Instituto Tecnológico Costa Rica

Escuela de Ingeniería en Computadores

Algoritmos y Estructura de Datos I

(CE - 1103)

# Proyecto III - AirWar

Profesor:

Leonardo Araya Martínez

Estudiantes:

Ariel Saborio Álvarez

2024101248

Sofía Xie Xie

2024100513

II Semestre

2024

## Tabla de Contenido

# 1. Introducción

Este proyecto tiene como objetivo el desarrollo de un juego de guerra aérea denominado *AirWar*, en el cual se aplicarán los principios de la Programación Orientada a Objetos (POO), así como estructuras de datos y algoritmos de búsqueda y ordenamiento. El proyecto tiene el objetivo de reforzar y demostrar habilidades técnicas en el diseño de software, utilizando el lenguaje de programación C# y frameworks como WPF.

En *AirWar*, el jugador controla una batería antiaérea para defenderse de aeronaves que circulan entre aeropuertos y portaaviones generados aleatoriamente en el mapa. El juego cuenta con una lógica compleja que incluye la asignación de rutas y la gestión de recursos de combustible, tanto en aeropuertos como en aviones. Además, se han implementado módulos de inteligencia artificial para la autonomía de los aviones, lo que permite la simulación de toma de decisiones y el cálculo de rutas óptimas.

# 2. Diseño

2.1 Requerimientos - Historia de Usuario

| | |
|---|---|
| **Identificador del US** | US 000 |
| **Nombre del US** | Lenguaje de Programación |
| **Descripción del US** | Como desarrollador, quiero que sea programado en C# utilizando WPF, para asegurar que el juego se ejecute en un entorno visual y funcional adecuado. |
| **Criterios de Aceptación** | ❖ El código debe de estar implementado en el lenguaje de C#<br>❖ Los elementos deben de estar reflejados en una interfaz.<br>❖ La ventana de juego debe ser interactiva y responder a las entradas del jugador de manera fluida y sin errores. |
| **Prioridad** | Alta |

| | |
|---|---|
| **Identificador del US** | US 001 |
| **Nombre del US** | Objetivo del Juego |
| **Descripción del US** | Como jugador, quiero poder destruir la mayor cantidad de aviones posibles en un tiempo determinado, para lograr la máxima puntuación y cumplir el objetivo del juego. |
| **Criterios de Aceptación** | ❖ Al iniciar la partida, el temporizador debe mostrarse en pantalla y comenzar a contar el tiempo de juego.<br>❖ El sistema debe registrar cada avión destruido y actualizar la puntuación en tiempo real.<br>❖ Al finalizar el tiempo, el juego debe mostrar la puntuación final y permitir al jugador la opción de reiniciar o salir del juego.<br>❖ La puntuación debe actualizarse cada vez que un avión es destruido, y el total de aviones destruidos debe ser visible en la interfaz. |
| **Prioridad** | Alta |

| | |
|---|---|
| **Identificador del US** | US 002 |
| **Nombre del US** | Generación de Estructuras y Rutas |
| **Descripción del US** | Como jugador, quiero ver aeropuertos y portaaviones generados aleatoriamente en el mapa, conectados por rutas visibles para que el juego sea más visual y estratégico. |
| **Criterios de Aceptación** | ❖ Las rutas deben generarse automáticamente, conectando aeropuertos y portaaviones mediante un grafo que use listas de adyacencia.<br>❖ Las rutas deben visualizarse en pantalla como líneas o trayectorias, conectando cada aeropuerto y portaaviones de manera clara y distinta. |
| **Prioridad** | Alta |

| | |
|---|---|
| **Identificador del US** | US 003 |
| **Nombre del US** | Control de la Batería Antiaérea |
| **Descripción del US** | Como jugador, quiero ser capaz de controlar los disparos de una batería antiaérea, además que este se mueva de manera constante de izquierda a derecha, para que el juego sea dinámico y realista. |
| **Criterios de Aceptación** | ❖ La batería debe moverse de forma automática de izquierda a derecha sin salir de la pantalla.<br>❖ El jugador debe poder controlar la salida de las balas en línea recta hacia arriba desde la posición de la batería.<br>❖ La velocidad de la bala debe ser mayor cuanto más tiempo se mantenga el clic antes de disparar.<br>❖ Las balas deben desaparecer al salir de la pantalla o al impactar con un avión enemigo. |
| **Prioridad** | Alta |

