COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

# Proyecto Final:
# Tower Defense


Jorge Moratalla Vita
Marcos Alconchel Fernández


Paradigmas y Técnicas de programación
3ºA Grado en Ingeniería Matemática e Inteligencia Artificial

# ÍNDICE

# Introducción

En este trabajo hemos desarrollado un videojuego del tipo **Tower Defense**, un tipo de juego que combina estrategia, gestión de recursos y acción en tiempo real. Las principales dificultades del proyecto han sido la implementación de las mecánicas del juego, tratando de conseguir funcionamiento adecuado del mismo, como, por ejemplo, la generación aleatoria de los enemigos, los diferentes niveles de dificultad y la integración de diferentes interfaces de juego, que brindan al jugador una gran experiencia de juego. También hemos implementado conceptos avanzados como la programación orientada a objetos, el uso de patrones de diseño y el trabajo colaborativo a través de Git.

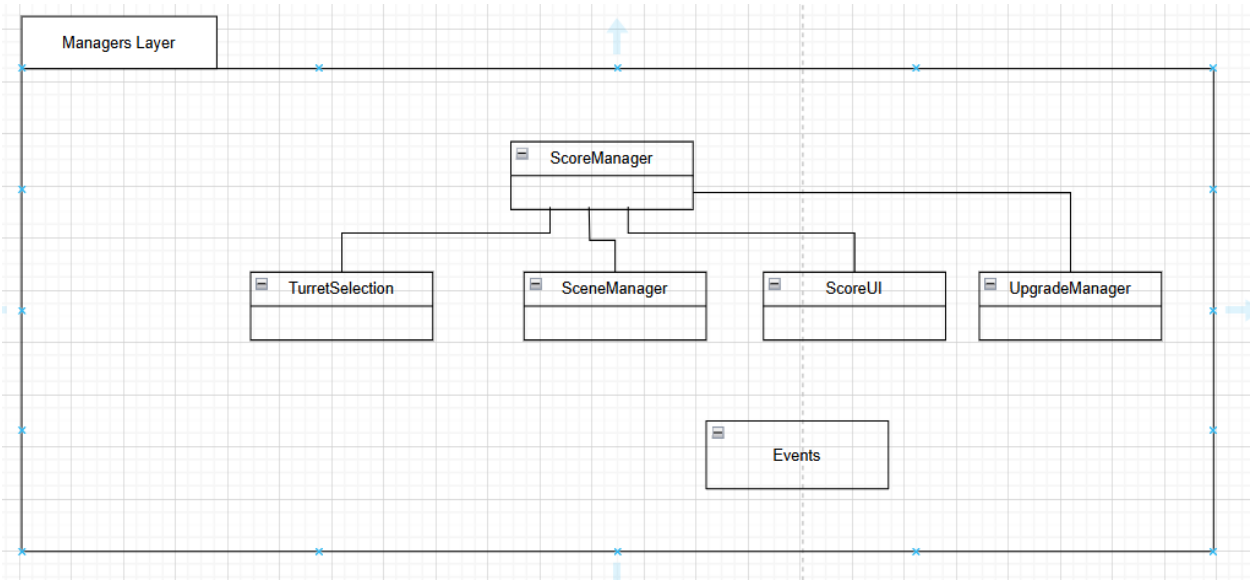
A lo largo del desarrollo se han planteado diferentes funcionalidades, como, por ejemplo, la creación de enemigos de diferentes niveles, variando la vida de estos, o la creación de torretas con la posibilidad de mejorarlas. También se ha implementado el conteo de los puntos dependiendo de si ganas puntos por matar enemigos o si los gastas al poner o mejorar torretas. Además, hemos implementado otros componentes como menús de pausa y las animaciones en los elementos del juego.

