Proyecto: Air War.

Integrantes: Jiacheng Tan He y Justin Solano.

Curso: Algoritmos y Estructuras de Datos I (CE 1103).

Profesor: Leonardo Araya.

25/11/2024.

Tabla de contenido

Introducción.

El proyecto Air War consiste en el desarrollo de un videojuego que combina elementos estratégicos y técnicos, implementado en C# mediante el uso de estructuras de datos avanzadas y algoritmos eficientes. Este proyecto tiene como principal objetivo consolidar los conocimientos adquiridos en Programación Orientada a Objetos (POO), mediante la resolución de un problema práctico que simula un escenario jugable de guerra aérea.

En el juego, el jugador debe operar una batería antiaérea para destruir aviones enemigos, mientras interactúa con un entorno dinámico que incluye aeropuertos, portaaviones y rutas aéreas generadas aleatoriamente. Además, el uso de estructuras como grafos para modelar rutas, junto con técnicas de ordenamiento como Merge Sort y Selection Sort, refuerza la comprensión de conceptos clave en algoritmos.

A través de este proyecto, se busca desarrollar habilidades en el diseño de sistemas orientados a objetos, el manejo de estructuras de datos, y la resolución creativa de problemas complejos en ingeniería informática. Al finalizar este, se trabajaron las capacidades de los estudiantes para integrar soluciones técnicas con una interfaz gráfica funcional y documentar todo el proceso de desarrollo de manera profesional.

Diseño.

Listado de requerimientos (Formato user stories).

ID	User Storie	Criterios de Aceptación
000	Como desarrollador, quiero implementar el juego en C# utilizando WPF, MAUI o Unity para garantizar una experiencia gráfica amigable y eficiente.	• El juego debe desarrollarse completamente en la plataforma seleccionada. • Todas las funcionalidades deben estar integradas correctamente en la interfaz.
001	Como jugador, quiero destruir la mayor cantidad de aviones en un tiempo limitado para maximizar mi puntuación.	• Se debe implementar un temporizador que limite la duración de la partida. • Debe contabilizarse el número de aviones derribados para la puntuación.
002	Como sistema, quiero generar aleatoriamente aeropuertos, portaaviones y rutas para garantizar diversidad en cada partida.	• Aeropuertos y portaaviones deben ubicarse aleatoriamente en el mapa. • Las rutas deben modelarse como un grafo usando listas de adyacencia.
003	Como jugador, quiero controlar una batería antiaérea que se mueva de forma lineal entre izquierda y derecha y dispare balas para destruir aviones enemigos.	• La batería debe moverse con velocidad constante. • Las balas deben dispararse con una trayectoria recta y velocidad variable según el tiempo de clic.
004	Como avión, quiero calcular la mejor ruta hacia mi destino considerando los pesos de las rutas para optimizar mi trayecto.	• Los pesos deben considerar distancia, tipo de destino (aeropuerto o portaaviones) y si la ruta es interoceánica o continental. • El avión debe seleccionar la ruta más eficiente en cada despegue.
005	Como avión, quiero recargar combustible al aterrizar y evitar quedarme sin combustible durante el vuelo.	• El tiempo de recarga y la cantidad de combustible deben ser aleatorios. • Los aeropuertos deben implementar una lógica para racionar el combustible.
006	Como aeropuerto, quiero construir aviones respetando la capacidad de mis hangares para mantenerme dentro de los límites permitidos.	• Cada aeropuerto debe tener un límite de aviones basado en su capacidad. • Los aviones deben tener IDs únicos generados mediante GUIDs.
007	Como avión, quiero funcionar de manera autónoma gracias a módulos de inteligencia artificial que trabajen en conjunto para manejar diferentes tareas.	• Cada avión debe contar con módulos para Pilot, Copilot, Maintenance y Space Awareness. • Cada módulo debe tener atributos como ID, rol y horas de vuelo.

