Proyecto Semestral, Algoritmos II

Título: Simulador de Rescate

(equipos de 4 estudiantes solamente)

Objetivos Generales

- Desarrollar una aplicación de simulación para coordinar operaciones de rescate terrestre en un mapa 2D con múltiples tipos de vehículos
- Implementar algoritmos eficientes para la navegación autónoma, evitando obstáculos y optimizando la recolección de recursos
- Desarrollar algoritmos y estructuras de datos para maximizar el puntaje mientras se minimizan los riesgos

Introducción

El simulador de rescate es un juego estratégico donde dos jugadores compiten para recolectar la mayor cantidad de personas y mercancías de un campo de batalla utilizando diferentes tipos de vehículos. El campo está poblado con recursos valiosos pero también contiene minas peligrosas que pueden destruir los vehículos.

El juego se desarrolla en un **mapa 2D** que contiene los siguientes elementos fundamentales:

Vehículos

Cada jugador dispone de una flota de 10 vehículos distribuidos de la siguiente manera:

1. **3 Jeeps**: Pueden recoger todo tipo de carga y realizar hasta 2 viajes sin necesidad de volver a su base
2. **2 Motos**: Pueden recoger solo personas. Una vez recogida una persona tienen que volver a la base antes de iniciar otro viaje
3. **2 Camiones**: Pueden recoger todo tipo de carga y realizar hasta 3 viajes sin necesidad de volver a la base
4. **3 Autos**: Pueden recoger personas y cargas. Una vez recogido una carga tienen que volver a la base antes de iniciar otro viaje

Recursos a Recolectar

El mapa contiene un total de **60 elementos** distribuidos aleatoriamente:

- **10 Personas** (50 puntos cada una)
- **50 Mercancías** distribuidas en cuatro tipos:
  - **Ropa**: 5 puntos por unidad
  - **Alimentos**: 10 puntos por unidad
  - **Medicamentos**: 20 puntos por unidad

- **Armamentos**: 50 puntos por unidad

Obstáculos - Minas

El terreno contiene 5 tipos diferentes de minas con áreas de efecto específicas:

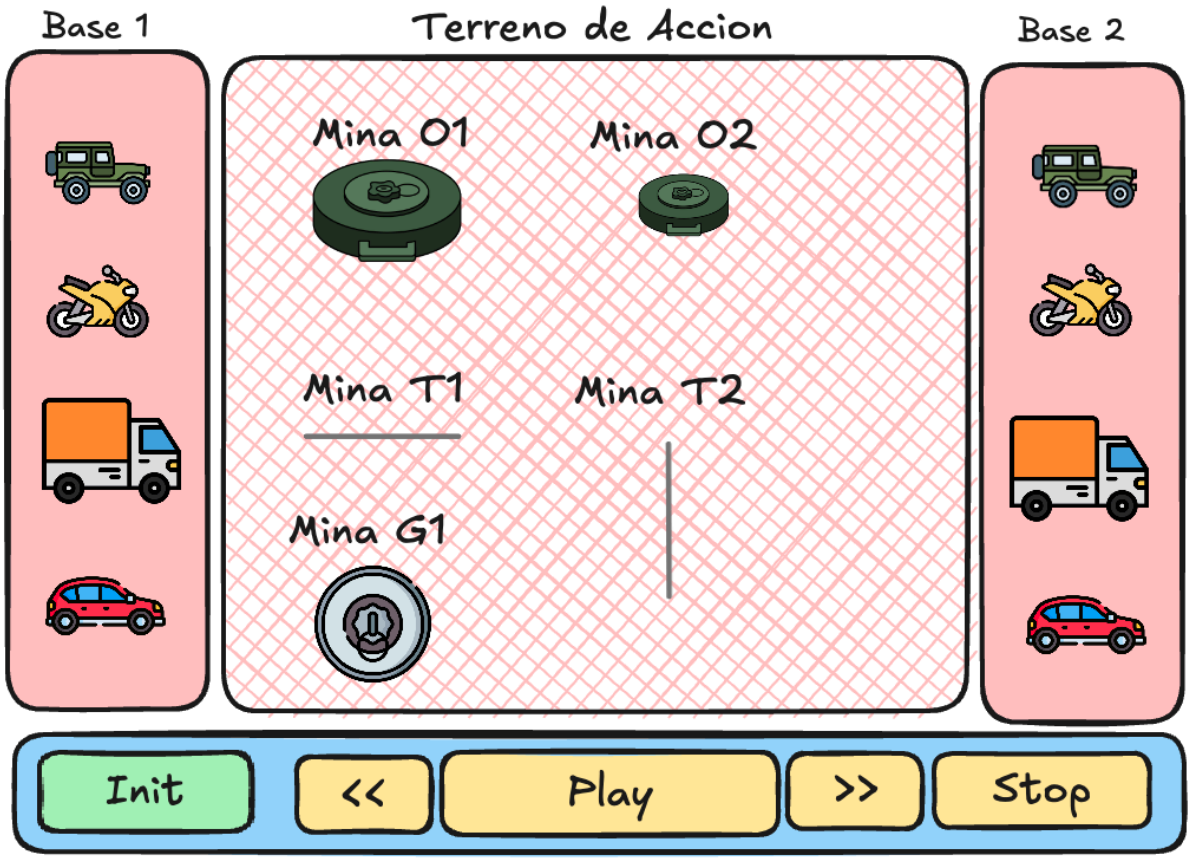
1. **Mina O1**: Radio circular de 10 unidades (estática)
2. **Mina O2**: Radio circular de 5 unidades (estática)
3. **Mina T1**: Radio horizontal de 10 unidades (estática)
4. **Mina T2**: Radio vertical de 5 unidades (estática)
5. **Mina G1**: Radio circular de 7 unidades (**móvil** - aparece y desaparece en el terreno por cada 5 instancias de tiempo)

