Spike Outcome Report

Number: 06

Spike Title: Graphs and Search

Personal: Peter Argent (7649991)

Goals:

Demonstrate the use of Dijkstra's (search for item) and A* (search for position):

- Modify the lab 10 code
- Add an agent that follows a path for a successful search result
    - Movement doesn't have to be fancy (same movement as planet wars is fine)
- Add the ability to search for an item or for points on the map
- Display the cost of the path
- Demonstrate the need for both algorithms

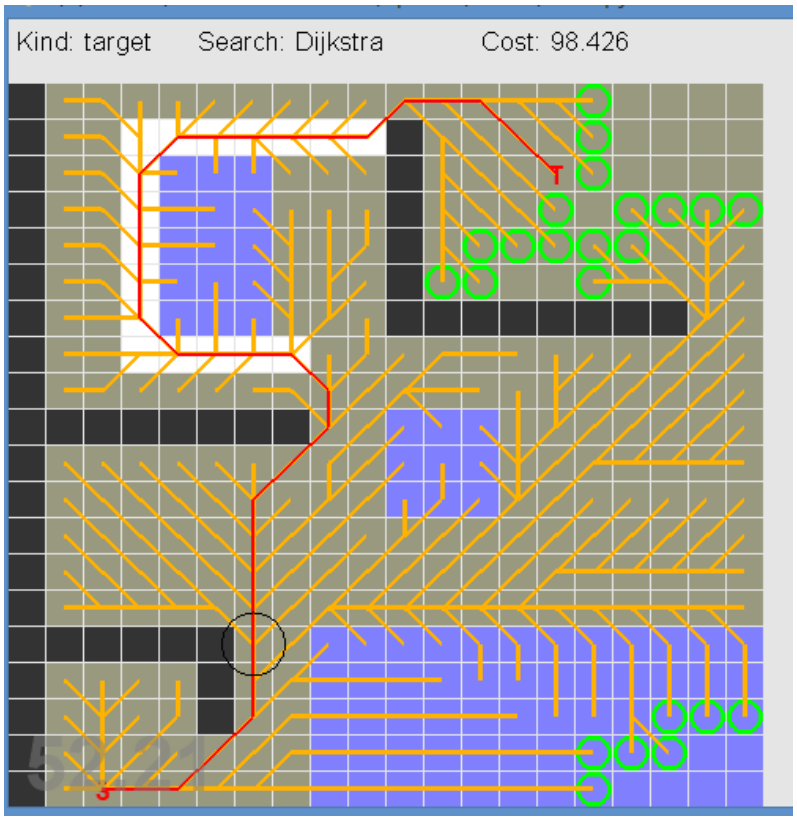
Technologies, Tools, and Resources used:

- Visual Studio 2017 with Python 3 installed
- Pyglet Documentation here: http://pyglet.readthedocs.io/en/pyglet-1.3-maintenance/
- Help from peers.
- Python 3 Documentation http://docs.python.org/
- Lab 10 Work as Base

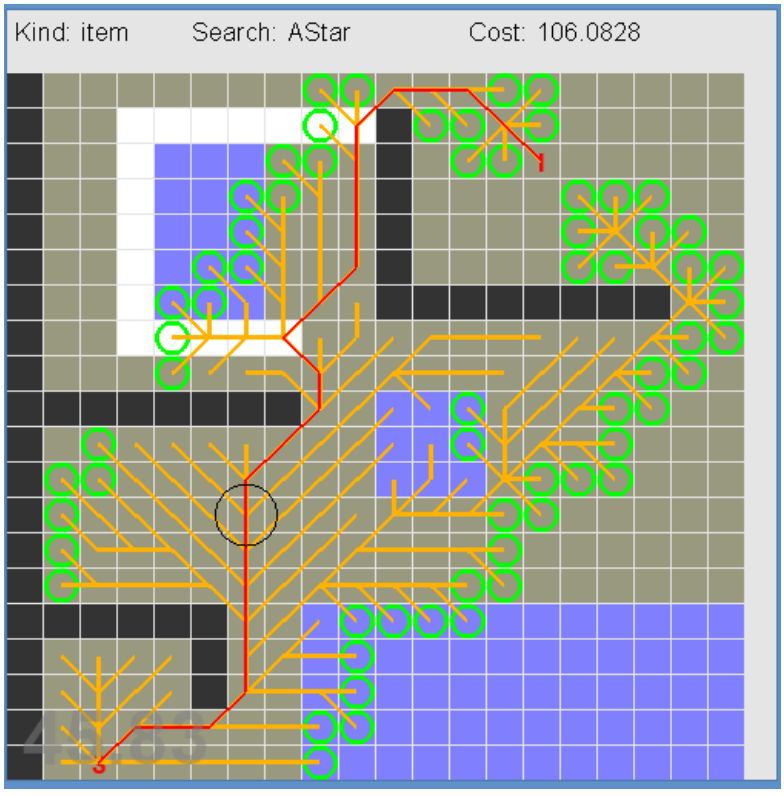
Tasks undertaken:

