

Universidad Argentina De la Empresa

Maximo Samperio

25 de septiembre de 2023

Programación de Sistemas

Análisis de patrones de diseño - Protect de Grove

# Introducción

Protect The Grove es un videojuego del género tower defense 3D. Esta cuenta actualmente con un solo nivel y es el objetivo de este documento realizar un análisis de dicho nivel, su composición en términos de código y ver como se incorporan diferentes patrones de diseño al videojuego. Para lograr esto, iré enumerando diferentes patrones y como estos están presentes en la estructura del juego.

## Flyweight

El patrón conocido como “Flyweight” busca, según mi entendimiento, minimizar el uso de la memoria usando variables constantes o en común para diferentes instancias de un mismo objeto o varios objetos con variables en común.

Yo aplique este patrón principalmente mediante el uso de un Scriptable Object llamado “EnemyType”, el cual se encarga de manejar cualquier tipo de enemigo que se quiera agregar al juego.

A screenshot of a computer program

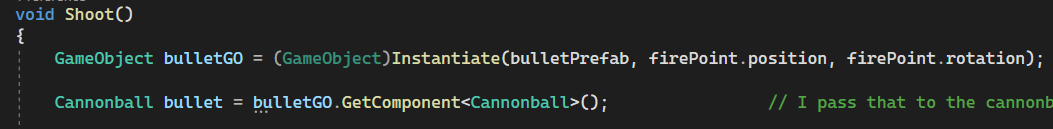
Description automatically generated

“EnemyType” almacena los diferentes datos que serán constantes en todos los enemigos e independientemente de su instanciación, pero que si varía según el tipo de enemigo. Estos datos son: la velocidad de movimiento, la vida máxima (la vida con la que comienza), el valor en términos de dinero que le otorga al jugador al morir y su efecto de muerte (cada tipo de enemigo tiene uno diferente).

El script que utiliza estos datos es “EnemyMovement”, el cual quería que se encargue de la parte lógica como el movimiento de los enemigos y de seguir el camino preasignado, las funciones de recibir daño, entre otros.

## Prototype

Prototype trata sobre la clonación y la instanciación de objetos. Esto esta presente en el juego con la ayuda del método “instantiate” de Unity. Esto está presente de varias formas diferentes en mi proyecto. Una de ellas es la instanciación de balas cada vez que la torreta va a disparar, donde se accede al GameObject de la bala, se la “castea” a un GO (para luego destruirla) y se la instancia en el “firepoint” (punta del cañon) con dirección al enemigo más cercano:



Otra forma en la que implemente el método prototype es claramente la generación de enemigos, donde cada vez que se matan a todos los enemigos presentes se instancian más.

A black screen with white text

Description automatically generated

A screen shot of a computer

Description automatically generatedPor último, también utilice este método para la instanciación de efectos de muerte cada vez que muere un enemigo. Si bien esta variable es almacenada por el Scriptable Object “Enemy Type”, esta es utilizada en el script “EnemyMovement” cuando muere un enemigo. También se realiza este proceso cuando se crea una torreta

## Façade + Singleton

El patrón de Façade, según mi entendimiento, busca crear una interfaz fácil de utilizar que actúe como un manager que centraliza diferentes funcionalidades del juego. Esto lo aplique de diferentes formas. En primer lugar, tengo un Game manager propiamente dicho que se encarga de manejar las condiciones de victoria y derrota, pero más importantemente, tengo un “BuildManager” script que hace uso de un singleton que en este caso es una instancia del susodicho BuildManager.

Este BuildManager accede a datos de diferentes scripts como “PlayerStats”, que almacena las vidas y el dinero actuales, accede a los diferentes valores de las torretas (actualmente estoy trabajando en una segunda torreta), accede a los diferentes nodos del mapa (los cuadrados donde se podrían posicionar las torretas) y centraliza todos estos datos para chequear si: el jugador tiene el dinero suficiente para comprar la torreta, si el cursor esta posicionado sobre un nodo valido y si no existe ya una torreta sobre ese nodo y si todas estas condiciones se cumplen, habilita la construcción de una torreta sobre el nodo clickeado.