Good Catch User Guide

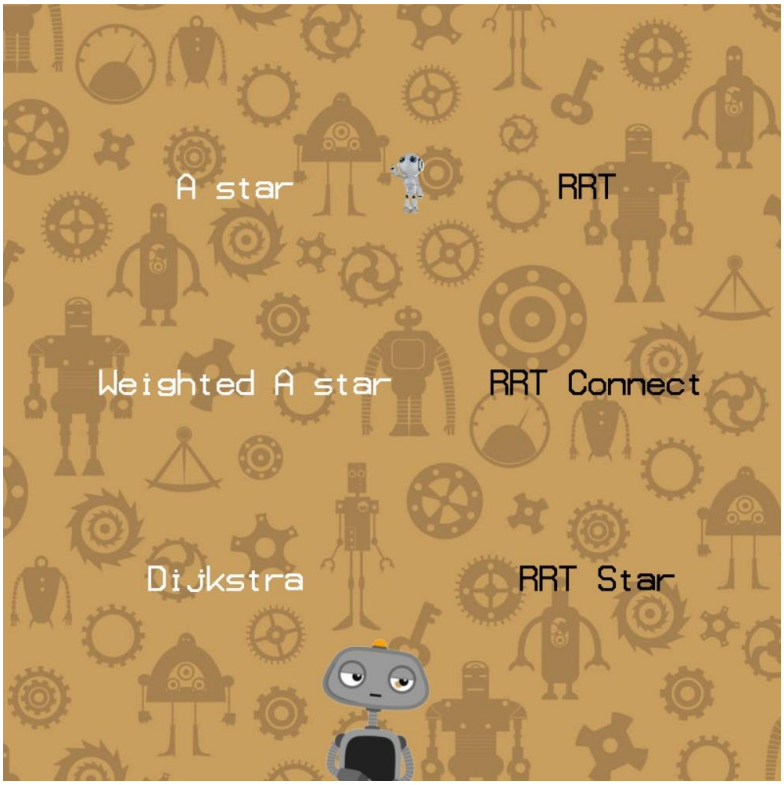
Product Name: Good Catch
Team Name: AStar
Team Members: Sai Abhishek Siddhartha Namburu, Ninaad Damis, Vishnu MH, William Lee, Aishwarya Ravi

Introduction:
Path Planning basically implies finding the shortest possible distance between the start and end points, usually used in the context of autonomous mobile robots. There are two main components necessary for path planners, first is the world or the map of the environment which needs to be traversed to find the optimal distance and the second is awareness of location of the robot in the map.

In this product, two types of path planning are implemented, Grid Based Planning and Sampling based Planning. For Grid Based Planning AStar, Weighted AStar and Djikstra algorithms are implemented and visualized, while for Sampling Based planners.

Rapidly exploring random trees (RRT), exploring random trees connect (RRT connect), Rapidly exploring random tree star (RRT*) are implemented.

UI: A Screen pops up to select the planner, select the planner as desired.

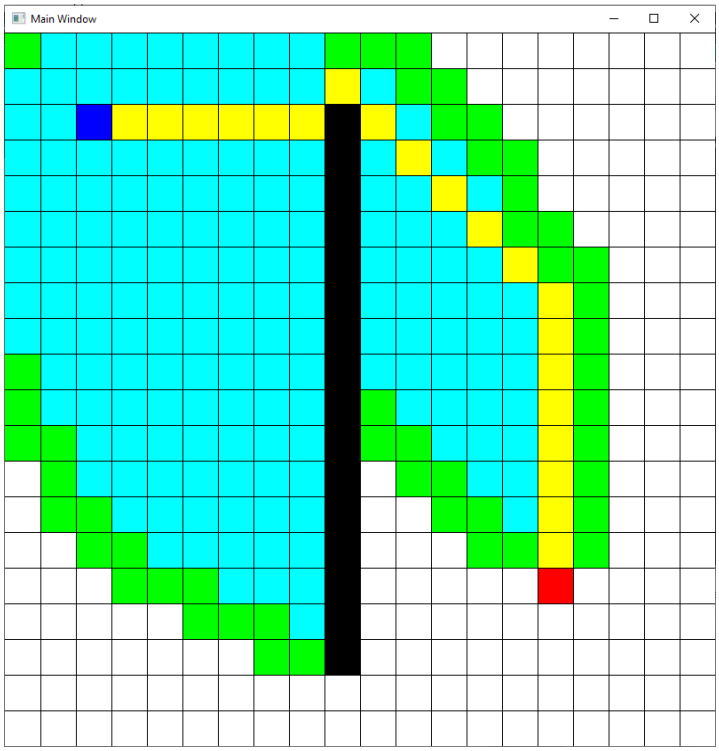


GRID BASED PLANNERS:

1. **A-Star**

   A-Star path planning algorithm utilizes a heuristic such as Euclidean distance, in this case, to compute the estimate from the current vertex to the end position and sums it with the cost from the starting node to a particular vertex to determine the optimal path.