| | |
|---|---|
| **Identificador del US** | US 004 |
| **Nombre del US** | Rutas con Costos Variables |
| **Descripción del US** | Como jugador, quiero que cada ruta tenga un costo basado en la distancia y el destino, de modo que la elección de rutas afecte la estrategia del juego, para defenderme y obtener puntos. |
| **Criterios de Aceptación** | ❖ Cada ruta debe tener un peso calculado basado en la distancia entre los puntos que conecta.<br>❖ El peso debe incrementarse si la ruta es interoceánica o si el destino es un portaaviones.<br>❖ Los aviones deben calcular sus rutas considerando el costo total de cada trayecto disponible. |
| **Prioridad** | Medio |

| | |
|---|---|
| **Identificador del US** | US 005 |
| **Nombre del US** | Decisiones de Vuelo Autónomas |
| **Descripción del US** | Como jugador, quiero que los aviones seleccionen destinos y rutas automáticamente, seguidamente que despeguen nuevamente después de un tiempo de espera aleatorio, para que el juego sea dinámico. |
| **Criterios de Aceptación** | ❖ Al despegar, cada avión debe elegir un destino aleatorio.<br>❖ El sistema debe calcular y asignar la ruta óptima según los pesos de las rutas hacia el destino.<br>❖ Al aterrizar, cada avión debe esperar un tiempo aleatorio (entre 5 y 10 segundos) antes de despegar de nuevo.<br>❖ Durante el tiempo de espera, el avión debe recargar una cantidad aleatoria de combustible |
| **Prioridad** | Medio |

| | |
|---|---|
| **Identificador del US** | US 006 |
| **Nombre del US** | Racionamiento de Combustible |
| **Descripción del US** | Como jugador, quiero que el combustible de los aviones esté limitado y que los aeropuertos racionen su suministro, para que el juego tenga un elemento de riesgo. |
| **Criterios de Aceptación** | ❖ Cada aeropuerto debe tener una cantidad limitada de combustible disponible.<br>❖ Los aviones deben consumir combustible en función de la distancia de cada trayecto y el peso de la ruta.<br>❖ Si un avión queda sin combustible antes de aterrizar, debe caer y mostrarse como derribado en la interfaz.<br>❖ Los aviones caídos deben eliminarse del juego y registrarse en una lista de aviones derribados. |
| **Prioridad** | Baja |

| | |
|---|---|
| **Identificador del US** | US 007 |
| **Nombre del US** | Construcción de Nuevos Aviones |
| **Descripción del US** | Como jugador, quiero que los aeropuertos generen nuevos aviones con una capacidad máxima determinada, para controlar el tráfico aéreo en el juego. |
| **Criterios de Aceptación** | ❖ Cada aeropuerto debe construir un nuevo avión cada cierto intervalo de tiempo.<br>❖ El número de aviones generados debe estar limitado por la capacidad del hangar de cada aeropuerto.<br>❖ Cada avión nuevo debe tener un ID único generado mediante GUID para identificarlo de manera única en el juego.<br>❖ La cantidad de aviones en el hangar debe actualizarse en tiempo real en la interfaz. |
| **Prioridad** | Medio |

| | |
|---|---|
| **Identificador del US** | US 008 |
| **Nombre del US** | Aviones Autónomos con Módulos AI |
| **Descripción del US** | Como jugador, quiero que cada avión esté equipado con módulos de AI para simular la toma de decisiones autónoma. |
| **Criterios de Aceptación** | ❖ Cada avión debe tener cuatro módulos AI: Pilot, Copilot, Maintenance y Space Awareness.<br>❖ Cada módulo debe tener un ID de tres letras generado aleatoriamente al crearse.<br>❖ Cada módulo debe tener un rol específico (Pilot, Copilot, etc.) y registrar sus horas de vuelo acumuladas.<br>❖ Los módulos deben cumplir sus roles: el Copilot toma el control si el Pilot falla, y el módulo de Maintenance intenta solucionar problemas detectados.<br>❖ Se detalla una lista ordenada por Mergesort para cada avión derribado. |
| **Prioridad** | Alta |

