Búsqueda Informada (A*) – Simulación Interactiva con Costos Múltiples

Inteligencia Artificial

Autor: **Aaron Rodrigo Ramos Reyes**

Fecha de entrega: 29 de Octubre de 2025

1. Introducción

El objetivo de esta práctica es implementar y analizar el algoritmo de búsqueda informada **A* (A-star)** aplicado a un entorno simulado mediante **pygame**. El entorno se modela como un mapa de celdas donde cada celda puede representar un obstáculo o una zona con distintos costos de desplazamiento.

A diferencia de un A* tradicional que solo considera la distancia, este proyecto combina varios **criterios de costo** para encontrar rutas más realistas:

- **Tiempo**: zonas donde el desplazamiento es más lento (por ejemplo, terreno difícil o tráfico).
- **Peligro**: áreas que representan mayor riesgo o dificultad.
- **Escénico**: zonas agradables o estéticamente preferibles.

El propósito del programa es permitir al usuario manipular los pesos de estos criterios y observar, de manera interactiva, cómo cambia la ruta óptima en tiempo real.

2. Descripción general del programa

El programa, desarrollado en **Python** usando la biblioteca **pygame**, genera un mapa bidimensional de tamaño configurable. Cada celda del mapa puede tener tres capas de información de costo:

1. **Mapa base de ocupación**: define qué zonas son transitables (color gris oscuro o negro = obstáculo).
2. **Mapa de tiempo**: determina la lentitud de las zonas (azul).
3. **Mapa de peligro**: representa áreas de alto riesgo (rojo).
4. **Mapa escénico**: zonas agradables visualmente (verde).

El algoritmo A* utiliza una función de costo compuesta:

$$f(n) = g(n) + h(n)$$

donde:

$$g(n) = \text{distancia} \times (1 + w_t \cdot C_t + w_p \cdot C_p + w_e \cdot C_e)$$

y

$$h(n) = \text{distancia euclidiana al objetivo}$$

con:

- C_t: costo de tiempo en la celda destino.
- C_p: costo de peligro.
- C_e: costo escénico (en realidad, penalización $1 -$ belleza).
- w_t, w_p, w_e: pesos ajustables para cada tipo de costo.

La heurística utilizada es la **distancia euclidiana**, adecuada para movimientos con 8 vecinos.

3. Cómo ejecutar el programa

3.1. Requisitos

- Python 3.10 o superior.
- Librerías: `pygame`, `numpy`, `pillow`.

3.2. Instalación

Desde la terminal, ejecutar:

```
pip install pygame numpy pillow
```

3.3. Ejecución

Guardar el archivo `astar_interactivo.py` y ejecutar:

```
python astar_interactivo.py
```

Aparecerá una ventana con el mapa, el punto de inicio (verde), el destino (amarillo) y la ruta (roja). La parte inferior mostrará los pesos actuales de cada criterio.

4. Controles del entorno interactivo

Tecla / Acción	Descripción
T / G	Aumentar / disminuir peso del **tiempo**.
Y / H	Aumentar / disminuir peso del **peligro**.
U / J	Aumentar / disminuir peso del **escénico**.
R	Regenerar mapa aleatorio.
Click izquierdo	Fijar nuevo punto de **inicio**.
Click derecho	Fijar nuevo **objetivo**.
ESC	Salir del programa.

Cada vez que se modifica un peso o se cambia un punto, el algoritmo A* recalcula automáticamente la nueva ruta óptima según los criterios activos.

5. Interpretación de los colores

El entorno utiliza colores para representar tanto los mapas de costo como la ruta:

- **Negro / Gris oscuro**: obstáculos o zonas no transitables.
- **Azul**: zonas lentas (alto costo de tiempo).
- **Rojo**: zonas peligrosas (alto riesgo).
- **Verde**: zonas escénicas (más agradables, bajo costo estético).
- **Rojo brillante**: ruta óptima actual.

- **Verde brillante**: punto de inicio.
- **Amarillo**: destino.

La intensidad de cada color depende del peso actual asignado a ese tipo de costo. Por ejemplo, si el peso de "peligro" es alto, las zonas rojas se vuelven más influyentes y la ruta tenderá a evitarlas.