   1) Select A*.
   2) A lattice map opens up. Press S to select the start position in the grid, it will be marked in blue.
   3) Press E to select the final position in the grid, it will be marked in red.
   4) Press and hold O to select the obstacle points in the map. You can select any number of obstacles and it will be marked in black.
   5) Press Enter to begin the search.
   6) The pixels that appear as green indicates the open list, blue indicates the closed list and the final optimal path will turn up as yellow.

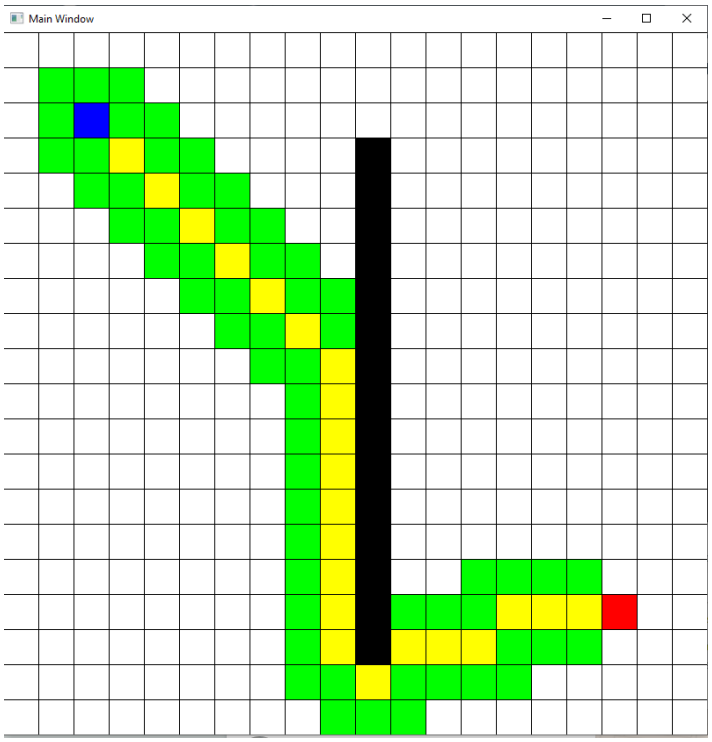


A* Visualization

2. **Weighted A-Star**

   Weighted A-Star path planning algorithm is the same as the A-Star implementation, however there is one thing that's different, which is that the f-cost and the g-cost are multiplied by a certain weight to calculate the total cost, which performs the search quicker by giving up the optimality.

1)Select Weighted A*.
2)A lattice map opens up. Press S to select the start position in the grid, it will be marked in blue.
3)Press E to select the final position in the grid, it will be marked in red.
4)Press and hold O to select the obstacle points in the map. You can select any number of obstacles and it will be marked in black.
5)Press Enter to begin the search.
6)The pixels that appear as green indicates the open list, blue indicates the closed list and the final optimal path will turn up as yellow.



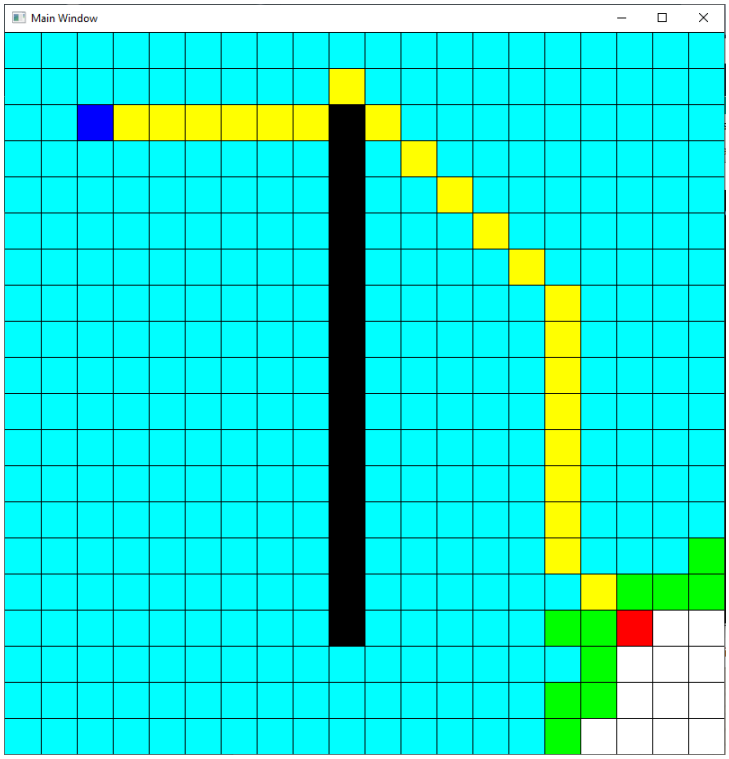
Weighted A* Visualization

3. Djikstra

This planner does not use a heuristic to perform the search. It checks for the edge cost of every neighbor of the node and finds the lowest cost amongst them all to then proceed to do the search from that node to the corresponding neighbors. Finally the optimal path is found by traversing the nodes with the lowest edge cost.