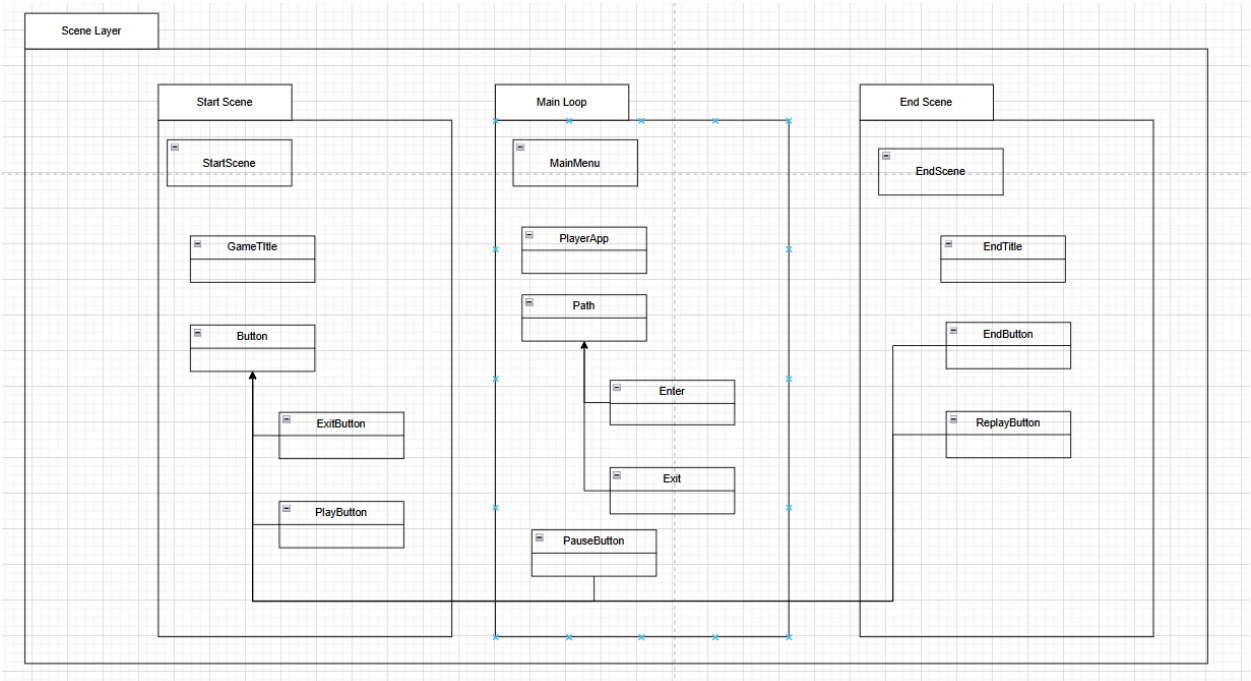
El desarrollo de este proyecto ha implicado una aplicación de conocimientos técnicos en programación y diseño de videojuegos, junto con la capacidad activa para resolver problemas complicados y administrar conflictos en el código. Estas habilidades se complementan y permiten a los profesionales de TI adaptarse a las necesidades cambiantes de la implementación. La memoria en este documento presenta un resumen completo del proceso, desde la concepción inicial hasta la implementación final, documentando las decisiones tomadas, los retos enfrentados y las soluciones adoptadas.

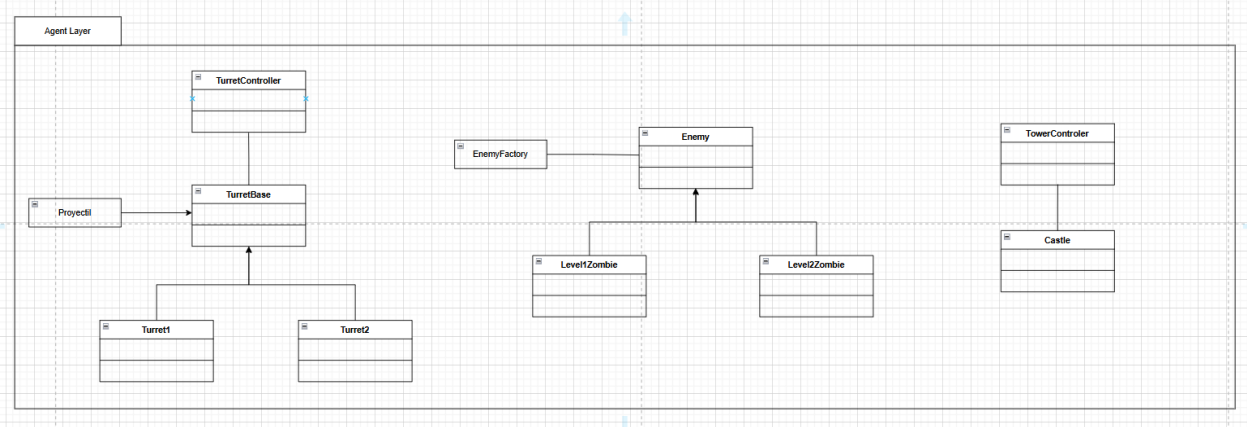
En este documento vamos a mostrar los progresos, como afrontamos los problemas, la evolución y el resultado final obtenido

