Hybrid A\* algorithm:

Function roundState:

Responsibility: Herman

Use: Rounds a continuous state to a cell in the search space.

Functions exists:

If the continuous state of the successor rounds to the continuous state of the predecessor, we should return true, otherwise false.

Start algorithm(start state, goal state):

1. Initialize open list
2. Initialize closed list
3. Predecessor(start state) = null
4. Push start state onto open list
5. While open list is not empty
6. Pop least cost node in open list
7. Push the node into the closed list
8. If the roundstate of the node is the same as the roundstate of the goal node:
9. Return node (this node will contain pointers back to the start)
10. Else
11. For every possible motion
12. Add motion to node to generate a child
13. If the child is not in the closed list
14. Calculate the cost-so-far (g) of the child as the sum of the cost to the predecessor and the cost from the predecessor to the child
15. If the child is not in the open list or the cost of the child is lower than the cost of a vertex in the same cell
16. Assign the previous node (x) as the predecessor of the child (x\_succ)
17. Assign the cost-so-far of the child g to the cost of the cell g(x\_succ)
18. Calculate the heuristic of the child based on the distance between the child and the goal
19. If the child is not in the open list, add it to the open list
20. Else update the priority of x\_succ in the open list
21. Return null

Function apply\_motion:

Successor. State = Predecessor. State + motion.motion

Successor.g = Predecessor.g + motion.l

Function l:

Length of arc + penalties (maybe add later)

Penalties:

+ Turning penalty

+ Reverse penalty

+ Changing direction penalty