6. Capturas y análisis

A continuación se presentan capturas de pantalla que muestran cómo cambia la ruta según los pesos de los costos:

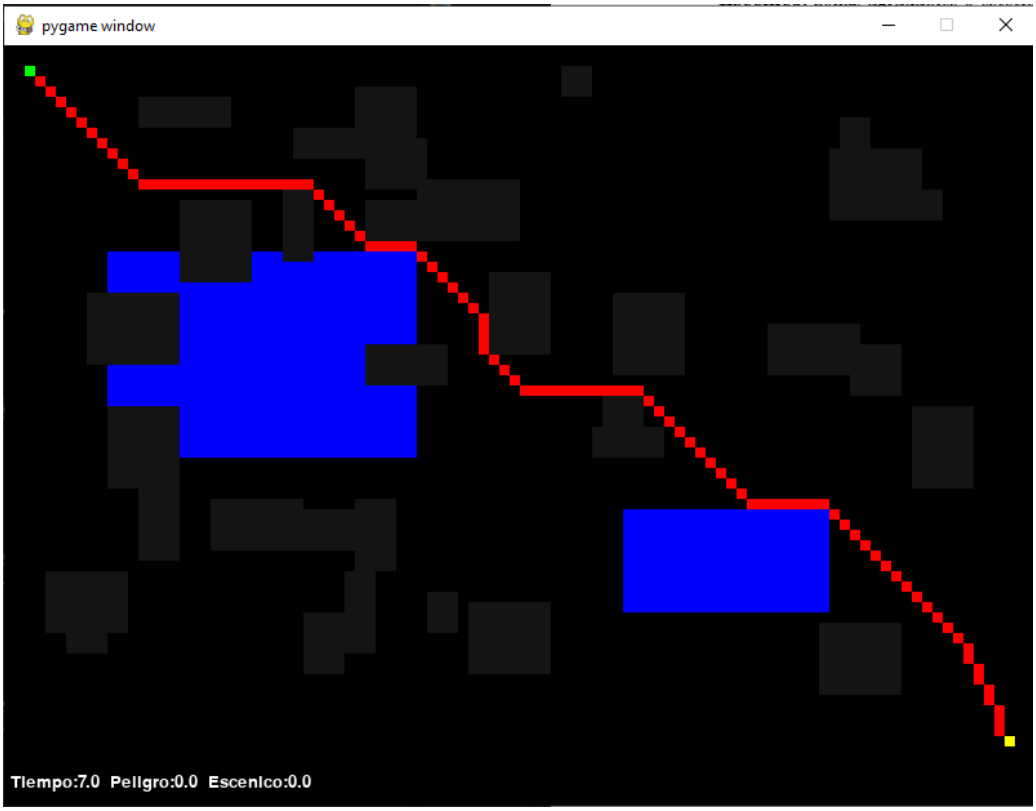

Figura 1: Ruta con peso de tiempo alto (w_t grande). La ruta evita zonas azules lentas.

