TEC | Tecnológico de Costa Rica

Tercer Proyecto

Airwar

Escuela de Ingeniería en Computadores

Algoritmos y Estructuras de Datos I

Profesor:

Leonardo Araya

Estudiantes:

Dylan Guerrero González | 2022016016
Paula Melina Porras Mora | 2023082886

25 de noviembre del 2024

Tabla de contenidos.

Introducción.

El presente documento detalla el desarrollo y diseño de AirWar, el cual es un juego de guerra aérea diseñado para aplicar conceptos de Programación Orientada a Objetos (POO) en C#. En este juego, el jugador controla una batería antiaérea con la cual debe derribar aviones en rutas aéreas generadas de forma aleatoria.

AirWar permite experimentar con estructuras de datos complejas y su aplicación en la resolución de problemas, por ejemplo, uno de los enfoques centrales es el uso de grafos para modelar los aeropuertos, portaaviones y las rutas que los conectan, a través de esta representación en grafo, el sistema simula la complejidad de un entorno de tráfico aéreo, donde cada ruta tiene un peso que varía según el tipo de destino (aeropuerto o portaaviones) y si la ruta es continental o interoceánica.

Por lo cual el uso de grafos en AirWar es fundamental, ya que permite organizar y optimizar la conectividad entre puntos de origen y destino en el mapa de juego. Estos representan una estructura natural para modelar redes de transporte y son una herramienta poderosa para encontrar rutas óptimas, en este contexto de AirWar, los aviones deben calcular la mejor ruta hacia su destino, tomando en cuenta la distancia y el tipo de conexión, lo que simula un problema real de gestión de rutas en sistemas de transporte. Este diseño no solo añade complejidad estratégica al juego, sino que también ofrece un entorno en el que el jugador experimenta la importancia de los costos y decisiones de ruta, ajustados dinámicamente a las condiciones del entorno.

Por lo cual, el grafo en AirWar está implementado mediante listas de adyacencia, una estructura que permite gestionar eficientemente las conexiones entre los diferentes nodos (aeropuertos y portaaviones) y soportar la búsqueda de rutas optimizadas mediante algoritmos de búsqueda, los cuales son esenciales para el movimiento autónomo de los aviones.

Diseño:

Historias de usuario.

- **Como jugador**, quiero controlar una batería antiaérea que se mueva de izquierda a derecha a velocidad constante, para intentar derribar la mayor cantidad de aviones posible en un tiempo limitado.
- **Como jugador**, quiero que el tiempo de presión del clic determine la velocidad de la bala lanzada, para tener control sobre la precisión y efectividad de cada disparo.
- **Como usuario del sistema**, quiero que los aeropuertos y portaaviones se generen en posiciones aleatorias al inicio del juego, para que cada partida tenga un mapa diferente y desafiante.
- **Como jugador**, quiero que las rutas entre aeropuertos y portaaviones tengan un peso basado en la distancia y el tipo de destino (aeropuerto, portaaviones, ruta continental o interoceánica), para reflejar los distintos niveles de dificultad al planear rutas de vuelo.
- **Como avión autónomo**, quiero seleccionar un destino aleatorio y calcular la ruta óptima, teniendo en cuenta el peso de cada ruta para asegurar una navegación eficiente.
- **Como avión autónomo**, quiero que mi tiempo de espera en el destino sea aleatorio y recargar combustible durante este tiempo, para poder continuar mis vuelos de manera realista.
- **Como aeropuerto**, quiero gestionar mi suministro de combustible y distribuirlo de forma adecuada entre los aviones, de modo que pueda satisfacer sus necesidades según mi capacidad.
- **Como jugador**, quiero ver cómo un avión puede caer si se le acaba el combustible antes de llegar a su destino, para reflejar las consecuencias de una mala gestión de recursos en el juego.
- **Como aeropuerto**, quiero construir nuevos aviones cada cierto tiempo y limitar la cantidad de aviones activos según la capacidad de mi hangar, para mantener el control de mi flota.
- **Como aeropuerto**, quiero asignar a cada avión un ID único generado con GUIDs, para poder identificar y gestionar de forma precisa cada avión en mi flota.
- **Como avión autónomo**, quiero estar equipado con cuatro módulos de inteligencia artificial (Pilot, Copilot, Maintenance, Space Awareness) que trabajen en conjunto para asegurar mi vuelo y respuesta ante problemas.

