Project: Food Hunting Go!

Team members: Zeyang Li, Yifan Zhang, Chongming Wang

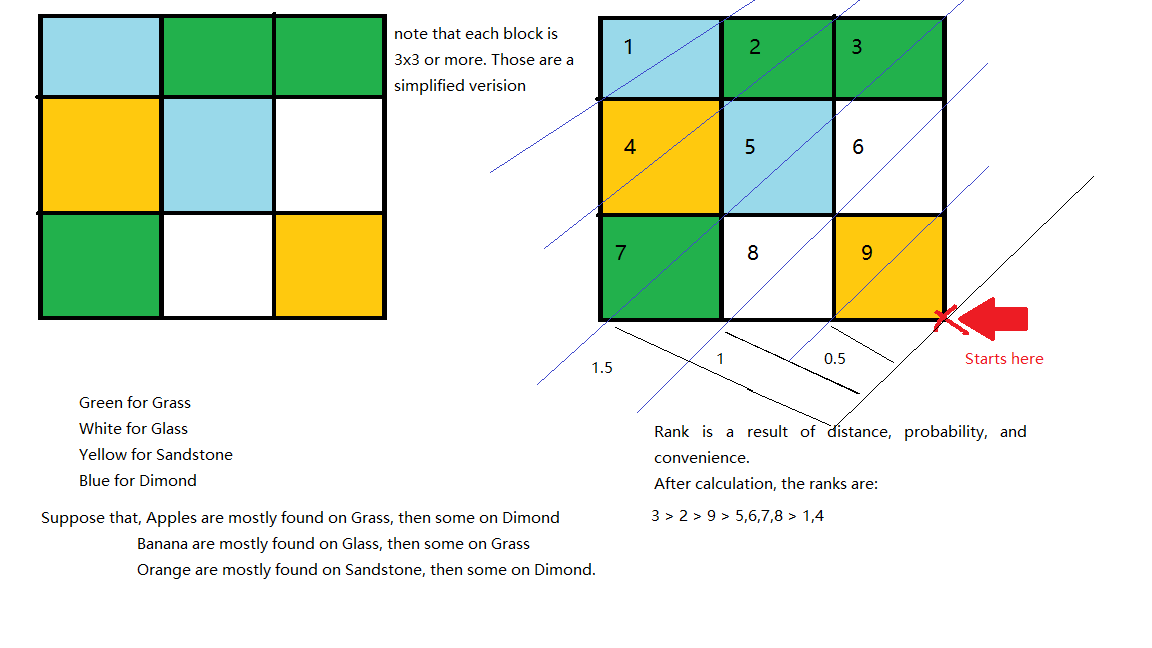
Project Summary:

We complete the searching algorithm and map generator. The agent now can discover and retrieve objects on any given map without dying. Moreover, the agent is capable of identifying and relating objects with the landscape. Therefore, on any new maps, the agent will look for places with the highest probability, and searching in a fastest route.

Approach:

For the searching part, we use depth-first search. With a stack of actions recording the last action, the agent can correctly trace back to the last state if it runs into a dead end. If it meets a dead end, the agent will break the loop, pop this current action off the stack, and push back in a new direction/action. On the other hand, the agent itself has a list of lists of 1’s and 0’s as it travels through the map. 1 and 0 are for accessibility of a block, and the coordinates of that block are the relative position in this list, for example, L[0][4] stands for the (1, 5) on map.