# Diseño UML

Tras la realización del proyecto, el diagrama UML resultante es el siguiente:



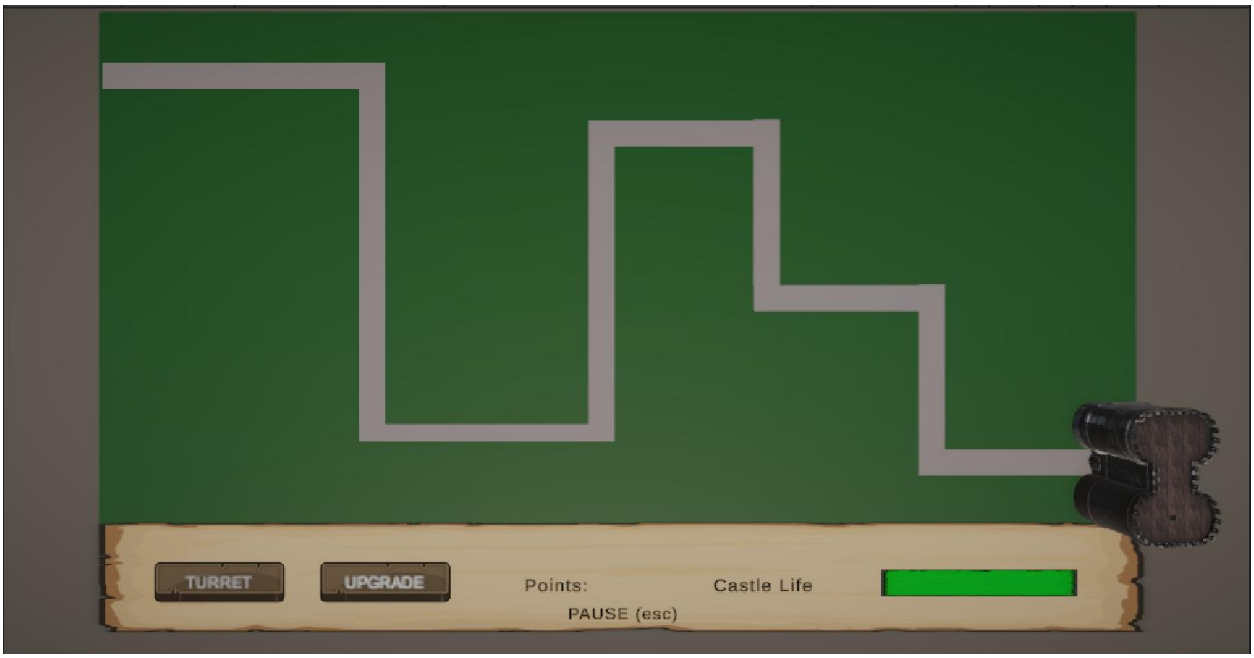






Respecto al UML inicial, resulta muy similar. Se han cambiado sobre todo los nombres de las clases. Además, se han reducido tanto el número de tipos de torretas como el de tipos de enemigos a dos. Esto, principalmente se debe a no complicar demasiado la jugabilidad con muchos tipos de elementos distintos que no suponen ninguna diferencia relevante en el nivel, calidad o aprendizaje de la implementación del videojuego.

