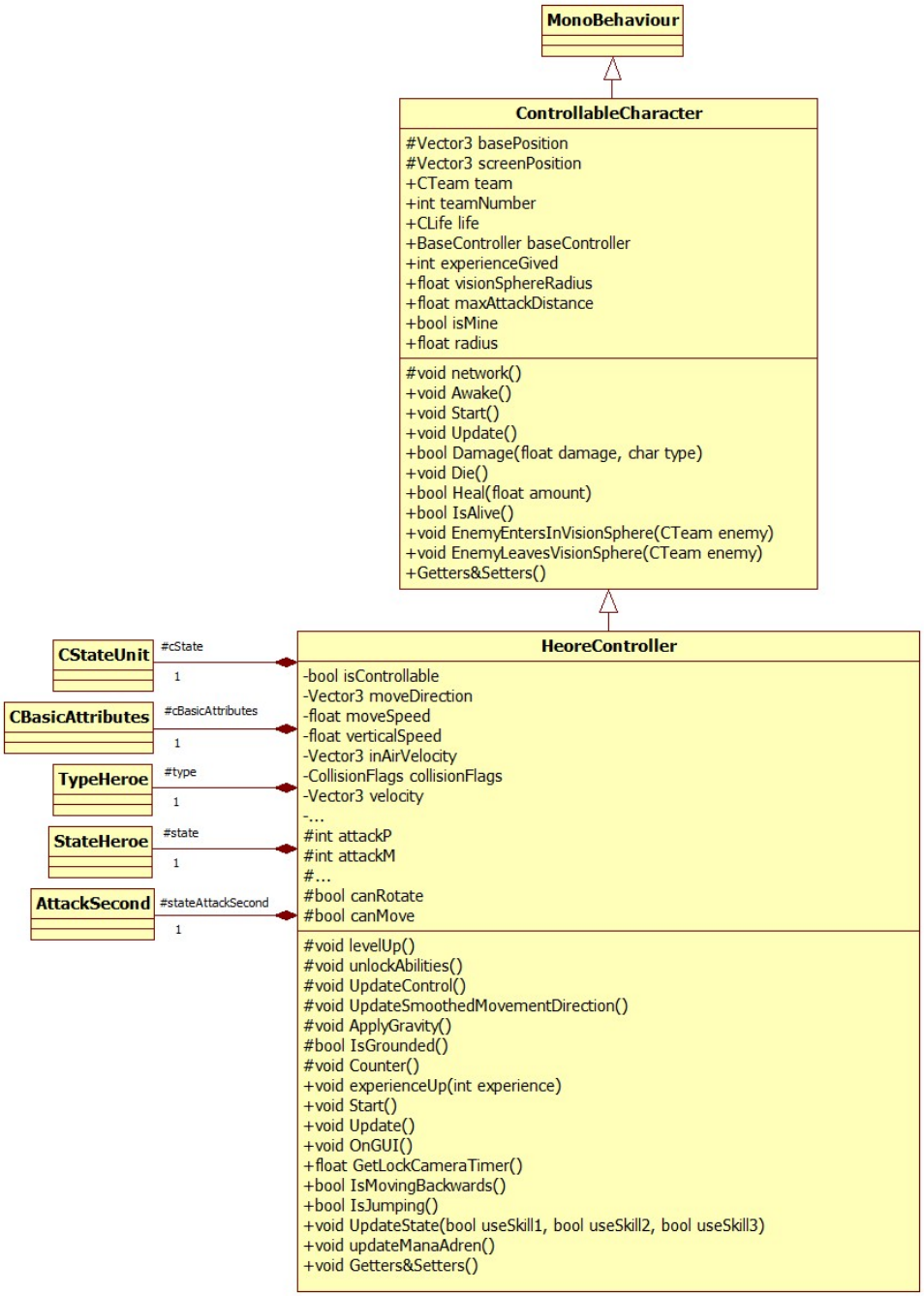


Figura 4.30: UML del script *HeroeController.*

2. BasicAttack

Para manejar el daño del ataque básico de ambos héroes se les incluye un colisionador en sus armas de ataque. Cada uno de los colisionadores tiene los siguientes componentes:

- *Transform*: esta componente almacena la posición, rotación y escalación.
- *BoxCollider*: esta componente es el colisionador en sí.
- *RigidBody*: esta componente sirve para que se detecte las colisiones contra los objetos.
- *BasicAttak*: esta componente es un script que se encarga de activar / desactivar el colisionador cuando sea necesario.
- *PhotonView*: esta componente es necesaria para instanciar el daño por red.
- *BasicAttackNetwork*: esta componente es necesario para activar los componentes necesarios, dependiendo de si la instancia es la nuestra o no.

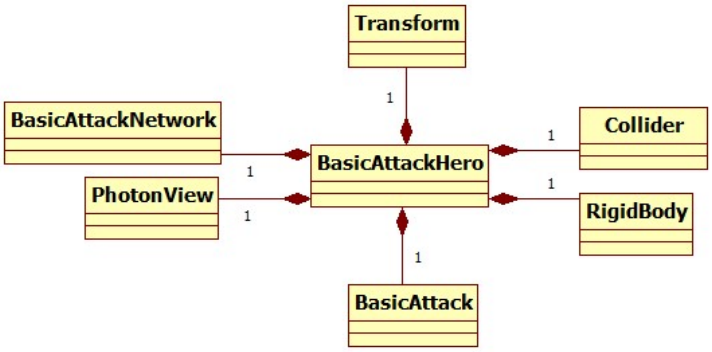


Figura 4.31: Componentes del ataque básico.

El diagrama UML de este script se muestra en la Figura 4.32.

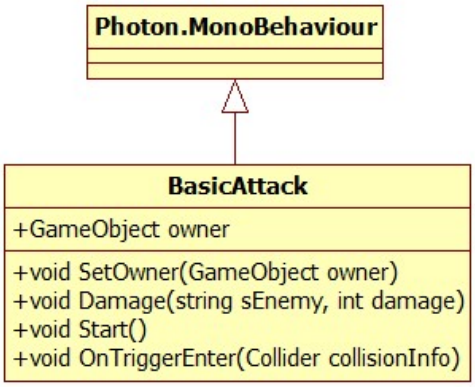


Figura 4.32: UML del script *BasicAttack.cs.*

3. Skills: SkillAttack y SkillDefense

Para realizar las habilidades de los héroes se distinguen 2 tipos, habilidades activas y habilidades pasivas. Las habilidades activas son aquellas que realizan daño, y las pasivas son aquellas que modifican los atributos del héroe.

En las habilidades activas, a la hora de realizar el daño, lo primero que se hace es detectar la colisión contra un enemigo, y luego se le aplica el daño correspondiente. Para detectar la colisión contra un enemigo se hace de dos formas distintas:

- *OnTriggerEnter*: esta detección se hace cuando las partículas no tienen activadas la opción de colisión, y entonces se les añade un componente *collider*.
- *OnParticleCollision*: esta detección se hace cuando las partículas tienen activadas la opción de colisión.

Las componentes que se usan en los dos casos anteriores son:

- *Transform*: este componente lleva la posición, rotación y escalación de las instancias de las habilidades.
- *ParticleSystem*: este componente es el que da forma a las partículas.
- *Collider*: este componente es el colisionador. Solo esta presente en las habilidades activas que no usan las partículas como colisión.
- *RigidBody*: este componente es necesario para detectar la colisión, y va ligado a *Collider*.
- *SkillAttack / SkillDefense*: este componente es el script que maneja la lógica de cada habilidad.
- *PhotonView*: este componente es necesario para instancias las habilidades por red.
- *SkillAttackNetwork / SkillDefenseNetwork*: este componente es el script que se encarga de decidir los componentes que están activos en cada instancia.

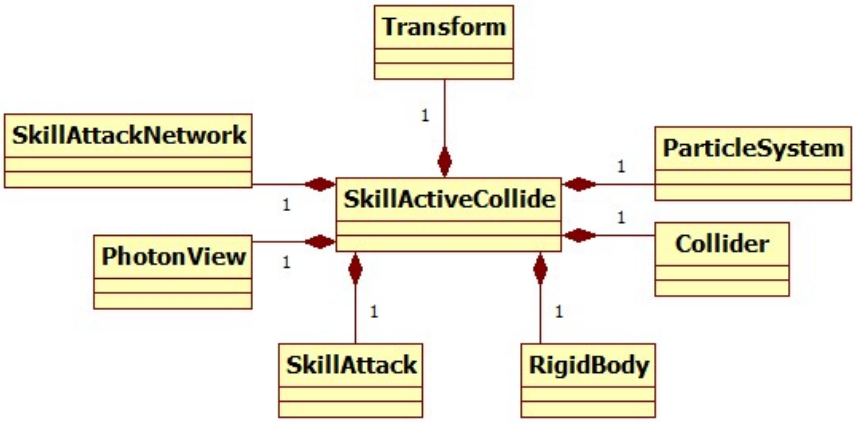


Figura 4.33: Componentes de las habilidades activas con colisionador.

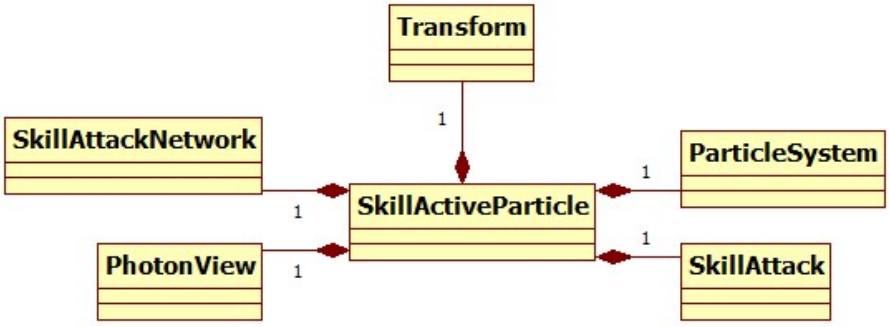


Figura 4.34: Componentes de las habilidades activas con colisionador de partículas.

El diagrama UML del script *SkillAttack* se muestra en la Figura 4.35.

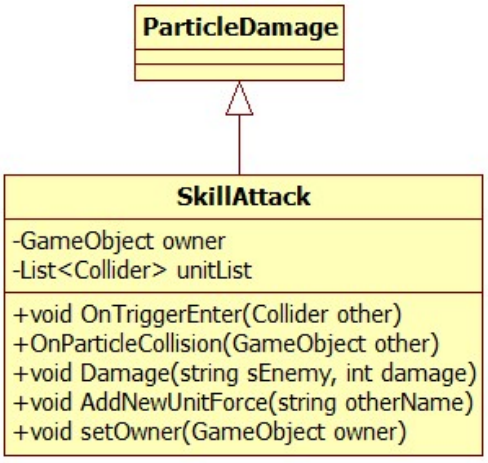


Figura 4.35: UML del script *SkillAttack.*

En las habilidades pasivas se utilizan los siguientes componentes:

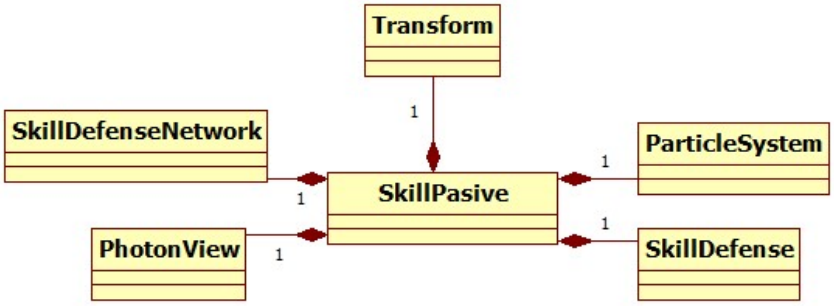


Figura 4.36: Componentes de las habilidades pasivas.

El diagrama UML del script *SkillDefense* se muestra en la Figura 4.37.

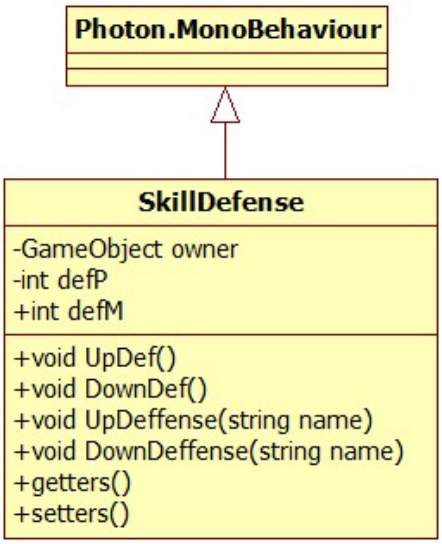


Figura 4.37: UML del script *SkillDefense.*

4. ParticleDamage

Todas las habilidades activas heredan del script *ParticleDamage.cs*, que es el encargado de almacenar la cantidad de daño que se realiza. El diagrama UML se muestra a en la Figura 4.38.

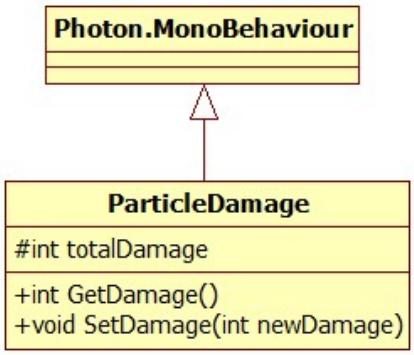


Figura 4.38: UML del script *ParticleDamage.cs.*

5. Diseño de los controladores

Se hace uso del script *OrcController.cs* y *RobotController.cs* para manejar sus animaciones.

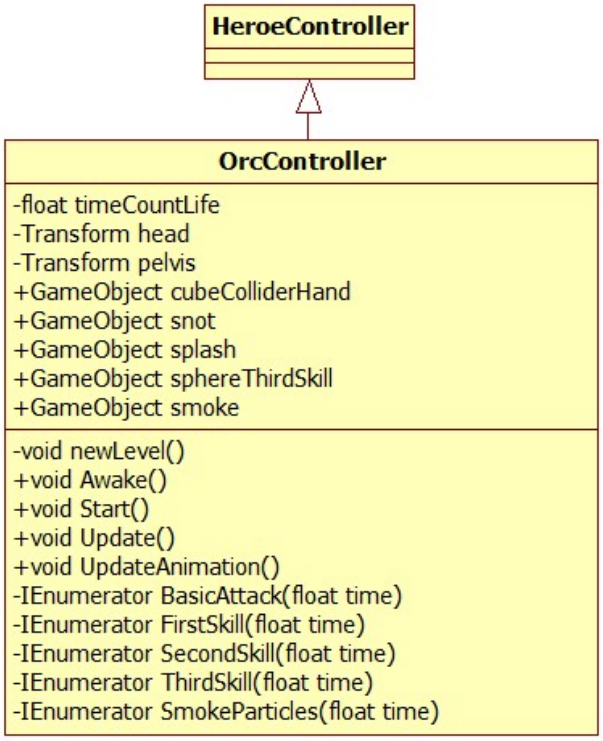


Figura 4.39: UML del script *OrcController.*

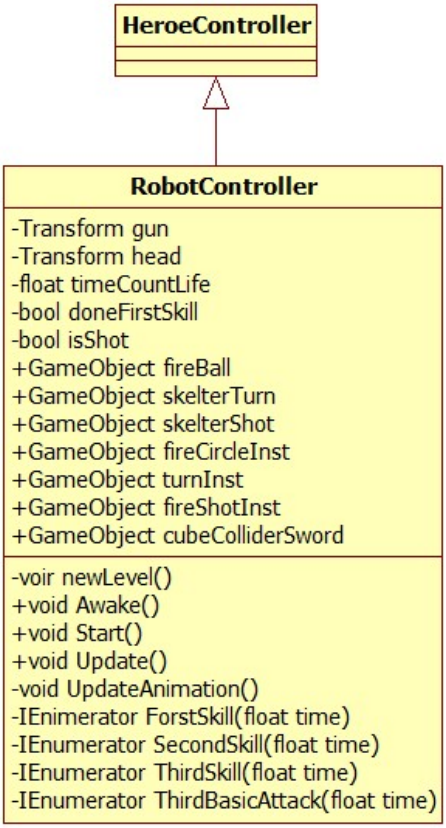


Figura 4.40: UML del script *RobotController*

5. Herramientas y Tecnologías

A la hora de elegir las herramientas tecnológicas se decidió utilizar Unity por la facilidad de uso, por su documentación oficial, por su amplia comunidad y por la potencia de su editor. Se eligió *C#* porque en Unity es más popular debido a estar mas documentado y ser de propósito general. Como herramienta de conexión se eligió Photon por su facilidad de uso, coste gratuito y ofrecer suficientes conexiones simultáneas.

En referencia a herramientas de soporte se han utilizado herramientas de control como *Trello*, herramientas de contenido como *Dropbox* o *Google Drive* y el repositorio *Github*.

a. Introducción a Unity

La tecnología principal que se ha usado para desarrollar este proyecto ha sido el motor para videojuegos Unity, utilizando además como lenguaje de programación *C#*. Unity es una herramienta para crear videojuegos y otras aplicaciones gráficas que requieran visualizaciones y animaciones en 3D en tiempo real. Cuenta con una potente interfaz gráfica que presenta un editor con múltiples herramientas que facilitan mucho el desarrollo de videojuegos en 3D. Además, Unity permite la ejecución de código en tiempo real, no teniendo que parar el juego para recompilar scripts.

En Unity siempre se trabaja sobre proyectos, y así mismo, sobre escenas, donde se pueden visualizar los avances. Cada escena puede considerarse un nivel del proyecto en cuestión conteniendo los *GameObjects* del juego y siempre podrán ser organizadas o accedidas mediante la carpeta Assets.

Para el manejo de Unity, y como ya hemos mencionado, está el editor. El editor es la clave de Unity, ya que es el elemento que marca la diferencia respecto a otras herramientas de creación de videojuegos gracias a sus numerosas opciones. El editor de Unity está compuesto por:

- Project Browser (explorador del proyecto): vista desde donde se puede acceder y gestionar los archivos que pertenecen al proyecto.
- Hierarchy (jerarquía): contiene los objetos de la escena actual.
- Toolbar (barra de herramientas): controles para manejarse en la escena.
- Scene View (vista de la escena): es la vista de carácter interactivo donde se puede seleccionar, posicionar y manipular todos los *GameObjects de la escena.*
- Game View (vista del juego): representa, en modo ejecución, el juego creado. Se pueden acceder a los *GameObjects* y manipularlos desde el editor, pero estos cambios son temporales y apreciables únicamente en ejecución.
- Inspector: muestra todas las características y propiedades de los *GameObjects*, donde se pueden modificar y configurar sus atributos.

Una de las mayores cualidades de esta plataforma de creación de videojuegos es la arquitectura por componentes, que podemos definir a grandes rasgos como la capacidad de añadir o quitar componentes a los *GameObjects* otorgándoles así nuevas funcionalidades o características. Se pueden añadir cualquier número de componentes a un *GameObject*, así como activar o desactivar dichos componentes en ciertos momentos, otorgando al motor gran versatilidad, y, sobre todo, una gran simplificación en el diseño software.

Como ya se mencionó al comienzo del apartado, el lenguaje elegido para implementar el proyecto es C#, el cual, no se ha estudiado por ninguno de los miembros durante la carrera y se ha necesitado un proceso inicial de aprendizaje.

A la hora de implementar se han utilizado dos entornos:

- Monodevelop: es un entorno (de desarrollo integrado) multiplataforma y principalmente diseñado para C# que se sincroniza con Unity de manera que se puede depurar el código con *breakpoints*, *watches*, etc… Este entorno se utilizó sobre todo en las primeras etapas del proyecto.
- VS 2012 (*Microsoft Visual Studio 2012*): es un entorno con el que ya habíamos tenido contacto durante la carrera, ofrece las mismas funcionalidades que Monodevelop, aunque esta en desventaja a nivel de depuración. Por este motivo se usa el plugin UnityVS, que ofrece esta capacidad.

Por último, la conexión en red se ha llevado a cabo mediante la plataforma de conexión para videojuegos en tiempo real Photon Network, que nos permitía acceso gratuito de hasta 20 conexiones paralelas. El especial aliciente para su uso es debido a que Unity tiene integración con Photon Network. Además, se ha dispuesto de tutoriales para cualquier usuario.

Para conocer más, las tecnologías utilizadas durante el proceso de desarrollo de este proyecto han sido ampliamente documentadas en la memoria *Tecnología e Implementación* (Gálvez Ruiz, Miranda Esteban, & Monasterio Martín).

b. Herramientas de organización

Algo imprescindible en un proyecto de tanta envergadura y con un número elevado de miembros es la sincronización y la organización. Por esto se han tenido que investigar y usar diferentes herramientas.