Zonas de Base

Cada jugador tiene una **zona de salida y retorno** ubicada en extremos opuestos del mapa, desde dónde:

- Salen todas sus vehículos al inicio de la misión
- Deben regresar para entregar la carga recolectada
- Se contabiliza el puntaje final

Ejemplo gráfico de las componentes del juego:



Mecánica del Juego

Fase de Preparación

Antes de iniciar la simulación:

1. **Distribución Aleatoria**: Al presionar el botón **Init** las minas y recursos se ubican automáticamente en posiciones aleatorias dentro del terreno de acción (mapa). Cada vez que se presione el botón se distribuyen los elementos. Una vez que se inicia la simulación (botón **Play**) esta opción queda deshabilitada
2. **Programación de Estrategias**: Cada jugador debe haber implementado las estrategias de navegación y recolección para cada tipo de vehículo. Al presionar **Start** inicia la simulación.
3. **Validación**: El sistema verifica que todos los vehículos tengan estrategias definidas

Fase de Simulación Automática

Al presionar **"Start"**, comienza la simulación donde:

1. **Despegue**: Todas los vehículos salen simultáneamente de sus respectivas bases
2. **Navegación Inteligente**: Cada vehículo ejecuta su estrategia programada para:
    - Detectar y evaluar recursos cercanos
    - Decidir qué recoger basándose en capacidad y valor
    - Evitar minas detectando sus áreas de efecto
    - Esquivar colisiones con otros vehículos
    - Atacar otros vehículos enemigos provocando colisiones
3. **Recolección**: Los vehículos recogen recursos según sus capacidades específicas
4. **Retorno**: Una vez completada la carga o agotados los viajes permitidos, regresan a base
5. **Entrega**: Solo se contabiliza el puntaje de aeronaves que regresan exitosamente

Condiciones de Falla

Un vehículo es destruido automáticamente si:

- **Entra en área minada**: Cualquier contacto con el radio de efecto de una mina
- **Colisiona con otro vehículo**: Ambas aeronaves involucradas son destruidas y pierden toda su carga

Funcionalidades a Implementar

El sistema debe proporcionar las siguientes capacidades:

1. Gestión del Mapa

- Generar mapas aleatorios con distribución balanceada de recursos y minas
- Visualizar en tiempo real la posición de vehículos, recursos y minas
- Detectar colisiones y áreas minadas con precisión