008	Como jugador, quiero ver una lista de aviones derribados y que estén ordenados por su identificador para entender mi progreso.	• La lista debe generarse usando Merge Sort. • Los aviones deben ordenarse por su ID en orden ascendente.

Problemas y soluciones.

Problema 1: Detección de colisiones entre balas y aviones

Requisito relacionado:

- **Requerimiento 003 del proyecto:**
  "El jugador presiona click para disparar balas con trayectoria recta y debe tratar de destruir los aviones en ruta."

Alternativas de solución:

1. **Implementación actual: Iterar sobre nodos y aviones**
   - Revisar las colisiones entre todas las balas y aviones iterando a través de los nodos del grafo y sus listas de aviones.
   - **Ventajas:**
     - Es fácil de implementar y entender.
     - Se ajusta a un número moderado de aviones y balas.
   - **Desventajas:**
     - La complejidad es alta (O(n×m)) en escenarios con muchos aviones y balas.
2. **Optimización con estructuras espaciales (Quadtrees o grillas):**
   - Dividir el espacio en celdas para reducir el número de comparaciones necesarias.
   - **Ventajas:**
     - Mayor rendimiento en escenarios con muchos elementos en pantalla.
   - **Desventajas:**
     - Requiere mayor memoria y complejidad en la implementación.

Selección:

Se selecciona la **opción 1 (implementación actual)** porque cumple con los requisitos del proyecto y la escala del juego no requiere una optimización avanzada.

Problema 2: Consumo de combustible durante el vuelo

Requisito relacionado:

- **Requerimiento 006:**
  "Un avión puede caerse si se le acaba el combustible antes de aterrizar."

Alternativas de solución:

1. **Implementación actual: Consumo fijo por tick del temporizador**
    - Reducir el combustible en cada tick de manera constante, simulando el consumo durante el vuelo.
    - **Ventajas:**
        - Fácil de implementar y entender.
        - Garantiza que el consumo de combustible ocurra de manera uniforme.
    - **Desventajas:**
        - No considera la distancia recorrida, lo que podría ser menos realista.
2. **Consumo basado en la distancia recorrida:**
    - Calcular el combustible según la distancia que el avión recorre en cada tick.
    - **Ventajas:**
        - Más realista al vincular consumo con movimiento real.
    - **Desventajas:**
        - Mayor complejidad computacional.

Selección:

Se selecciona la **opción 1 (implementación actual)** porque cumple el requisito de hacer que los aviones se caigan por falta de combustible de forma sencilla.

Problema 3: Generación de rutas con pesos

Requisito relacionado:

- **Requerimiento 004:**
  "El peso de una ruta considerará la distancia y el tipo de destino (aeropuerto o portaaviones)."

Alternativas de solución:

1. **Implementación actual: Pesos aleatorios entre nodos**
    - Generar rutas con pesos asignados aleatoriamente para simular diferentes costos de viaje.
    - **Ventajas:**
        - Fácil de implementar.
        - Permite crear rutas sin depender de cálculos complejos.
    - **Desventajas:**
        - No refleja costos específicos como rutas interoceánicas o tipos de destino.
2. **Pesos calculados basados en heurísticas:**
    - Asignar pesos según una fórmula que combine distancia y tipo de ruta.
    - **Ventajas:**
        - Representa mejor los costos reales de las rutas.
    - **Desventajas:**
        - Requiere cálculos adicionales y ajustes para balancear el juego.

Selección:

Se selecciona la **opción 1 (implementación actual)** porque genera rutas que cumplen con la funcionalidad básica requerida para el grafo.

Problema 4: Movimiento de aviones en tránsito

Requisito relacionado:

- **Requerimiento 005:**
  "Cuando un avión aterriza, espera una cantidad aleatoria de segundos antes de retomar una nueva ruta."