1)Select Djikstra by pressing.
2)A lattice map opens up. Press S to select the start position in the grid, it will be marked in blue.
3)Press E to select the final position in the grid, it will be marked in red.

4)Press and hold O to select the obstacle points in the map. You can select any number of obstacles and it will be marked in black.
5)Press Enter to begin the search.
6)The pixels that appear as green indicates the open list, blue indicates the closed list and the final optimal path will turn up as yellow.



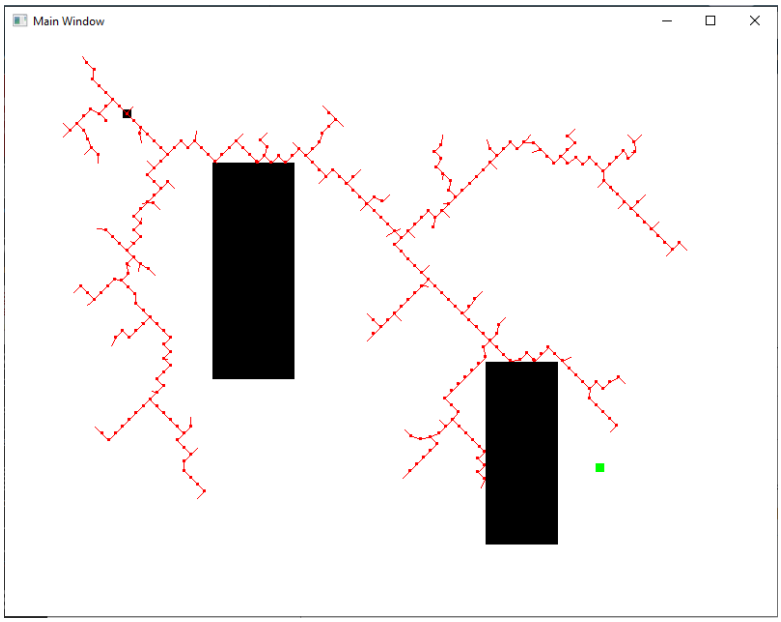
Dijkstra Visualization

SAMPLING BASED PLANNERS:

1. **RRT**

This path planning algorithm is utilized in the case where the grid based planners become computationally expensive like in the case when the searching space is large. RRT uses an iterative method to traverse a large search space using a randomized approach wherein it connects the initial vertex node to a random point in space while keeping track of the cost-to-start and biasing those nodes that are closer to the target.

1)Select RRT.
2)An empty map opens up. Press S to select the start position in the map, it will be marked in black.
3)Press E to select the final position in the map, it will be marked in green.
4)Press O to get the left top corner of an obstacle, Press t to get the right bottom corner of the same obstacle. The obstacle is then displayed on the map
5)Press Enter to begin the search.
6)The tree is connected by means of nodes displayed in red and the optimal path is found.



RRT visualization

2. **RRT***

RRT* is same as RRT with two main differences being recording the cost of the vertex from the parent vertex and rewiring performed amongst the neighbors to make sure that the next node being connected has a reduced cost. In this manner this method finds the shortest path.

1)Select RRT*.

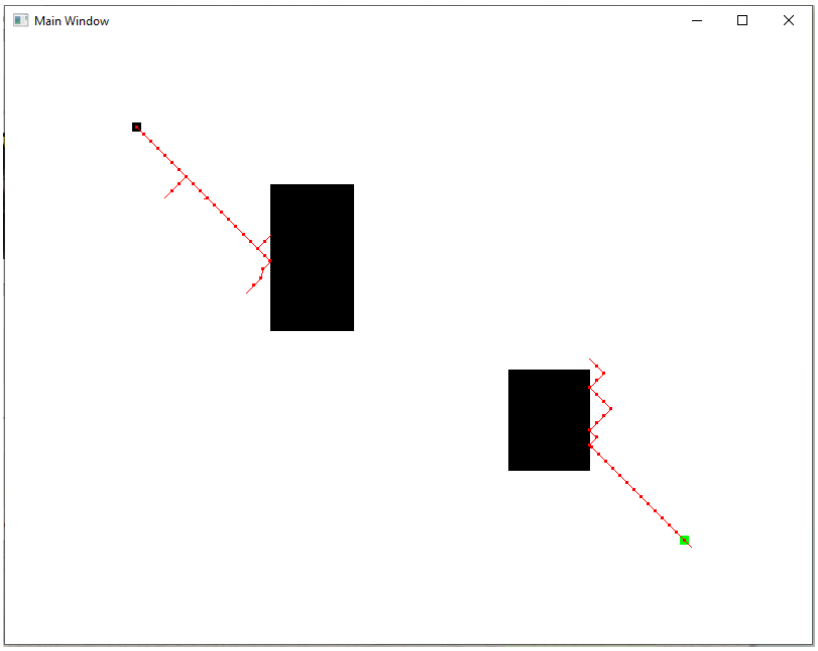
2)An empty map opens up. Press S to select the start position in the map, it will be marked in black.