2. Sistema de Vehículos

- Simular el comportamiento específico de cada tipo de vehículo
- Implementar las restricciones de carga y número de viajes
- Calcular trayectorias óptimas considerando obstáculos

3. Algoritmos

- Algoritmos de navegación segura
- Estrategias de priorización de recursos basadas en valor/riesgo
- Sistemas de evasión de minas y vehículos enemigos

4. Sistema de Puntuación

- Calcular puntajes en tiempo real durante la simulación
- Determinar el ganador al finalizar todas las misiones
- Generar estadísticas detalladas de desempeño

5. Modo Replay

- Grabar y reproducir simulaciones completas
- Análisis frame-by-frame para depuración de estrategias
- **(Opcional)** Exportar estadísticas detalladas en formato CSV

Requerimientos Técnicos

Lenguaje y Estructura

- **Python 3**: Lenguaje de programación obligatorio
- **Archivo Principal**: `rescue_simulator.py`
- **Arquitectura Modular**: Separación clara entre lógica de juego, visualización y estrategias

Bibliotecas Permitidas

- Las bibliotecas permitidas serán detalladas en el canal #proyecto del Slack
- **Restricción**: Todas las estructuras de datos principales deben ser implementadas por el equipo

Persistencia de Datos

- **Configuraciones**: Guardado automático de mapas y estrategias
- **Resultados**: Historial de simulaciones y estadísticas
- **Recuperación**: Capacidad de reanudar simulaciones interrumpidas

Evaluación del Proyecto

Factores de Evaluación (100 puntos total)

1. **Comprensión Técnica**
    - Perfecto entendimiento del código por todos los integrantes

    - Capacidad de explicar decisiones algorítmicas
2. **Funcionamiento Correcto**
    - Simulación completa sin errores
    - Cumplimiento de todas las reglas del juego
    - Manejo adecuado de los distintos casos
3. **Calidad del Código**
    - Código limpio y bien documentado
    - Arquitectura modular y extensible
    - Manejo de errores robusto
4. **Algoritmos y Estructuras**
    - Elección apropiada de estructuras de datos
    - Implementación eficiente de los algoritmos necesarios
5. **Eficiencia**
    - Rendimiento óptimo en simulaciones grandes
    - Uso eficiente de memoria
    - Escalabilidad del código

Criterios Adicionales

- **Creatividad**: Bonificación por implementaciones innovadoras
- **Visualización**: Interfaz gráfica clara y atractiva
- **Testing**: Cobertura de pruebas unitarias

Estructura de Archivos

```
rescue_simulator/
├── rescue_simulator.py         # Archivo principal
├── config/
│   ├── default_config.json     # Configuración por defecto
│   └── strategies/
│       ├── player1_strategies.py # Estrategias del jugador 1
│       └── player2_strategies.py # Estrategias del jugador 2
├── src/
│   ├── game_engine.py          # Motor del juego
│   ├── aircraft.py             # Clases de aeronaves
│   ├── map_manager.py          # Gestión del mapa
│   ├── pathfinding.py          # Algoritmos de navegación
│   └── visualization.py        # Interfaz gráfica
├── data/
│   ├── simulations/            # Simulaciones guardadas
│   └── statistics/             # Estadísticas y análisis
└── tests/
    ├── test_aircraft.py        # Pruebas unitarias
    ├── test_pathfinding.py
    └── test_game_engine.py
```

Entrega y Demostración

Fecha de Entrega

- **Código completo**: [Fecha a definir]
- **Documentación**: Manual de usuario y técnico
- **Video demostración**: 10 minutos mostrando funcionalidades principales

Demostración Oral

- **Duración**: 30 minutos por equipo
- **Contenido**:
    - Explicación de arquitectura (10 min)
    - Demo en vivo (15 min)
    - Preguntas técnicas (5 min)

Formato de Entrega

- **Repositorio Git**: Código fuente versionado
- **README.md**: Instrucciones de instalación y uso
- **requirements.txt**: Dependencias del proyecto
- **Ejemplos**: Configuraciones de prueba incluidas
