Github link: https://github.com/Ibinarriaga8/Proyecto-Final-Paradigmas

Historias de Usuario

Como jugador (Player), quiero conducir un Sedan para huir de la policía y mantenerme fuera de su alcance

- El jugador controla un vehículo tipo Sedan y debe escapar de la policía. El objetivo es evadir el Police Car mientras conduce por la calle.

Como policía (Police), quiero conducir un Police Car para perseguir y capturar al jugador que está huyendo

- La policía, controlada por un agente de inteligencia artificial (AI Agent), persigue al jugador utilizando el Police Car como un obstáculo móvil.

Como jugador, quiero recoger elementos de mejora (Buf**) en el camino para aumentar mi vida y mejorar mis posibilidades de escape**

- Los Bufs son obstáculos que, al ser recogidos, aumentan la vida del jugador, ayudándole a resistir más tiempo durante la persecución.

Como jugador, quiero evitar los elementos de debilitación (Debuf**) que reducen mi velocidad y dificultan mi huida de la policía**

- Los Debuf son obstáculos que reducen temporalmente la velocidad del jugador, dificultando su huida y dándole una ventaja al Police Car.

Como policía (AI Agent), quiero poder esquivar obstáculos y civiles mientras persigo al jugador

- La policía, como un AI Agent, debe evitar los civiles y otros obstáculos (como Bufs y Debufs) para no ralentizar su persecución.

Como civil (Civilian), quiero interactuar con el entorno, manteniendo mis propias rutas en la calle

- Los Civilians en la calle son controlados por agentes de inteligencia artificial y se mueven de forma autónoma o permanecen en posiciones específicas, sin perseguir al jugador ni a la policía.

Patrones de diseño empleados en el diagrama UML:

1. Flyweight en Obstacle

Obstacle usa Flyweight para reducir el uso de memoria compartiendo datos comunes (como propiedades visuales o efectos) entre instancias de obstáculos similares (PoliceFence, Buf, Debuf). Esto permite tener múltiples obstáculos en el mapa sin duplicar datos innecesarios.

2. Factory Method en Clases Factory

Las clases PoliceCarFactory, SedanFactory, PoliceFactory, y CivilianFactory implementan el Factory Method para crear instancias específicas de vehículos y personas. Cada fábrica encapsula la lógica de creación, permitiendo instanciar estos objetos sin especificar la clase concreta en el código cliente.

3. Bridge en Controllers

Los Controllers (PlayerController, PoliceController, CivilianController) actúan como puentes, separando la lógica de control de cada entidad (Player, Police, Civilian) de su implementación. Esto permite modificar la lógica de control sin afectar a las clases principales de cada personaje o vehículo.