Introducción

El videojuego está ambientado en el conflicto existente entre dioses y titanes dentro de la mitología griega. Este tipo de juego es denominado tower defense, se basa en la colocación de torres a lo largo de un camino para evitar que las tropas lleguen a la base aliada.

La historia se centra en una supuesta segunda titanomaquia en la que varios titanes intentan llegar al monte olimpo, y Zeus y otros dioses se encargarán de intentar evitarlo.

Mecánicas de juego

-Jugabilidad:

El jugador deberá elegir la torre que quiera y colocarla en el lugar que estime oportuno, siempre y cuando en esa posición no haya ninguna otra torre. Además el jugador podrá mejorar las torres que ya tiene un máximo de 3 veces, aumentando el daño que hacen. Además se podrán vender las torres que ya están colocadas.



-Elementos:

El juego consta de tres tipos de torres y tres tipos de esbirros:

.Esbirros:

Gorgona: es el esbirro básico del juego con vida y velocidad de movimiento equilibradas.

Fauno: es el esbirro más rápido del juego, pero tiene una cantidad de vida considerablemente menor que la que tiene la Gorgona.

Ciclope: es el esbirro tanque del juego tiene una gran cantidad de vida, pero una pésima velocidad de movimiento.

Torres:

Flechas: poseen una buena cadencia de tiro. Son las torres básicas.

Catapulta: son las torres de tiro potente, poseen gran daño al impactar, pero tiene una velocidad de ataque bastante baja.

Hielo: es una torre que tiene un daño bajo, pero ralentiza al impactar.

Creación de la parte de programación

-Creación de los capítulos, súbditos y torres:

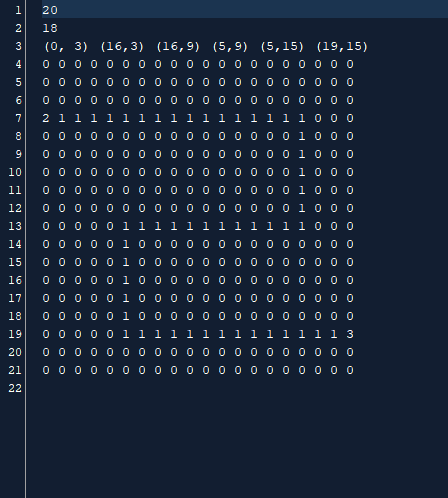
Para crear los capítulos se ha utilizado una herencia simple con varias clases, en estas clases se guarda toda la información que sea exclusiva de un determinado capitulo, como puede ser la vida inicial, el oro inicial y las oleadas de súbditos. Este mismo mecanismo es utilizado para la creación de los súbditos y de las torres.

-Creación de los menús:

Para la implementación de los distintos menús se ha utilizado un sistema basado en estados. Dichos estados se inicializan en la clase main y se va accediendo de uno a otro según se pulse el botón con el que se accede a un menú o a otro.

-Creación del mapa:

Para la implementación de los mapas en los que se juega, y el camino por el que irán los súbditos se realiza mediante la lectura en una clase de un txt guardado en la carpeta mapSaves.

 Como puede observarse en este ejemplo, la primera línea es la que indica cual es el ancho en tiles del mapa a crear. La segunda línea es la que indica cual es la altura en tiles del mapa. Y la tercera línea indica el camino que seguirán los súbditos, capa punto indica las coordenadas x e y de cada punto en el camino en el que hay un giro hacia izquierda o derecha. Además, lo que hace el programa al leer el archivo va uniendo los puntos dados para crear el camino. Toma como referencia el primer punto que será utilizado para hace el inicio del camino y el último punto dado que será el final del camino.

Además, como puede observarse tras esta lista de puntos hay un mapa de números, este mapa se utiliza solo cuando la persona que cree el mapa quiere hacerse una referencia de cómo quedará el mapa. Los 0 son los lugares donde pueden quedarse las torres, los 1 son el camino que recorrerán los súbditos, el 2 es el punto de inicio del respawn de los súbditos y el 3 es el final del recorrido.

-Movimiento de los súbditos:

Este apartado se trata en la clase principal **súbdito** de la que heredan después el resto de súbditos, más concretamente en el método move(). Este método comprueba donde se encuentra el próximo tile al que el súbdito puede ir, y al obtener dicha dirección se cambia el valor dirección del súbdito en cuestión. Además, al llegar al final el súbdito desaparecerá y se restará un punto de vida de la vida del jugador.

-Disparos de las torres:

Esto se controla mediante la comprobación de las coordenadas x e y. Primero se comprueba si un súbdito ha entrado en el rango de la torre, si es así la torre dispara al súbdito, con su animación y sonidos que pertenezcan a la torre. Además, calcula el ángulo de disparo para rotar la torre. Cada torre solo puede disparar a un único súbdito, que será el primero que este a rango, o el que más cerca este en caso de que el súbdito objetivo anterior muera o salga del rango. En ese caso será el que más cerca de la torre esté.

-Colocación de las torres:

Para la colocación de las torres solo se usa una comprobación de coordenadas, cada torre ocupa un único tile, y en cada tile solo puede haber una torre. Primero se comprueba si en el lugar en el que se ha hecho click se puede poner una torre, si es parte del camino que recorren los súbditos no se puede y si hay otra torre en ese tile tampoco puede colocarse tampoco. Además, si el jugador no dispone del oro suficiente para comprar la torre tampoco podrá colocarla.

Problemas encontrados

El principal problema que nos hemos encontrado es a la hora de intentar inicializar el juego en un equipo nuevo, lo que ocurre es que para buscar la carpeta de mapSaves en la clase LoadFiles como hay que poner la ruta completa en la que está dicha carpeta hay problemas cuando se inicializa en otro equipo y para solventarlo el nuevo usuario tiene que cambiar la ruta que esta puesta en la clase loadFiles.

Otro problema que hemos encontrado en la creación de una torre que haga daño en área, era muy complicada la implementación de de ese efecto asi que lo suplimos haciendo una torre que tuviese mucho daño de ataque.