GitHub + Git Extensions

El código del proyecto está almacenado en el repositorio software *Github (*Figura 5.1*)*, es de licencia libre y cualquier persona lo puede descargar. Además se ha trabajado con la herramienta *Git Extensions*, la cual da posibilidad de trabajar en paralelo con distintas ramas y operaciones. Es un entorno con el que el grupo del proyecto ya estaba familiarizado, ya que en el año anterior se usó para hacer la asignatura de Ingeniería del Software. De ese año se supo que era mejor trabajar en diferentes ramas, ya que sino, a la hora de hacer *push*, se encontraría con muchos *merge*, algo que puede ser muy complejo y dar muchos problemas.

En general trabajar con Unity y Github + Git Extensions es muy recomendable, a pesar de que si no se ha usado nunca un repositorio y un control de versiones, al principio puede ser muy complejo, pero cuando se conoce bien es de gran utilidad. Se han creado diferentes proyectos para poder trabajar por un lado con la parte *MOBA* y por otro con la parte *RTS*, además estas se han dividido a veces en distintos proyectos para investigar diferentes tecnologías o sistemas, como por ejemplo *pathfinding, algoritmo de bandada,* etc. Muchas veces varias personas están trabajando en el mismo proyecto, es decir con los mismos scripts, GameObjects, etc. Cuando se hace de esta manera hay un problema con las escenas: cuando se programaba sobre una escena y se hace *commit + push*, si se está trabajando por otro lado se tiene que hacer merge porque cambia la escena; por esto si se va a trabajar sobre un mismo proyecto continuadamente por varias personas se crean clones de las escenas, en las que cada miembro trabaja en una, y al hacer *commit + push* se actualizan bien todos los scripts, materiales, GameObjects, etc.

Se elaboró un sencillo .gitignore con los archivos que tiene que ignorar *Git + Extensions* a la hora de hacer *commit*, archivos que no son necesarios. Contiene básicamente estos: .DS_Store, Library, Temp,*.csproj, *.pidb, *.unityproj, *.sln, *.userprefs.

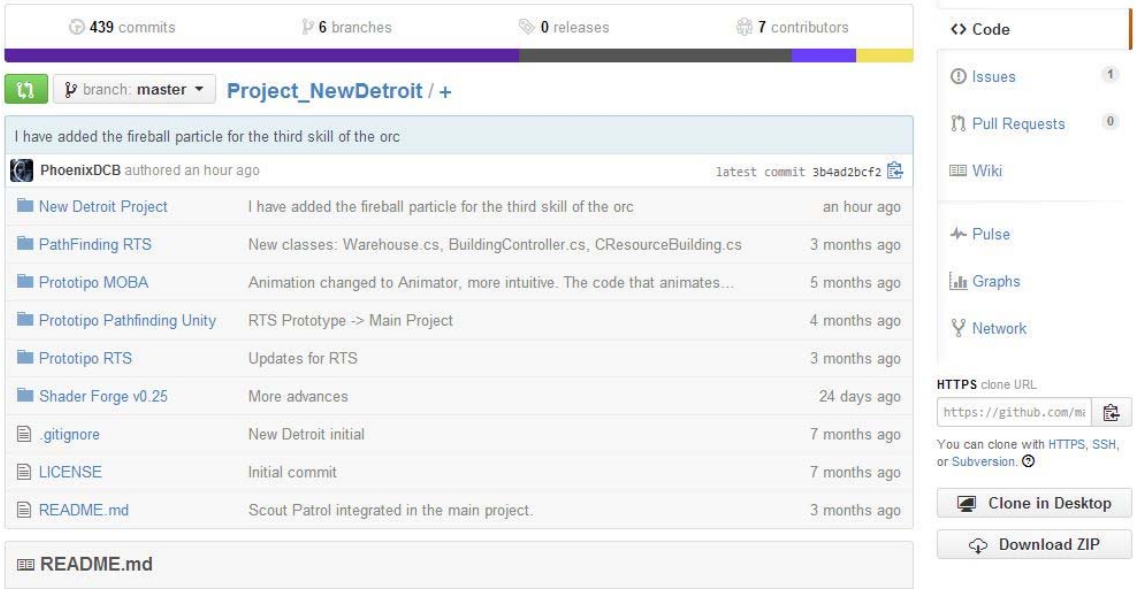


Figura 5.1: Aspecto del proyecto en *Github.*

Trello

Trello es una herramienta de organización y gestión de proyectos. En una metodología de desarrollo ágil como es SCRUM, es de gran utilidad, ya que esta herramienta deja crear nuevas tareas (basadas en los sprints por ejemplo), asignándose miembros y definiéndose una fecha de cierre.

La organización del trello del proyecto es muy intuitiva, cómoda y eficaz. En el proyecto (Figura 5.2) hay cinco listas en las que hay tareas:

- BackLog: tareas definidas pendientes de hacer, que no están en un sprint.
- To Do: tareas definidas en un sprint que están pendiente de hacer.
- Doing: tareas que se están realizando actualmente.

- Done: tareas que se han acabado, pero que no se ha comprobado su correcto funcionamiento en reuniones.
- Archived: tareas acabadas y comprobado su correcto comportamiento durante una reunión.

De esta forma, un ciclo completo de una tarea es: creación de la tarea en *BackLog*, después la tarea se define en el sprint con una fecha final y unos miembros asignados, con esto ya se pasa a *To Do* y una vez un miembro empieza a hacer la tarea se mueve a *Doing*; en cuanto finaliza la tarea a realizar se pasa a *Done*, después se comprueba en una reunión que la tarea realizada sea correcta y no haya errores y si es así se pasa a *Archived*.

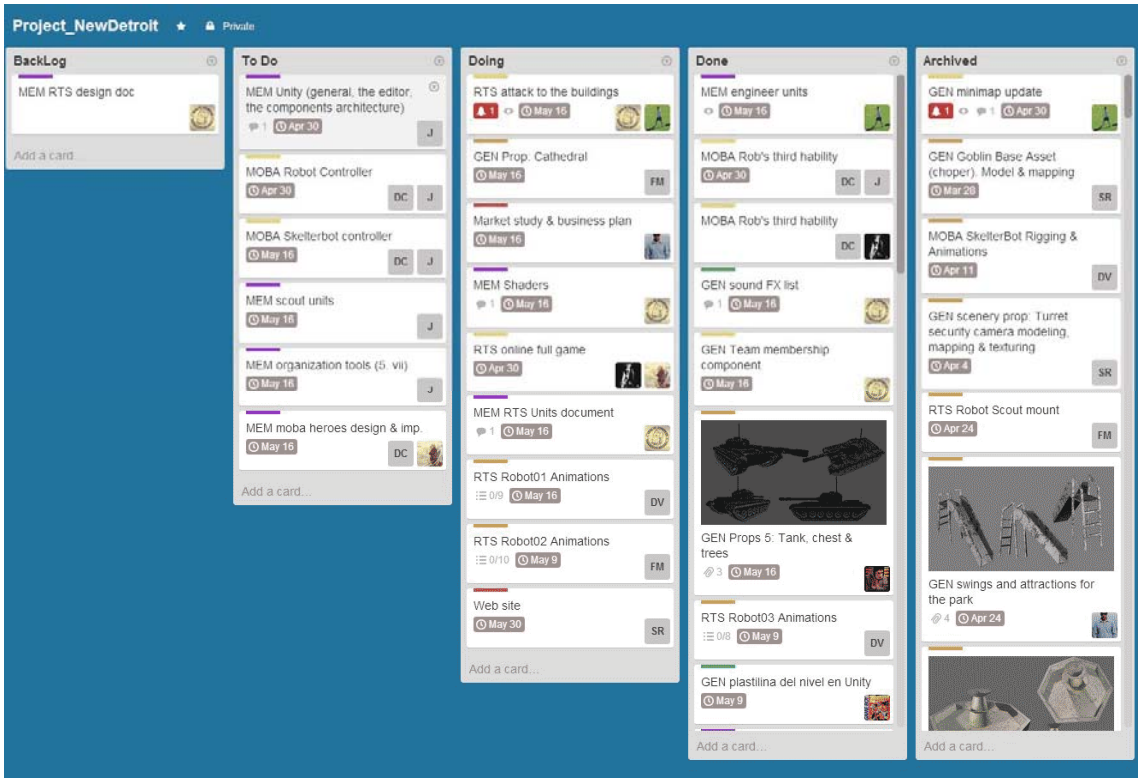


Figura 5.2: Aspecto del *Trello* del proyecto.

Google Drive

Google Drive es la herramienta usada tanto de repositorio para copias como para organización de la documentación. Una vez se ha ido desarrollando o investigando una nueva tecnología se ha ido creando un documento con toda la información relevante de esa tecnología. Como puede verse en la (Figura 5.3) el directorio *Project New Detroit* se ha dividido en diferentes carpetas:

- Art: carpeta dividida en *Art*, *Concept Art* y el listado de assets de audio y props gráficos. En la carpeta *Art* se encuentran los *assets* con sus correspondientes carpetas y archivos, y se utiliza como repositorio para copias de seguridad de los archivos de arte que se encuentran actualizados en *Dropbox*.
- Design: carpeta donde se encuentra toda la documentación relacionada con el diseño del videojuego, como por ejemplo: *GDD* (Documento de Diseño del Juego), Game Atributes, RTS Gameplay, etc.

- ESNE Resources: carpeta donde se guardan todos los archivos y recursos de la universidad de ESNE, así como la producción de esta parte.
- FDI Resources: carpeta donde se guarda todo lo relacionado con la memoria, como esquemas de memoria, normativa de Sistemas Informáticos, etc. No se guardan aquí los textos que se incluyen en la memoria.
- Game Engine: carpeta donde se encuentra la documentación para todas las herramientas y tecnologías utilizadas, divididas en diferentes documentos, como por ejemplo: *Fog of War* (niebla de guerra), algoritmo de bandada, física en Unity, manual de Photon, herramienta de medición de distancias, etc.
- Market Study: carpeta donde se encuentra toda la documentación acerca del estudio de mercado.
- Organization: carpeta donde se encuentran todas las actas de reuniones, con información de cada una y los sprints realizados. Además hay una hoja de contacto con toda la información de los integrantes del equipo.
- Production: carpeta donde se encuentran los documentos y archivos de producción, como por ejemplo: EDT (Estructura de División de Trabajo), *Burndown chart*, etc.
- Web page: carpeta con toda la información relevante de las páginas web del proyecto, como por ejemplo: información del servidor alojado, claves, correo, etc.

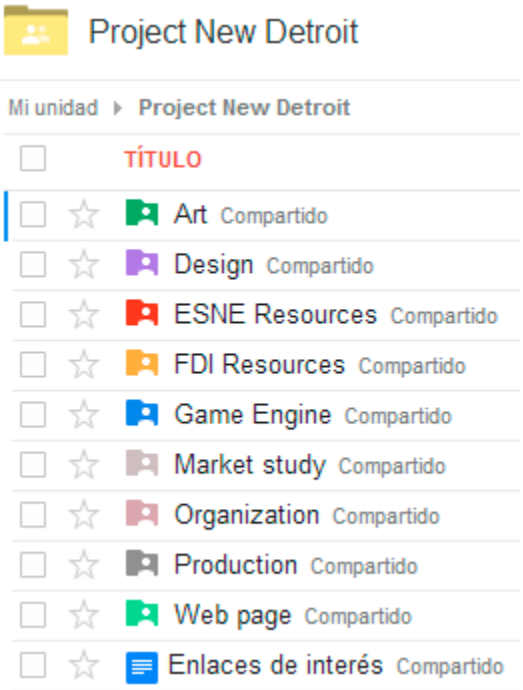


Figura 5.3: Aspecto del *Google Drive* del proyecto.

Dropbox

Dropbox (Figura 5.4) ha sido la principal herramienta a la hora de compartir los archivos de *assets* con los artistas que han ayudado en el proyecto. Además se guarda gran cantidad de documentación y ficheros útiles del proyecto. Se ha usado de manera más directa que *Google Drive*, ya que todos los componentes del equipo lo tenían instalado en su ordenador y a la hora de actualizar cualquier archivo se podía hacer con solo arrastrar el archivo de una carpeta a

otra. Tiene un control de versiones, de manera que si un fichero o documento se ha actualizado por uno erróneo, se puede restablecer en cualquier momento.

La estructura que se ha seguido en el directorio es la siguiente:

- Graphic resources: carpeta principal que contiene todo el material para *assets* gráficos.
  - Building_Models: *assets* de edificios.
  - Character_Models: *assets* de personajes.
  - Concept Art: archivos de arte conceptual o previo.
  - Other Resources: otros archivos sin cabida en el resto de carpetas.
  - Props_Models: *assets* que no son edificios ni personajes, básicamente muebles urbanos.
    - Cada una de esta carpeta se encuentra subdividida en varias carpetas según los *assets* que contengan, estas son:
      - Geom (con los archivos que contienen las mallas con geometría (los modelos 3D) originales (ya sean de ficheros de Autodesk Maya o Autodesk 3D Studio Max).
      - Render: con las imágenes renderizadas de los ficheros de geometría desde algunos de los programas dichos en el punto anterior o desde Marmoset Toolbag[20].
      - Export: con los archivos de geometría exportados para incluir en Unity (ya sean ficheros .obj o .fbx).
      - Maps: con archivos que son mapas de textura de photoshop y el mapa de coordenadas de texturas (uv map).
      - Textures: con las texturas terminadas y exportadas para incluir en Unity (normalmente en .tga).
- Audio resources: carpeta principal que contiene todo el material para *assets* de audio.
  - Music: archivos de música.
  - SFX: archivos con efectos de audio.
- Detroit references: contiene imágenes y otras referencias de la ciudad de Detroit.
- ESNE y FDI: material específico para las asignaturas que están cursando los miembros del equipo que estudian en estas universidades.
- Photon server: archivos necesarios para ejecutar *Photon Server*.
- Production: material sobre producción, como imágenes promocionales, logotipos, etc.

[20] http://www.marmoset.co/

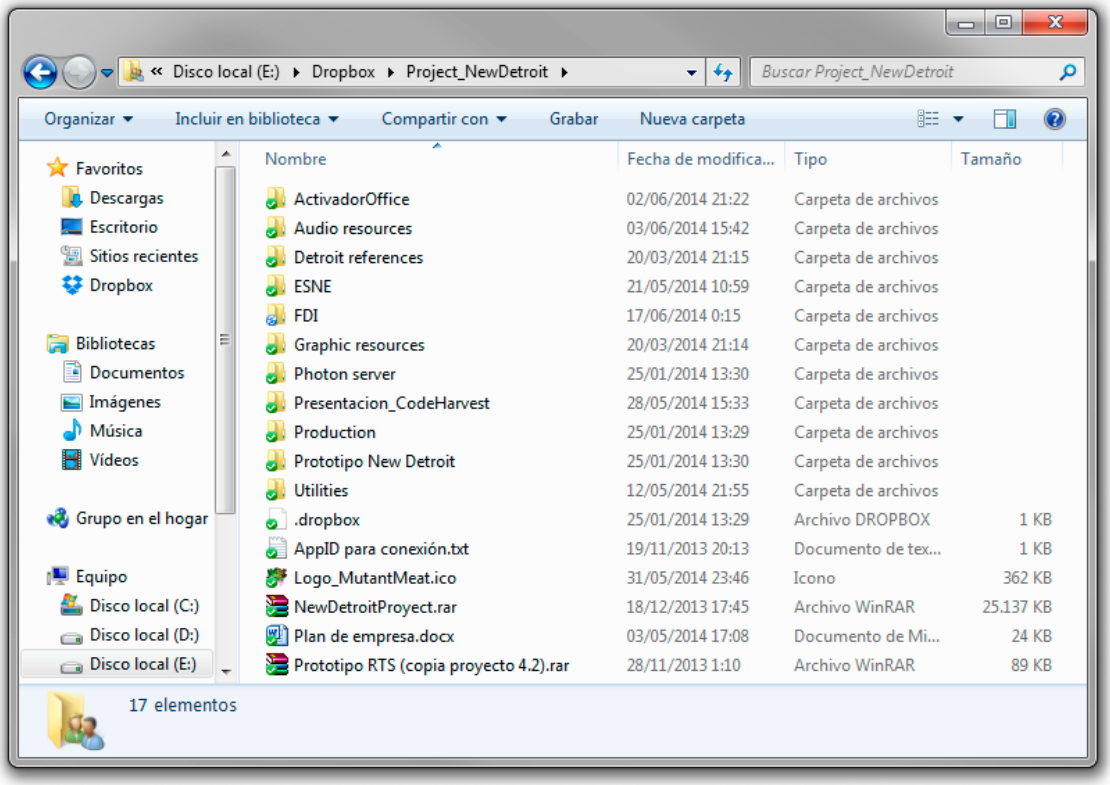


Figura 5.4: Aspecto de la carpeta principal de *Dropbox* del proyecto.

6. Discusión

Al comienzo de este proyecto se realizó un estudio de mercado de muchos juegos los cuales se probaron a conciencia para extraer las características que se consideraron más importantes, y en relación a la popularidad de estos en el mundo del videojuego se han seleccionado las jugabilidades más adictivas.

El videojuego *League Of Legends* es el juego online más jugado en el mundo, por lo que ha sido la referencia más relevante. Entre los aspectos más destacados de este juego está la gran gama de personajes con distinta experiencia de juego, lo que nos impulsó a diseñar héroes con distintas habilidades. Como aspecto negativo, en el transcurso de una partida los jugadores deben permanecer en esta hasta el final, ya que si abandonan en mitad de esta, el juego se descompensa y repercute directamente a los jugadores. En el diseño esto no ha sido contemplado, dado que no se ha considerado un aspecto tan importante.

Así mismo las mecánicas del videojuego *Smite* han sido importantes dado que rompía un poco el esquema tradicional de este tipo de juegos, ya que en este caso la cámara sigue al personaje de cerca haciendo que el control de la visión cambie y, por lo tanto, la experiencia de juego. En este caso se ha optado por un tipo de cámara similar a esta diferenciándolo más del otro modo.

En cuanto a la otra mecánica del juego (RTS) se ha cogido, como principal referencia, el videojuego *StarCraft II*, ya que en su género es el más popular actualmente. Destaca principalmente la diferencia entre las facciones jugables de este juego, y es por esto que en el diseño se ha tratado este tema, de modo que haya diferenciación entre las facciones pero no de ventajas a una facción en concreto.

La planificación y la metodología que usan las empresas de los juegos mencionados tienen en cuenta un mayor tiempo y personal, por lo que a la hora de tomarlos como referencia no se podía optar a los resultados de estas. Como consecuencia, algunas de las tareas planificadas al principio del proyecto no se han desarrollado, dejándolas como trabajo futuro o desechándolas. Tampoco se tenía la experiencia necesaria para hacer una correcta planificación en comparación a los años que nos llevan de ventaja las grandes empresas.

Teniendo en cuenta la división de tareas, esta se ha diseñado como una jerarquía modular de modo que se distinguiese claramente los géneros de este juego, así como de la interfaz de los menús. A mayor profundidad de la jerarquía, otra de las divisiones importantes ha sido la distinción entre arte, diseño e implementación, contemplando la relación entre ellos. De este modo se ha podido paralelizar el trabajo para obtener un mayor consumo de horas del proyecto.

El proyecto cuenta con altas expectativas de mercado dada la popularidad de los géneros tratados y la novedad de combinarlos. Por otra parte, la dificultad de esto reside en la aceptación por parte del jugador y la inversión necesaria para comercializarlo con garantías. Otro aspecto que representa dificultad es la promoción del juego, para lo cual se ha participado en los siguientes eventos:

- Zerouno: se trata de un evento de inversores, jugadores aficionados y desarrolladores de videojuegos. Una de las partes del evento consistía en un concurso de presentación[21] de videojuegos con un video y una breve explicación, todo de un minuto y medio. Se asistió y se mostró el video[22] del juego al público del evento, nuestro juego resultó ganador y, posteriormente, los representantes del evento hicieron una video entrevista[23] a un miembro de los desarrolladores y a otro del grupo de artistas.
- Gamelab: es una de las ferias mundiales más importantes de desarrolladores de videojuegos, se celebra en Barcelona y nuestro proyecto participó con un vídeo[24] y una demo en el concurso a mejor proyecto de videojuegos universitarios[25].

A parte, se consiguió la colaboración de un conocido artista *Youtuber* llamado Rush Smith, gracias a la mediación de ESNE, que ha realizado un videoclip con imágenes y animaciones del videojuego realizado en este proyecto. Este video[26], en tan solo 4 días llevaba más de 35.000 visitas en YouTube y más de 4.000 *me gusta*.