- **Como jugador o desarrollador**, quiero poder obtener una lista de todos los aviones derribados y ordenarlos por su ID usando el algoritmo de Merge Sort, para visualizar de forma organizada la información de las bajas.
- **Como usuario del sistema**, quiero poder ver la tripulación de cada avión derribado y ordenarla por ID, rol o horas de vuelo, utilizando Selection Sort, para analizar la información relevante de cada módulo de AI.
- **Como jugador**, quiero ver todos los datos relevantes del juego, incluyendo rutas calculadas, pesos de cada ruta y atributos de los aviones, para tener un contexto claro de la situación en el mapa.

Problemas y soluciones.

Problema 1: Generar rutas con pesos basados en distancia y tipo de destino

Alternativa 1: Asignar pesos manualmente en una tabla de configuración.

- **Ventajas:**
    1. Control total sobre los valores de peso.
    2. Fácil de ajustar si se necesitan cambios rápidos.
- **Desventajas:**
    1. Escalabilidad limitada si se agregan más rutas o tipos de destinos.
    2. No considera cambios dinámicos en las distancias durante el juego.

Alternativa 2: Calcular los pesos en tiempo real según una fórmula basada en la distancia y tipo de destino.

- **Ventajas:**
    1. Más adaptable y dinámico; no requiere configuraciones manuales.
    2. Asegura coherencia entre todas las rutas, incluso en mapas generados aleatoriamente.
- **Desventajas:**
    1. Mayor complejidad de implementación.
    2. Puede consumir más recursos si el cálculo es frecuente.

Selección:
Se selecciona la **Alternativa 2** porque es más escalable y adecuada para un juego dinámico donde las rutas y destinos cambian con frecuencia.

Problema 2: Calcular la ruta óptima para un avión autónomo

Alternativa 1: Implementar el algoritmo de Dijkstra para rutas más cortas.

- **Ventajas:**
    1. Alta eficiencia para calcular rutas con pesos positivos.
    2. Es un algoritmo bien documentado y probado.
- **Desventajas:**
    1. Requiere preprocesar las conexiones entre nodos en listas de adyacencia.
    2. No es ideal si se esperan cambios dinámicos en las rutas.

Alternativa 2: Usar el algoritmo A* para encontrar rutas óptimas.

- **Ventajas:**
    1. Integra una heurística, lo que lo hace más eficiente para mapas grandes.
    2. Se adapta mejor si el mapa cambia dinámicamente.
- **Desventajas:**
    1. Más complejo de implementar correctamente.
    2. La calidad de la heurística puede afectar el rendimiento.

Selección:

Se eligió la **Alternativa 1** por su simplicidad, eficiencia, control sobre la experiencia del jugador y adecuación al problema al ofrecer una implementación directa y ajustable.

Problema 3: Derribar un avión cuando se le acaba el combustible

Alternativa 1: Calcular el consumo de combustible por unidad de tiempo.

- **Ventajas:**
    1. Sencillo de implementar y entender.
    2. Permite un control preciso sobre el tiempo que el avión puede volar.
- **Desventajas:**
    1. No considera factores externos como viento o rutas más largas.
    2. Comportamiento puede parecer artificial si es muy predecible.

Alternativa 2: Usar eventos aleatorios para determinar el consumo de combustible.

- **Ventajas:**

1. Introduce variabilidad y realismo al juego.
2. Simula condiciones de vuelo impredecibles.

- **Desventajas:**
    1. Difícil de equilibrar; puede generar frustración en el jugador.
    2. Requiere más pruebas y ajustes para evitar injusticias.

Selección:
Se selecciona la **Alternativa 1** porque ofrece un control más directo sobre la mecánica del juego y reduce la frustración del jugador al permitirle anticipar el comportamiento.

Problema 4: Crear aviones con un ID único generado con GUIDs

Alternativa 1: Generar GUIDs utilizando métodos incorporados del lenguaje (por ejemplo, Guid.NewGuid() en C#).

- **Ventajas:**
    1. Generación automática y garantizada de unicidad.
    2. Fácil de implementar.
- **Desventajas:**
    1. Los GUIDs pueden ser difíciles de leer o depurar por su longitud.
    2. No permiten personalización específica del formato.

Alternativa 2: Crear IDs personalizados basados en prefijos y contadores secuenciales.

- **Ventajas:**
    1. IDs más legibles y fáciles de rastrear.
    2. Posibilidad de añadir prefijos específicos (como "AV-" para aviones).
- **Desventajas:**
    1. Mayor riesgo de colisiones si no se controla bien el contador.
    2. Requiere más lógica y pruebas para asegurar unicidad.