| | |
|---|---|
| **Identificador del US** | US 009 |
| **Nombre del US** | Ordenamiento de la Tripulación por Atributos |
| **Descripción del US** | Como jugador, quiero poder ver y ordenar la tripulación de cada avión derribado por ID, rol o horas de vuelo, para entender su rendimiento. |
| **Criterios de Aceptación** | ❖ La tripulación (módulos AI) de cada avión derribado debe mostrarse en una lista al seleccionar el avión en la interfaz.<br>❖ El jugador debe poder ordenar esta lista por ID, rol o horas de vuelo usando el algoritmo de Selection Sort.<br>❖ La interfaz debe reflejar los cambios de orden de inmediato y sin errores visuales.<br>❖ Cada módulo AI en la lista debe mostrar su ID, rol y horas de vuelo acumuladas. |
| **Prioridad** | Baja |

| | |
|---|---|
| **Identificador del US** | US 010 |
| **Nombre del US** | Visualización en Pantalla de Información del Juego |
| **Descripción del US** | Como jugador, quiero ver todos los datos relevantes del juego en pantalla, para entender mejor las decisiones de los aviones y mi progreso en el juego. |
| **Criterios de Aceptación** | ❖ La interfaz debe mostrar las rutas de los aviones, pesos de las rutas y los atributos de cada avión de forma clara y organizada.<br>❖ La información de cada avión debe incluir su ID, destino, combustible restante y ruta seleccionada.<br>❖ Los datos deben actualizarse en tiempo real y reflejar cualquier cambio en el juego (nuevos aviones, derribos, cambios de ruta).<br>❖ La interfaz debe ser intuitiva y no sobrecargar al jugador con información innecesaria. |
| **Prioridad** | Media |

2.2 Soluciones Extras

**Problema 1 - Generación de Estructuras y Rutas (US 002)**

| | **SOLUCIÓN #1 - Kruskal** | **SOLUCIÓN #2 - Red Aleatoria** |
|---|---|---|
| **Descripción** | Se generan aleatoriamente nodos sobre el mapa, seguidamente se emplea el algoritmo de Kruskal para conectar la estructura y generar las rutas. | Generar nodos en posiciones aleatorias dentro del mapa y hacer una variable random que los conecte con cierta posibilidad. |
| **Ventajas** | - Se genera un árbol mínimo sin ciclos cerrados.<br>- Puede manejar grandes cantidades de nodos. | - No requiere algoritmos complejos, es más simple. |
| **Desventajas** | - Es complejo de implementar y su diseño de lógica es difícil de comprender.<br>- Las conexiones pueden ser complejas de modo que se dificulte su comprensión. | - La distribución de nodos puede ser afectada, ya sea teniendo nodos aislados o muy cercano a otro.<br>- Las conexiones pueden ser costosas si el número de nodos y rutas es muy alto.<br>- Las rutas no son lo más eficiente. |
| **Elección** | | Se eligió crear rutas con distintas posibilidades de establecerse ya que al ser un proyecto pequeño, el algoritmo Kruskal es muy complejo de comprender, además, el juego es pequeño, por lo tanto no se espera generar muchos nodos o rutas. |

**Problema 2 - Rutas con Costos Variables (US 004)**

| | SOLUCIÓN #1 - Pitágoras | SOLUCIÓN #2 - Algoritmos |
|---|---|---|
| **Descripción** | Emplear fórmulas matemáticas para calcular el costo de cada ruta. Incluyendo factores como la distancia, ruta interoceánica y destino. | Los costos de cada ruta se ajustan en tiempo real basándose en factores dentro del mapa. Empleando algoritmos de corrimiento. |
| **Ventajas** | - Es fácil de implementar y entender.<br>- Cálculo rápido de los costos de las rutas.<br>- No requiere mucha memoria dichos cálculos simples. | - Es más flexible ante cambios que se puedan presentar en el juego. |
| **Desventajas** | - Depender de una fórmula es muy rígido, no se va a poder adaptar a cambios, por lo tanto lo vuelve menos flexible. | - Más complejo de implementar. |
| **Elección** | - Para este pequeño proyecto se montó una fórmula sencilla para manejar las rutas, ya que es fácil y menos compleja. | |