[21] https://www.youtube.com/watch?v=m97gaXmV1kQ
[22] https://www.youtube.com/watch?v=vMgNFLCOVHA
[23] https://www.youtube.com/watch?v=f3OCu8mJXPk
[24] https://www.youtube.com/watch?v=f0HNy2u1nhQ
[25] http://gamelab.es/premios2014/candidates/view-student/?id=36
[26] http://youtu.be/a2j-IivbSCc

7. Conclusiones

El trabajo de diseño realizado al principio del proyecto ha sido útil a la hora de su desarrollo, de modo que este ha ido cambiando a lo largo del mismo para adaptarse a los problemas que han ido surgiendo. Para ello se ha contado con una planificación en la que se ha contemplado las horas estimadas, lo que ha ayudado tener una visión global de la evolución del proyecto. Los diagramas UML y los diagramas de las máquinas de estado han ayudado a la comprensión del comportamiento de las distintas unidades, simplificando y acelerando el proceso de implementación.

En cuanto a la planificación, se ha ganado experiencia para los próximos proyectos que se lleven a cabo por este grupo. Se ha visto la correlación entre las tareas previstas en la tabla EDT y el trabajo que finalmente se ha realizado y, sobre todo, se ha visto claramente cómo una buena estimación de las tareas a realizar influye de manera sustancial en el buen desarrollo del proyecto.

La metodología de trabajo *Scrum* ha resultado ser una opción adecuada, debido a que permitió la distribución de las distintas tareas planificadas en el EDT a lo largo de diferentes *sprints*, además de poder adaptarse a los frecuentes cambios que han ido surgiendo a lo largo de la realización del proyecto y, sobre todo, a la hora de organizar un equipo tan numeroso y con gente ajena a la facultad. Además ha permitido ver el ritmo de trabajo a lo largo del año y la reorganización de los distintos grupos de trabajo en función del trabajo requerido por las distintas tareas.

El videojuego desarrollado en este proyecto cuenta con una belleza estética muy viva y guarda una gran coherencia, además cuenta con una variada gama de opciones de juego. De cara al futuro, los principales trabajos de diseño se centrarían en crear nuevos héroes y más unidades de cada facción, aportando así mayor versatilidad en el juego. Se añadirían los trabajos de la planificación que han sido pospuestos, así como los ítems del juego y nuevos mapas que aportarían mayor variedad y, por lo tanto, los jugadores optarían a nuevas estrategias.

Habría que invertir esfuerzo en la mejora del rendimiento de la red, de modo que esto no sea un elemento disuasorio para jugar. Para ello habría que realizar un diseño de una aplicación propia para la aplicación de servidor que gestione las partidas online, ya que el proyecto cuenta con una limitación de número de conexiones.

Uno de nuestros objetivos es llegar a promocionar el juego mediante la asistencia a eventos o salones de videojuegos, de modo que nos demos a conocer. También sería importante conseguir financiación mediante inversores o estrategias que siguen los juegos *free-to-play*, como la comercialización de contenido exclusivo y la adición de publicidad.

Por último, se desea destacar que el proyecto está alojado en el repositorio de código libre GitHub[27].

[27] https://github.com/maxi-jp/Project_NewDetroit

8. Referencias Bibliográficas

Blackman, S. (2013). *Beginning 3D Game Development with Unity 4* (2nd Edition ed.). Apress.

Ferguson, J., Patterson, B., Beres, J., Boutquin, P., & Gupta, M. (2003). *La Biblia de C#.* Anaya Multimedia.

Gálvez Ruiz, D., Miranda Esteban, M., & Monasterio Martín, F. (s.f.). *Producción de un Videojuego Multijugador en Unity Combinando los Géneros MOBA y RTS - Tecnología e Implementación.*

Holub, A. (2011). Retrieved Junio 2014, from Allen Holub's UML Quick Reference: http://www.holub.com/goodies/uml/

J. Deitel, P., & M. Deitel, H. (2010). *C# 2010 for Programmers* (4th Edition ed.). Deitel Developer Series.

Norton, T. (2013). *Learning C# by Developing Games with Unity 3D Beginner's Guide.* Birmingham: Packt Publishing Ltd.

R. Stagner, A. (2013). *Unity Multiplayer Games.* Birmingham: Packt Publishing.

Rollings, A., & Morris, D. (1999). *Game Architecture and Design.* Coriolis Group Books.

S. Rubin, K. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process.* Addison-Wesley.

Schell, J. (2008). *The Art of Game Design: A book of lenses.* CRC Press.

Sithu Kyaw, A., Peters, C., & Naing Swe, T. (2013). *Unity 4.x Game AI Programming.* Birmingham: Packt Publishing.

Smith, M., & Queiroz, C. (2013). *Unity 4.x Cookbook.* Birmingham: Packt Publishing.

Unity. (n.d.). *Unity Scripting API*. Retrieved Junio 2014, from http://docs.unity3d.com/ScriptReference/index.html

9. Apéndices

a. Manual de Instrucciones Básicas

Cuando se ha iniciado el juego y elegido una sala para empezar a jugar, se presenta un menú en el que se puede elegir entre dos modos de juego:

MOBA mode (*Hero*)

En este modo de juego se controla a uno de los dos héroes Rob Render o, en el otro bando, al héroe Skelterbot. Para ello se controla su movimiento mediante las teclas WASD para moverlo hacia adelante, atrás y girar. Para combatir a sus enemigos vale con mantener pulsado el botón izquierdo del ratón para golpear a los que se tengan justo delante. Con el clic derecho del ratón se puede posicionar justo detrás la cámara desde la que se observa al héroe. También cuenta con tres habilidades para abatir a los enemigos más duros, para activarlos hace falta tener suficiente nivel y pulsar una de las teclas 1,2 o 3.

RTS mode (*Army*)

En este modo se controla al resto del ejército de Rob, o de Skelterbot. Se puede seleccionar la base haciendo clic con el botón izquierdo del ratón poniendo previamente el cursor encima de esta. Después de seleccionarla se tornará del color del equipo y se podrán desplegar unidades al punto de inicio (éste se puede cambiar con el clic derecho del ratón mientras la base esté seleccionada). Mientras esté seleccionada la base se crean unidades con las teclas 1, 2, 3, 4 y 5. Para mandar las órdenes a las unidades desplegadas primero hemos de seleccionarlas (más adelante se ve cómo hacerlo). Después se selecciona el enemigo o el lugar con el que queremos que interactúe posicionando el cursor encima del sitio y pulsando el clic derecho del ratón.

A continuación se explican las características de las unidades que se pueden desplegar, para usar sus habilidades recordamos que han de estar seleccionadas antes:

1. **Unidad recolectora**: esta unidad es capaz de recolectar cristal de las minas y llevarlo a la base para conseguir recursos que utilizaremos en el despliegue de unidades. También pueden curar a las unidades amigas.
2. **Unidad artillera ligera**: esta unidad es capaz de disparar a las otras unidades a distancia.
3. **Unidad artillera pesada***:* esta unidad es más potente que la anterior y además puede desplegarse en un sitio para hacer más daño a sus oponentes y aumentar su rango. Para hacerlo hay que tener esta unidad seleccionada y pulsar la tecla D.
4. **Unidad ingeniera**: esta unidad es capaz de construir torres que defienden a tus unidades y centros de recolección de recursos para que los recolectores vayan a depositar los recursos a estos. Para ello debe tenerse el ingeniero seleccionado y pulsar las teclas T y W respectivamente, y después seleccionar el lugar donde se

quiera que se construya. Esta unidad también puede reparar edificios dañados, ayudar a completar los edificios en construcción y conquistar las torres neutrales.

5. **Unidad exploradora**: esta unidad es más rápida y más barata que las demás, pero más débil por lo que nos sirve para explorar el terreno en busca de enemigos. Además tiene la habilidad de recorrer rutas marcadas por el jugador. Esto se hace marcando puntos con el clic izquierdo del ratón mientras tenemos pulsada la tecla P. Estas unidades también pueden saltar ríos.

Se puede eliminar a tus propias unidades si pulsamos la tecla *suprimir* con la unidad seleccionada.

Existe un modo de ataque automático con las unidades de artillería (ligera y pesada). Este se activa con la tecla A presionada mientras hacemos clic izquierdo en un lugar del mapa, de esta manera los artilleros que estuvieran seleccionados se desplazaran hasta ese punto aniquilando a todo enemigo que se encuentren a su paso.

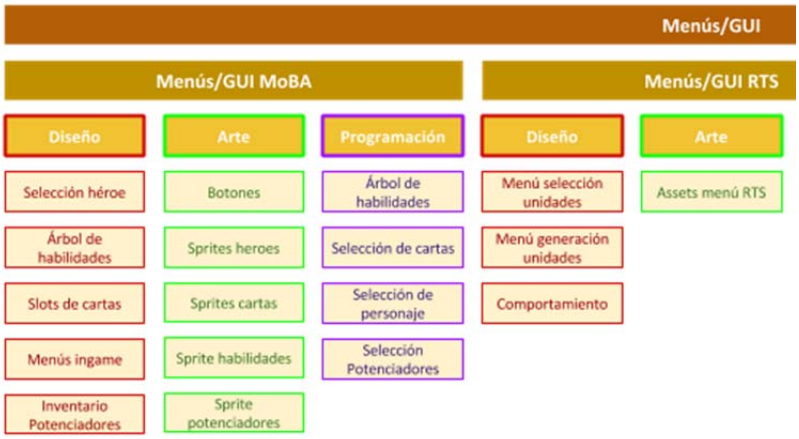
En cuanto a la selección de unidades:

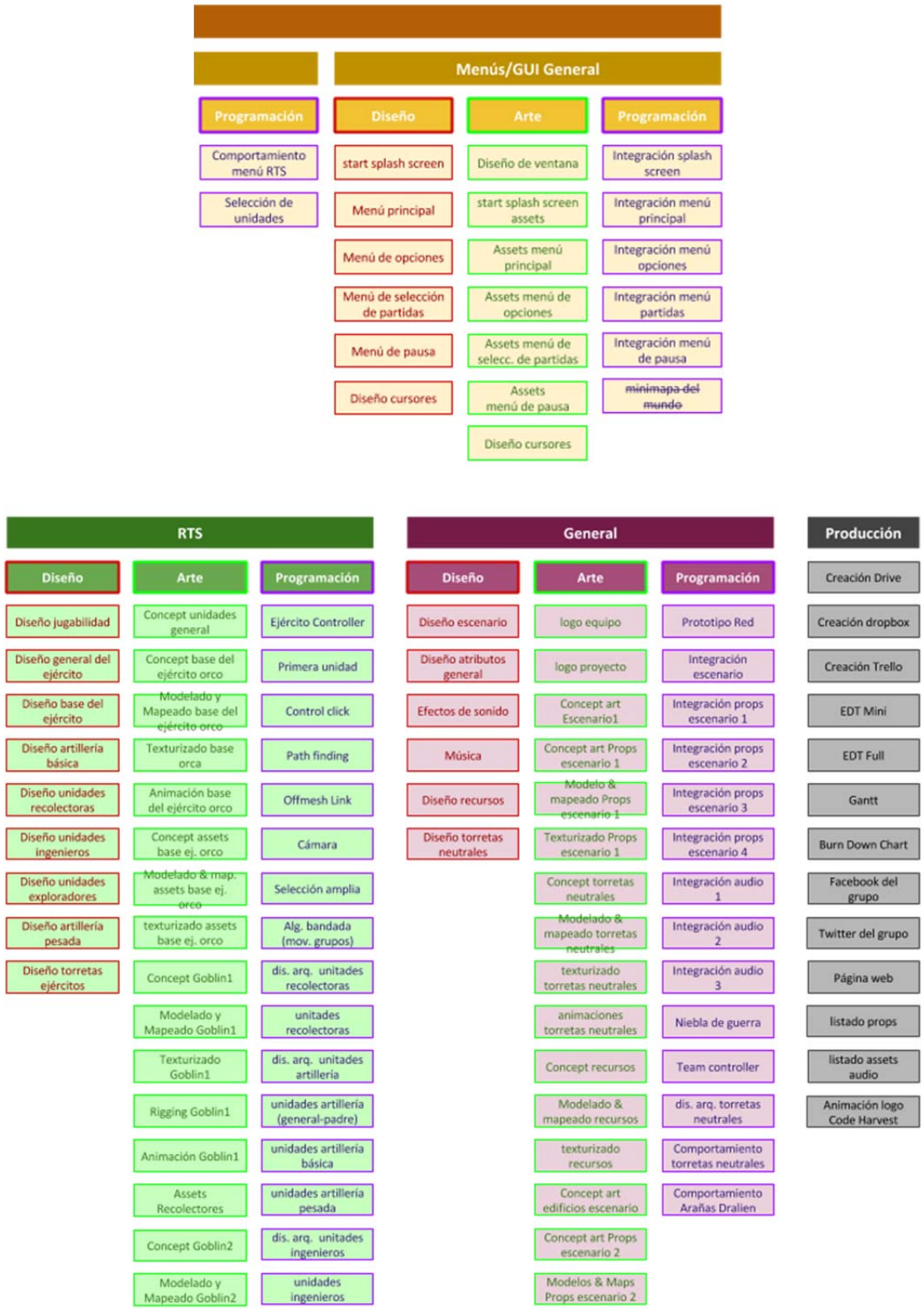
- Se puede realizar selección múltiple dejando pulsado el botón izquierdo del ratón y desplazando el puntero.
- Se puede realizar selección múltiple haciendo doble clic con el botón izquierdo del ratón, y así se seleccionan todas las unidades del mismo tipo.
- Se puede realizar selección múltiple mediante la tecla *Ctrl* del teclado y seleccionando la unidad que se quiera, de manera que se mantienen seleccionadas las unidades que se tenían y se añade esta última.

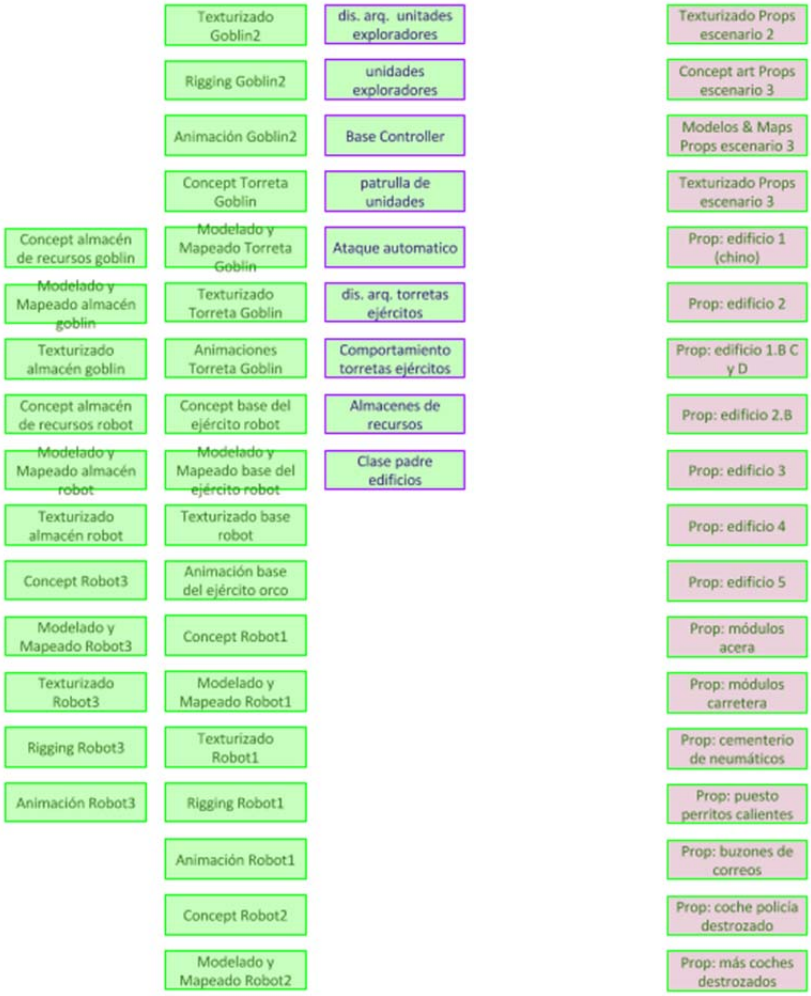
El objetivo de la partida es destruir la base rival sin que el enemigo destruya la tuya antes.

b. EDT

A continuación se muestra el documento de EDT al completo.




Menús/GUI General
Programación
Comportamiento menú RTS
Selección de unidades
Diseño
start splash screen
Menú principal
Menú de opciones
Menú de selección de partidas
Menú de pausa
Diseño cursores
Arte
Diseño de ventana
start splash screen assets
Assets menú principal
Assets menú de opciones
Assets menú de selecc. de partidas
Assets menú de pausa
Diseño cursores
Programación
Integración splash screen
Integración menú principal
Integración menú opciones
Integración menú partidas
Integración menú de pausa
minimapa del mundo
RTS
Diseño
Diseño jugabilidad
Diseño general del ejército
Diseño base del ejército
Diseño artillería básica
Diseño unidades recolectoras
Diseño unidades ingenieros
Diseño unidades exploradores
Diseño artillería pesada
Diseño torretas ejércitos
Arte
Concept unidades general
Concept base del ejército orco
Modelado y Mapeado base del ejército orco
Texturizado base orca
Animación base del ejército orco
Concept assets base ej. orco
Modelado & map. assets base ej. orco
texturizado assets base ej. orco
Concept Goblin1
Modelado y Mapeado Goblin1
Texturizado Goblin1
Rigging Goblin1
Animación Goblin1
Assets Recolectores
Concept Goblin2
Modelado y Mapeado Goblin2
Programación
Ejército Controller
Primera unidad
Control click
Path finding
Offmesh Link
Cámara
Selección amplia
Alg. bandada (mov. grupos)
dis. arq. unitades recolectoras
unitades recolectoras
dis. arq. unitades artillería
unidades artillería (general-padre)
unidades artillería básica
unidades artillería pesada
dis. arq. unitades ingenieros
unidades ingenieros
General
Diseño
Diseño escenario
Diseño atributos general
Efectos de sonido
Música
Diseño recursos
Diseño torretas neutrales
Arte
logo equipo
logo proyecto
Concept art Escenario1
Concept art Props escenario 1
Modelo & mapeado Props escenario 1
Texturizado Props escenario 1
Concept torretas neutrales
Modelado & mapeado torretas neutrales
texturizado torretas neutrales
animaciones torretas neutrales
Concept recursos
Modelado & mapeado recursos
texturizado recursos
Concept art edificios escenario
Concept art Props escenario 2
Modelos & Maps Props escenario 2
Programación
Prototipo Red
Integración escenario
Integración props escenario 1
Integración props escenario 2
Integración props escenario 3
Integración props escenario 4
Integración audio 1
Integración audio 2
Integración audio 3
Niebla de guerra
Team controller
dis. arq. torretas neutrales
Comportamiento torretas neutrales
Comportamiento Arañas Dralien
Producción
Creación Drive
Creación dropbox
Creación Trello
EDT Mini
EDT Full
Gantt
Burn Down Chart
Facebook del grupo
Twitter del grupo
Página web
listado props
listado assets audio
Animación logo Code Harvest


Concept almacén de recursos goblin
Modelado y Mapeado almacén goblin
Texturizado almacén goblin
Concept almacén de recursos robot
Modelado y Mapeado almacén robot
Texturizado almacén robot
Concept Robot3
Modelado y Mapeado Robot3
Texturizado Robot3
Rigging Robot3
Animación Robot3
Texturizado Goblin2
Rigging Goblin2
Animación Goblin2
Concept Torreta Goblin
Modelado y Mapeado Torreta Goblin
Texturizado Torreta Goblin
Animaciones Torreta Goblin
Concept base del ejército robot
Modelado y Mapeado base del ejército robot
Texturizado base robot
Animación base del ejército orco
Concept Robot1
Modelado y Mapeado Robot1
Texturizado Robot1
Rigging Robot1
Animación Robot1
Concept Robot2
Modelado y Mapeado Robot2
dis. arq. unitades exploradores
unidades exploradores
Base Controller
patrulla de unidades
Ataque automatico
dis. arq. torretas ejércitos
Comportamiento torretas ejércitos
Almacenes de recursos
Clase padre edificios
Texturizado Props escenario 2
Concept art Props escenario 3
Modelos & Maps Props escenario 3
Texturizado Props escenario 3
Prop: edificio 1 (chino)
Prop: edificio 2
Prop: edificio 1.B C y D
Prop: edificio 2.B
Prop: edificio 3
Prop: edificio 4
Prop: edificio 5
Prop: módulos acera
Prop: módulos carretera
Prop: cementerio de neumáticos
Prop: puesto perritos calientes
Prop: buzones de correos
Prop: coche policía destrozado
Prop: más coches destrozados