Alternativas de solución:

1. **Implementación actual: Movimiento por pasos constantes**
   - Dividir la distancia entre el origen y el destino en pasos uniformes para mover al avión.
   - **Ventajas:**
     - Sencillo de implementar y controlar.
     - Asegura que los aviones lleguen al destino en un tiempo razonable.
   - **Desventajas:**
     - El movimiento puede parecer menos natural para distancias largas o cortas.
2. **Interpolación basada en tiempo o velocidad:**
   - Calcular la velocidad o el tiempo total necesario y ajustar el movimiento dinámicamente.
   - **Ventajas:**
     - Movimiento más fluido y realista.
   - **Desventajas:**
     - Mayor complejidad de implementación.

Selección:

Se selecciona la **opción 1 (implementación actual)** porque asegura un movimiento funcional y sencillo de manejar.

Problema 5: Reinicio del juego

Requisito relacionado:

- **Requerimiento general:**
  "El juego debe permitir reiniciar la partida manteniendo el grafo generado."

Alternativas de solución:

1. **Implementación actual: Restaurar atributos básicos (combustible, aviones, etc.)**
   - Vaciar listas de balas y aviones, y recargar el combustible a los valores iniciales.
   - **Ventajas:**

        - Cumple con el requisito de mantener el grafo mientras reinicia el estado del juego.
        - Fácil de implementar y controlar.
    - **Desventajas:**
        - No permite recuperar un estado inicial exacto si se modificaron atributos dinámicos.
2. **Guardar y restaurar el estado inicial del grafo:**
    - Tomar una copia inicial del grafo al generarlo y restaurarlo al reiniciar.
    - **Ventajas:**
        - Permite restaurar estados exactos.
    - **Desventajas:**
        - Requiere mayor uso de memoria y lógica adicional.

Selección:

Se selecciona la **opción 1 (implementación actual)** porque cumple con los requisitos del proyecto y permite reiniciar el juego sin complejidad adicional.

Diagrama de Clases.

