Tarea: PathFinding

1. Complejidad de y Explicación de Cada Método de Búsqueda

1.1. Dijkstra

El algoritmo de Dijkstra es uno de los métodos más conocidos para encontrar el camino más corto en un grafo ponderado. Su complejidad en el peor de los casos es $O((V+E)\log V)$, ya que, en el peor de los casos, visitamos todos los nodos. Aunque no es óptimo, ya que no hay nada que nos impida ir por ramas que obviamente no son válidas, el costo de visitar los vértices es V, y el costo de acceder a cada vértice es $\log V$ si tratamos la cola de prioridad como un heap (como se mostró en clase).

Comenzamos desde el origen, asignamos todas las distancias a infinito y ponemos el destino en la cola de prioridad. Luego, iterativamente, calculamos la distancia de sus vecinos en base al peso del nodo más el peso de la arista (el peso del nodo es cuánto costó llegar del `src` hasta él). Si la nueva distancia es menor a la que ya tenía el nodo, lo colocamos de nuevo en la cola de prioridad. Visitamos todos los nodos hasta llegar al destino y, al final, obtenemos el peso del destino.

1.2. Greedy BFS

El algoritmo de búsqueda primero en amplitud (BFS) con enfoque codicioso (Greedy BFS) es una versión más simple y rápida, pero no siempre garantiza la respuesta correcta. Este método de búsqueda se basa fundamentalmente en el uso de heurísticas, específicamente la heurística euclidiana (que se explica más a fondo en A*), para decidir el siguiente nodo a explorar.

Se empieza desde el nodo `src` y, en base a la heurística, se añaden los nodos a la cola de prioridad, avanzando siempre por el nodo cuya heurística sea menor respecto al destino.

1.3. A*

El algoritmo A* es una combinación del Dijkstra previamente mencionado con el Greedy BFS. Aunque es más lento que el Greedy BFS, A* es mucho más certero. En este caso, para calcular el peso de los nodos, el algoritmo primero calcula la distancia al `src` usando el mismo método de comparación de distancias (el peso del nodo más el peso de la arista), pero a la hora de guardar los nodos en la cola de prioridad, combina la heurística del Greedy BFS.

En este caso, utilizamos la heurística euclidiana, es decir, usamos el teorema de Pitágoras con las coordenadas de los nodos para calcular la distancia en línea recta hacia el destino, lo que nos permite decidir cuál será el siguiente nodo a visitar.

1.4. Implementación Práctica de A*

En la práctica, esta versión de A* se formaliza en el método `a_star` de `PathFindingManager`. En este método, cada nodo mantiene un `g-score` (la distancia acumulada real) y se calcula un `f-score`, que es la suma de la heurística euclidiana (implementada en `euclid()`) antes de insertarlo en la cola de prioridad (`priority_queue<Entry>`).

Cada entrada `Entry` encapsula el vértice, su `g-score` y la prioridad (`f-score`), de modo que el `min-heap` extrae siempre el nodo con el menor costo estimado total. Para evitar reprocesar vértices ya cerrados, se utiliza un conjunto no ordenado `unordered_set<Node*>visited`, y la exploración de vecinos se filtra respetando la dirección de las aristas (`one_way`). De esta forma, garantizamos la optimalidad sin ciclos innecesarios.

Además, se añaden optimizaciones en la visualización: solo cada 50 iteraciones se repinta la ventana y se dibujan las aristas exploradas, lo que aligera el renderizado sin sacrificar la lógica interna. Finalmente, el resultado se reconstruye con `set_final_path`, que retrocede por el mapa `parent` hasta el origen.

2. Integrantes del Grupo

- Gianfranco Cordero Aguirre (202310325)
- David A. Salaverry Cuzcano (202310285)
- Federico Iribar Casanova (202319321)
- José Huamaní (202310632)