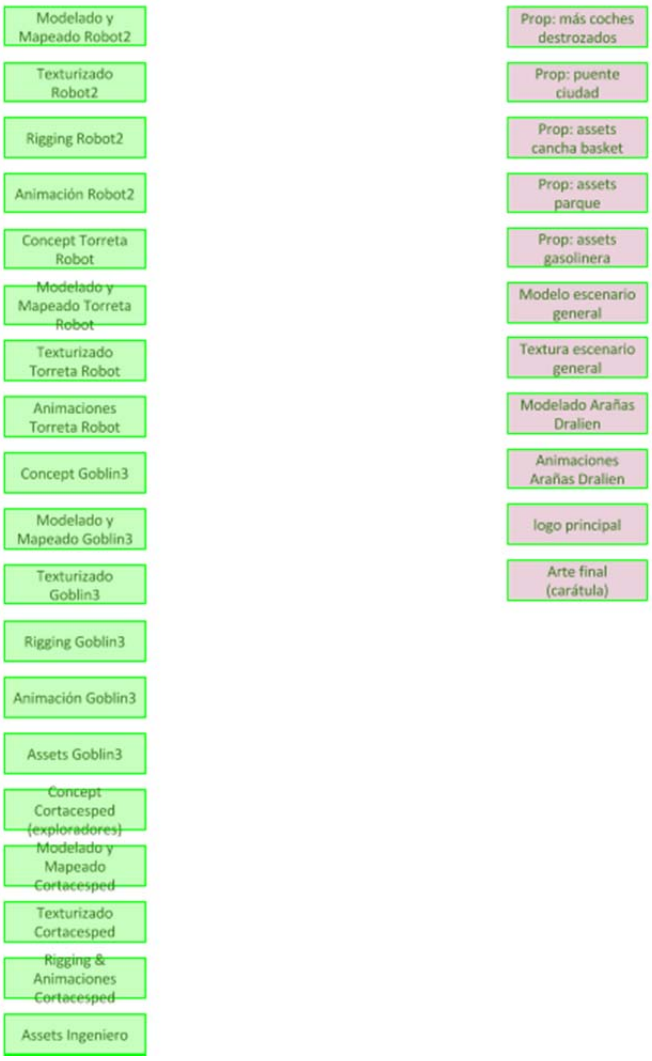
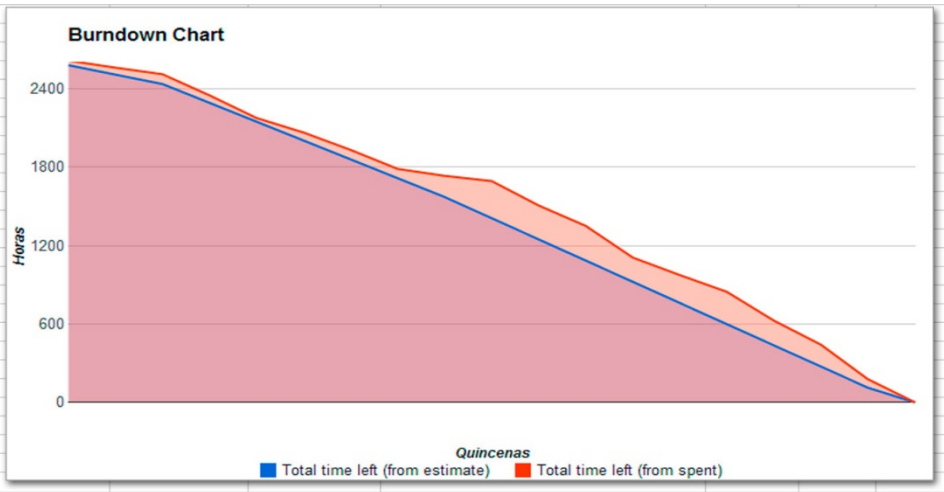


Figura 9.1: Estructura de Descomposición de Trabajo completo.

c. Burndown chart

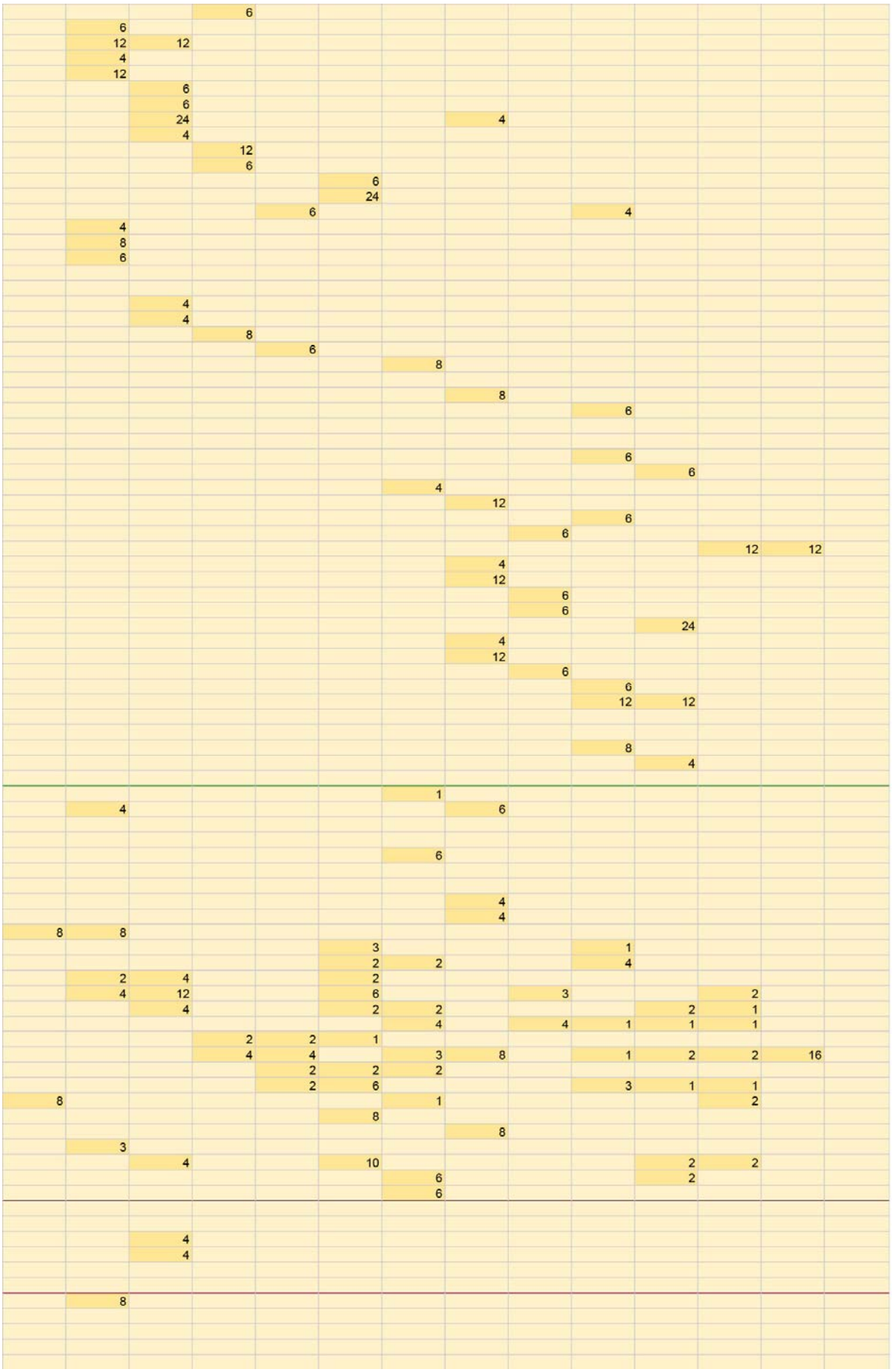
En las siguientes páginas se muestra el documento Burndown chart al completo.



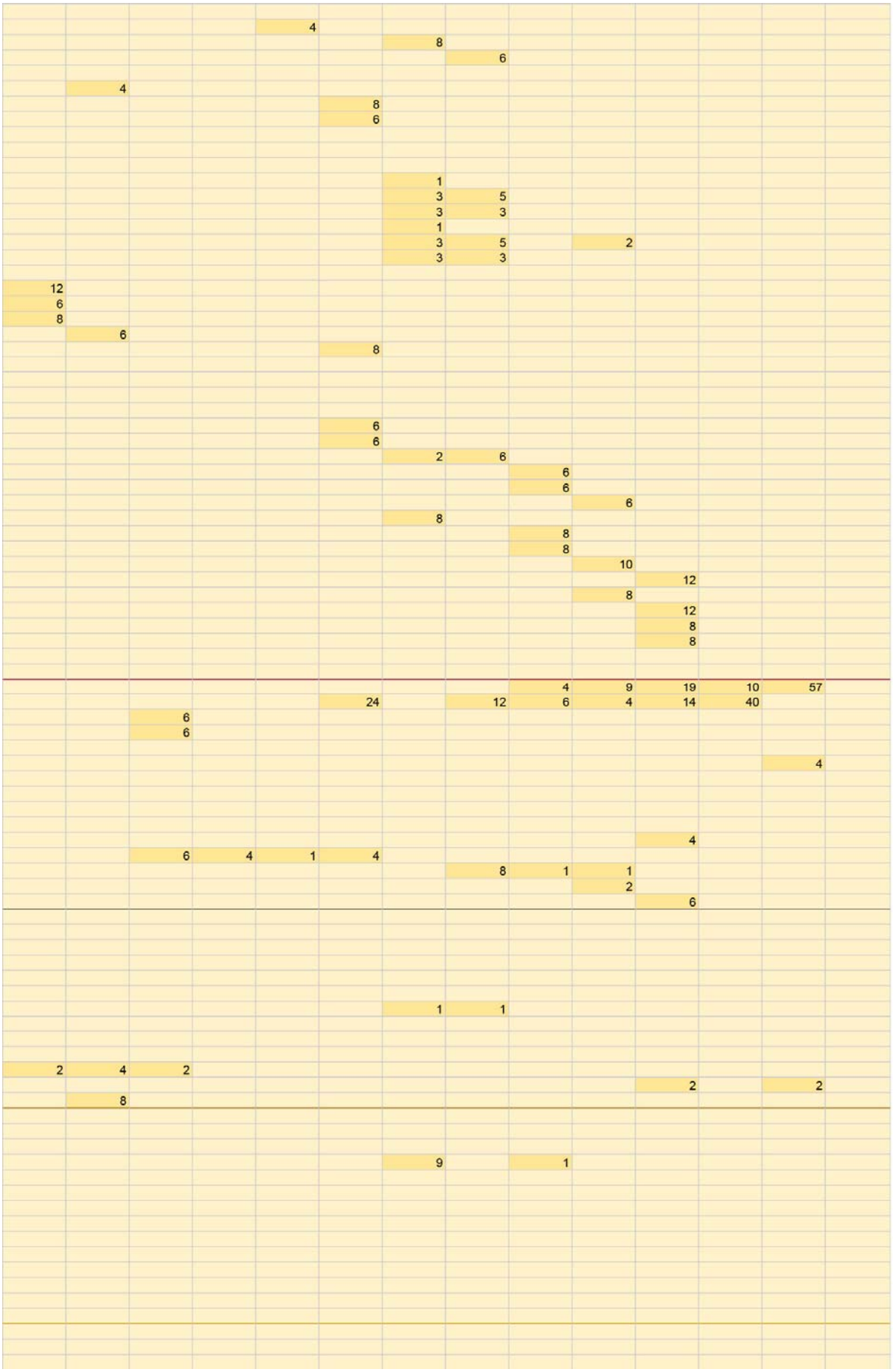
Task(1)	Task(2)	Task(3)	Task(4)	Time (estimated)	Time (spent)	12/10/2013	26/10/2013	9/11/2013	23/11/2013	7/12/2013
	MoBA	Diseño	Diseño jugabilidad	24	24	12	12			
			Diseño combate Orco	13	13				12	
			Diseño evolución Orco	8	8					
			Árbol de habilidades Orco	6	6					
			Diseño combate Robot	12	12					
			Diseño evolución Robot	12	6					
			Árbol de habilidades Robot	12	0					
			Tesoros	12	0					
			Cartas	12	0					
			Subidas de nivel	12	1					
		Arte	Concept Art Orco	6	6				6	
			Modelo Héroe Orco	12	12				12	
			Mapeado Modelo Héroe Orco	6	6				6	
			Texturizar modelo Héroe Orco	6	6					6
			Riging modelo Héroe Orco	6	6					6
			Animación Héroe Orco	24	24					12
			Concept Art Robot	6	6				6	
			Modelo Héroe Robot	12	12					
			Mapeado Héroe Robot	6	6					
			Texturizado Héroe Robot	8	8					
			Rigging Héroe Robot	6	6					
			Animación Héroe Robot	24	6					
			Assets árbol de habilidades orco	0	0					
			Assets árbol de habilidades robot	0	0					
		Programación	Cámara	32	32				16	16
			Hero Controller	10	35					1
			Orc Controller	10	15					1
			Combate Orco simple	6	6					
			Combate Orco habilidades	20	20					
			Integración Orco en escenario	32	31					
			Robot Controller	10	45					1
			Combate Robot simple	10	0					
			Combate robot habilidades	20	0					
			Integración Robot en escenario	12	1					
			Depurar combate	25	0					
			prueba combate en red	50	29					
	RTS	Diseño	Diseño jugabilidad	26	26	12	12			
			Diseño general del ejército	12	12			12		
			Diseño base del ejército	6	6				6	
			Diseño unidades recolectoras	4	4					4
			Diseño unidades ingenieros	4	4					
			Diseño unidades exploradores	4	4					
			Diseño artillería pesada	4	4					
			Diseño artillería ligera	4	4					
			Diseño torretas ejércitos	4	4					
			Partículas de los ingenieros	3	3					
			Partículas de los exploradores	6	6					
				0	0					
		Arte	Concept unidades general	6	6		6			
			Concept base del ejército orco	4	4					
			Modelado base del ejército orco	6	6					
			Mapeado y texturizado base orco	6	6					
			Animación base del ejército orco	0	0					
			Concept Assets base del ejército orco	4	4					
			Mod. y map. Assets base del ej. orco	8	8					
			Texturizado Assets base del ej. orco	8	8					
			Concept almacén recursos ejército orco	2	2					
			Mod. y map almacén recursos ej. orco	6	6					
			Texturizado almacén recursos ej. orco	6	6					
			Concept Goblin01	4	4					
			Modelado y Mapeado Goblin01	12	12					

21/12/2013	10/01/2014	24/01/2014	7/02/2014	21/02/2014	7/03/2014	21/03/2014	4/04/2014	18/04/2014	2/05/2014	16/05/2014	30/05/2014	13/06/2014	19/06/2014
			1			8							
						6							
							12						
						6							
			1										
12													
							12						
							6						
								8					
							6						
							6						
3					5	20	4					2	
3			2		1			8					
		5			1								
		4		1	12	2				1			
6	6	10	1		4	2	2						
3												41	
										1			
		10	2		4		12	1					
					2								
4													
4													
4													
4													
4													
							3						
							4		2				
4													
	6												
	6												
				4									
							8						
							8						
						2							
							6						
										6			
4													
6	6												

			Texturizado Goblin01	6	6					
			Rigging Goblin01	6	6					
			Animaciones Goblin01	24	24					
			Concept Goblin02	4	4					
			Modelado y Mapeado Goblin02	12	12					
			Texturizado Goblin02	6	6					
			Rigging Goblin02	6	6					
			Animaciones Goblin02	28	28					
			Concept Goblin03	4	4					
			Modelado y Mapeado Goblin03	12	12					
			Texturizado Goblin03	6	6					
			Rigging Goblin03	6	6					
			Animaciones Goblin03	24	24					
			Assets recolectores Goblin	10	10					
			Concept Torreta Goblin	4	4					
			Modelado y Mapeado Torreta Goblin	8	8					
			Texturizado Torreta Goblin	6	6					
			Animaciones Torreta Goblin	0	0					
			Assets Artillería Goblin	0	0					
			Assets Ingenieros Goblin	6	4					
			Concept cortacesped (exp. Goblin)	4	4					
			Modelado y Map. cortaces. (exp. Goblin)	8	8					
			Texturizado cortacesped (exp. Goblin)	6	6					
			Rig. y Anim. cortacesped (exp. Goblin)	8	8					
			Concept base del ejército robot	0	0					
			Modelado base del ejército robot	8	8					
			Mapeado y texturizado base robot	6	6					
			Animación base del ejército robot	0	0					
			Concept almacén recursos ejército robot	0	0					
			Mod. y map almacén recursos ej. robot	6	6					
			Texturizado almacén recursos ej. robot	6	6					
			Concept Robot01	4	4					
			Modelado y Mapeado Robot01	12	12					
			Texturizado Robot01	6	6					
			Rigging Robot01	6	6					
			Animaciones Robot01	24	24					
			Concept Robot02	4	4					
			Modelado y Mapeado Robot02	12	12					
			Texturizado Robot02	6	6					
			Rigging Robot02	6	6					
			Animaciones Robot02	24	24					
			Concept Robot03	4	4					
			Modelado y Mapeado Robot03	12	12					
			Texturizado Robot03	6	6					
			Rigging Robot03	6	6					
			Animaciones Robot03	24	24					
			Assets recolectores Robots	6	0					
			Concept Torreta Robots	0	0					
			Modelado y Mapeado Torreta Robots	8	8					
			Texturizado Torreta Robots	4	4					
			Animaciones Torreta Robots	0	0					
		Programación	Ejército controller	16	17				8	8
			Unit Controller (clase padre)	12	14				4	
			Control click	6	6				6	
			Path finding (navemesh)	16	16				8	8
			Offmesh Link (navemesh)	6	6					
			Cámara	8	6					6
			Selección amplia 1 (cuadrado)	8	8				8	
			Selección amplia 2 (doble click o ctrl)	4	4					
			Selección amplia 3 (shift)	4	4					
			Alg. bandada (mov. grupos)	18	18					2
			Dis. Arquitectura Unidades recolectoras	9	10					6
			Unidades recolectoras	8	16					8
			Dis. Arquitectura Unidades artillería	8	8					
			Unidades artillería (general padre)	24	27					
			Unidades artillería básica	6	11					
			Unidades artillería pesada	8	11					
			Dis. Arquitectura Unidades ingenieros	6	5					
			Unidades ingenieros	24	40					
			Dis. Arquitectura Unidades exploradores	6	6					
			Unidades Exploradores	8	13					
			Base controller	12	15					4
			patrulla de unidades	8	8					
			Movimiento con ataque (A)	8	8					
			Dis. Arquitectura torretas ejércitos	3	3					
			Comportamiento torretas ejércitos	16	18					
			Almacenes de recursos	8	8					
			Clase padre edificios	6	6					
	General	Diseño	Diseño escenario	12	12		12			
			Diseño atributos general	24	24			12	6	6
			Diseño recursos	4	4					
			Diseño torretas neutrales	4	4					
			Efectos de sonido	6	0					
			Música	6	0					
		Arte	Logo equipo	8	8					
			Logo proyecto	4	4				4	
			Concept Art escenario 1	4	4				4	
			Concept Props escenario 1	4	4				4	
			Modelado y Mapeado Props escenario 1	12	12				4	8


6
6
12 12
4
12
6
6
24
4
12
6
6
24
6 4
4
8
6
4
4
8
6
8
8
6
6
6
4
12
6
6
12 12
4
12
6
6
4
12
6
12 12
8
4
1
4 6
6
4
4
8 8
3 1
2 2 4
2 4 2
4 12 6 3 2
4 2 2 2 1
4 4 1 1 1
2 2 1
4 4 3 8 1 2 2 16
2 2 2
2 6 3 1 1
8 1 2
8
8
3
4 10 2 2
6 2
6
4
4
8