- Adjust the search file to only have A* and Dijkstra's algorithms (commenting out the code is good idea)
- Add an agent that follows a completed path
    - Store the boxes from the world in the agent
    - When a successful path is made add the path to the agent
    - Once working then add circle (or any other object) that is drawn as the agent moves
- Add the ability to place items
    - If you place an item, make the search algorithm Dijkstra's
    - If you place the target position, make the search algorithm A*
    - Then It calculates the path automatically
- Changed the status text to display Cost when you have a successful path
- Make a custom map which is cost heavy
    - 20x20 map
    - Lots of mud
    - A couple pools of water
    - A clear path around a pool of water
    - Lots of walls blocking the start and target

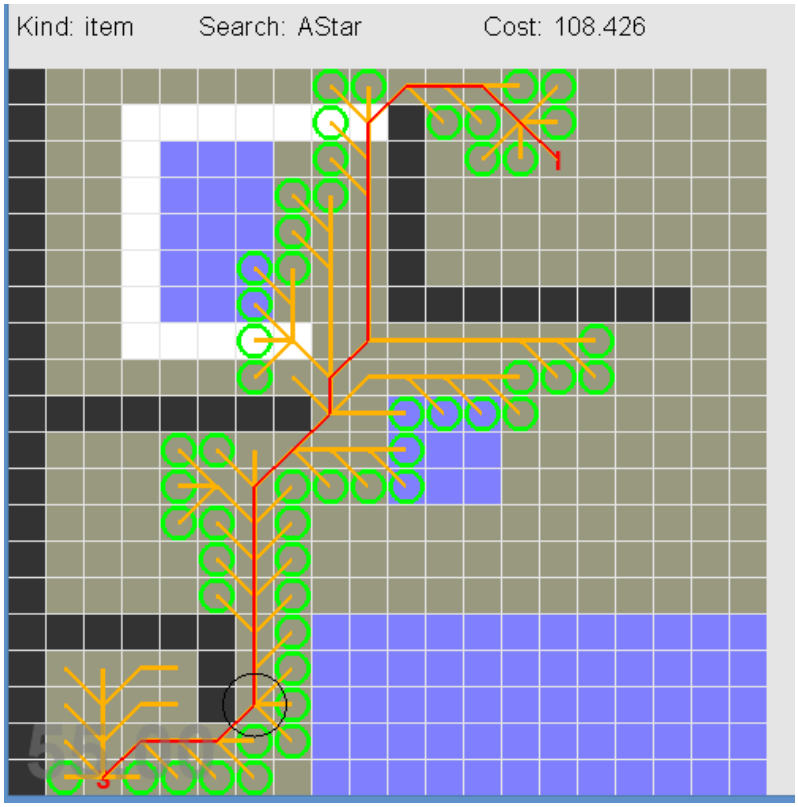
What we found out:



Search for Item using Dijkstra's



Search for Target with A with min_edge_cost of 5*



Search for Target with A With min_edge_cost of 10*

From the above graphics we are shown a few things:

- One, Dijkstra's Algoritm will search almost the entire map before deciding on the path to the Item.
- Two, A* will tend to be more direct than Dijkstra's purely because of the heuristic added to the search algorithm, Because of this the higher the min_edge_cost the more likely the directness of the path but with a higher cost associated.
- The major difference is that because the A* Algorithm has an extra heuristic to consider it will find out a good path quicker (on average) than Dijkstra but that path will end up being costlier.

From this we can ascertain three things:

- One, in games which algorithm required will change depending on circumstances, is the agent wanting to get to a specific position, or one of several positions (eg Item Spawn locations). Ultimately the choice of algorithm will be dependent on other circumstances for which one will be better to use in a given situation
- Two, Because Dijkstra's search almost the entire map, it will give many different paths and defiantly choose the lowest costing path. This can be taken advantage of by making the cost be some sort of factor of safety, thus if an agent needs to get to a health pack they call this algorithm to find the safest route to any health pack
- Three, Because A* tends to be a quicker search method, even though the route to the might be less safe (costlier) than the Dijkstra's path we can set a heuristic that the agent tells the algorithm "hey I can handle this amount of danger (min_edge_cost)". It means that the agent moving will not be afraid of handling a little mud in the case of the above map and target.

Notes:

In the graphics above the Item and Target Kind labels are reversed, that bug has been fixed in the uploaded code.

Keys for use:

6 – Choose to place Target marker for A*

7- Choose to Place an Item Marker for Dijkstra's

Left Click will place the selected marker (or modify the map similar to lab 10)