Informe del Proyecto: Laberinto Saltarín

Introducción

Este proyecto implementa una solución al problema del laberinto saltarín, donde un agente debe encontrar el camino más corto desde un punto de inicio hasta un destino, moviéndose una cantidad fija de pasos definida por el valor de la celda en la que se encuentra.

Objetivo

El objetivo de este proyecto es implementar dos algoritmos de búsqueda (DFS y Búsqueda de Costo Uniforme) para encontrar el camino más corto en un laberinto saltarín y comparar su eficacia y eficiencia en diferentes laberintos.

Estructura del Proyecto Laberinto Saltarín

```
LaberintoSaltarin/
├── data/  # Archivos de entrada con laberintos
│   ├── entrada.txt
│   ├── entrada2.txt
│   └── entrada3.txt
└── src/
    ├── agent.py # Clase que representa al agente
    ├── game.py  # Clase que representa el juego
    ├── main.py  # Punto de entrada del programa
    ├── maze.py  # Clase que representa el laberinto
    ├── parseador.py # Clase que se encarga de parsear el archivo de entrada
    └── __pycache__/
├── .gitignore
├── README.md
└── Tarea 1 2024.pdf
```

Instrucciones de Ejecución

Para ejecutar el proyecto, sigue estos pasos:

1. Asegúrate de tener instalado Python y Pygame. Si necesitas instalar Pygame, puedes hacerlo ejecutando `pip install pygame` en tu terminal.
2. Navega hasta la carpeta `src/` donde se encuentra el archivo `main.py`.
3. Ejecuta el archivo `main.py` usando Python desde tu terminal con el siguiente comando:

```
python main.py
```

Descripción de los Algoritmos

Búsqueda en Profundidad (DFS)

- **Funcionamiento**: DFS explora el laberinto de manera exhaustiva, retrocediendo solo cuando se encuentra con un callejón sin salida.
- **Adecuación**: Aunque no garantiza encontrar el camino más corto, DFS es útil en laberintos sin caminos largos o donde el factor de ramificación es bajo.

Búsqueda de Costo Uniforme

- **Funcionamiento**: Este algoritmo explora los caminos de menor costo primero, asegurando que el primer camino que llega a la meta es el más corto.
- **Adecuación**: Es efectivo para garantizar la solución más óptima en términos de número de pasos, ya que todos los movimientos tienen el mismo costo.

Desafíos y Soluciones

- **Coordenadas**: Uno de los mayores desafíos fue manejar correctamente las coordenadas (x, y) en todas las operaciones.
- **Solución**: Aseguramos que todas las funciones respetaran un sistema de coordenadas consistente, pasando siempre coordenadas como (y, x) excepto en las operaciones de dibujo.

Ejemplos de Entradas y Salidas

- **Entrada**: Un laberinto donde el agente comienza en (0,0) y la meta está en (3,3).
- **Salida**: El agente encuentra el camino en 7 movimientos utilizando la búsqueda de costo uniforme.

Análisis de Rendimiento

- **DFS**: Rápido en laberintos pequeños pero ineficiente en laberintos grandes debido a su naturaleza exhaustiva.
- **Búsqueda de Costo Uniforme**: Más lento pero garantiza encontrar el camino más corto, ideal para laberintos donde cada paso tiene el mismo costo.

Conclusión

Este proyecto demostró la eficacia de diferentes algoritmos de búsqueda en el contexto de un laberinto saltarín y destacó la importancia de una heurística adecuada para mejorar la eficiencia de la búsqueda.

Apéndices

- Código fuente completo.
- Capturas de pantalla del juego en ejecución.

```
```