			Texturizado Props escenario 1	8	8					8
			Concept torretas neutrales	4	4					
			Modelado y Mapeado torretas neutrales	8	8					
			Texturizado torretas neutrales	6	6					
			Animaciones torretas neutrales	0	0					
			Concept mina de recursos	4	4					
			Modelado y Mapeado mina de recursos	8	8					
			Texturizado recursos	6	6					
			Concept Art Props escenario 2	4	4					4
			Modelado y Mapeado Props escenario 2	8	8					8
			Texturizado Props escenario 2	6	6					6
			Concept Art Props escenario 3	1	1					
			Modelado y Mapeado Props escenario 3	8	8					
			Texturizado Props escenario 3	6	6					
			Concept Art Props escenario 4	0	1					
			Modelado y Mapeado Props escenario 4	10	10					
			Texturizado Props escenario 4	6	6					
			Concepts edificios escenario	8	8				4	4
			Modelado y Map. Prop: edificio 1 (chino)	12	12					
			Texturizado Prop: edificio 1 (chino)	6	6					
			Modelado y Map. Prop: edificio 2	8	8					
			Texturizado Prop: edificio 2	6	6					
			Prop: edificio 1.B, C, D (variaciones)	8	8					
			Prop: edificio 2.1 (variaciones)	6	0					
			Prop: edificio 3	10	0					
			Prop: edificio 4	10	0					
			Prop: edificio 5	10	0					
			Prop: módulos aceras	6	6					
			Prop: módulos carretera	6	6					
			Prop: cementerio de neumáticos	8	8					
			Prop: puesto perritos calientes	6	6					
			Prop: coche policía destrozado	6	6					
			Prop: otro coche destrozado	6	6					
			Prop: assets cancha de basket	8	8					
			Prop: puente Detroit	8	8					
			Prop: assets parque	6	8					
			Prop: assets gasolinera	10	10					
			Props: pack 5 (cofre, arbol y tanque)	12	12					
			Arañas Dralien (Modelado y texturizado)	8	8					
			Arañas Dralien (Animaciones)	12	12					
			Modelo escenario general	8	8					
			Textura escenario general	8	8					
			logo principal	0	0					
			Arte final (carátula)	0	0					
		Programación	Motor Online	150	147		8	12	16	12
			Integración escenario	100	100					
			Integración Props escenario 1	6	6					
			Integración Props escenario 2	6	6					
			Integración Props escenario 3	6	0					
			Integración audio 1	4	4					
			Integración audio 2	4	0					
			Integración audio 3	4	0					
			Niebla de guerra	16	16				4	12
			Team controller	24	0					
			Comportamiento Arañas Dralien	8	4					
			Comportamiento torretas neutrales	15	15					
			Herramienta para la medición de dist.	12	10					
			Componente CLife	2	2					
			Componente CTeam	6	6					
		Producción	Creación Drive	1	1	1				
			Creación Dropbox	2	2	2				
			Creación Trello	2	2	2				
			EDT Mini	2	2			2		
			EDT Full	12	12			8	4	
			Gantt	4	4				2	2
			Burndown Chart	12	20				8	10
			Facebook del grupo	2	2	2				
			Twitter del grupo	2	2	2				
			Página web	12	0					
			listado de props	8	8					
			listado de assets de audio	6	4					
			Animación Logo Code Harvest	8	8					
Menús/GUI	Menús/GUI MOBA	Diseño	Selección héroe	12	0					
			Árbol de habilidades	12	0					
			Slots de cartas	12	0					
			Menús ingame	12	10					
			Inventario Potenciadores	12	0					
		Arte	Botones	4	0					
			Sprites heroes	8	0					
			Sprites cartas	6	0					
			Sprite habilidades	6	0					
			Sprite potenciadores	4	0					
		Programación	Árbol de habilidades	10	0					
			Selección de cartas	10	0					
			Selección de personaje	10	0					
			Selección Potenciadores	10	0					
	Menús/GUI RTS	Diseño	Menú selección unidades	12	0					
			Menú generación unidades	12	0					
			Comportamiento	12	0					


4
8
6
4
8
6
1
3 5
3 3
1
3 5 2
3 3
12
6
8
6
8
6
6
2 6
6
6
6
8
8
8
10
12
8
12
8
8
4 9 19 10 57
24 12 6 4 14 40
6
6
4
4
6 4 1 4
8 1 1
2
6
1 1
2 4 2
2 2
8
9 1

		Arte	Assets menú RTS	6	0						
		Programación	Comportamiento menú RTS	12	0						
			Selección de unidades	12	0						
	Menús/GUI General	Diseño	start splash screen	8	0						
			Menú principal	12	13						
			Menú de opciones	12	0						
			Menú de selección de partidas	12	0						
			Menú de pausa	12	0						
			Diseño cursores	12	0						
			Diseño de ventana	4	0						
		Arte	start splash screen assets	4	0						
			Assets menú principal	4	0						
			Assets menú de opciones	4	0						
			Assets menú de selecc. de partidas	4	0						
			menú de pausa	4	0						
			Diseño cursores	6	0						
		Programación	Integración splash screen	6	0						
			Integración menú principal	6	0						
			Integración menú opciones	6	0						
			Integración menú partidas	6	0						
			Integración menú de pausa	6	0						
			Minimapa del mundo	8	16						
		Memoria	Trabajo referente a la memoria de SI	200	386	4	4	4	4	4	
		Otros	horas no previstas en las tareas previas	82	82	0	0	0	0	0	
			TOTAL	2650	2391	37	54	50	162	173	
			Daily burnout			72	72	72	144	144	
			Total time left (from estimate)			2578	2506	2434	2290	2146	
			Total time left (from spent)			2613	2559	2509	2347	2174	

												13	
								8	8				
6	6	4	4	8	12	11	12	16	14	51	105	117	177
0	0	0	0	0	12	12	6	22	4	20	6		
111	133	145	53	40	187	156	242	134	128	220	184	264	177
144	144	144	144	162	162	162	162	162	162	162	162	162	112
2002	1858	1714	1570	1408	1246	1084	922	760	598	436	274	112	0
2063	1930	1785	1732	1692	1505	1349	1107	973	845	625	441	177	0

d. Actas

En este apartado se puede ver el trabajo realizado desde el primer día de trabajo, ya que los sprints se han quedado plasmados en las actas de reuniones y sirven como ejemplo perfecto de la metodología Scrum seguida durante todo el proceso de desarrollo.

1. Acta del 2 de octubre de 2013 (UCM)

Fecha	2 de octubre de 2013
Concepto	Comentar primeras ideas sobre el proyecto
Asistentes	Todos
Duración	2 horas
Conclusiones	Establecidos primeros aspectos del diseño del juego. Se opta por un MOBA+RTS. Quedaremos la semana que viene para comenzar las reuniones con los tutores y cada miembro del equipo nos pondremos a hacer tutoriales de Unity.

a. Puntos tratados

- Ideas generales para el juego del proyecto.
- Brain storming.

b. Desarrollo:

Juegos pensados:

- Diego: RTS en el que el mapa es un campo de fútbol (o de baloncesto, o de lo que sea), las unidades se crean en base a recursos (generados por tiempo o por recogida en el mapa). Las unidades cuestan más o menos según sus capacidades (defensas con más ataque y vida, jugadores que van más rápido a por el balón, pero tienen menos precisión, etc.). El objetivo es marcar más que tu enemigo en lo que dura una partida. Las bases que generan unidades pueden ser destruidas y esto hace que se generen menos recursos o ciertos tipos de unidades. Se regenerarían con el tiempo para no hacer tediosa la experiencia de juego.
- Maxi: juego de tipo Tower Defense, en el que un jugador tiene que defender un bosque de un ejército de leñadores. En principio solo hay unidades y ataques básicos (leñadores, animales protectores del bosque) y según avanza la partida se desbloquean unidades especiales (tractores, señores gordos con motosierras por un bando y por el otro, ataques usando meteorología, héroes como super hippies, etc.).
- Guille: se maneja una unidad en el mapa y el objetivo es encontrar tesoros repartidos por el escenario. Se puede mezclar con mecánicas clásicas de juegos de estrategia (tipo Warcraft 3).

- Maxi: RTS en el que los mapas se generan a partir de códigos QR, las casillas de control (las grandes de las esquinas) son las bases de las facciones. Los recursos se consiguen abriéndose camino (destruyendo las casillas negras del QR) o por el paso del tiempo.

Brain storming:

- Mezclar recogida de tesoros con las razas (tener que recoger recursos diferentes en función de la raza) y después de recoger tesoros, posibilidad de crear héroes.
- Tener héroes e ir evolucionándolos.
- Algo estilo comandos.
- StarCraft simplificado + LOL.
- Distintos jugadores manejan unidades diferentes de la misma facción (uno el héroe y otros un ejército de minions).
    - Solo hay una raza, pero varios tipos de héroes. Cada uno tiene unos atributos característicos y pueden potenciar atributos propios de los minions.
    - Unidades recolectoras del ejército que tienen la capacidad de destruir rocas (cuadrados por el escenario) y recolectar así recursos. Las unidades bélicas también pueden destruir rocas para abrir camino, pero lo harán de una forma mucho más rápida, pero como punto negativo las rocas así destruidas ya no darán recursos nunca más.
    - Es posible jugar partidas sin héroes (sería como un Starcraft clásico simplificado).
    - Posibilidad de controlar el ejército y el héroe un mismo jugador, ¿por qué no?

2. Acta del 11 de octubre de 2013 (UCM)

Fecha	11 de octubre de 2013
Concepto	Primera reunión con los profesores del proyecto
Asistentes	Todos
Duración	2 horas
Conclusiones	Desarrollar para la semana que viene un documento de concepto.

a. Puntos tratados

- Ideas generales para el juego del proyecto, planteadas en una reunión grupal anterior.
- Proyectos similares y documentación a consultar.
- Cómo organizarse de cara a la memoria del proyecto.
- Desarrollo de un documento de concepto.
- Próxima reunión: dentro de 2 semanas.

b. Desarrollo:

- Al final de curso se tendrán que presentar 2 memorias diferentes, pueden coincidir en cosas, pero no en más del 30%.

- Es mejor no cerrar el juego a partidas de 1 ejército + 1 héroe vs 1 ejército + 1 héroe. Por ejemplo: 3 ejércitos vs 1 héroe.
- Desarrollar un documento de concepto del juego con las opciones y fundamentos de este, que explique de forma concisa y breve el juego, mecánicas, ítems, etc… Mandarlo por email a los profesores para la semana que viene.
- Plantearse si será necesario el desarrollo de herramientas para Unity.
- Generación de terrenos: de momento mejor no planteárselo.
- Licencias Pro para Unity: consultar si hay disponibles en la facultad.
- Tener muy claro lo que es un *RTS* (mirar Wikipedia).
- Ideas para el juego:
    - ¿Es viable y buena opción usar vehículos?
    - Comunicación chat y mensajes: muy importante para este tipo de juegos.
    - Si muere el héroe queda desactivado y hay que curarlo.
    - Tutorial ingame.
- Proyectos similares o que pueden resultar interesantes:
    - Digital Chocolate.
    - RTS del máster de VJu de la fdi (*seven unit* o algo así). Mirar en la web del master.

3. Acta del 23 de octubre de 2013 (UCM)

Fecha	23 de octubre de 2013
Concepto	Iniciar el Game Design Document (GDC), repositorio, control de versiones, entorno de programación y hoja de contactos.
Asistentes	Todos
Duración	2 horas
Conclusiones	Todos vamos a usar la versión 4.2.2 de Unity y como repositorio BitBucket, como control de versiones vamos a usar Sourcetree. Vamos a usar como entorno de programación Visual Studio 2012. Desarrollo del documento de concepto.

a. Puntos tratados

- Se inicia el Game Design Document.

b. Desarrollo:

- Desarrollo del Game Design Document por todos los miembros.

4. Acta del 25 de octubre de 2013 (ESNE)

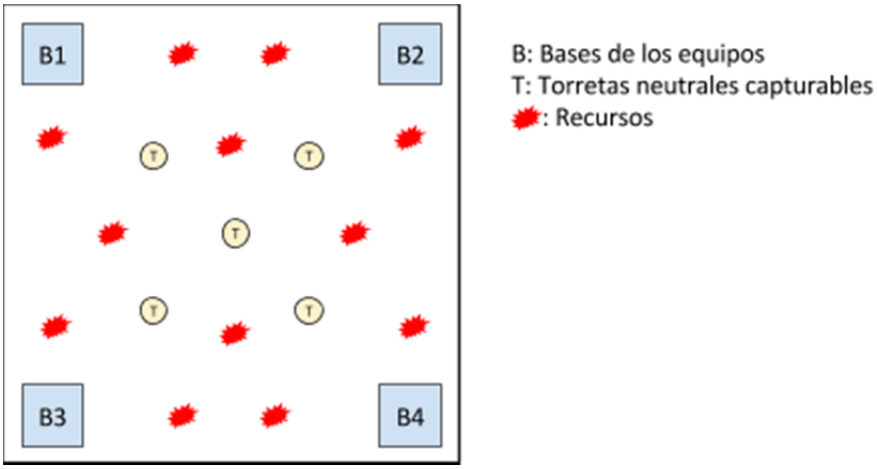
Fecha	25 de octubre de 2013
Concepto	Reunión con el equipo de artistas. Aspectos del diseño del juego.
Asistentes	Todos
Duración	1 hora
Conclusiones	En un principio se va a concentrar los esfuerzos de diseño de niveles a uno solo basado en una estética similar a la película *1997: escape from New York*. Jacobo desarrollará para la próxima reunión unos *mockups* para testear el aspecto general de los héroes del juego. Próxima reunión: miércoles de 11 a 13 para discutir sobre la estética en base a los mockups preparados y comenzar con el diseño de los héroes.

a. Puntos tratados

- Diseño del mapeado del primer escenario.
- Planteamiento de la estética del juego y de otros aspectos (tipos de héroes).
- Próxima reunión.

b. Desarrollo:

Mapa del juego: en principio vamos a desarrollar un único mapa para el juego, este tendrá una estética similar al de películas como *1997: escape from New York*. El mapa tendrá la siguiente distribución:



Ideas para el escenario:

- Ciclos día/noche, lluvia, nieve…
- Límites naturales (ej.: calles cortadas, vallas policiales, edificios en ruinas…) + niebla.
- Recursos:
  - Entradas al metro
  - Tiendas semiderruidas.

- Un tipo de recurso básico que hay que recoger por el mapa + otro que se recoge automáticamente (como una moneda, economía).

Héroes:

- Estéticas:
  - Cubistas (minecraft).
  - Semi-low poly (Final Fantasy VII).
  - Medium poly (LOL).
- Posibles modelos:
  - Orco (Warcraft 3 con un toque ochentero)
  - Robot (yo robot).

5. Acta del 25 de octubre de 2013 (UCM)

Fecha	25 de octubre de 2013
Concepto	Reunión con los profesores del proyecto.
Asistentes	Todos
Duración	1 hora
Conclusiones	Se dividirá el equipo en 3 grupos y se asignarán tareas (prototipo, jugabilidad, motor del juego). Hay que definir responsabilidades dentro del equipo. Rellenar el documento de visión del mdv lo antes posible. Elección de repositorio GitHub y de control de versiones Git + Extensions.

a. Puntos tratados

- Aspectos generales de la jugabilidad.
- Organización a corto plazo, definición de prioridades.
- Cambio de repositorio y control de versiones.

b. Desarrollo:

- Rellenar plantilla del máster de videojuegos de la UCM de documento concepto (nos lo enviará Federico).
- Mirar y estudiar más jugabilidades de MOBAs:
  - LOL, ESDLA: Guardianes de la Tierra Media, DOTA, Smite, Overlord, Combate del WOW (World of Warcraft), Y Buscar juegos parecidos…
- Muy importante que funcionen las diferentes jugabilidades:
  - 1 héroe vs 1 héroe.

    - 1 ejército vs 1 ejército.
    - 1 héroe + 1 ejército vs 1 héroe + 1 ejército.
- Justificar la estética del juego: ¿por qué todo es cómo es? ¿Cuál es el *espíritu* o *alma* del juego?
- Prototipo del juego:
    - Para probar jugabilidad.
    - Plantearse preguntas que el prototipo tiene que responder y cuánto creemos que vamos a tardar en responderlas:
        - Cómo se ataca y cómo se defiende.
        - Control del héroe.
        - Control de la cámara.
        - Cuál es la relación de tamaño entre el héroe y las unidades del ejército.
- Reparto de tareas para objetivos a corto plazo (3 grupos):
    - Jugadores, estudio del arte, mecánicas de lucha de MOBAs.
    - Prototipo de la jugabilidad.
    - Comienzo con el motor del juego:
        - mirar plugins.
        - niebla de guerra.
        - online.
- Prioridades:
    - Definir equipos, responsabilidades, (arte, líder,...).
    - Temática, espíritu del juego.
    - Prototipo
    - Documento de visión (plantilla de Federico).
- Elección de repositorio GitHub y de control de versiones Git + Extensions.

Quedamos para dentro de dos semanas (viernes 8 de noviembre).

6. Acta del 30 de octubre de 2013 (UCM)

Fecha	30 de octubre de 2013
Concepto	Reflexiones grupales sobre la última reunión con los tutores.
Asistentes	Todos
Duración	1 hora
Conclusiones	Se establece la división en 3 grupos. Se elige una estética cubista, tanto para las unidades del ejército como para los héroes.

a. Puntos tratados

- División del equipo en 3 grupos.
- Visión de los mockups para establecer una estética.

b. Desarrollo:

- Sobre la división en 3 grupos:
  - Estado del arte, jugar a mobas, WOW, overlord, etc. Para estudiar el sistema de combate. Asignado a Javi y Diego.
  - Prototipo en Unity para probar combate que conteste a las siguientes preguntas:
    - Como se ataca y cómo se defiende, controles básicos.
    - Control de la cámara.
    - Relación del tamaño del héroe y las unidades del ejército.
    - Asignado a Maxi y Fernando.
  - Motor del juego: montar un escenario, mirar sistemas de niebla de guerra y online. Asignado a Dani y Guille.
- Se presentan los mockups preparados por Jacobo, elegimos héroes cúbicos.
  - tarea: hacer un héroe orco y un héroe robot.
- Hay que crear un trello: lo hace Guille.
- Se decide establecer un número de niveles de 5.
- Posibles temáticas:
  - Orcos.
  - robots (yo robot).
  - aliens (héroe Ripley).
  - dinosaurios.
  - Samuráis japoneses.
  - ninjas.

7. Acta del 12 de noviembre de 2013 (UCM)

Fecha	12 de noviembre de 2013
Concepto	Planificación anual.
Asistentes	Todos
Duración	1 hora
Conclusiones	Hemos puesto todas las tareas que queremos desarrollar durante el año. Le hemos dado una vuelta a la evolución del héroe.

a. Puntos tratados

- Diseño.
- Arte.
- Programación.

b. Desarrollo:

Evolución del héroe:

- Outgame:
  - Seleccionamos el árbol de habilidades del héroe.
  - Seleccionamos cartas para el héroe que potencian atributos.
- Ingame:
  - Según evolucionas al héroe eliges mejorar atributos.
  - Los objetos que recogemos potencian nuestras habilidades, y son activables.

8. Acta del 15 de noviembre de 2013 (ESNE)

Fecha	15 de noviembre de 2013
Concepto	Reunión de artistas y planificación arte hasta diciembre
Asistentes	Fernando, Jacobo, Maxi
Duración	1 hora
Conclusiones	Sprints para las próximas 4 semanas

a. Puntos tratados

- Planificación arte para las próximas semanas.

b. Desarrollo:

- 22 noviembre:
  - Concept general del escenario (Jacobo)
  - Modelo 3D del Orco (Fernando)
  - Mapeado del Orco (Fernando)
- 29 noviembre:
  - Concept props escenario 1 (Jacobo)
  - Texturizado del Orco (Jacobo)
  - Modelado 3D de props 1 (Fernando)
  - Mapeado 3D de props 1 (Fernando)
- 6 diciembre:
  - Texturizado de props 1 (Jacobo)

- Rigging Orco (Fernando)

- 13 diciembre:
  - Montar escenario (Fernando)
  - Animar Orco (Jacobo)
- 20 diciembre:
  - Unidades básicas RTS

Importante: para animar el orco es indispensable tener el combate bien diseñado y definido.

9. Acta del 26 de noviembre de 2013 (UCM)

Fecha	26 de noviembre de 2013
Concepto	Reunión de programadores y semi planificación de dos sprints
Asistentes	Todos
Duración	1 hora
Conclusiones	Sprints para las próximas 2 semanas

a. Puntos tratados

- Planificación de programadores para las próximas semanas.

b. Desarrollo

- 29 noviembre:
  - Niebla de guerra (1): asignado a Guille y Dani.
  - Cámara RTS/Cámara MoBA: asignado a Diego.
  - Selección y *goto* de unidades básicas 'RTS': asignado a Maxi y Fernando.
  - Diseño atributos específicos MOBA, unidades y torretas: asignado a Javier.
- 6 diciembre:
  - Pathfinding de RTS: asignado a Maxi y Fernando.
  - Niebla de guerra (2): asignado a Guille y Dani.