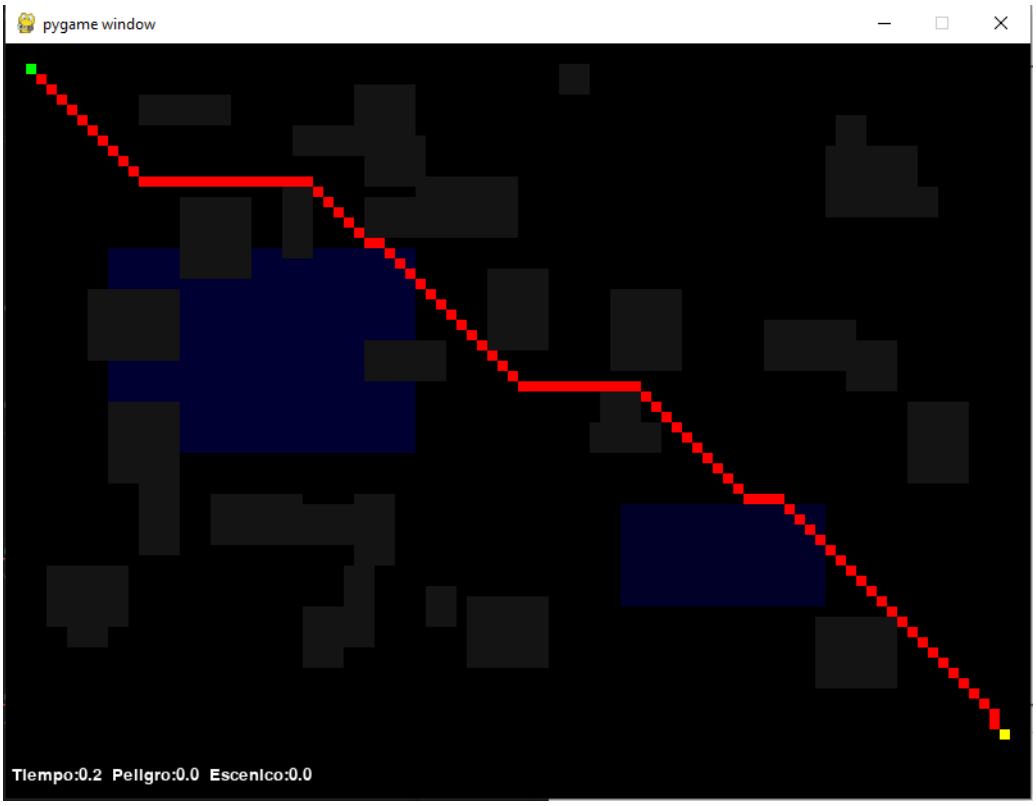

Figura 2: Ruta con peso de tiempo bajo. El agente atraviesa zonas lentas.

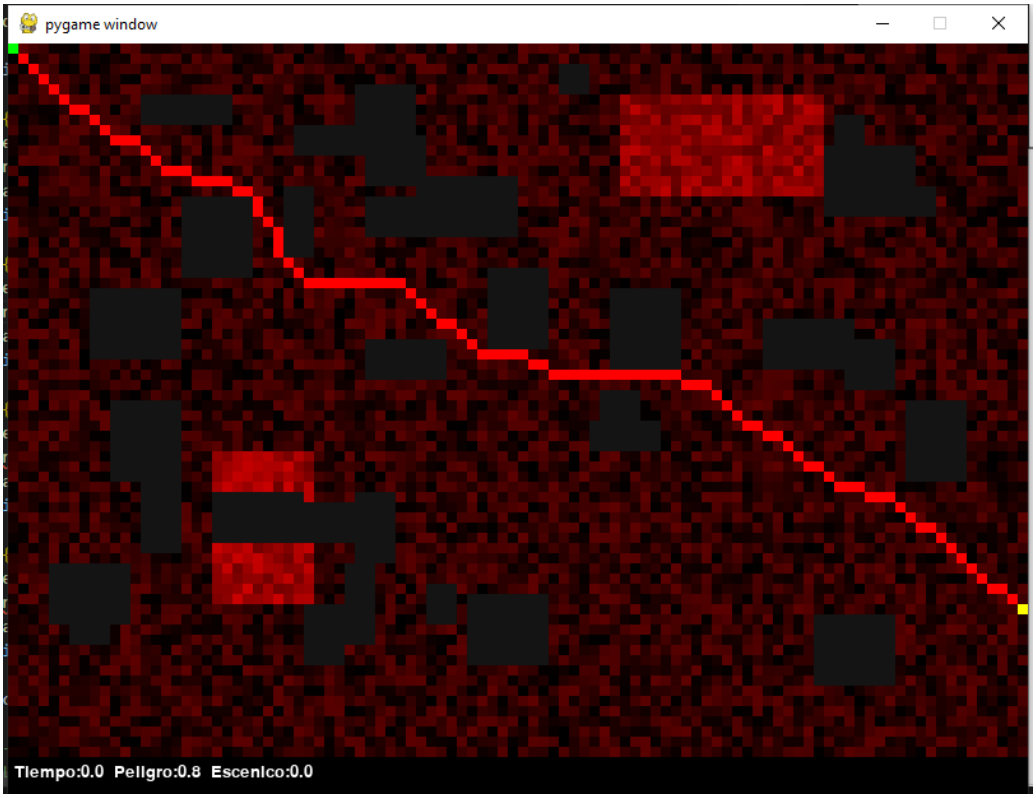

Figura 3: Ruta con peso de peligro alto. La trayectoria evita zonas rojas.

