Atributos del curso

Instituto Tecnológico de Costa Rica

Algoritmos Y Estructuras De Datos

Sede Central Cartago

Profesor: Jose Isaac Ramirez Herrera

Integrantes:

Jose Fabian Viquez Elizondo

Sebastián Bolaños Zamora

Ciclo Lectivo: II Semestre del año 2024

Tabla de contenidos

Introduccion

El presente documento describe el desarrollo y análisis del proyecto AirWar, un juego de guerra aérea diseñado para aplicar principios del Paradigma Orientado a Objetos y la implementación de algoritmos de búsqueda y estructuras de datos. Este trabajo busca cumplir con los objetivos académicos de la materia Algoritmos y Estructuras de Datos I del Instituto Tecnológico de Costa Rica.

En el proyecto, se presenta una solución técnica al problema planteado, acompañada de la respectiva documentación detallada, que incluye historias de usuario, propuestas de soluciones a problemas específicos y diagramas UML. Además, se valida que la implementación cumpla con los requerimientos del juego, así como con los atributos de acreditación definidos en la rúbrica del curso. Este documento busca servir como una evidencia integral de las habilidades técnicas y de diseño adquiridas durante el proyecto.

Historias de usuario

HU-001: Destrucción de aviones

Como jugador, **quiero** destruir la mayor cantidad de aviones posible dentro de un periodo de tiempo definido, **para** maximizar mi puntuación en el juego.

Criterios de éxito:

- El sistema registra correctamente la cantidad de aviones destruidos dentro del tiempo límite.
- El temporizador funciona sin errores y termina el juego al alcanzar el tiempo establecido.

Criterios de fallo:

- No se registra correctamente la cantidad de aviones destruidos.
- El temporizador no detiene el juego al alcanzar el tiempo límite.

HU-002: Generación de aeropuertos, portaaviones y rutas

Como jugador, **quiero** que los aeropuertos, portaaviones y rutas se generen aleatoriamente al iniciar el juego, **para** tener una experiencia de juego única en cada partida.

Criterios de éxito:

- Los aeropuertos y portaaviones se generan en posiciones aleatorias y no se superponen.
- Las rutas generadas conectan correctamente los aeropuertos y portaaviones.

Criterios de fallo:

- Los elementos se generan en posiciones fijas o superpuestas.
- Las rutas contienen errores, como conexiones inválidas o pesos negativos.

HU-003: Control de la batería antiaérea

Como jugador, **quiero** mover la batería antiaérea de izquierda a derecha con velocidad constante y disparar balas ajustando su velocidad según el tiempo de carga, **para** atacar a los aviones con precisión.

Criterios de éxito:

- La batería se mueve continuamente entre los límites de la pantalla sin detenerse.
- Las balas se disparan en línea recta con una velocidad ajustada correctamente al tiempo de carga.

Criterios de fallo:

- La batería se detiene o se mueve de forma errática.
- Las balas no responden al tiempo de carga o tienen trayectorias incorrectas.

HU-004: Generación de rutas con pesos variables

Como jugador, **quiero** que las rutas entre aeropuertos y portaaviones se generen aleatoriamente con pesos que consideren distancia, tipo de destino y tipo de ruta, **para** reflejar costos realistas en el juego.

Criterios de éxito:

- Todas las rutas generadas tienen pesos coherentes con las condiciones establecidas.
- Las rutas se muestran correctamente en el mapa.

Criterios de fallo:

- Los pesos asignados no reflejan las condiciones dadas (tipo de destino, distancia, tipo de ruta).
- Las rutas no aparecen correctamente en el mapa.

HU-005: Decisión de ruta y recarga de combustible

Como avión, **quiero** elegir un destino aleatorio y calcular la mejor ruta considerando los pesos, además de recargar combustible durante la espera en el aeropuerto, **para** completar mis trayectos sin interrupciones.

Criterios de éxito:

- El avión selecciona siempre la ruta más óptima según los pesos.
- Durante la espera, el avión recarga una cantidad aleatoria de combustible correctamente.