10. Acta del 29 de noviembre de 2013 (UCM)

Fecha	29 de noviembre de 2013
Concepto	Reunión de todos para definir los movimientos del orco
Asistentes	Todos
Duración	1 hora
Conclusiones	Hay que animar, mucho curro

a. Puntos tratados

- Habilidades y movimientos del orco

b. Desarrollo:

- Ataques:
  - Ataque con un bate.
  - Ataque con un bate 2.
  - Ataque patada.
  - Encadenamiento de estos 3 ataques físicos.
- Habilidades
  - Escupe y ralentiza al contrario.
  - Saca un radiocasete grande y se pone a hacer gestos de cuernos. Deja a los minions que tenga alrededor X segundos paralizados y X-Y segundos paralizado al otro héroe.
  - Carga como un toro y avanza unos metros llevándose todo por delante.
  - Golpe al suelo (a lo donkey kong), aprovechando lo grande que son sus brazos. Lanza a los enemigos que estén en el radio de alcance hacia atrás.
  - Eructa y se pone un mechero en la boca, sale una llamarada de fuego como si estuviera escupiendo fuego.
- Movimientos del orco (Animaciones):
  - Idle.
  - Idle plus (cuando lleva varios segundos sin recibir órdenes realiza esta acción solo).
  - Walk.
  - Run.
  - 3 ataques.
  - escupir.
  - eructar y colocarse el mechero en la boca.
  - Golpe al suelo.
  - Cargar como un toro.

11. Acta del 5 de diciembre de 2013 (UCM)

Fecha	5 de diciembre de 2013
Concepto	Sprint, evaluación sprints anteriores y preparación del siguiente
Asistentes	Todos
Duración	1 hora
Conclusiones	Establecemos un sprint para el viernes 13 de diciembre

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas del siguiente sprint.

b. Desarrollo:

Tareas previstas hasta la fecha:

- Programación
  - Se ha creado un documento de atributos de unidades y héroes para ayudar en la implementación.
  - RTS:
    - Selección de unidades.
    - Controlador de la base y del ejército.
    - Recolección (hay que darle otra vuelta).
    - Cámara.
    - Niebla de guerra (falta integrarla en el proyecto).
    - Pathfinding: Fer se está pegando con ello, falta darle vueltas.
  - MoBA:
    - Cámara e integración con el online.
- Arte:
  - Se han realizado múltiples assets para el escenario (modelado y texturizado).
  - El héroe orco esta modelado, texturizado y riggeado.
- Otros:
  - Gantt: se ha comenzado, falta darle unas vueltas.
  - Burn Down Chart: se ha comenzado, queda pendiente reunión con Javi.

Próximo sprint para el 13 de diciembre:

- Diego: seguirá con la cámara y con el online.
- Fer: seguirá con el pathfinding.
- Maxi: mirará algoritmos de bandada, cómo se mueven los grupos.

- Javi: terminará el Burn Down Chart.
- Cuestas (Guille y Dani): comenzarán con la implementación de los héroes (combate, atributos…).

12. Acta del 13 de diciembre de 2013 (ESNE)

Fecha	13 de diciembre de 2013
Concepto	Sprints de arte para las próximas semanas.
Asistentes	Jacobo y Maxi.
Duración	1 hora
Conclusiones	Preparamos 7 sprints (hasta febrero) y dividimos el trabajo. Además creamos un documento para listar los props del escenario (enlace).

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas del siguiente sprint.

b. Desarrollo:

El objetivo es tener un ejército de goblins modelados y animados para finales de enero, además de muchos más props para el escenario.

Hacemos un documento para listar todos los props necesarios.

Sprints en cada semana:

- Viernes 20 Diciembre.
  - Concept goblin (1) (Jacobo).
  - Texturizado edificio (David).
  - Modelado y mapeado props (El Otro edificio sin la formación alienígena) (Fer).
- Viernes 27 Diciembre.
  - Modelado y Mapeado goblin 1 (Fer).
  - Texturizado props (Edificio 2) (Jacobo).
  - Concepts ayuntamiento (base goblin), torreta goblin y almacén de recursos goblin (Jacobo).
- Viernes 3 Enero.
  - Texturizado goblin 1 (Jacobo).
  - Riggin goblin 1 (David).
  - Modelado y mapeado ayuntamiento (base goblin) (Fer).
- Viernes 10 Enero.
  - Animación goblin (1) (David)
  - Concept goblin (2) (Jacobo).

- Texturizado ayuntamiento (base goblin) (Jacobo).
- Modelado y mapeado torreta goblin (Fer).
- Texturizado torreta goblin (Jacobo).

- Viernes 17 Enero.
  - Modelado goblin (2) Y mapeado (Fer).
  - Texturizado Base orco (ayuntamiento) (David).
  - Modelado y mapeado y texturizado de props (Jacobo).
- Viernes 24 Enero.
  - Texturizado goblin (2) Y Riggin (Jacobo).
  - Modelado y mapeado y texturizado de props (Fer).
  - Texturizado Torreta orca (David).
- Viernes 31 Enero.
  - Animación goblin (2) (David),
  - Diseño del arte de la hud (Jacobo).
  - Diseño de logo y el arte de la interface del menú (Jacobo).
  - Más props (Fer).

13. Acta del 18 de diciembre de 2013 (UCM)

Fecha	18 de diciembre de 2013
Concepto	Sprint, evaluación sprints anteriores y preparación del siguiente.
Asistentes	Todos
Duración	2 horas
Conclusiones	Establecemos un sprint para el viernes 11 de enero

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas del siguiente sprint.

b. Desarrollo:

Tareas realizadas desde el último sprint:

- Los Cuesta están con el Hero controller, y la niebla ya está 100% integrada y funcional (habría que crear una malla mejor triangulada). Hacen falta más pruebas al jugar con otros jugadores, necesitará otra vuelta.
- Javi ha terminado el Burn Down Chart. Hay que comunicar a los de arte que rellenen sus trabajos aquí.
- Diego ha terminado de implementar en un mismo proyecto el RTS y el héroe.
- Maxi y Fernando han implementado el navemesh, y refinado cosas del RTS.

Próximo sprint para el 29 de diciembre:

- Fer**:** algoritmo para recolocar las posiciones de destino cuando se seleccionan múltiples unidades RTS y se las envía al mismo punto (algoritmo de bandada).
- Cuestas: seguirán con la jugabilidad del MOBA.
- Diego: ayudará a los Cuesta y consultar sistema online.
- Maxi: unidades bélicas del RTS (máquina de estados e implementación).
- Javi: comienzo del comportamiento de las torretas, definir experiencia del nivel de los héroes y su recompensa de experiencia en combate.

14. Acta del 10 de enero de 2014 (UCM)

Fecha	10 de enero de 2014
Concepto	Sprint, evaluación sprints anteriores y preparación del siguiente. Planificación a largo plazo.
Asistentes	Todos
Duración	2 horas
Conclusiones	Establecemos un sprint para el viernes 24 de enero

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas del siguiente sprint.
- Planificación a largo plazo.

b. Desarrollo:

Tareas realizadas desde el último sprint:

- Los Cuesta no han podido terminar.
- Javi no ha podido hacer lo de las torretas.
- Diego ha integrado el orco y ha cambiado las animaciones (componente Animation por animator).
- Fernando ha realizado el algoritmo de bandada para recolocar en formación las unidades del RTS.
- Maxi ha ajustado lo que hizo Fernando para que la formación rote en función de las posiciones origen de las unidades. También ha empezado con las unidades bélicas, pero no ha terminado.

Próximo sprint para el 24 de enero:

- Fer y Javi: comenzarán las torretas goblins.
- Cuestas y Diego: arreglarán bugs generales (fog plane, animación orco, etc.), avanzarán también con el combate del orco.

- Maxi: terminará las unidades bélicas del RTS (comportamiento básico).

Planificación:

- Hay que hacer un documento para listar assets de audio.
- Preparar un emisor de partículas para la habilidad de escupir del orco.
- Pensar cómo hacer los avisos de ataques a distancia (ej.: lanzar una granada, ver dónde va a caer, la parábola que seguirá).
- Para próximos sprints:
  - Empezar a meter assets terminados al nivel.
  - Menús.

15. Acta del 17 de enero de 2014 (ESNE)

Fecha	17 de enero de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación de los dos siguientes.
Asistentes	Fer, David, Jaco y Maxi
Duración	2 horas
Conclusiones	Establecemos dos sprints para los viernes 24 y 31 de enero

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas de los dos siguientes sprint.
- Se hace necesario documentar en una tabla el reparto de modelos 3D y animaciones para el ejército goblin.
- Otras ideas.

b. Desarrollo:

Para las unidades de exploradores se decide realizar un cortacésped ligero, en el que encima haya un Goblin01. Esto simplificará las animaciones, a una breve vibración (cuando esté en idle, o se desplace) una de ataque (retrocede y embiste levemente) y de muerte (el goblin cae al suelo).

Pensamos que estaría muy chulo que aparte de la animación de muerte, los goblin tuvieran una animación más de pos-morten, que consistiría en el espíritu del goblin elevándose al cielo y desapareciendo (con shaders).

Se establece nombres en clave para los modelos del ejército goblin, además de su disposición en las diferentes unidades que componen el ejército:

- *Goblin01*: el modelo goblin *clásico*, tipo *gollum*.
  - Recolectores.
  - Exploradores (encima de la máquina cortacésped).

- *Goblin02*: el modelo con el pelo afro.
    - Artillería ligera (dos variaciones de colores, solo cambios en la textura).
    - Ingenieros (con una mochila).
- *Goblin03*: el modelo de goblin grande, que tiene un cañón en la espalda.
    - Artillería pesada.

Próximo sprint para el 24 de enero:

- Jacobo: concept-Art del Goblin03 + Concept-Art del cortacésped + textura en otro color del Goblin02.
- Fernando: mochila para el ingeniero (modelado + texturizado) + montar más edificios.
- David: assets para los recolectores (modelado + texturizado) (cascos, conos, gafas 3d, cigarrillos) + animar Goblin02 (copiar las animaciones básicas del Goblin01 + animaciones de ataque de artillería ligera).

Sprint para el 31 de enero:

- Jacobo: texturizar Goblin01 (2 variantes: recolectores & exploradores que irán en el cortacésped).
- Fernando: modelo 3D Cortacésped (máquina + agregar Goblin01) + mapeado.
- David: modelo 3D Goblin03 (unidad artillería pesada) + mapeado.

Apuntes para los siguientes Sprints:

- Texturizar Goblin03.
- Texturizar cortacésped.
- Animaciones unidades exploradores (cortacésped + Goblin01).
- Animaciones unidades artillería pesada (Goblin03).
- Animaciones post-morten de todos los modelos (Goblin01, Goblin02 y Goblin03).
- Concept-Art torretas neutrales (del escenario) + modelado + mapeado + texturizado.
- Tiles de carretera, aceras, etc.
- Assets para la base del ejército goblin (chopers, vallas, Mustang, porche).

16. Acta del 7 de febrero de 2014 (ESNE)

Fecha	7 de febrero de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación de los dos siguientes.
Asistentes	Fer, David, Jaco y Maxi
Duración	1 hora
Conclusiones	Establecemos dos sprints para los viernes 21 y 28 de marzo

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas de los dos siguientes sprint.
- Otras ideas.

b. Desarrollo:

Tareas realizadas en los sprints previos:

- Goblin03, esta modelado y texturizado (falta animarlo).
- Cortacésped, modelado y mapeado (falta texturizar).
- Goblin01 y goblin02 tienen una textura, falta variación.
- Animaciones goblin01 y goblin02 ya terminadas.

Sobre las torretas neutrales: mejor hacer algo que *case* con la ciudad, decidimos hacer edificios altos pero más delgados. Hay que hacer arte primero y luego modelarlo.

Animaciones post-morten de todos los modelos (Goblin01, Goblin02 y Goblin03). Esto mejor lo hacemos en Unity.

Sprint 14 febrero:

- Fer: texturizar cortacésped.
- Jaco: concept torreta neutral, concepts de assets para la base del ejército goblin (chopers, vallas, Mustang, porche).
- David: animación Goblin03.

Sprint 28 febrero:

- Fer: tiles de carreteras, aceras, etc. (lo que le de tiempo)
- Jaco: minas de recursos, Modelo, mapeado y textura.
- David: modelos assets para la base del ejército goblin (chopers, vallas, Mustang, porche).

Cosas para siguientes sprints

- montar goblin01 en el cortacésped y animarlo.
- variaciones texturas goblin01, goblin02 y goblin03.

- Interfaz RTS.
- Texturizar assets base del ejército goblin.

17. Acta del 19 de febrero de 2014 (UCM)

Fecha	19 de febrero de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación de los dos siguientes.
Asistentes	Maxi, Diego, Javi, Dani, Guille, Fernando
Duración	1 hora
Conclusiones	Establecemos sprint para el viernes 28 de febrero

a. Puntos tratados

- Tareas realizadas en el último mes.
- Próximo sprint para el 28 de febrero, y primer esbozo del siguiente sprint (7 de marzo).
- Otros temas.

b. Desarrollo:

Tareas realizadas en los sprints previos:

- Fernando y Javi: lógica de las torres neutrales que son conquistadas, reparadas y atacan a los enemigos, nueva unidad ingeniería (las unidades ya pueden conquistar las torres y repararlas).
- Cuestas: han continuado con el HeroController (ya muere y reaparece en la posición de origen) y han aumentado con la documentación (tablas de experiencia).
- Diego: Ataque a distancia del héroe orco y ataque del pisotón.
- Maxi: documentación de diseño.

Necesitamos realizar un mapa de todo el escenario. Podemos meter los modelos de las unidades del ejército goblin. Hay que hacer una super clase.

Quedamos mañana (20 de febrero) para hacer el esquema de la memoria.

Sprint 28 de febrero:

- Cuestas: seguirán y finalizarán el HeroController y el OrcController:
  - Evolución del orco.
  - Elección de habilidades.
  - Usar habilidades.
  - Curación del héroe.
  - Golpes a las unidades del RTS.
- Maxi:
  - Crear mapeado de colisiones para la navemesh.
  - Crear una clase padre de todas las unidades (ejército y héroe).

    - Continuar con la documentación de diseño (extensión y concreción de varios puntos que siguen estando un tanto ambiguos).
    - Continuar con la clase base de la artillería RTS.
    - Comenzar con el diseño avanzado del mapeado.
- Diego: arreglar las cosas que se han roto al juntar lo nuevo del prototipo RTS con el proyecto general.
    - Hacer que la onda de choque de Rob haga daño.
    - Partículas para la carga de Rob.
- Fernando y Javi:
    - Arreglar la casilla rosa de reparación/conquista
    - Agregar la clase padre de héroe/unidad a towerVision
- Todos:
    - Preparar esquema de la memoria para entregárselo a los tutores.

Sprint 7 de marzo:

- Cuestas:
    - Ya se verá
- Maxi:
    - Terminar la clase base de la artillería RTS.
    - Continuar con el diseño avanzado del mapeado.
- Fernando y Javi:
    - Recolectores que curen al héroe
    - Meter los goblins
    - Construcción de torretas goblin
- Diego:
    - Ya se verá
- Todos:
    - Diseño del mapeado.

18. Acta del 3 de marzo de 2014 (UCM)

Fecha	3 de marzo de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación de los dos siguientes.
Asistentes	Maxi, Diego, Javi, Dani, Guille, Fernando
Duración	1 hora
Conclusiones	Establecemos sprint para el viernes 14 de Marzo

a. Puntos tratados

- Tareas realizadas en el último sprint.
- Próximo sprint para el 14 de marzo.
- Otros temas.

b. Desarrollo:

Tareas realizadas en los sprints previos:

- Fernando:
  - Continuación con las máquinas de estados de la torre goblin.
  - Máquina de estados de las unidades ingenieras.
  - Shader sobre la selección de unidades y sobre la torre goblin.
  - Comienzo de la programación de la torre goblin y continuación con la programación de las unidades ingenieras.
  - Nueva clase padre para las torres.
- Javi:
  - Creación de unidades exploradoras y su máquina de estados.
- Cuestas:
  - Continuación del Hero Controller: hud de la vida, ganancia de experiencia por matar unidades, subidas de nivel y elección de habilidades por subida de nivel.
- Diego:
  - Bugs arreglados del RTS relacionados con Photon.
  - Documentación
  - Comienzo con el sistema de partículas para la tercera habilidad del Orco.
- Maxi:
  - Creada clase padre de las unidades.
  - Comienzo con el diseño del mapeado (creación de un mini-nivel) e integración de props: modelos de Goblin01 y Goblin02, modelos de edificios, tiles de carretera.

Sprint 14 de marzo:

- Cuestas:
  - Terminar la evolución de los atributos del Orco (armadura, ataques básicos), aplicar resistencia sobre ataques por armadura y terminar ataques secundarios para que tengan su efecto de daño.
- Maxi:
  - Crear mapeado de colisiones para la navemesh.
  - Terminar la clase base de la artillería RTS.
  - Continuar con el diseño avanzado del mapeado.
  - Continuar con la documentación de diseño (extensión y concreción de varios puntos que siguen estando un tanto ambiguos).

- Fernando:
  - Comportamiento de las torretas goblin (programación y construcción).
- Javi:
  - Curación de unidades recolectoras y terminar detalles de los exploradores (a ampliar).
- Diego:
  - Investigación del sistema de partículas.

19. Acta del 7 de marzo de 2014 (UCM)

Fecha	7 de marzo de 2014
Concepto	Reunión con los tutores de SI
Asistentes	Maxi, Diego, Javi, Dani, Guille, Fernando Monasterio, Federico y Fernando Rubio
Duración	1 hora
Conclusiones	Hay que apretar, volvemos a quedar para el 21 de marzo (o para una semana después).

a. Puntos tratados

- Visión general del estado actual del proyecto.
- Correcciones sobre el esquema de la memoria y algo de la documentación.
- Otros temas.

b. Desarrollo:

Comentarios sobre la demo:

- Hay que definir más la jugabilidad del MOBA, especificar más todo y si hace falta copiar mecánicas de otro juego que sabemos que funcionan.
- El RTS se ve muy avanzado, pero molaría mucho ver todo más integrado (orco pegando a goblins).

Comentarios sobre el avance del proyecto:

- Realizamos mucho trabajo en tareas no previstas en el Burndown chart (investigación, pruebas, etc.), hay que añadir tareas adicionales para contabilizar ahí horas dedicadas a imprevistos.

Comentarios sobre el esquema de la memoria que preparamos:

- En el primer punto, antes que explicar las motivaciones, destacar la novedad del proyecto. Después ya ponemos algo de motivaciones.
- Separar el 5º punto (requisitos/objetivos) en dos independientes: tecnologías y diseño son suficientemente grandes como para ser dos entes independientes.

- Los puntos 6 y 7 (diseño UML / implementación): plantearse si es mejor tenerlos separados o unirlos en uno solo.
- Hay documentos mal redactados, hay que usar formas impersonales e intentar tener el mismo estilo en todos los documentos. A la hora de pasarlos a la memoria habrá que quitar cabeceras y demás.
- Introducir en los documentos qué se va a explicar. Por ejemplo: en el del Photon introducir qué es con un párrafo.
- Sobre la documentación de Unity que meteremos en la memoria: ideas básicas, qué se puede hacer con Unity y luego concretar en base a nuestro proyecto, a lo que nosotros hemos hecho.

20. Acta del 12 de marzo de 2014 (ESNE)

Fecha	12 de marzo de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación de los dos siguientes.
Asistentes	Fer, David, Jaco y Maxi
Duración	2 horas
Conclusiones	Establecemos dos sprints para los viernes 21, 28 de marzo, 4 y 11 de abril.

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas de los dos siguientes sprint.
- Otras ideas.

b. Desarrollo:

Tareas realizadas en los sprints previos:

- Goblin03, ya está animado y con una segunda textura morada.
- Cortacésped ya está texturizado (falta animar).
- Animaciones goblin01 y goblin02 ya terminadas.
- Torretas neutrales (Jacobo hizo el concept).
- Assets de la base del ejército Goblin (Jacobo hizo el concept).
- Tiles de suelo (Fer).

Siguientes tareas a planificar sobre el ejército Goblin:

- Animaciones del explorador (Goblin01 montado en cortacésped): harían falta tres animaciones:
  - Una para cuando se mueve y está quieto que sea básicamente una breve vibración (muy sencillo).