Selección:
Se selecciona la **Alternativa 1** porque los GUIDs aseguran unicidad sin necesidad de implementar lógica adicional, lo que es crucial en un entorno donde los aviones son generados dinámicamente.

Problema 5: Ordenar la tripulación de un avión derribado por ID, rol o horas de vuelo

Alternativa 1: Usar un algoritmo de Selection Sort.

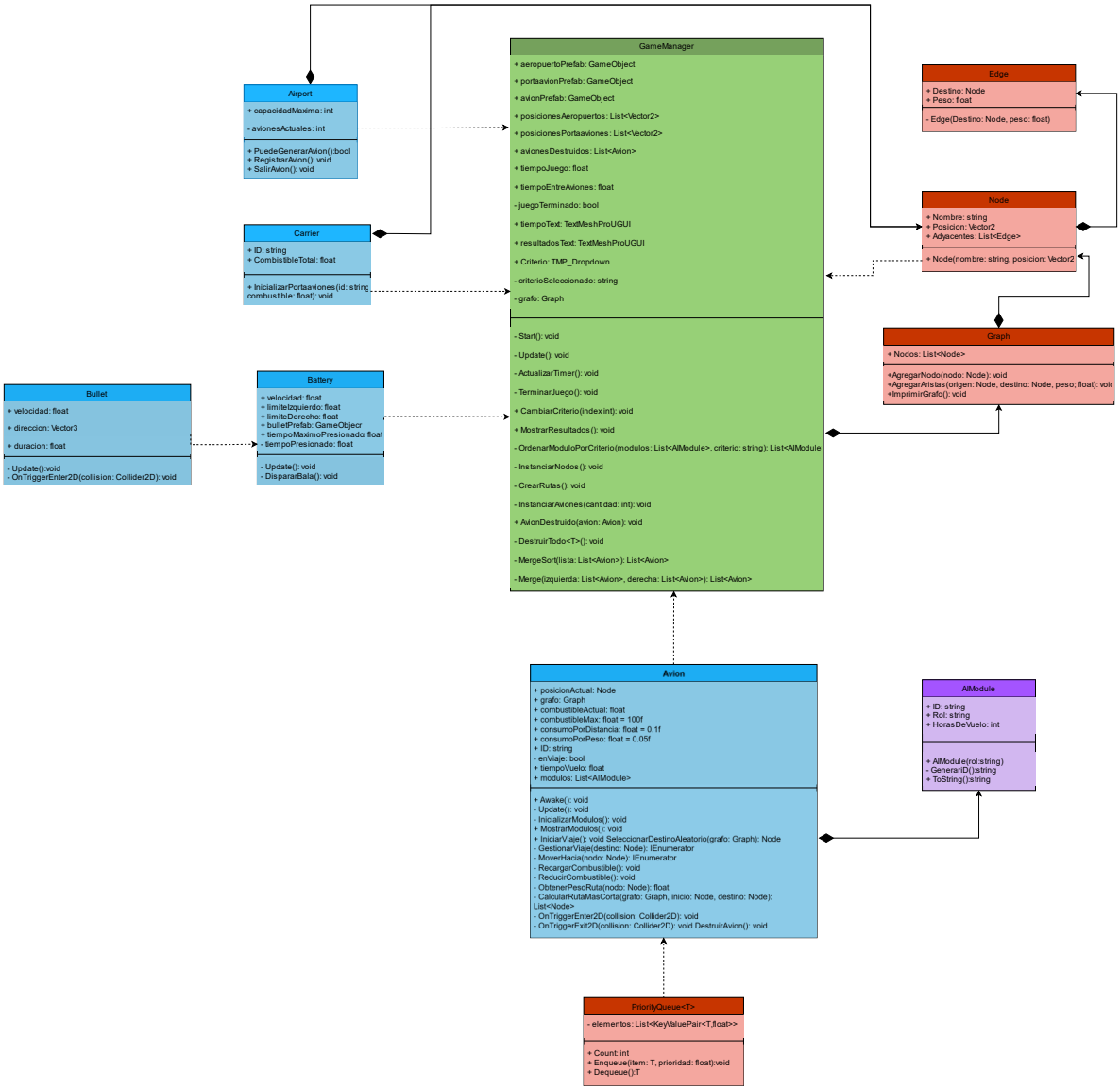
- **Ventajas:**
    1. Algoritmo simple de implementar y entender.
    2. Adecuado para listas pequeñas, como la tripulación de un avión.
- **Desventajas:**
    1. Ineficiente para listas más grandes (complejidad $O(n^2)$).
    2. No es fácilmente adaptable si se desea optimizar el rendimiento.