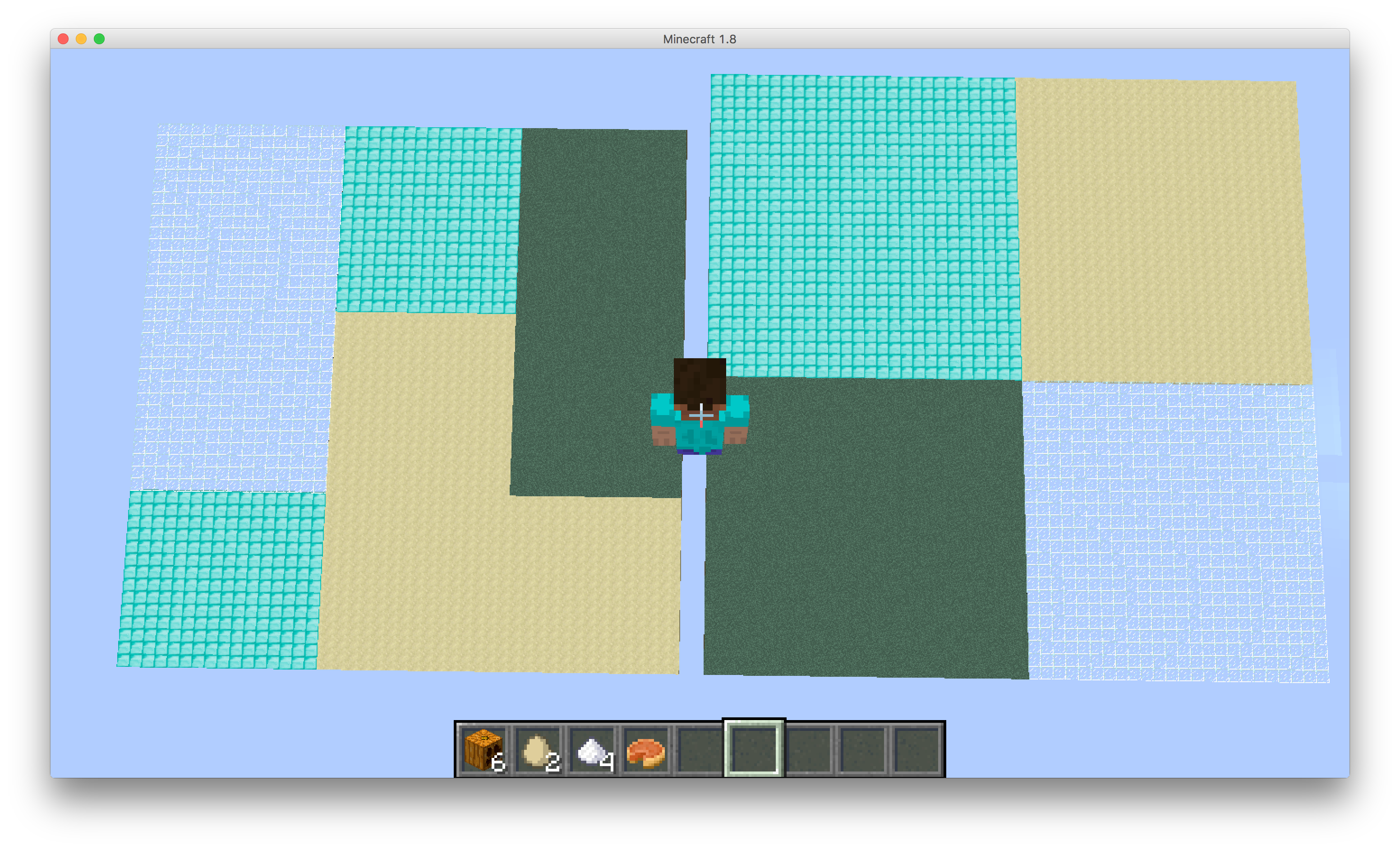
The agent will auto-collect objects from 9 blocks around it. After exploring the map, the agent will come back to the pot. If the agent does not have enough material, one of the possible states is that the objects are surrounded by lava. If so, the agent will carry a bucket with water to pave a path off lava to discover places it had not visited.



For the training map, the agent searches via depth-first search. It will check if that block is visited or available. The agent will only take blocks that are un-visited and not causing death, such as lava and air. When the agent is traveling through the map, it collects objects and records on which blocks they are found. Therefore, it will have a dictionary that looks like {(“apple”, “grass”): 64, (“apple”, “sandstone”): 34, (“apple”, “glass”): 2}. The values mean how many times such object is found on this block, so we can conclude the chance to find apples on grass is 64/(64+34+2) = 64%, 24% on sandstone, and 2% on glass. In a test map, we will have a ranking system to evaluate every block of the map. Those blocks are rated based on three elements: how many objects and how often those objects can be found on this kind of terrain, and distance towards agent. For example, with same distance, grass will have higher rank than the other two. However, if there is a banana with {(“banana”, “diamond”): 40, (“banana”, “sandstone”): 30, (“banana”, “diamond”): 10}, and the agent will consider to visit sandstone first because two objects are likely to be found on sandstone instead of visiting grass for apple and diamond for banana.

We hope to apply this method on a larger scale with 3D obstacles, so the agent really has to think about its path.

Evaluation:



The agent takes the training result and optimizes a better way of retrieving those objects on the new map.

The agent will rank terrain with probability from the dictionary. In other word, it will go to the closest block with highest probability. In the demo, the task to make a pumpkin pie, which is made of a pumpkin, sugar, and an egg.

Therefore, the agent will first find a pumpkin on Sandstone. Since it is currently on Sandstone, it will begin a DFS in this area. Then it will look for egg on Diamond, which is on the East side of its current position. Sugar is on Glass, and the closest glass is on South. Hence, it will go south for the last piece to craft a pumpkin pie.

Remaining Goals and Challenges:

Future Aims:

This prototype is limited to its ability on known maps and terrains; it does not know how to act on unseen situations. A real agent should know boundaries and optimize its route while it is searching on training map because the agent can collect objects within 9 blocks around it. It does not have to travel the whole map to know everything on the map.

For future project, we are looking for a functioning agent that can handle a very complex and large-scale map with height and lava situation. That will be more likely to a real-life case. On some hills, it is hard to climb from one side but easier on the other. In addition to that, it is possible to just jump off from the cliff, so a hill can be considered into path under certain conditions. This agent can be used on robots that can search and discover on lands that people cannot step a foot on, such as seafloor, jungles, or the surface of Mars!

For what we are having here, the agent only decides where to go before its action, but we want the agent to gradually learn about its position and reconsider its possible action as it goes. Moreover, it has to learn about forgetting places with fewer values or visited to save memory. It is updating as it goes. However, it is hard to re-calculating route with landscapes in real-time, and it is difficult to “guess” un-known map.

In the later project, we will largely decrease the number of objects on the map, so agent will have to find/guess some more possible places to look around, yet it needs to figure out how to climb mountains and swim over seas. Putting simply, this agent is capable of training itself on any map. It is training and searching on the same time. This will be the most difficult part of our project.