II Ciclo 2025
EIF-207 Estructuras de Datos
Segundo Proyecto Programado

Courier Quest

1. Introducción

En esta segunda fase del proyecto, extenderán el juego **Courier Quest** desarrollado en la Parte 1 incorporando un **jugador CPU (inteligencia artificial)** que competirá contra el jugador humano para realizar entregas dentro de la ciudad.

El objetivo es diseñar e implementar tres niveles de dificultad de IA, cada uno utilizando diferentes técnicas de búsqueda, estructuras de datos y estrategias de decisión.

La información del mundo de juego (ciudad, pedidos, clima, etc.) se mantiene igual que en el primer proyecto:

https://tigerds-api.kindflower-ccaf48b6.eastus.azurecontainerapps.io

2. Objetivos de aprendizaje

- Aplicar estructuras lineales y no lineales (colas, árboles, grafos, colas de prioridad)
- Implementar algoritmos de decisión y búsqueda adaptados al contexto del juego
- Analizar la eficiencia de distintos enfoques de IA
- Desarrollar un agente autónomo que se comporte de manera coherente y competitiva

3. Jugabilidad (Gameplay)

El sistema debe incluir al menos **un jugador controlado por IA**, que:

- Reciba la misma información del mapa, clima y solicitudes de entrega que el jugador humano
- Pueda desplazarse por la ciudad, recoger y entregar pedidos siguiendo las reglas del juego
- Tenga su propia barra de resistencia, reputación y capacidad de carga
- Pueda ser configurado en **tres niveles de dificultad**: *fácil, medio, difícil*.

Dificultad	Descripción	Técnica sugerida	Conceptos
Fácil	El jugador CPU toma decisiones aleatorias	Elección aleatoria (Random Walk / Random Choice)	Listas, colas, control básico de movimiento
Media	La IA evalúa movimientos futuros en función de un puntaje heurístico	Búsqueda Greedy / Mini-max / Expactimax	Árboles, heurísticas, toma de decisiones
Difícil	La IA busca rutas óptimas entre entregas considerando costos y el clima	Algoritmos de grafos (Dijkstra, A*, BFS ponderado, TSP aproximado)	Grafos, colas de prioridad, planificación

Fácil — Heurística aleatoria

Objetivo

Lógica probabilística simple y al uso de colas

Comportamiento

- Elige un trabajo disponible al azar
- Elige una dirección aleatoria de calle adyacente en cada tick (evita edificios)
- Ocasionalmente, vuelve a lanzar el objetivo después de un tiempo límite o al completar una entrega

Medio — Evaluación ambiciosa o de estilo minimax

Objetivo

Introducir la evaluación de estados y una anticipación limitada

Comportamiento

- Mantiene un horizonte de anticipación pequeño (2-3 acciones por delante)

- Evalúa movimientos potenciales con una función de puntuación simple, por ejemplo:

  score = α*(expected payout) – β*(distance cost) – γ*(weather penalty)

- Selecciona el movimiento con la puntuación máxima

Opciones de implementación

"**Greedy best-first**" (más simple que el "minimax" completo, pero con la misma filosofía)

Minimax si se desea que se considere al jugador humano como un oponente que compite por las entregas

Consejo: en lugar de un árbol de juego completo, **expectimax** funciona muy bien y es mucho más fácil de programar

Difícil: Optimización basada en grafos (ruta más corta)

Objetivo

Aplicar algoritmos de grafos clásicos

Opciones de implementación

- Representar la cuadrícula de la ciudad como un grafo ponderado (peso de superficie como costo de arista)
- Usar Dijkstra o A* para la búsqueda de rutas
- Opcionalmente, integrar la replanificación dinámica:
  - Si el clima empeora, aumentar el coste de borde
  - Si la resistencia es demasiado baja, elegir desvíos más cortos
- Añadir lógica de decisión
  - Elegir la secuencia de entregas que minimice los desplazamientos y maximice las ganancias

4. Consideraciones

- El proyecto debe realizarse en grupos de máximo 3 personas. No se permite la entrega individual. Además, pueden trabajar con personas de otro horario

- El proyecto debe estar alojado en un repositorio de Github (a la hora de la entrega, solamente deben proporcionar el link donde está el código fuente del programa)

- Está permitido consultar documentación y pedir ayuda a un asistente de IA, pero deben registrar en un archivo de bitácora los **prompts utilizados** y las **modificaciones realizadas**

- Se penalizará la copia directa sin explicación del código

- El código debe cumplir las normas PEP8, y debe estar debidamente documentado utilizando Python Docstrings

- Fecha de entrega: Lunes 17 de noviembre a las 11:59pm

15. Criterios de evaluación (100%)

- Algoritmos y rendimiento (55%)
- Jugabilidad (30%)
- Código y Documentación (15%)