Report

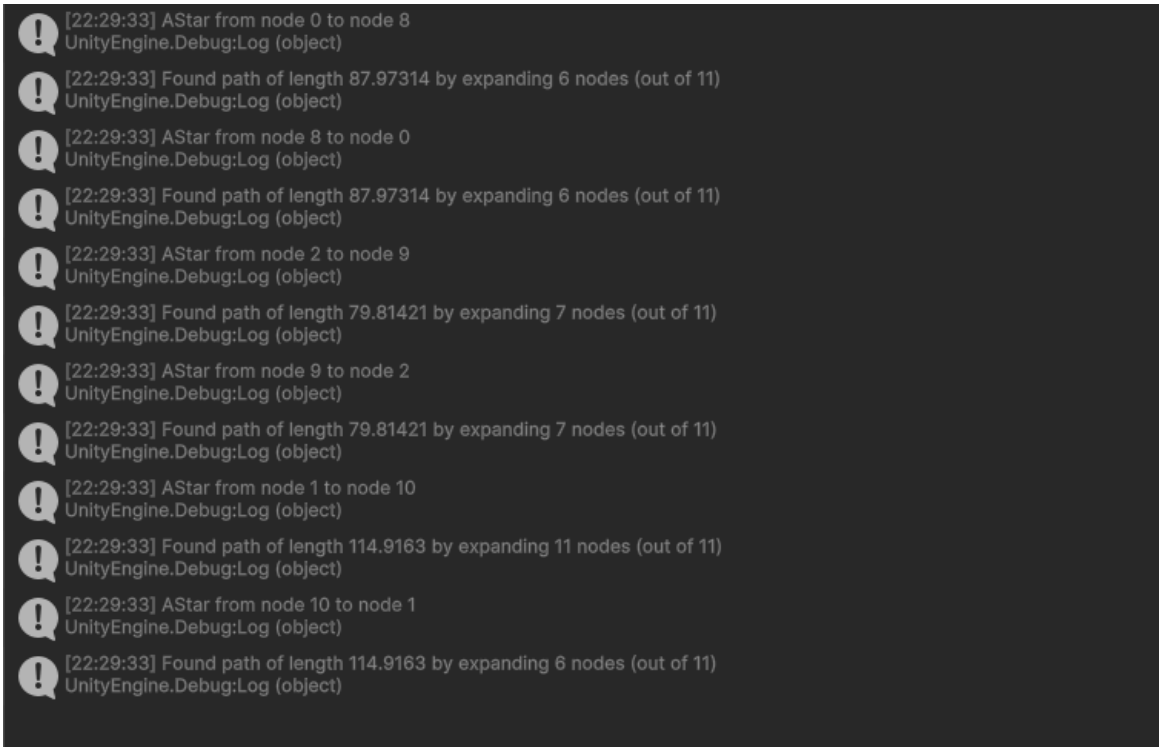
Mihai Lache and Anuar Romo

Briefly explain the outline of your A* implementation

Our A* implementation keeps track of which nodes have already been seen and which ones we still need to look at. We used a regular list instead of a priority queue and looped through it each time to find the node with the best total cost (formula we used: *distance so far + estimated distance to the goal*). For each node, we look at its neighbors, and update their costs if we find a better way to get to them, and keep track of where we came from and what wall connects them. Once we reached the destination, we traced our steps back through the nodes and collected the midpoints of the walls we passed through. Then we reverse that path and add the final target point to the end.

Did it find the optimal paths on the standalone test cases?

Yes. The logs in the Unity console matched all the expected path lengths exactly. Here is a screenshot of the results passing the test cases:



Did you add any of your own test maps, or standalone test cases? What do they look like?

No, we just focused on getting everything working with the provided maps. They already covered the main edge cases and gave us enough variety to test the algorithm well.

Did you encounter any particular challenges during the assignment?

- At first, our code wasn't actually using the midpoints, and the car was trying to walk through walls. Took a bit to realize we had to use neighbor.GetWall().midpoint to build the path properly.
- Sometimes we were storing extra or wrong points in the path because we didn't reverse it or forgot to include the final target. We fixed that by walking backwards through the cameFrom dictionary and reversing it at the end.
- Earlier versions would mess up if we clicked on a new target without restarting because the variables didn't reset. Once we re-initialized everything fresh in AStar, that problem went away.

Retrospective

Mihai: I focused on the logic for picking the best next node during the search, managing the dictionaries to keep track of paths, and putting together the final list of midpoints. I also handled some of the distance math and made sure the cost calculations made sense. It helped me understand how A* works step by step and what can break it if you're not careful.

Anuar: I also worked through the logic for choosing the next best node in the A* loop and making sure the dictionaries were updating correctly. I got a better understanding of how A* ties together and how important it is to track where you came from and how the cost values affect the search. I also troubleshooted the bugs we had when something wasn't working.

We used ChatGPT during debugging to help understand why certain parts weren't working as expected / understanding Unity