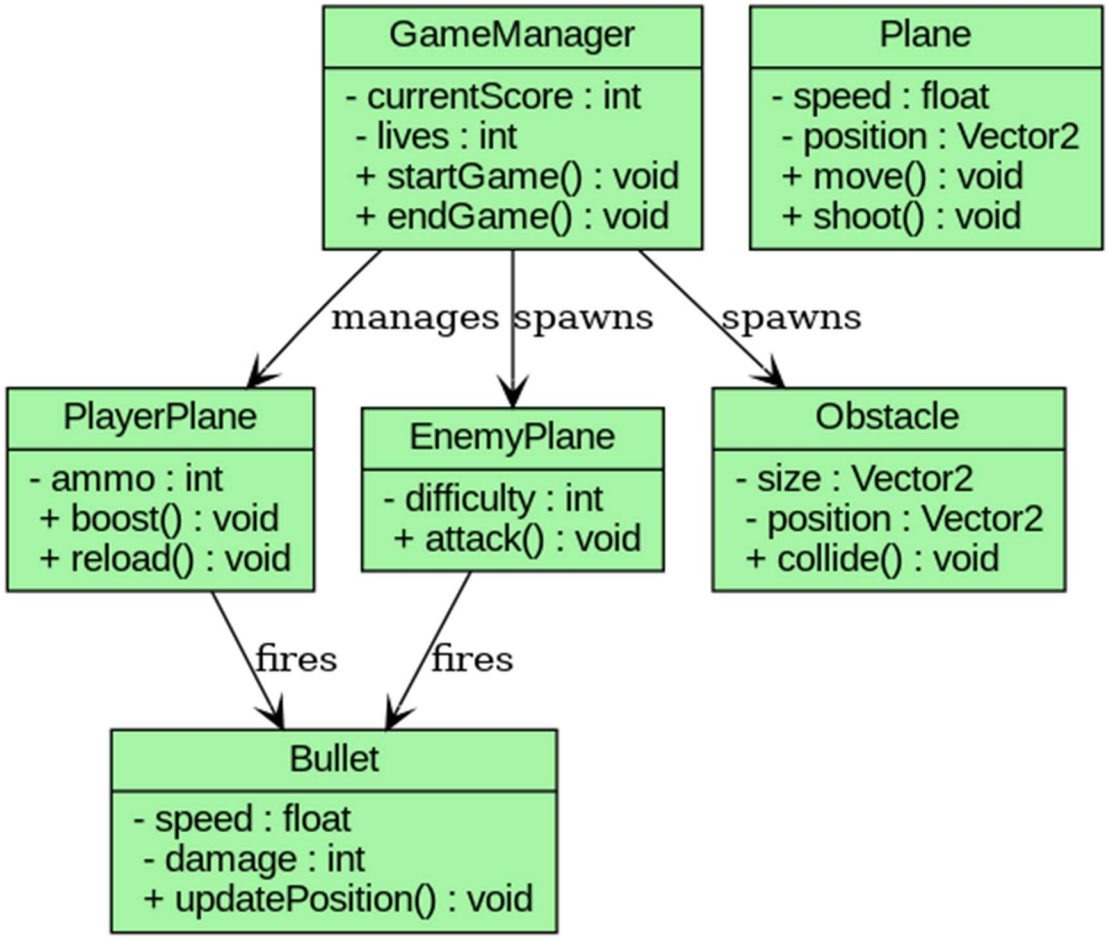
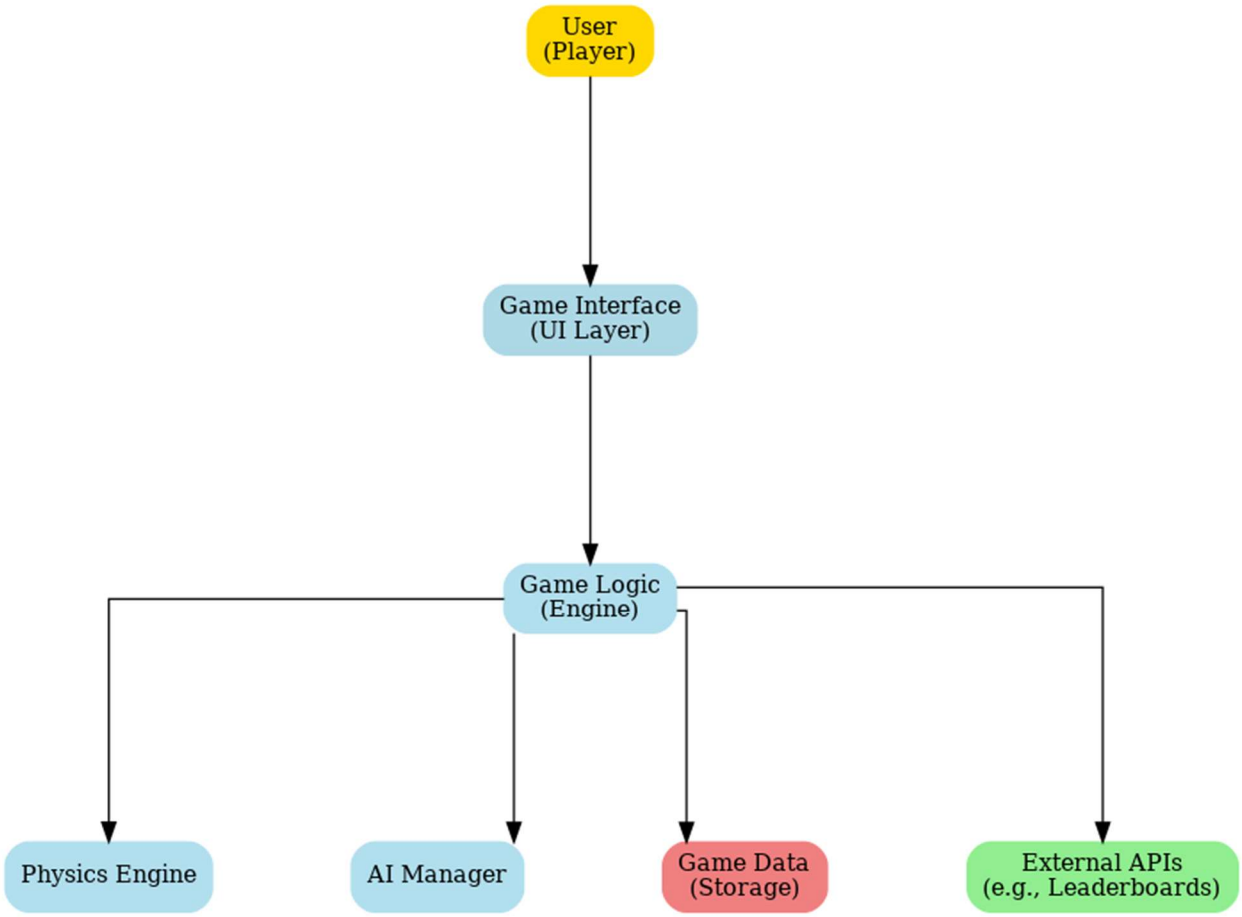