Criterios de fallo:

- Se selecciona una ruta no óptima cuando existen mejores alternativas.
- El combustible no se recarga correctamente en el aeropuerto.

HU-006: Racionamiento de combustible en aeropuertos

Como aeropuerto, **quiero** racionar el combustible disponible entre los aviones, **para** optimizar su uso y evitar que los aviones se queden sin combustible antes de aterrizar.

Criterios de éxito:

- El combustible se distribuye de forma equitativa o estratégica según lo definido.
- Los aviones no se caen salvo que realmente no sea posible abastecerlos.

Criterios de fallo:

- La distribución de combustible es errática o no se realiza.
- Los aviones se quedan sin combustible debido a un mal manejo del racionamiento.

HU-007: Construcción de nuevos aviones

Como aeropuerto, **quiero** generar nuevos aviones en intervalos regulares sin exceder la capacidad del hangar, asignándoles un ID único, **para** mantener una flota activa en el juego.

Criterios de éxito:

- Los aviones se generan con IDs únicos y dentro del límite del hangar.
- Los atributos de los aviones se asignan correctamente al crearlos.

Criterios de fallo:

- Se generan más aviones de los permitidos en el hangar.
- Los IDs de los aviones no son únicos o los atributos son inválidos.

HU-008: Visualización de datos del juego

Como jugador, **quiero** visualizar todos los datos relevantes del juego, como rutas, pesos y atributos de los aviones, **para** entender las estadísticas y tomar decisiones informadas.

Criterios de éxito:

- Los datos se muestran de forma clara, precisa y actualizada en tiempo real.
- Los elementos visuales son intuitivos y legibles.

Criterios de fallo:

- Los datos no se actualizan correctamente o muestran información incorrecta.
- La interfaz gráfica es confusa o difícil de interpretar.

Problemas

Problema 1: Movimiento de la batería antiaérea

Alternativa 1: Uso de un temporizador para actualizar la posición a intervalos constantes.

- **Ventajas:**
    - Implementación sencilla.
    - Requiere menos cálculos, lo que puede mejorar el rendimiento en sistemas con recursos limitados.
- **Desventajas:**
    - Movimiento puede no ser fluido si el temporizador tiene errores en la sincronización.
    - Puede causar desincronización si otros elementos del juego dependen del tiempo global.

Alternativa 2: Actualización basada en el tiempo transcurrido en cada frame.

- **Ventajas:**
    - Movimiento más suave y sincronizado con el resto de la lógica del juego.
    - Ajustable a diferentes tasas de frames (FPS).
- **Desventajas:**
    - Requiere más recursos computacionales para realizar los cálculos en cada frame.

Selección: La alternativa 1, debido a que era lo mas simple de implementar y lo que ocupaba menos recursos computacionales.

Problema 2: Generación aleatoria de aeropuertos, portaaviones y rutas

Alternativa 1: Generación previa de todos los elementos al inicio del juego.

- **Ventajas:**
    - Evita cálculos durante el juego, mejorando el rendimiento en tiempo de ejecución.
    - Fácil de depurar, ya que todo se genera de una vez.
- **Desventajas:**
    - Puede consumir mucho tiempo inicial si hay muchos elementos que calcular.
    - El mapa no se adapta a eventos dinámicos del juego.

Alternativa 2: Generación dinámica según las necesidades del juego.

- **Ventajas:**
    - Permite un juego más dinámico y adaptable.
    - Reduce la carga inicial del sistema.
- **Desventajas:**
    - Puede generar lag o interrupciones si no se optimiza adecuadamente.
    - Implementación más compleja.

Selección: La alternativa 1, ya que garantiza estabilidad y una experiencia de juego más predecible.

Problema 3: Racionamiento de combustible en los aeropuertos

Alternativa 1: Uso de un sistema de asignación equitativa (por ejemplo, dividir el combustible disponible entre todos los aviones).

- **Ventajas:**
    - Simple de implementar.
    - Garantiza que todos los aviones reciban combustible.
- **Desventajas:**
    - No optimiza el uso del recurso, ya que puede asignar combustible a aviones que no lo necesitan urgentemente.