# Desarrollo y Funcionalidades implementadas

El juego básicamente consiste en evitar que los enemigos, que siguen un camino determinado, destruyan tu base, un castillo en nuestro caso, atravesando el camino entero sin morir. Para evitarlo se pueden colocar diferentes torretas, en diferentes zonas, las cuales se encargan de proteger la base eliminando a los enemigos. La distribución del mapa es la siguiente:




A continuación, se van a detallar las principales funcionalidades implementadas:

- Generación de enemigos.

  En esta parte se utiliza un sistema dinámico que se encarga de instanciar copias de los enemigos a partir de un prefab base en puntos específicos del mapa. A medida que se van generando los enemigos, se van generando de manera un poco más rápida y cuando han pasado una cantidad determinada de enemigos de nivel 1, se aumenta el nivel y empiezan a salir enemigos de nivel 2, que están mejorados teniendo más vida y siendo más difícil matarlos.

- Interfaz de usuario UI.

  Se incluye una barra de salud del castillo, la cual va a determinar, en el caso de que se vacíe, si pierdes o no la partida, un marcador de puntos además de los botones para colocar y mejorar torretas. También se implementa un menú de pausa durante la partida pulsando la tecla de escape.

- Movimiento de los enemigos.

  Para el movimiento de los enemigos, hemos indicado varios puntos a los cuales se dirigen, de uno en uno y van cambiando a medida que van avanzando hasta llegar al final, el castillo, donde se destruyen quitando vida a la base.

- Colocación y mejora de torretas.

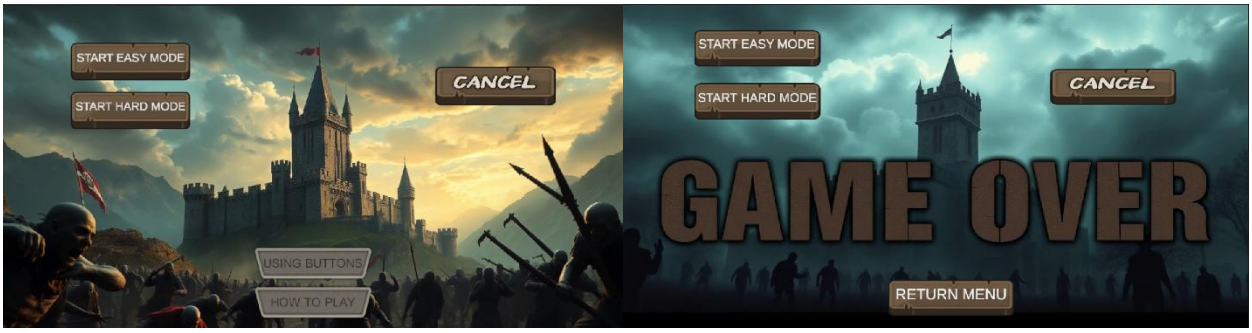
  Para la colocación y la mejora de las torretas se implementan dos botones que se encargan de detectar el lugar en el que se está pulsando y comprueba si el lugar es correcto para colocar una torreta, no está encima del camino, y comprueba si se ha hecho clic encima de una torreta para una posible mejora.

- Sonidos y Animaciones.

  El juego incluye también una música con el fin de ambientar y mejorar la experiencia del jugador, además de incluir algunas animaciones por parte de los elementos del juego como la manera de andar de los enemigos, además de los disparos de la torreta.

- Escenas

  Además de lo anterior, implementamos varias escenas para conseguir una jugabilidad más intuitiva y al gusto del usuario, introducimos una pantalla de inicio en la que se explican las reglas básicas del juego además de la manera de jugar, y una escena de final que permite al jugador jugar una nueva partida o salir del juego.




- Generación de Escena y Rutas de Enemigos

  Los enemigos se desplazan por rutas predefinidas que los llevan hacia el castillo. Estas rutas se establecen mediante puntos de camino (waypoints) distribuidos a lo largo del mapa. Con la aparición de más enemigos, la dificultad se incrementa, lo que obliga al jugador a tomar decisiones más estratégicas sobre la ubicación de sus torres.