Diagrama de arquitectura.



Check list de las user story implementados.

ID	User Storie	Check
000	Como desarrollador, quiero implementar el juego en C# utilizando WPF, MAUI o Unity para garantizar una experiencia gráfica amigable y eficiente.	√
001	Como jugador, quiero destruir la mayor cantidad de aviones en un tiempo limitado para maximizar mi puntuación.	√
002	Como sistema, quiero generar aleatoriamente aeropuertos, portaaviones y rutas para garantizar diversidad en cada partida.	√
003	Como jugador, quiero controlar una batería antiaérea que se mueva de forma lineal entre izquierda y derecha y dispare balas para destruir aviones enemigos.	√
004	Como avión, quiero calcular la mejor ruta hacia mi destino considerando los pesos de las rutas para optimizar mi trayecto.	☐
005	Como avión, quiero recargar combustible al aterrizar y evitar quedarme sin combustible durante el vuelo.	√
006	Como aeropuerto, quiero construir aviones respetando la capacidad de mis hangares para mantenerme dentro de los límites permitidos.	√
007	Como avión, quiero funcionar de manera autónoma gracias a módulos de inteligencia artificial que trabajen en conjunto para manejar diferentes tareas.	☐ (Parcialmente implementada)
008	Como jugador, quiero ver una lista de aviones derribados y que estén ordenados por su identificador para entender mi progreso.	√

Conclusiones y más.

Aprendizajes Técnico.

Este proyecto ha permitido fortalecer aún más el uso de estructuras de datos como grafos e implementar algoritmos que son efectivos para ciertos problemas, como el cálculo de rutas y la identificación de datos.

Fortalecimiento de las habilidades de diseño.

La implementación de la POO en un entorno funcional ha fortalecido la capacidad de crear soluciones modulares, escalables y reutilizables que son fundamentales para el desarrollo de software.

Configuración de piezas integrales.

La interfaz gráfica, que funciona en conjunto con Game Logic, demostró la importancia de integrar diferentes capas de software, desde la lógica hasta la presentación.

Trabajo en Equipo.

Trabajar en parejas y usar herramientas como Git/GitHub mejora las habilidades de colaboración y gestión de proyectos.

Sugerencias generales para futuros proyectos:

Aún con los objetivos abarcados, algunos puntos adicionales al proyecto pueden incluir:

- Optimizar la gestión de combustible y recursos en los aeropuertos.
- Mejoras en la inteligencia de los aviones autónomos.
- Extensiones como la de Arduino promueven una experiencia más inmersiva.

Esto con el fin de mejorar la calidad del trabajo realizado.

En general, AirWar fue un desafío integral que consideró no solo habilidades técnicas, sino también habilidades de pensamiento crítico y resolución creativa de problemas.