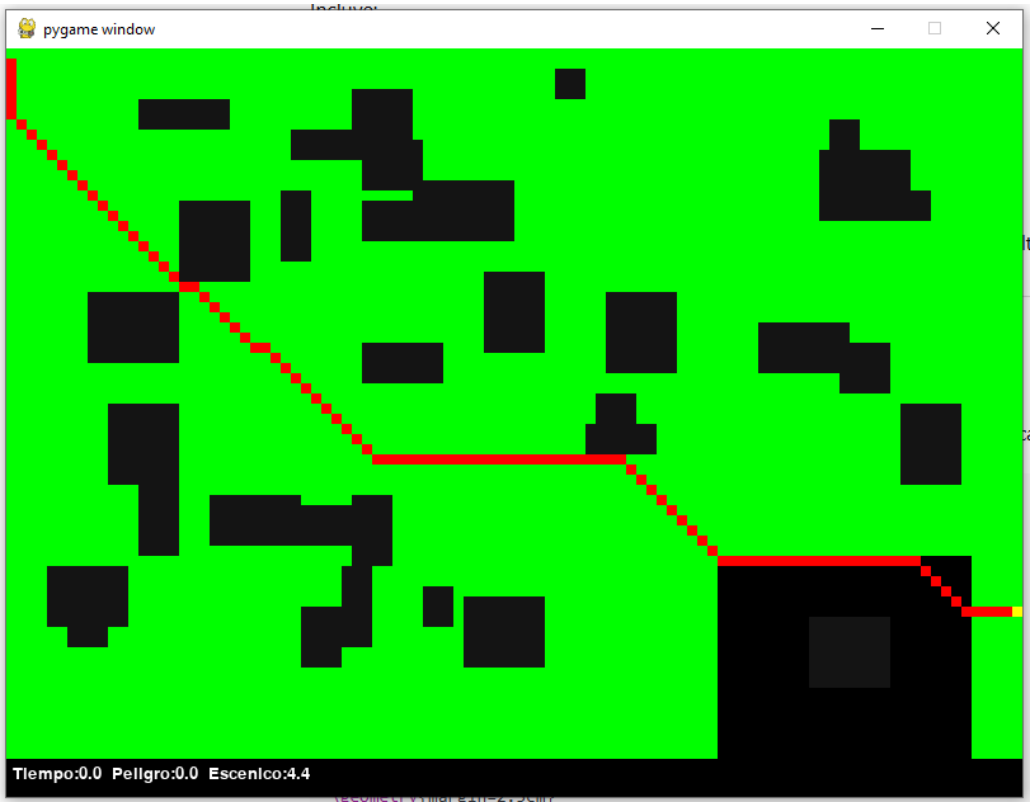

Figura 4: Ruta con peso escénico alto. Se favorecen las zonas verdes agradables.

Estas capturas demuestran cómo el algoritmo A* combina múltiples criterios para producir rutas óptimas de acuerdo con los valores establecidos por el usuario.

7. Conclusiones

El experimento permitió observar de manera visual e interactiva cómo el algoritmo A* puede adaptarse a entornos complejos con múltiples factores de costo.

Se comprobó que:

- Aumentar el peso de un criterio influye directamente en el trayecto seleccionado.
- La combinación ponderada de costos genera comportamientos realistas y ajustables.
- El uso de pygame facilita la comprensión intuitiva del funcionamiento de A* en entornos dinámicos.

En síntesis, la versión interactiva demuestra cómo la búsqueda informada puede extenderse más allá de la simple distancia, incorporando criterios contextuales como seguridad, velocidad o preferencia estética.

Fin del documento.