Alternativa 2: Usar una estructura de datos como un árbol de búsqueda binario para ordenar.

- **Ventajas:**
    1. Mayor eficiencia para listas dinámicas y más grandes.
    2. Fácil de extender si se requieren búsquedas frecuentes.
- **Desventajas:**
    1. Más complejo de implementar en comparación con Selection Sort.
    2. Puede ser excesivo para una lista pequeña como la tripulación.

Selección:
Se selecciona la **Alternativa 1** porque es suficiente para el caso específico de ordenar una tripulación pequeña y no requiere una estructura adicional compleja.

Diagrama de flujo



CheckList

Historia de usuario	Estado
Como jugador, quiero controlar una batería antiaérea que se mueva de izquierda a derecha a velocidad constante, para intentar derribar la mayor cantidad de aviones posible en un tiempo limitado.	Implementada
Como jugador, quiero que el tiempo de presión del clic determine la velocidad de la bala lanzada, para tener control sobre la precisión y efectividad de cada disparo.	Pendiente
Como usuario del sistema, quiero que los aeropuertos y portaaviones se generen en posiciones aleatorias al inicio del juego, para que cada partida tenga un mapa diferente y desafiante.	Pendiente
Como jugador, quiero que las rutas entre aeropuertos y portaaviones tengan un peso basado en la distancia y el tipo de destino (aeropuerto, portaaviones, ruta continental o interoceánica), para reflejar los distintos niveles de dificultad al planear rutas de vuelo.	Implementada
Como avión autónomo, quiero seleccionar un destino aleatorio y calcular la ruta óptima, teniendo en cuenta el peso de cada ruta para asegurar una navegación eficiente.	Implementada
Como avión autónomo, quiero que mi tiempo de espera en el destino sea aleatorio y recargar combustible durante este tiempo, para poder continuar mis vuelos de manera realista.	Implementada
Como aeropuerto, quiero gestionar mi suministro de combustible y distribuirlo de forma adecuada entre los aviones, de modo que pueda satisfacer sus necesidades según mi capacidad.	Pendiente
Como usuario del sistema, quiero que los aeropuertos y portaaviones se generen en posiciones aleatorias al inicio del juego, para que cada partida tenga un mapa diferente y desafiante.	Pendiente

Como jugador, quiero que las rutas entre aeropuertos y portaaviones tengan un peso basado en la distancia y el tipo de destino (aeropuerto, portaaviones, ruta continental o interoceánica), para reflejar los distintos niveles de dificultad al planear rutas de vuelo.	Implementada
Como avión autónomo, quiero seleccionar un destino aleatorio y calcular la ruta óptima, teniendo en cuenta el peso de cada ruta para asegurar una navegación eficiente.	Implementada
Como avión autónomo, quiero que mi tiempo de espera en el destino sea aleatorio y recargar combustible durante este tiempo, para poder continuar mis vuelos de manera realista.	Implementada
Como aeropuerto, quiero gestionar mi suministro de combustible y distribuirlo de forma adecuada entre los aviones, de modo que pueda satisfacer sus necesidades según mi capacidad.	Pendiente
Como jugador, quiero ver cómo un avión puede caer si se le acaba el combustible antes de llegar a su destino, para reflejar las consecuencias de una mala gestión de recursos en el juego.	Implementada
Como aeropuerto, quiero construir nuevos aviones cada cierto tiempo y limitar la cantidad de aviones activos según la capacidad de mi hangar, para mantener el control de mi flota.	Implementada parcialmente
Como aeropuerto, quiero asignar a cada avión un ID único generado con GUIDs, para poder identificar y gestionar de forma precisa cada avión en mi flota.	Implementada
Como avión autónomo, quiero estar equipado con cuatro módulos de inteligencia artificial (Pilot, Copilot, Maintenance, Space Awareness) que trabajen en conjunto para asegurar mi vuelo y respuesta ante problemas.	Implementada
Como jugador o desarrollador, quiero poder obtener una lista de todos los aviones derribados y ordenarlos por su ID usando el algoritmo de Merge Sort, para	Implementada

visualizar de forma organizada la información de las bajas.	
Como usuario del sistema, quiero poder ver la tripulación de cada avión derribado y ordenarla por ID, rol o horas de vuelo, utilizando Selection Sort, para analizar la información relevante de cada módulo de AI.	Implementada
Como jugador, quiero ver todos los datos relevantes del juego, incluyendo rutas calculadas, pesos de cada ruta y atributos de los aviones, para tener un contexto claro de la situación en el mapa.	Pendiente