    - Ataque: simplemente como una leve embestida (se echa un poco hacia atrás y se mueve hacia adelante de forma violenta). Comentario reunión: mejor que se dé la vuelta, se levanta y lanza residuos desde la parte de abajo.
    - Muerte: el orco cae del cortacésped y queda tendido en el suelo.
- Goblin02 versión ingeniero: faltan las animaciones de esta unidad (sin armas):
    - Idle, en espera.
    - Idle-Wait.
    - Walk: desplazamiento.
    - Ataque: copiado del documento de diseño: *los ingenieros pueden atacar a distancia lanzando bombas fétidas, este ataque se desarrolla de la siguiente manera: la unidad se agacha y comienza a construir algo con sus manos en el suelo durante 1 o 2 segundos, después se levanta y lo lanza como si fuera una granada*.
    - Conquista y construcción: como la de picar de los recolectores (hará falta un martillo o algo similar). Sacan un portátil, hacen algo con él y luego golpean con el portátil.
    - Muerte (esta creemos que valdría la que ya está hecha).
- Goblin02 versión artillería: falta variación de textura morada.
- Goblin01: falta variación de textura morada.
- Goblin02 versión ingeniero: falta una variación (2 contando la variación morada) para distinguir el modelo de los artilleros.
- Almacén de recursos: concept, modelo y textura.

Siguientes tareas sobre assets del escenario:

- Torretas neutrales: ya está el concept art, falta modelar, mapear y texturizar.
- Cementerio de neumáticos: concept art (muy rápido y simple), modelado y texturizado.
- Assets de la base de ejército goblin: modelos y texturas del porche y de las chopers.
- Algunos tiles más, faltan de césped.
- Props para la cancha de básquet (vallas, mejor que sea un plano con alpha, suelo y canastas): concept (creo que ni haría falta), modelado y texturizado.

Siguientes tareas sobre el equipo de SkelterBot:

- Concept de las unidades del ejército 3 robots:
    - Robot01: para los recolectores y los exploradores (irán subidos en algún vehículo).
    - Robot02: para la artillería ligera y los ingenieros.
    - Robot03: para la artillería pesada.
- Modelado, mapeado y texturizado de SkelterBot.
- Rigeado y Animaciones de SkelterBot.

Sprint 21 de marzo:

- Fer: tiles que faltan (césped), modelado y mapeado de las torretas neutrales.
- Jaco: concept rápido del cementerio de neumáticos, props para la cancha de básquet, concept del almacén de recursos del ejército goblin concept del Robot01.
- David: animación Goblin02 versión ingeniero y animaciones explorador (Goblin01 en cortacésped).

Sprint 28 de marzo:

- Fer: modelado y mapeado de los assets de la base del ejército goblin (porche con mesas y sillas y moto chopper).
- Jaco: concepts del Robot02 y del Robot03.
- David: modelado, mapeado y texturizado del cementerio de neumáticos. Modelado mapeado y texturizado del almacén de recursos del ejército goblin.

Sprint 4 de abril:

- Fer: modelado y mapeado de SkelterBot.
- Jaco: texturizado de los assets de la base del ejército goblin y del almacén de recursos del ejército goblin, y si da tiempo hacer un coche (Mustang).
- David: modelado y mapeado de Robot01.

Sprint 11 de abril:

- Fer: modelado y mapeado de Robot02.
- Jaco: texturizado de Skelterbot y de Robot01.
- David: rigeado y comienzo de la animaciones de Skelterbot.

Sprint 18 de abril:

- Fer: modelado y mapeado de Robot03.
- Jaco: texturizado de Robot02.
- David: continuar con las animaciones de Skelterbot.

Tareas para siguientes sprints:

- Texturizar Robot03.
- Rigeado y animaciones de Robot01.
- Rigeado y animaciones de Robot02.
- Rigeado y animaciones de Robot03.
- Concept, modelado y texturizado de la base del ejército robot.
- Concept, modelado y texturizado del almacén de recursos del ejército robot.

Quedaría la mitad de Abril y todo Mayo (6 semanas) para depurar cosas, trabajar en la GUI y hacer algún asset más para el escenario.

Nota: mirar la generación de terrenos.

21. Acta del 14 de marzo de 2014 (UCM)

Fecha	14 de marzo de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación de los dos siguientes.
Asistentes	Maxi, Diego, Javi, Dani, Guille, Fernando
Duración	1 hora
Conclusiones	Establecemos sprint para el viernes 21 de Marzo.

a. Puntos tratados

- Tareas realizadas en el último sprint.
- Próximo sprint para el 21 de marzo.
- Otros temas.

b. Desarrollo:

Tareas realizadas en los sprints previos:

- Fernando:
  - Finalización de la programación y comportamiento de Tower, TowerGoblin, TowerNeutral.
  - Las unidades ingenieras construyen y reparan la TowerGoblin.
  - Offmesh link para las unidades exploradoras.
  - Documentación: Navigation y pathfinding.
- Javi:
  - Patrulla de las unidades exploradoras.
  - Habilidad de curación de las unidades recolectoras.
- Cuestas: continuación del Hero Controller:
  - Terminar la evolución de los atributos del Orco (armadura, ataques básicos).
  - Aplicar resistencia sobre ataques por armadura.
  - Comienzo del panel inferior del orco.
  - Pasar animación del orco a código.
  - Añadir al hud del orco el maná y la adrenalina.
- Diego:
  - Comienzo de ataques secundarios para que tengan su efecto de daño.
- Maxi:
  - Crear mapeado de colisiones para la navemesh.
  - Continuar con la clase base de la artillería RTS y artillería pesada.
  - Continuar con el diseño avanzado del mapeado.
  - Una valla muy chula y texturas moradas para los goblin.

- Continuación de artillería básica e inicio de artillería pesada (modo de despliegue).
- Documentación.

Sprint 21 de marzo:

- Cuestas: continuación del Hero Controller:
  - Terminar el panel inferior del orco.
  - Cambiar en el orco el animator por animation.
- Maxi:
  - Continuación de la artillería.
  - Mapeado
- Fernando:
  - Construcción de almacenes de recursos por los ingenieros.
  - Que los harvesters vayan después de recolectar al almacén más cercano.
  - Comportamiento de los almacenes.
  - Crear clase padre llamada Building.cs de la que hereden todas las edificaciones que sean destruibles (base, tower, warehouse)
  - Actualizar la clase tower con lo de building
- Javi:
  - Documentación.
  - Diseño de las habilidades outgame.
- Diego:
  - Continuar con los ataques secundarios del orco.
  - Documentación.

22. Acta del 25 de marzo de 2014 (UCM)

Fecha	25 de marzo de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación de los dos siguientes.
Asistentes	Maxi, Diego, Javi, Dani, Guille, Fernando
Duración	1 hora
Conclusiones	Establecemos sprint para el viernes 27 de Marzo

a. Puntos tratados

- Tareas realizadas en el último sprint.
- Próximo sprint para el 27 de marzo.
- Otros temas.

b. Desarrollo:

Tareas realizadas en los sprints previos:

- Cuestas: continuación del Hero Controller:
  - Terminar el panel inferior del orco.
  - Cambiar en el orco el animator por animation.
  - Animaciones del orco reparadas.
- Maxi:
  - Props.
  - Integración de animaciones.
  - Mejoras en el armyController y baseController.
  - Correcciones en shaders y materiales.
  - Integración de la patrulla.
- Fernando:
  - Construcción de almacenes de recursos por los ingenieros.
  - Comportamiento de los harvesters: tienen que ir después de recolectar al almacén más cercano o a la base.
  - Comportamiento de los almacenes.
  - Clases padre Building.cs y CResourceBuilding.cs de las que hereden todas las edificaciones que sean destruibles (base, tower, warehouse).
  - Actualizar las clases de edificios.
  - Rehacer y actualizar comportamiento de ingenieros.
- Javi:
  - Documentación.
  - Diseño de las habilidades outgame.
- Diego:
  - Continuar con los ataques secundarios del orco.
  - Documentación.

Sprint 27 de marzo:

- Cuestas:
  - continuación del Hero Controller:
  - Funcionamiento de los orcos con Photon.
- Maxi:
  - Malla del mapa.
- Fernando:
  - Integración del modelo del ingeniero con las animaciones.
- Javi:
  - Investigación de partículas.

- Diego:
  - Mantenimiento del Photon.
  - 3ª habilidad del orco.

Todos: hacer memoria.

23. Acta del 3 de abril de 2014 (ESNE)

Fecha	3 de abril de 2014
Concepto	Scrum, evaluación sprint anteriores y preparación del siguiente.
Asistentes	Fer, David, Jaco, L. Suárez, L. Álvarez y Maxi
Duración	1 hora
Conclusiones	Establecemos un sprint para el viernes 11 de abril

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas del siguiente sprint.
- Otras ideas.

b. Desarrollo:

Tareas realizadas en los sprints previos:

- David:
  - Animaciones Skelterbot (a medias).
- Jacobo:
  - Modelar y mapeado el Robot03.
  - Modelado de la base del ejército robot.
  - Textura de las torretas neutrales.
- Fer:
  - Modelado y mapeado del Robot01.
  - Modelado y mapeado del Robot02.
- L. Suárez:
  - buzón de correos, farolas, banco (modelado y texturizado).
- L. Álvarez:
  - Asset para el parque: escenario abierto (bioshock infinite).

Tareas que quedan sin hacer en el sprint previo:

- Puesto de perritos calientes: modelado y mapeado.
- Torreta con cámara de seguridad modelado, mapeado y texturizado.

Siguientes tareas sobre assets del escenario:

- Assets de la base de ejército goblin: textura las chopers (estaría bien hacer 2 variaciones de color).
- Props en general: coche destrozado, señales de tráfico, una fuente, papeleras, etc.
- La malla del escenario.

Siguientes tareas sobre el equipo de SkelterBot:

- Texturizado de SkelterBot.
- terminar animaciones de SkelterBot.
- Texturas de los Robots01, 02 y 03.
- Riggear y animar los Robots01, 02 y 03.
- Textura de la Base.
- Concept, modelado y textura de la Torreta.
- Concept, modelado y textura del almacén de recursos.

Tareas previamente previstas para el sprint del 11 de abril:

- Jaco: Texturizado de Skelterbot y de Robot01.
- David: Rigeado y animaciones de Skelterbot.
- L. Suárez: Props para el parque: fuente, arbustos, farola y papelera (modelado y texturizado).

Sprint 11 de abril:

- David:
  - Terminar animaciones Skelterbot.
  - (si da tiempo) modelos de las armas de Skelterbot.
- Jacobo:
  - Textura del Skelterbot.
  - Textura del Robot02.
- Fer:
  - Riggear Robot01.
  - Riggear Robot02.
- L. Suárez:
  - Assets: papelera y coche abandonado.
- L. Álvarez:
  - Assets: Puente, modelo y mapeado del puesto de perritos calientes.

Sprint 18 de abril:

- Fer: Animaciones de Robot01 versión recolector.

- Jaco: Concept de la base del ejército robot, concept de las torretas del ejército robot, concept del almacén de recursos del ejército robot.
- David: Animaciones de Robot02 versión artillería ligera.
- L. Suárez: Texturas Robot02 (dos versiones de colores, como los goblin verdes y morados).
- L. Álvarez: Texturas Robot03 (dos versiones de colores, como los goblin verdes y morados).

Sprint 25 de abril:

- Fer: Animaciones de Robot02 versión ingenieros.
- Jaco: Animaciones del Robot03 (artillería pesada).
- David: Animaciones de Robot01 versión explorador.
- L. Suárez: Modelado y texturizado de la base del ejército robot.
- L. Álvarez: Modelado y texturizado de las torretas del ejército robot.

Tareas para siguientes sprints:

- Modelado y texturizado del almacén de recursos del ejército robot.
- Animaciones del monstruo de la fuente del parque.
- Modelado de la malla del suelo del escenario.

Quedaría la mitad de todo Mayo (5 semanas) para depurar cosas, trabajar en la GUI y hacer algún asset más para el escenario.

24. Acta del 4 de abril de 2014 (UCM)

Fecha	4 de abril de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación del siguiente.
Asistentes	Maxi, Javi, Dani, Guille, Fernando, Diego
Duración	1 hora
Conclusiones	Establecemos sprint para el jueves 10 de Abril

a. Puntos tratados

- Tareas realizadas en el último sprint.
- Próximo sprint para el 10 de abril.
- Otros temas.

b. Desarrollo:

Establecemos hacer aparte de las tareas previstas, algo de la memoria y cada uno hará una parte.

Tareas realizadas en los sprints previos (4 de abril):

- Dani:
  - Ha corregido la animación de Rob por red (ataque secundario).
  - Documentación.
- Guille:
  - Conexión red del orco.
  - Documentación.
  - Funcionalidad de resta de maná y adrenalina (ataques de Rob).
  - Cooldown de las habilidades.
  - Corrección de errores.
- Maxi:
  - Ataque básico de todas las tropas.
  - Distances Measurer Tool: clase para controlar las distancias entre todas las unidades. (Eliminados componentes RigidBody).
  - Continuar con la artillería, movimiento con ataque (A).
  - Documentación.
- Fernando:
  - Nuevos modos de selección de unidades.
  - Nuevo disparo del ingeniero con adición de fuerzas.
  - Documentación.
- Javi:
  - Partículas del explorador.
  - Documentación.
- Diego:
  - Mantenimiento del Photon.
  - Actualización del golpe del orco contra el suelo con adición de fuerzas.

Sprint 10 de abril:

- Cuestas: continuación del Hero Controller:
  - Afinar el orco.
- Maxi:
  - Ataque físico de la artillería básica.
  - Listar sonidos.
- Fernando:
  - Ataque en modo desplegado de la artillería pesada.
- Diego:
  - Partida RTS online.

25. Acta del 10 de abril de 2014 (UCM)

Fecha	10 de abril de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación del siguiente.
Asistentes	Maxi, Javi, Dani, Guille, Fernando, Diego
Duración	1 hora
Conclusiones	Establecemos sprint para el viernes 25 de Abril

a. Puntos tratados

- Tareas realizadas en el último sprint.
- Próximo sprint para el 25 de abril.
- Otros temas.

b. Desarrollo:

Establecemos hacer aparte de las tareas previstas, algo de la memoria, cada uno hará una parte.

Tareas realizadas en los sprints previos (10 de abril):

- Cuestas:
  - Afinar el orco.
  - Tercer ataque de Rob.
- Maxi:
  - Ataque físico de la artillería básica.
  - Integración de nuevos props.
- Fernando:
  - Minimapa.
- Diego:
  - Partida RTS online. Ha empezado, tiene mucha tela.

Tareas que quedan sin hacer el último sprint:

- Listado de efectos de sonido.
- Ataque en modo desplegado de la artillería pesada.

Sprint 10 de abril:

- Cuestas:
  - Seguir afinando el orco.
  - Comenzar con el comportamiento de Skelterbot.
- Maxi:
  - Ayudar a diego con el online RTS.
  - Listar sonidos.

- Documentación.
- Integrar props y los nuevos modelos que vayan saliendo.

- Fernando:
  - Ataque en modo desplegado de la artillería pesada.
  - Documentación.
  - Minimapa: añadir la niebla.
- Javi:
  - Ayudar a los cuestas.
  - Documentación.
- Diego:
  - Partida RTS online.
  - Documentación.

26. Acta del 10 de abril de 2014 (ESNE)

Fecha	10 de abril de 2014
Concepto	Scrum, evaluación sprint anteriores y preparación del siguiente.
Asistentes	Fer, David, Jaco, L. Suárez, L. Álvarez, Santi y Maxi
Duración	1 hora
Conclusiones	Establecemos un sprint para el viernes 25 de abril

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas de los siguientes sprints.
- Otras ideas.

b. Desarrollo:

Tareas realizadas en los sprints previos:

- David:
  - Animaciones Skelterbot.
- Jacobo:
  - Textura del Skelterbot.
  - Textura del Robot02.
- Fer:
  - Riggear Robot01.
  - Riggear Robot02.
- L. Suárez:

    - Assets: papelera y coche abandonado.
- L. Álvarez:
    - Assets: Puente, y puesto de perritos calientes (modelo, mapeado y textura).
- Santiago Rico:
    - Cámara de vigilancia.

Siguientes tareas sobre assets del escenario:

- Assets de la base de ejército goblin: textura las chopers (estaría bien hacer 2 variaciones de color).
- Props en general: señales de tráfico, una fuente, papeleras, etc.
- La malla del escenario.

Siguientes tareas sobre el equipo de SkelterBot:

- Texturizado de SkelterBot.
- Texturas de los Robots01 y 03.
- Animar los Robots01, 02 y 03.
- Textura de la Base.
- Concept, modelado y textura de la Torreta.
- Concept, modelado y textura del almacén de recursos.

Tareas previamente previstas para el sprint del 18 de abril:

- Fer: Animaciones de Robot01 versión recolector.
- Jaco: Concept de la base del ejército robot, concept de las torretas del ejército robot, concept del almacén de recursos del ejército robot.
- David: Animaciones de Robot02 versión artillería ligera.
- L. Suárez: Texturas Robot02 (dos versiones de colores, como los goblin verdes y morados).
- L. Álvarez: texturas Robot03 (dos versiones de colores, como los goblin verdes y morados).

Tareas previamente previstas para el sprint del 24 de abril:

- Fer: animaciones de Robot02 versión ingenieros.
- Jaco: animaciones del Robot03 (artillería pesada).
- David: animaciones de Robot01 versión explorador.
- L. Suárez: modelado y texturizado de la base del ejército robot.
- L. Álvarez: modelado y texturizado de las torretas del ejército robot.

Sprint 24 de abril:

- David:
    - Animaciones del Robot03.

- Jacobo:
  - Textura del Robot01.
  - Textura del Robot03.
  - Animaciones del Robot01.
- Fer:
  - Animaciones del Robot02.
  - Cohete para los exploradores robot (Robot02).
- L. Suárez:
  - Fuente para el parque.
  - Columpios y atracciones para el parque.
- L. Álvarez:
  - Modelo y mapeado de las arañas Dralien.
  - Set the assets para los harvester (Goblin01, casco, gafas, mochila).
- Santiago Rico:
  - Plastilina en Unity del nivel.
  - Assets de gasolinera.

Tareas para siguientes sprints:

- Modelado y texturizado del almacén de recursos del ejército robot.
- Animaciones del monstruo de la fuente del parque.
- Modelado de la malla del suelo del escenario.

Quedaría la mitad de todo Mayo (5 semanas) para depurar cosas, trabajar en la GUI y hacer algún asset más para el escenario.

27. Acta del 24 de abril de 2014 (UCM)

Fecha	24 de abril de 2014
Concepto	Reunión con los tutores de SI
Asistentes	Maxi, Diego, Dani, Guille, Fernando Monasterio, Federico y Fernando Rubio
Duración	1 hora
Conclusiones	Apuntes y correcciones sobre el primer borrador de la memoria. A partir de ahora nos reuniremos todas las semanas, comenzamos el 9 de mayo a las 12.

a. Puntos tratados

- Apuntes y correcciones sobre el primer borrador de la memoria.

b. Desarrollo:

Apuntes y correcciones sobre el primer borrador de la memoria entregado a los tutores:

- Capítulo de implementación: mejor mezclarlo con el 5. Análisis y diseño del proyecto, apartado b) Diseño del software. Si este punto crece demasiado lo sacamos del apartado para pasar a ser un tema independiente.
- Sería muy interesante añadir un glosario de términos en los apéndices.
- Figuras (imágenes): tienen que tener una referencia (a pie de imagen) numeradas y luego un índice de figuras al principio de la memoria.
- El tema 1. c) Objetivos: explicar la gracia del proyecto, qué es lo que le hace especial, por qué mola. Un breve párrafo en este tema, después extenderlo y detallarlo más en el 3. a) Objetivos y material
- El tema 4 (Documento de diseño de juego) si no se extiende mucho sería mejor meterlo en el tema 3. c).
- Navemesh: separar la parte más *pura* de Unity, para que pase a un apartado del tema 5. a. ii. de la parte más de nuestro juego (5. b. i. 5).
- Arte: dejar un apartado como apéndice, muy chulo y visual.
- El tema 8 (Pruebas) si no es muy extenso lo metemos en el tema de Discusión.
- Eliminamos los temas de Contribución y de Resultados.
- Documento de Estado del arte: la lista de juegos parece muy caótica, sin ningún orden aparente. Poner un índice con los que se van a explicar después y después una lista con el resto a modo de referencia.

Otros comentarios sobre la memoria:

- Hay que ser consistentes con el uso del español-inglés.
- *Comillas* vs *cursiva*, hay que ser coherente, todo en *cursiva*.

Otros asuntos:

- Estaría bien hacer un calendario de implementación y que cuando se toque algo probar que no se haya roto nada antes.
- Volvemos a quedar el día 9 de mayo a las 12.

