COS30002 Lab 5 Notes: Graphs, Paths & Search

Marco Giacoppo - COS30002 - Task 05

🔧 Setup & Initial Exploration

- Used the updated files provided on 24-03-2024

- Installed Pyglet using pip to ensure GUI functionality

- Launched the program and experimented with map editing, search depth control, and switching between algorithms

🧠 Initial Observations (Unmodified Code)

- DFS always follows a fixed path regardless of tile cost. It dives deep in one direction before checking others, often producing long or inefficient paths.

- BFS explores layer by layer and typically finds the shortest path in terms of steps, but it doesn't care about tile cost.

- Dijkstra's algorithm avoids expensive tiles like mud and water, even if it means taking a longer route to reduce overall cost.

- A* finds a balance between short paths and low cost. However, with the original min_edge_cost value, it prioritizes path length more than cost, sometimes choosing faster but more expensive paths.

🔍 Comparison of Search Methods

- DFS: Ignores cost completely, long and inefficient paths, always similar shape.

- BFS: Prioritizes shortest path in terms of steps, doesn't consider tile cost at all.

- Dijkstra: Fully cost-based, prefers longer routes with cheaper tiles.

- A*: Hybrid approach, uses both actual and estimated cost. More efficient than Dijkstra when the heuristic is accurate.

🛠️ Fixes & Modifications

✅ Fixed mouse interaction bug caused by float indices in `get_box_by_pos()` by casting the index to int.

✅ Changed `min_edge_cost` from 10.0 to 1.0 in `box_world.py` to fix incorrect A* behavior.

✅ Uncommented diagonal edge logic and switched the heuristic to `_hypot` to allow accurate diagonal pathfinding.

📈 A* Cost Tuning – Observations

Before changing `min_edge_cost`, A* heavily favored short paths even if expensive tiles were involved. After setting it to 1.0, the algorithm began preferring slightly longer but cheaper routes.

In one test (map1), placing start at 0 and target at 5 with mud at positions 3 and 4:

- Original min_edge_cost = 10.0 → Path: [0,1,2,9,10,5], Cost: 5.8284, Steps: 6

- New min_edge_cost = 1.0 → Path: [0, 1, 2, 8, 9, 10, 11, 5], Cost: 7.0, Steps: 11

This proves that the updated cost makes A* more sensitive to expensive tiles.

📐 Diagonals & Heuristics

Enabling diagonals adds realistic 8-directional movement, with diagonal edges costing ~1.41. A* and Dijkstra both use this to avoid extra clear tiles when a diagonal is more efficient.

Originally, A* used Manhattan distance, which assumes no diagonal paths. This was replaced with the `_hypot` heuristic to match the new movement model.

Heuristic Overview:

- `_manhattan`: Suitable for N/S/E/W movement. Calculates |x1 - x2| + |y1 - y2|.

- `_hypot`: Uses Pythagorean theorem. Best for 8-directional movement with diagonal edges.

- `_max`: Chebyshev distance. Works well when diagonal and straight move costs are equal.

✅ Final Thoughts

This lab gave me a clear understanding of how different search algorithms behave in practice. Modifying the heuristic and topology made it clear how much influence the cost model has on pathfinding performance.