- Interacciones

Se implementó un sistema de rayos para permitir al jugador seleccionar las torres con un clic del ratón. Cuando el jugador hace clic en una torre, se selecciona y se permite realizar mejoras, también este sistema nos permite saber exactamente el lugar donde se colocan las torretas, ya que detecta la posición del clic y eso permite poder realizar las comprobaciones antes de colocar la torreta en esa posición.

# Dificultades

Una vez explicado cómo funciona exactamente y los elementos implementados, explicados resumidamente, en el apartado anterior. Nos vamos a centrar, ahora, en la parte de los conflictos y dificultades que hemos tenido en ciertos momentos para implementar algunos elementos y algunos que pensamos implementar, aunque no fuimos capaces de hacerlo finalmente.

Primero, nuestra primera idea era introducir elementos de decoración del mapa, algo que no fuimos capaces de implementar ya que intentamos con algunas herramientas crear algún tipo de shader para poder agregar al mapa base, con el fin de dar una apariencia más elegante e inmersiva, pero no fuimos capaces por lo que abandonamos eso ya que pensamos que sería más importante implementar correctamente las diferentes funcionalidades del juego que la parte visual.

Otro de los problemas que más tiempo nos llevó solucionar, que también tuvimos al principio fue la manera en la que debíamos hacer el respawn de los enemigos, ya que en primera instancia, lo que hicimos fue, a partir del prefab de enemigo que teníamos creado hacer copias y agregarles el camino que debían de seguir, algo que funcionó de primeras, pero que al introducir la mecánica de matar a los enemigos con las torretas, estábamos destruyendo el enemigo sobre el cual se estaba realizando las copias. Después nos dimos cuenta de que clonar enemigos, no era una buena idea y decidimos implementar una factoría, la cual recibe como argumentos los dos niveles de enemigos creados y realiza la lógica para instanciar uno u otro desentendiendo de diferentes parámetros.

Otro problema que nos surgió, esta vez mas enfocado en la parte de las torretas era a la hora de disparar. Para disparar la torreta, creamos unos colliders en las torretas activando is trigger ya que queríamos que detectara a los enemigos cuando entraba en el rango de disparo sin bloquear el paso. Entonces, al crear eso y el collider de la bala, nos daba un problema ya que la bala se destruía antes de salir de la torreta debido a que chocaba con el cañón, algo que nos costó darnos cuenta ya que no éramos capaces de entender cuál estaba siendo el problema, finalmente, lo arreglamos añadiendo un tag con el fin de que si colisionaba con un elemento con el tag “Torreta” que no impactara y siguiera adelante.

Luego, decidimos implementar también que la torreta disparara al enemigo más cercano, ya que al principio simplemente definimos que disparara al enemigo que entrara en el rango de disparo

y cambiaba muy rápido de objetivo y no era óptimo ya que de esa manera costaba mucho matar a los enemigos y se perdía muy rápido.

También, tuvimos varias dificultades a la hora de colocar las torretas en el mapa, ya que la posición de la cámara era bastante importante para detectar exactamente la posición que se estaba seleccionando con el ratón, y estuvimos probando diferentes maneras de colocarlas hasta que nos dimos cuenta de que, al colocar la cámara justamente perpendicular al mapa, podíamos hacerlo de la manera deseada, ya que al colocarla encima sí que se detectaba bien los puntos seleccionados.

Por último, otro contratiempo que tuvimos, el cual nos llevó bastante tiempo ajustar correctamente, fue definir los parámetros a la hora de jugar, es decir, todo lo que tiene que ver con los enemigos y las torretas, la vida y la velocidad movimiento de los enemigos y la velocidad de disparo y el daño de las torretas. Definir estos parámetros, es otra parte fundamental del juego, ya que de seleccionar unos parámetros incorrectos se puede hacer el juego o muy fácil o difícil y la idea es hacerlo de manera que se pueda aguantar tiempo, aunque en algún momento se pueda perder y tras diferentes pruebas conseguimos elegir los mejores parámetros.

# Enlace del repositorio

Enlace al repositorio de GitHub:

https://github.com/Markk40/TowerDefense.git