**Problema 3 - Decisiones de Vuelo Autónomas (US 005)**

| | SOLUCIÓN #1 - Aleatorio | SOLUCIÓN #2 - Dinámico |
|---|---|---|
| **Descripción** | El avión selecciona su destino aleatoriamente, calcula la ruta óptima y, al aterrizar, toma un tiempo random en volver a despegar de nuevo. | El tiempo de espera y el gas varía aleatoriamente dentro de un rango definido cada vez que un avión aterriza. |
| **Ventajas** | - Es fácil de implementar y entender.<br>- Bajo costo, no es necesario realizar más cálculos de lo necesario. | - Más dinámico, impredecible y variado. Haciéndolo más realista. |
| **Desventajas** | - Menor dinamismo: Los aviones siempre esperan el mismo tiempo, lo que puede hacer que el comportamiento sea predecible y no tan interesante. | - Es más complejo unir la generación de tiempos y cargas aleatorias con el sistema de rutas. |
| **Elección** | - Se decidió emplear una variable random para determinar los tiempos, en donde genera un random que determina dichas variables. | |

**Problema 4 - Racionamiento de Combustible (US 006)**

| | SOLUCIÓN #1 - Calcular | SOLUCIÓN #2 - Valor Fijo |
|---|---|---|
| **Descripción** | El consumo de combustible de un avión varía en función de la distancia de su trayecto y el peso de la carga que transporta, por lo tanto es calculada. | Cada avión posee un conjunto de valores que representa la cantidad de combustible gastado fijo, en cada trayecto se le resta o suma los valores. |
| **Ventajas** | - Es más realista<br>- El sistema puede ser fácilmente ampliado para incluir factores adicionales. | - Es más sencillo de implementar y entender. |
| **Desventajas** | - Los cálculos pueden volverse complejos.<br>- Interfaz más cargada: Los jugadores tendrán que ver y gestionar más detalles, lo que podría hacer que la interfaz de usuario sea más complicada y menos intuitiva. | - Es menos realista porque no tiene en cuenta el consumo de combustible basado en distancias y pesos específicos de los aviones. |
| **Elección** | - Proporciona una experiencia de juego más dinámica y realista. Aunque es más compleja de implementar. | |

**Problema 5 - Ordenamiento de la Tripulación por Atributos (US 009)**

| | SOLUCIÓN #1 - Selection Sort | SOLUCIÓN #2 - Ordenado |
|---|---|---|
| **Descripción** | Implementar el algoritmo de Selection Sort de forma directa sobre la lista de tripulación mostrada en la interfaz. | Emplear una lista interna (en memoria) que mantiene los módulos ordenados. En la interfaz se actualiza cuando el jugador haya terminado de realizar ajustes. Ejecutar el algoritmo en segundo plano. |
| **Ventajas** | - El algoritmo es fácil de implementar y entender.<br>- Se ven los cambios inmediatos en la interfaz. | - Mayor control sobre el rendimiento. |
| **Desventajas** | - La complejidad es de $O(n^2)$.<br>- Ineficiente y no es flexible. | - Las respuestas no se ven de forma inmediata.<br>- Más complejo de implementar. |
| **Elección** | - Es más simple y se dan resultados inmediatamente. | |

## 2.3 Diagrama de Clases UML

| GameLogic |
| --- |
| - gameCanvas = canvas;<br>- this.player = player;<br>- this.airportPositions = airportPositions;<br>- this.carrierPositions = carrierPositions; |
| ● StartGame()<br>● SMovimiento()<br>● STimer()<br>● HandleKeyPress(Key key, DateTime pressStartTime, DateTime pressEndTime)<br>● CrearEnemigosDesdePosiciones()<br>● EnemigosSpawn(double x, double y, string origen)<br>● MoverAviones()<br>● MoverAvion(Enemy enemigo)<br>● EliminarEnemigosMarcados()<br>● DrawRoutes()<br>● GetEnemigos() |

| Enemy |
| --- |
| - Rectangle rectangulo<br>- double x<br>- double y<br>- string origen<br>- Canvas gameCanvas |
| ● Enemy(Rectangle rectangulo, double x, double y, string origen, Canvas gameCanvas)<br>● StartDetentionTime()<br>● UpdateDetentionTimer()<br>● RefuelIncrement()<br>● ConsumeFuel(double amount)<br>● MarkForDeletion()<br>● Refuel()<br>● Destruir(Canvas gameCanvas, List<Enemy> enemigos) |

