For comparing the algorithms, we used the map in Image 1. There are two points, green on (191,0,191), as the start goal and red on (-112,0,106) as the goal. All algorithms were ran with the same setup:

* The mesh has 7879 nodes (only considering NavMeshEdge as those are the only ones used by the algorithms).
* 30 nodes analysed per frame.
* Each experiment was made 5 times and an average was calculated.

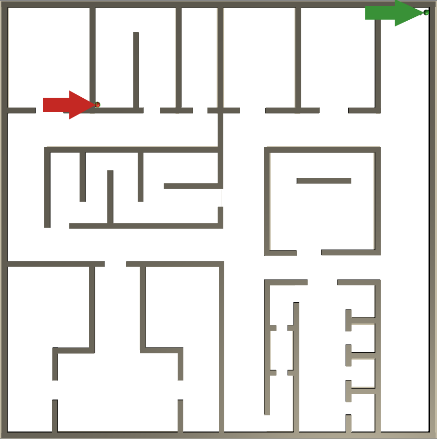


Figure 1 Map

This are the stats gathered:

|  |  |
| --- | --- |
| **A\*** | |
| Nodes Explored | 6860 |
| Fill | 87.07% |
| Total Time (ms) | 350,4 |
| Time/ node (ms) | 0.0601 |
| Max Open Size | 166 |

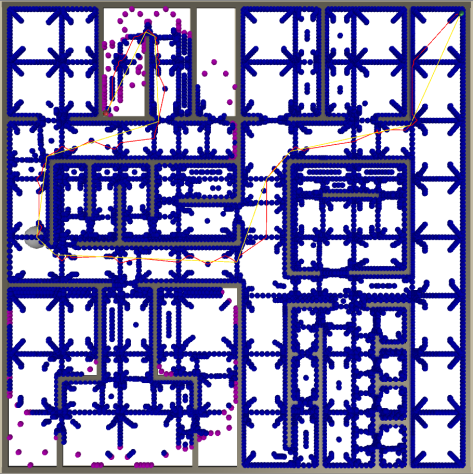


Figure A\*

|  |  |
| --- | --- |
| **NodeArrayA\*** | |
| Nodes Explored | 6860 |
| Fill | 87.07% |
| Total Time (ms) | 159,34 |
| Time/ node (ms) | 0.0230 |
| Max Open Size | 166 |

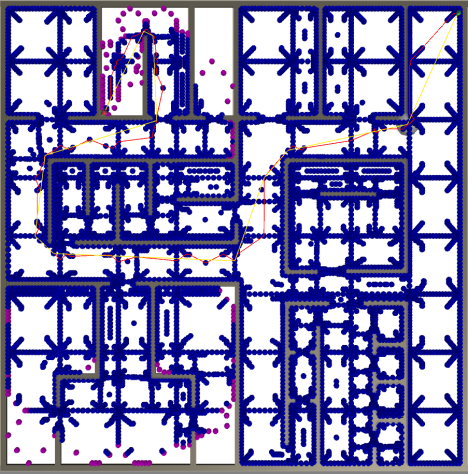


Figure NodeArrayA\*

|  |  |
| --- | --- |
| **GoalBoundsA\*** | |
| Nodes Explored | 142 |
| Fill | 1.80% |
| Total Time (ms) | 3.056 |
| Time/ node (ms) | 0.0215 |
| Max Open Size | 5 |
| Discarded Edges | 744 (76%) |
| Visited Edges | 230 (24%) |
| Total Edges | 974 |

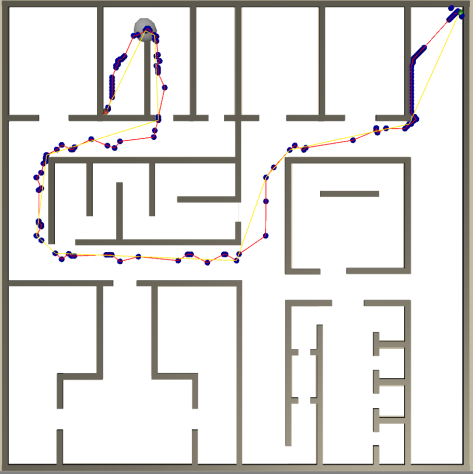


Figure 4 GoalBoundA\*

The difference between A\* and NodeArrayA\* is a trade off between memory and time, therefore ignoring the time stats they have the same values (as it is the same algorithm running, but with different data structures). However due to the efficiency of the NodeArray we get a time improvement of 120%. The main advantage comes from the time it takes to access a node( O(1) in the NodeArrayA\* vs O(n) in the A\*).

There is a huge run time improvement between GoalBounding and the previous two algorithms (11365% and 5102% respectively). The gigantic improvement comes with a cost though, which is the pre-processing of the scene, storing it in the disk and later loading it to memory. This provides us with the foresight of knowing which directions will never lead us to the goal optimally, thus allowing the algorithm to ignore them. This results in the fill property to be much much smaller (87.07% vs 1.80%) In the Figures 2, 3 and 4 we can see the fill property graphically (In blue the Visited Nodes and Purple the open Nodes). We are able to realise that in GoalBound the algorithm almost only explores the nodes included the optimal path to reach the goal.

**Path Smoothing**

The algorithm used was for any three sequential nodes (starting in the start