Alternativa 2: Priorización basada en la necesidad (aviones con menos combustible reciben más).

- **Ventajas:**
    - Maximiza las probabilidades de que los aviones lleguen a su destino.
    - Realista y estratégico.
- **Desventajas:**
    - Implementación más compleja.
    - Puede ser percibido como injusto si no se comunica claramente al jugador.

Selección: La alternativa 1, porque asi nos aseguramos de que todos los aviones reciban combustible.

Problema 4: Visualización de datos en pantalla

Alternativa 1: Mostrar todos los datos relevantes en una única vista de información estática.

- **Ventajas:**
    - Fácil de implementar.
    - Proporciona toda la información al jugador de manera inmediata.
- **Desventajas:**
    - Puede saturar la pantalla con demasiada información.
    - Dificultad para encontrar datos específicos.

Alternativa 2: Usar un sistema de ventanas o paneles desplegables para organizar la información.

- **Ventajas:**
    - Organiza mejor la información y facilita su búsqueda.
    - Mejora la experiencia del jugador al evitar saturación visual.
- **Desventajas:**
    - Requiere más tiempo y esfuerzo de implementación.
    - Puede ser confuso si la navegación no es intuitiva.

Selección: La alternativa 1, debido a que es lo más simple de implementar y proporciona toda la información de manera inmediata.

Problema 5: Creación de nuevos aviones en aeropuertos

Alternativa 1: Crear aviones en intervalos de tiempo constantes.

- **Ventajas:**
    - Fácil de implementar y predecible.
    - Garantiza una tasa constante de producción.
- **Desventajas:**
    - Puede resultar en una sobrepoblación si no se controla adecuadamente la capacidad del hangar.
    - No responde a las necesidades dinámicas del juego.

Alternativa 2: Crear aviones basados en la demanda (por ejemplo, si hay rutas con pocos aviones).

- **Ventajas:**
    - Se adapta a las necesidades del juego.
    - Evita una sobrepoblación innecesaria.
- **Desventajas:**
    - Implementación más compleja.
    - Menos predecible, lo que puede confundir al jugador.

Selección: La alternativa 1, porque asi nos aseguramos que siempre haya una creación constante de aviones por el mapa.