3)Press E to select the final position in the map, it will be marked in green.

4)Press O to get the left top corner of an obstacle, Press t to get the right bottom corner of the same obstacle. The obstacle is then displayed on the map.

5)Press Enter to begin the search.

6)The tree is connected by means of nodes displayed in red and the optimal path is found.



RRT connect visualization

3. RRT Connect

RRT connect is also performs the same functions as RRT but also grows the tree from the End towards the start without the distance epsilon restriction and tries to connect both the trees at every iteration and visualizes the search of both the trees and the optimal path.

1)Select RRT.

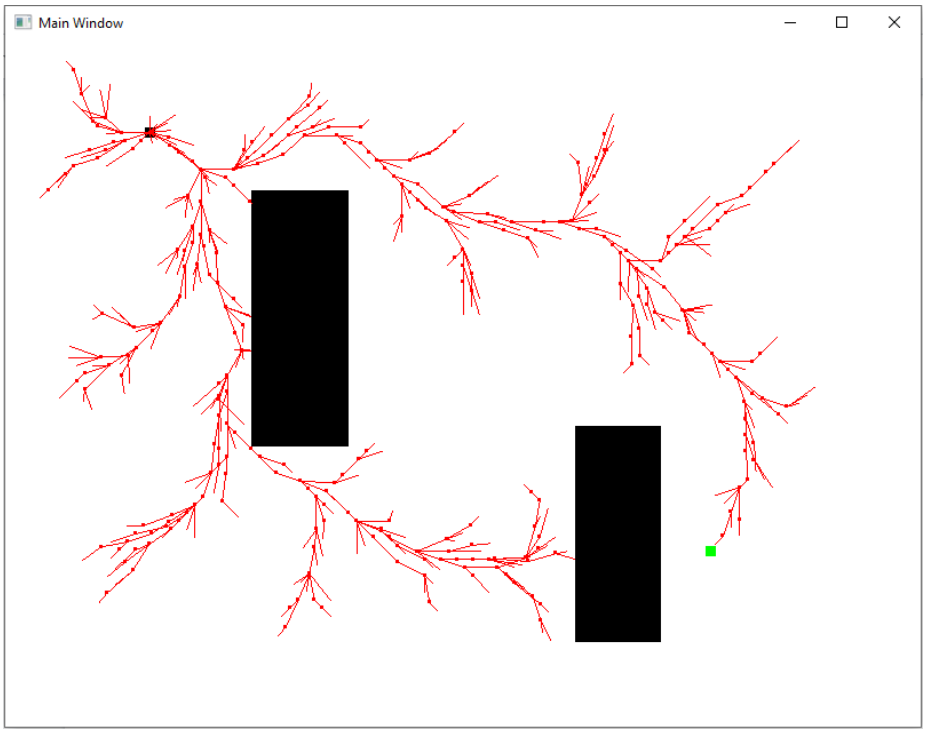
2)An empty map opens up. Press S to select the start position in the map, it will be marked in black.

3)Press E to select the final position in the map, it will be marked in green.

4)Press O to get the left top corner of an obstacle, Press t to get the right bottom corner of the same obstacle. The obstacle is then displayed on the map

5)Press Enter to begin the search.

6)The tree is connected by means of nodes displayed in red and the optimal path is found.



RRT Star visualization

Compiling Instructions:

→The executable rrtmain.exe can to be run.
→Add the following compiling files to the project in order to compile it :
cl rrtmain.cpp rrt.cpp rrtconnect.cpp rrtstar.cpp visualize_rrt.cpp astar.cpp map.cpp GridVisual.cpp fssimplewindow.cpp yssimplesound.cpp ysglfontdata.c yspng.cpp yspngenc.cpp
→Press the play button “Local Windows Debugger” to compile and run the project.

Libraries:

➔ **Predefined Libraries**

- Stdio
- Math
- Iostream
- List
- Set
- Cstdlib
- Vector
- Float
- Stack
- Limits
- Queue
- chrono

➔ **YS Libraries**

- fssimplewindow
- yssimplesound
- ygslfontdata

➔ **Custom Libraries**

- astar
- rrt
- rrtconnect
- rrtstar
- visualize_rrt
- GridVisual
- Map