28.Acta del 25 de abril de 2014 (ESNE)

Fecha	25 de abril de 2014
Concepto	Scrum, evaluación sprint anteriores y preparación del siguiente.
Asistentes	Fer, David, L. Suárez, L. Álvarez, Santi y Maxi
Duración	1 hora
Conclusiones	Establecemos un sprint para el viernes 2 de mayo

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas de los siguientes sprints.
- Otras ideas.

b. Desarrollo:

Tareas realizadas en los sprints previos:

- David:
  - Animaciones Skelterbot.
- Jacobo:
  - Textura del Robot01.
  - Textura del Robot03.
- Fer:
  - Cohete para los exploradores robot (Robot02).
- L. Suárez:
  - Fuente para el parque.
  - Columpios y atracciones para el parque.
- L. Álvarez:
  - Modelado, mapeado y textura de las arañas Dralien.
  - Set the assets para los harvester (Goblin01, casco, gafas, mochila).
- Santiago Rico:
  - Assets de gasolinera.

Cosas que han quedado sin hacer en el sprint previo:

- Plastilina en Unity del nivel.
- Animaciones Robot02.
- Animaciones del Robot03.

Siguientes tareas sobre assets del escenario:

- Assets de la base de ejército goblin: textura las chopers (estaría bien hacer 2 variaciones de color).
- Props en general: señales de tráfico, una fuente, papeleras, etc.
- La malla del escenario.

Siguientes tareas sobre el equipo de SkelterBot:

- Animar los Robots01, 02 y 03.
- Textura de la Base.
- Concept, modelado y textura de la Torreta.

- Concept, modelado y textura del almacén de recursos.

Tareas previamente previstas para el sprint del 18 de abril:

- Fer: Animaciones de Robot01 versión recolector.
- Jaco: Concept de la base del ejército robot, concept de las torretas del ejército robot, concept del almacén de recursos del ejército robot.
- David: Animaciones de Robot02 versión artillería ligera.
- L. Suárez: Texturas Robot02 (dos versiones de colores, como los goblin verdes y morados).
- L. Álvarez: Texturas Robot03 (dos versiones de colores, como los goblin verdes y morados).

Tareas previamente previstas para el sprint del 24 de abril:

- Fer: Animaciones de Robot02 versión ingenieros.
- Jaco: Animaciones del Robot03 (artillería pesada).
- David: Animaciones de Robot01 versión explorador.
- L. Suárez: Modelado y texturizado de la base del ejército robot.
- L. Álvarez: Modelado y texturizado de las torretas del ejército robot.

Sprint 2 de mayo:

- David:
  - Animaciones del Robot03.
- Jacobo:
  - Texturizar base del ejército Robot.
  - Texturizar el almacén de recursos del ejército robot
- Fer:
  - Modelar y mapear el almacén de recursos del ejército Robot.
  - Modelar y mapear las torretas del ejército Robot.
- L. Suárez:
  - Assets: otro coche.
  - Estudio de mercado en Facebook.
- L. Álvarez:
  - Texturas de la moto (2 colores) y de la cámara de vigilancia.
- Santiago Rico:
  - Plastilina en Unity del nivel.

Tareas para siguientes sprints:

- Animaciones del monstruo de la fuente del parque.
- Modelado de la malla del suelo del escenario.

29. Acta del 25 de abril de 2014 (UCM)

Fecha	25 de abril de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación del siguiente.
Asistentes	Maxi, Javi, Dani, Guille, Fernando, Diego
Duración	1 hora
Conclusiones	Establecemos sprint para el miércoles 30 de Abril

a. Puntos tratados

- Tareas realizadas en el último sprint.
- Próximo sprint para el 30 de abril.
- Otros temas.

b. Desarrollo:

Tareas realizadas en los sprints previos (10 de abril):

- Maxi:
  - Integrar props y los nuevos modelos que vayan saliendo.
  - Primera versión de la memoria.
- Fernando:
  - Ataque en modo desplegado de la artillería pesada.
  - Minimapa: añadir la niebla.
- Javi:
  - Documentación.
- Diego:
  - Partida RTS online. (ha seguido con ello, es muy chungo)

Tareas que quedan sin hacer el último sprint:

- Listado de efectos de sonido.
- Actualizar minimapa con edificios sin desaparecer y que se actualicen cuando se vean.
- Actualizar máquinas de estados y pasarlas a IEEE.
- Comenzar con el comportamiento de Skelterbot.

Sprint 30 de abril:

- Dani y Javi:
  - Seguir afinando el orco (tercera habilidad).
  - Comenzar con el comportamiento de Skelterbot.
- Maxi:
  - Listar sonidos.

- Integrar props y los nuevos modelos que vayan saliendo: nuevos props del escenario, nuevos accesorios de los harvester goblin, modelos y animaciones de los robots.

- Fernando:
  - Actualizar minimapa con edificios sin desaparecer y que se actualicen cuando se vean.
- Diego y Guille:
  - Online RTS.
- Javi:
  - Documentación.

30. Acta del 30 de abril de 2014 (UCM)

Fecha	30 de abril de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación del siguiente.
Asistentes	Maxi, Javi, Dani, Guille, Fernando, Diego
Duración	1 hora
Conclusiones	Establecemos sprint para el viernes 9 de mayo

a. Puntos tratados

- Tareas realizadas en el último sprint.
- Próximo sprint para el 9 de mayo.
- Otros temas.

b. Desarrollo:

Establecemos hacer aparte de las tareas previstas, algo de la memoria, cada uno hará una parte.

Tareas realizadas en los sprints previos (30 de abril):

- Dani y Javi:
  - Seguir afinando el orco (tercera habilidad).
  - Comenzar con el comportamiento de Skelterbot.
- Maxi:
  - Listar sonidos.
  - Integrar props y los nuevos modelos que vayan saliendo: nuevos props del escenario, nuevos accesorios de los harvester goblin, modelos y animaciones de los robots.
- Fernando:
  - Documentación.

- Javi:
  - Documentación.
- Diego y Guille:
  - Online RTS (A medias).

Tareas que quedan sin hacer el último sprint:

- Listado de efectos de sonido.
- Actualizar minimapa con edificios sin desaparecer y que se actualicen cuando se vean.
- Actualizar máquinas de estados y pasarlas a IEEE.
- Comenzar con el comportamiento de Skelterbot.
- Integrar props y los nuevos modelos que vayan saliendo: nuevos props del escenario, nuevos accesorios de los harvester goblin, modelos y animaciones de los robots.
- Seguir afinando el orco (tercera habilidad).
- Comenzar con el comportamiento de Skelterbot.
- Puntos 2b y 2c del esquema de Federico.
- Online RTS (A medias).
- Documentación.

Tareas para el próximo sprint 9 de mayo

- Dani y Javi:
  - Seguir afinando el orco. (tercera habilidad).
  - Comenzar con el comportamiento de Skelterbot.
- Maxi:
  - Listar sonidos.
  - Integrar props y los nuevos modelos que vayan saliendo: nuevos props del escenario, nuevos accesorios de los harvester goblin, modelos y animaciones de los robots.
- Fernando:
  - Resolver estado *flying*.
- Fernando y Maxi
  - Ataque a las estructuras.
- Javi:
  - Documentación.
- Diego y Guille:
  - Terminar el Online RTS (A medias) y testearlo mucho.

31. Acta del 8 de mayo de 2014 (ESNE)

Fecha	8 de mayo de 2014
Concepto	Scrum, evaluación sprint anteriores y preparación del siguiente.
Asistentes	Fer, David, L. Suárez, L. Álvarez, Santi y Maxi
Duración	1 hora
Conclusiones	Establecemos un sprint para el viernes 16 de mayo

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas de los siguientes sprints.
- Otras ideas.

b. Desarrollo:

Tareas realizadas en los sprints previos:

- David:
  - Animaciones del Robot03.
- Jacobo:
  - Nivel Unity.
- Fer:
  - Animaciones del Robot02.
- L. Suárez:
  - Estudio de mercado.
  - Plan de empresa (dafo, cosas de producción).
- L. Álvarez:
  - Texturizar la torreta del ejército Robot.
  - Texturizar el almacén de recursos del ejército robot.
- Santiago Rico:
  - Animaciones de la araña.

Siguientes tareas sobre assets del escenario:

- Props en general: señales de tráfico, una fuente, papeleras, etc.
- La malla del escenario.

Sprint 16 de mayo:

- David:
  - Animaciones del Robot01.

- Jacobo:
  - Empezar con la GUI.
- Fer:
  - Catedral.
- L. Suárez:
  - Continuar con los temas de producción.
- L. Álvarez:
  - más props: Cofre, tanque, árbol.
- Santiago Rico:
  - Comenzar el desarrollo de la web.

Tareas para siguientes sprints:

- Animaciones del monstruo de la fuente del parque.
- GUI y menús.
- Arreglar cosas.
- Más props, edificios, objetos para los minions…
- Pájaro mutante con 3 ojos.
- Assets para un cementerio.
- Tentáculo
- Monturas para los héroes

32.Acta del 9 de mayo de 2014 (UCM)

Fecha	9 de mayo de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación del siguiente.
Asistentes	Maxi, Javi, Dani, Guille, Fernando, Diego
Duración	1 hora
Conclusiones	Establecemos sprint para el viernes 16 de mayo

a. Puntos tratados

- Tareas realizadas en el último sprint.
- Próximo sprint para el 16 de mayo.
- Otros temas.

b. Desarrollo:

Establecemos hacer aparte de las tareas previstas, algo de la memoria, cada uno hará una parte.

Tareas realizadas en los sprints previos (30 de abril):

- Dani y Javi:
  - Resolver bugs.
- Maxi:
  - Integrar props (muchos muchos) y los nuevos modelos que vayan saliendo: nuevos props del escenario, nuevos accesorios de los harvester goblin, modelos y animaciones de los robots.
  - Arreglar update de los recolectores.
  - Arreglar curación de los recolectores.
  - Arreglar explosión de los exploradores.
  - Nuevo mapa (se supone que será el final).
- Fernando:
  - Resolver estado *flying*.
  - Actualizar Partículas.
  - Arreglar update de los ingenieros.
- Javi:
  - Puntos 2b y 2c del esquema de Federico (falta revisar).
  - Partículas de las arañas, de los minerales del metro y de picar de los recolectores.
- Diego y Guille:
  - Animaciones en el online.
  - Online RTS (A medias).

Tareas que quedan sin hacer el último sprint:

- Listado de efectos de sonido.
- Actualizar minimapa con edificios sin desaparecer y que se actualicen cuando se vean.
- Actualizar máquinas de estados y pasarlas a IEEE.
- Comenzar con el comportamiento de Skelterbot.
- Seguir afinando el orco. (tercera habilidad).
- Comenzar con el comportamiento de Skelterbot.
- Online RTS (A medias).

Tareas para el próximo sprint 16 de mayo

- Dani y Javi:
  - Seguir afinando el orco. (tercera habilidad).
  - Comenzar con el comportamiento de Skelterbot.
- Maxi:
  - Documentación.
  - Integrar props y los nuevos modelos que vayan saliendo.
  - Componente de pertenencia a equipo (CTeam).

- Listado de efectos de sonido.

- Fernando:
  - Documentación.
- Fernando, Guille y Diego:
  - Online del RTS.
- Fernando y Maxi:
  - Ataque a las estructuras.
- Javi:
  - Documentación.
- Cuestas:
  - Documentación.

33. Acta del 16 de Mayo de 2014 (UCM)

Fecha	16 de mayo de 2014
Concepto	Scrum, evaluación sprints anteriores y preparación del siguiente.
Asistentes	Maxi, Javi, Dani, Guille, Fernando, Diego
Duración	1 hora
Conclusiones	Establecemos sprint para el viernes 23 de mayo

a. Puntos tratados

- Tareas realizadas en el último sprint.
- Próximo sprint para el 23 de mayo.
- Otros temas.

b. Desarrollo:

Establecemos hacer aparte de las tareas previstas, algo de la memoria, cada uno hará una parte.

Tareas realizadas en los sprints previos (30 de abril):

- Dani y Javi:
  - Seguir afinando el orco (tercera habilidad).
  - Comenzar con el comportamiento de Skelterbot.
- Maxi:
  - Integrar props y los nuevos modelos que vayan saliendo.
  - Componente de pertenencia a equipo (CTeam).
  - Listado de efectos de sonido.

    - Muchos fixes
- Fernando:
    - Documentación.
- Javi:
    - Documentación.

Tareas que quedan sin hacer el último sprint:

- Quitar vida a edificios: añadir edificios a la matriz de distancias cuando se construye y conquistan.
- Quitar que lance un rayo al construir una torre o almacén: con canConstruct vale.
- El puente tiene que tener una meshCollider.
- Off-mesh link para saltar el río.
- NGUI.
- Online:
    - Instanciar edificios con componente Photon.
    - Quitar vida a edificios.
    - Quitar vida a unidades.
    - Actualización de recursos de las minas online (si quitas mina al remoto quitarle también).
    - Gestionar menús y que salgan bonitos.

Tareas para el próximo sprint 23 de mayo

- Todos, cuando acabes con tus tareas: NGUI.
- Dani y Javi:
    - Acabar con el Skelterbot.
- Maxi:
    - Contar a artista que tiene que hacer: el puente tiene que tener una meshCollider.
    - Documentación.
    - Meter sonidos.
- Fernando:
    - Quitar que lance un rayo al construir una torre o almacén: con canConstruct vale (RightClickOnSelected).
    - Off-mesh link para saltar el río.
- Guille y Diego:
    - Actualización de recursos de las minas online (si quitas mina al remoto quitarle también).
    - Quitar vida a enemigos.
- Fernando y Maxi:

- Quitar vida a edificios: añadir edificios a la matriz de distancias cuando se construye y conquistan.

- Javi:
  - Documentación.
  - Hacer partículas de las arañas.
- Cuestas:
  - Documentación.

34. Acta del 28 de Mayo de 2014 (ESNE)

Fecha	28 de mayo de 2014
Concepto	Scrum, evaluación sprint anteriores y preparación del siguiente.
Asistentes	Fer, David, L. Suárez, L. Álvarez y Maxi
Duración	1 hora
Conclusiones	Establecemos un sprint para el viernes 16 de mayo

a. Puntos tratados

- Evaluación tareas de anteriores sprints.
- Reparto de tareas de los siguientes sprints.
- Se discute la GUI del RTS, elementos, aspecto, etc.

b. Desarrollo:

Listados de assets.

Tareas realizadas en los sprints previos:

- David:
  - Seguir con las animaciones.
- Jacobo:
  - Ha empezado a diseñar la GUI.
- Fer:
  - Catedral (modelado y mapeado).
- L. Suárez:
  - Continuar con los temas de producción.
- L. Álvarez:
  - Más props: Cofre, tanque, árbol.
- Santiago Rico:
  - Comenzar el desarrollo de la web.

Sprint:

- David:
  - Animaciones.

- Jacobo:
  - *GUI* a saco.
- Fer:
  - Pájaro mutante con 3 ojos (con animaciones).
- L. Suárez:
  - Producción.
- L. Álvarez:
  - Tumbas, criptas, vallas y cosas para el cementerio, texturizar la catedral de Fer.
- Santiago Rico:
  - Páginas web.

e. Glosario de términos

- **Aliasing**: es el efecto visual que se produce al intentar representar una imagen con curvas y líneas inclinadas en una pantalla. Como la resolución es finita, se ve un efecto visual en el que las curvas se muestran dentadas debido a que están compuestas por pixeles.
- **As a service** (*Software as a service*): es un modelo de distribución de software. El soporte lógico y los datos se alojan en servidores a los que se accede por un navegador web a través de internet. La empresa contenedora de todo esto es una compañía de tecnología de información y comunicación (TIC), y ofrece un servicio de mantenimiento y del soporte de software usado por el cliente, así como actividades desde ubicaciones centrales en lugar de la sede de cada cliente, la distribución según el modelo uno-a-muchos (una instancia con múltiples usuarios) y actualizaciones centralizadas, lo cual elimina la necesidad de descargar paquetes por parte de los usuarios finales.
- **Asset**: cualquier archivo del proyecto, ya sean ficheros de material gráfico (como modelos 3D, texturas, materiales, etc.), ficheros de audio, scripts de código fuente o shaders.
- **Atlas**: textura de gran tamaño que se usa para proyectar las texturas que contiene **.**
- **DeltaTime**: tiempo que ha transcurrido desde el último *frame*.
- **Drawcall**: llamada a la función que tiene un objeto que está presente en la escena para que el motor gráfico lo pinte y pueda ser visible para el usuario.
- **Dummy**: marca una posición de un modelo 3D dentro de su jerarquía interna (p.e. un dummy en la mano para marcar la posición donde tiene que ir un martillo, de este modo el martillo seguiría la mano al moverse).
- **FixedDeltaTime**: tiempo (en segundos) que tarda en realizarse la física.
- **FPS** (*frames per second*): imagenes por segundo, es la frecuencia a la que un reproductor gráfico genera distintos fotogramas.
- **Gameplay**: *ver jugabilidad*.
- **GameObject:** clase base de la que heredan todos los elementos de una escena.
- **Gizmo**: se usa para depurar visualmente o como ayudas de configuración en la escena visual.
- **Idle**: estado que representa reposo.
- **Jugabilidad:** *aquello que hace el jugador*. En diseño y análisis de juegos es un término que describe la calidad del juego en términos de sus reglas de funcionamiento y de su diseño como juego. El famoso diseñador de videojuegos Sid Meier definió la *jugabilidad* como *Una serie de decisiones interesantes* (Rollings & Morris, 1999).
- **Layout**: en el ciclo de juego, es el evento que se envía antes que cualquier otra cosa.
- **Malla**: estructura de datos que contiene los datos necesarios para representar un objeto tridimensional como son los vértices, índices, normales y coordenadas de textura. En

videojuegos se suele utilizar este término para referirse a esta representación tridimensional de un objeto.

- **Minijuego:** un juego con una mecánica sencilla.
- **Minions:** personajes menos poderosos que los héroes que se controlan en la parte RTS del videojuego (*unidad de artillería, unidad exploradora, etc.*).
- **Mipmapping**: efecto básico en programación gráfica que se encarga de difuminar las texturas para que no *pixeles*, suavizándola, usando copias en menor resolución de la textura original.
- **MOBA**: (siglas en inglés de *Multiplayer Online Battle Arena*) videojuego de estrategia de acción en tiempo real, es un sub-género de los RTS en el que el jugador no gestiona un ejército sino un único héroe.
- **Prefab (Prefabricado)**[28]: es una copia de un *GameObject* convertido a *Asset* reusable.
- **Prop**: son objetos del juego formados por un conjunto de *Assets* (p.e. un personaje controlable con su modelo 3D y sus scripts asociados).
- **Repaint**: en el ciclo de juego, es el evento que se envía una vez por frame. Primero se envían los otros eventos y después se envía este.
- **Rigidbody:** componente físico de Unity que toma el control sobre la posición del objeto que lo tenga agregado y lo afecta de forma que simula la influencia de gravedad y calcula cómo responde a colisiones.
- **RTS**: (siglas en inglés de *Real-Time Strategy*) videojuego de estrategia en tiempo real en el que el jugador, entre otras cosas, gestiona un ejército en base a una economía.
- **Rush**: ataque rápido y por sorpresa.
- **Shader**: pequeños programas que se ejecutan en la GPU. Para más información consultar el capítulo de *Shaders*.
- **Shooter:** género o jugabilidad que consiste en disparar a otros personajes para poder conseguir un objetivo.
- **Skybox**: textura que envuelve a la escena para recrear el cielo y todo lo que hay en la lejanía.
- **Spawn:** en videojuegos, se utiliza el término *engendrar* para referirse a la aparición de un objeto en el juego en un momento determinado.
- **Sprite**: un tipo de mapa de bits que se dibujan sin generar cálculos adicionales en la CPU.
- **Tower defense:** género basado en defender una base de oleadas de enemigos que pretenden destruirla.
- **Update**: en programación de videojuegos es el método que se encarga de actualizar el estado y atributos de los objetos del juego. Es invocado con una determinada frecuencia (normalmente entre 30 y 60 veces por segundo).

[28] http://docs.unity3d.com/Manual/Prefabs.html

- **WASD:** esquema de control en videojuegos de ordenador, se refiere al uso de las teclas W, A, S, D para mover al avatar del jugador hacia adelante, izquierda, atrás y derecha respectivamente.
- **Yield**: suspende la ejecución de una corrutina para reanudarla en el siguiente frame desde el punto en que se suspendió, manteniendo los valores como estaban.