Diagrama de clases UML

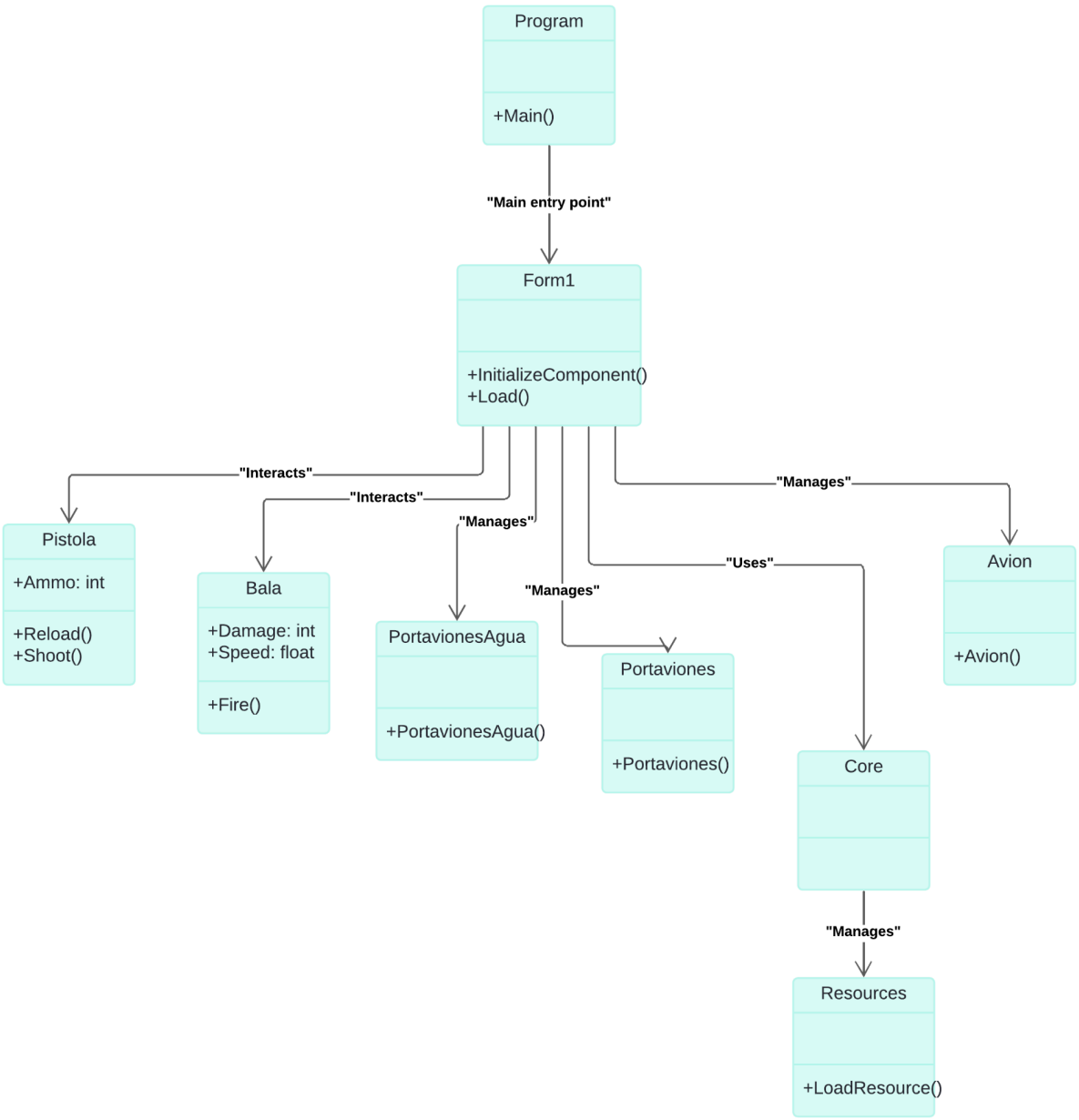
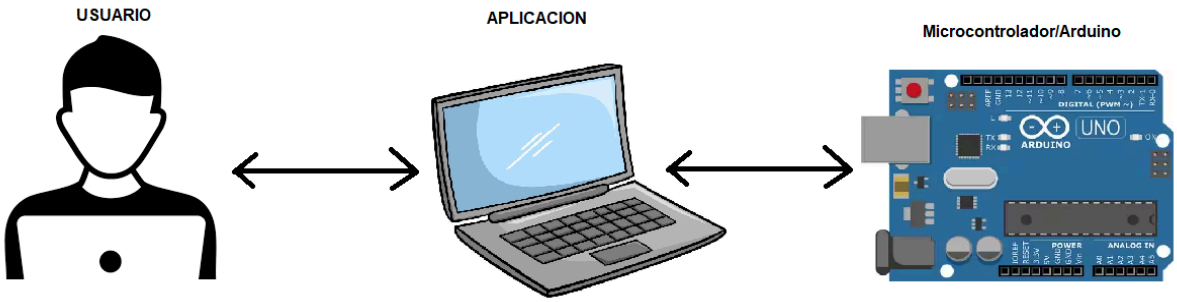


Diagrama de arquitectura



Completacion de Historias de Usuario

- ☑ ~~HU-001: Destrucción de aviones~~
- ☑ ~~HU-002: Generación de aeropuertos, portaaviones y rutas~~
- ☑ ~~HU-003: Control de la batería antiaérea~~
- ☑ ~~HU-004: Generación de rutas con pesos variables~~
- ☑ ~~HU-005: Decisión de ruta y recarga de combustible~~
- ☑ ~~HU-006: Racionamiento de combustible en aeropuertos~~
- ☑ ~~HU-007: Construcción de nuevos aviones~~
- ☑ ~~HU-008: Visualización de datos del juego~~