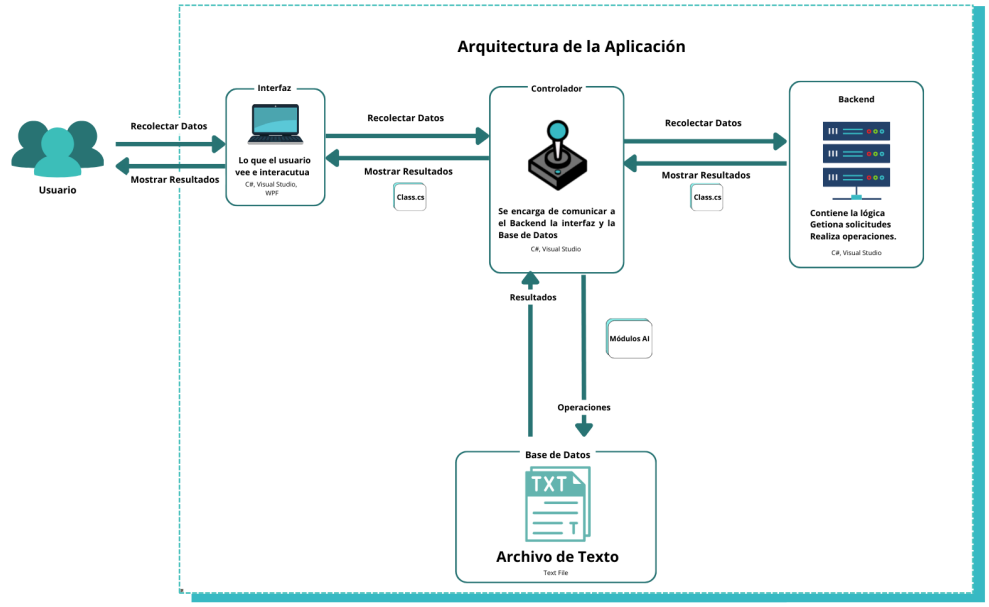
| MainWindow |
| --- |
| —---------- |
| • MainWindow()<br>• Window_KeyDown(object sender, KeyEventArgs e)<br>• Window_KeyUp(object sender, KeyEventArgs e) |

| Player |
| --- |
| - UIElement element<br>- Canvas canvas |
| • Player(UIElement element, Canvas canvas)<br>• Move()<br>• Shoot(double pressDuration, List<Enemy> enemigos)<br>• IsColliding(Rectangle bullet, Enemy enemigo)<br>• GetElement() |

| Grafo |
| --- |
| —---------- |
| • Grafo()<br>• PlaceNodesManually(List<(double X, double Y)> Aeropuerto, List<(double X, double Y)> Portaaviones, Canvas canvas)<br>• AddToCanvas(Nodo node, Canvas canvas)<br>• AddNode(Nodo node)<br>• AddEdge(Nodo from, Nodo to)<br>• CrearImagenEstructura(string imagePath, double width = 100, double height = 100)<br>• GenerateRandomRoutes(double connectionProbability) |

| Nodo |
| --- |
| - string name<br>- string type<br>- (double X, double Y) position<br>- UIElement shape |
| —---------- |

## 2.4 Diagrama de arquitectura




https://www.canva.com/design/DAGSnfwx9cw/_wIJuYwvfWesLAAD087EEA/view?utm_content=DAGSnfwx9cw&utm_campaign=designshare&utm_medium=link2&utm_source=uniquelinks&utlId=h0de6d2cd75

## 2.5 Historias de Usuarios Implementadas

| Historias de Usuario Realizadas | Historias de Usuario Pendientes |
|---|---|
| US 000 Lenguaje de Programación | US 008 Aviones Autónomos con Módulos AI |
| US 001 Objetivo del Juego | US 009 Ordenamiento de la Tripulación por Atributos |
| US 002 Generación de Estructuras y Rutas | US 010 Visualización en Pantalla de Información del Juego |
| US 003 Control de la Batería Antiaérea | |
| US 004 Rutas con Costos Variables | |
| US 005 Decisiones de Vuelo Autónomas | |
| US 006 Racionamiento de Combustible | |
| US 007 Construcción de Nuevos Aviones | |