TDQC

Mining

Jack Spence

30 November 2018

1. Write-Up

1.1 Requirements

Requirements were to write an API for a Zerg mining operation. The classes required were an Overlord class which will be command and control for subordinate drones of my own creation. Drones will have a cost associated with them and cost refined minerals to create. The drones will be deployed on maps with minerals and will scout and mine them as decided based on their job role.

1.2 Suggested Features

Suggested feature attempted in my submission is to implement A* pathing as well as having a more helpful display using colorful labels to signify what drones have found on maps.

1.3 Syntax

Usage: python3 begin_mining_expedition.py

My module is set up to allow being included from parent directories with the __init__.py

2. Project Design Plans

2.1 Initial Design Plans

My initial design plan was to have the drones only look into a list of commands that the overlord creates and just read the most recent command. This was the plan to help alleviate the time constraints on the drones as well as the overlord.

2.2 What didn’t work

The largest hurdle I ran into on this project was the tracking of currently deploying and returning drones. My problems stemmed from my initial design of the overlord tracking deployed vs. returning drones, the responsibility of updating the respective lists was undetermined which lead to a lot of headache which lead to the rewriting of a lot of the logic in that regards. This lead to a lot of rewriting of logic for clarity of flow and overall program design.

2.3 What went well

The separation of responsibilities between the drones and overlord worked out very well for timing constraints. With drones only looking at their local list of commands, they can handle their actions well within the time limit. I had tested with up to 3 actions, which is not implemented, and was still within the 1 millisecond time limit.

2.4 Conclusion

This project has brought up some unexpected problems that I had not had in the past. Mainly showcasing my struggles of not properly planning things and sticking with the first thing that worked, vs. verifying that it works in all scenarios before building on top of that.

3. Test Procedures

3.1 Testing

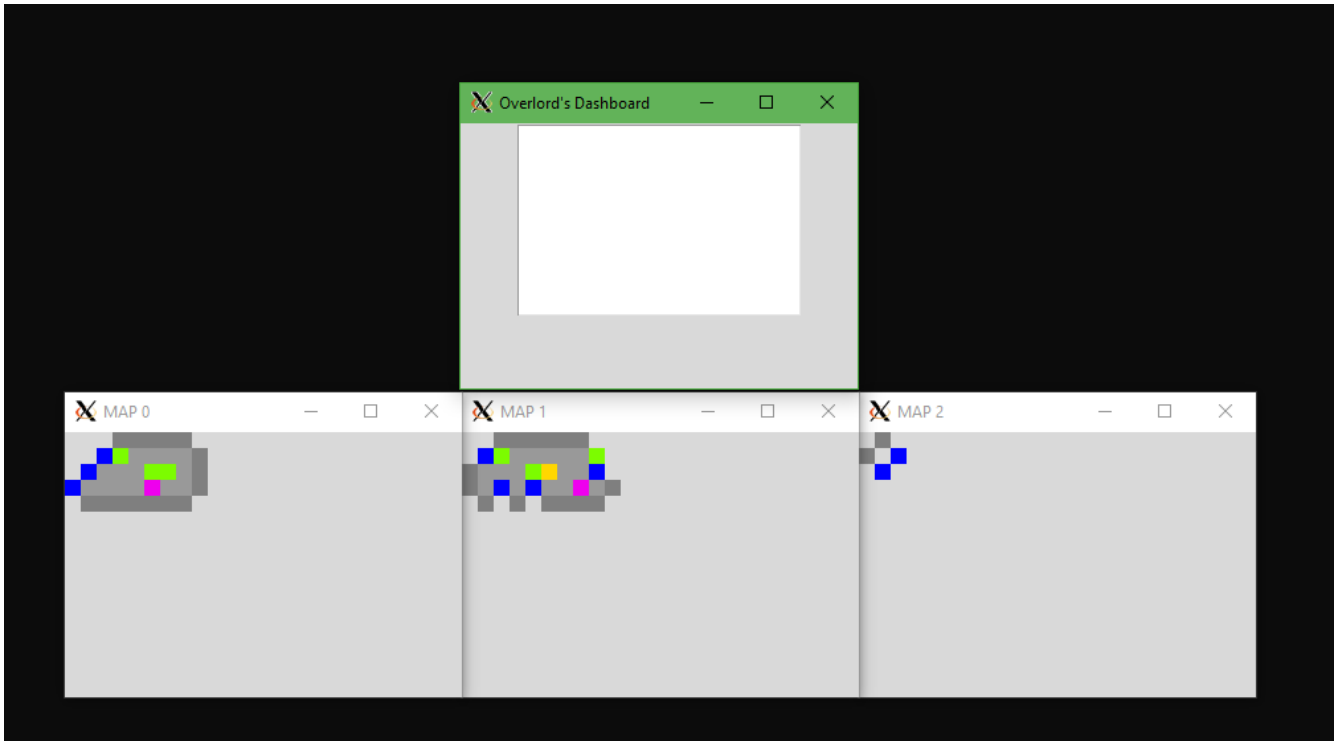
My testing procedures for this project was using the provided begin_mining_expedition.py and using different maps and gauging the minerals and paths taken. I'm aware that my implementation will take more ticks to gather the most resources, but in my testing it is pretty thorough and will mine as many minerals that are discovered. One problem that will occur is if the landing pad is surrounded by minerals the Scouts will not pick them up, thus not freeing the landing pad for a miner.

3.2 Testing Procedure

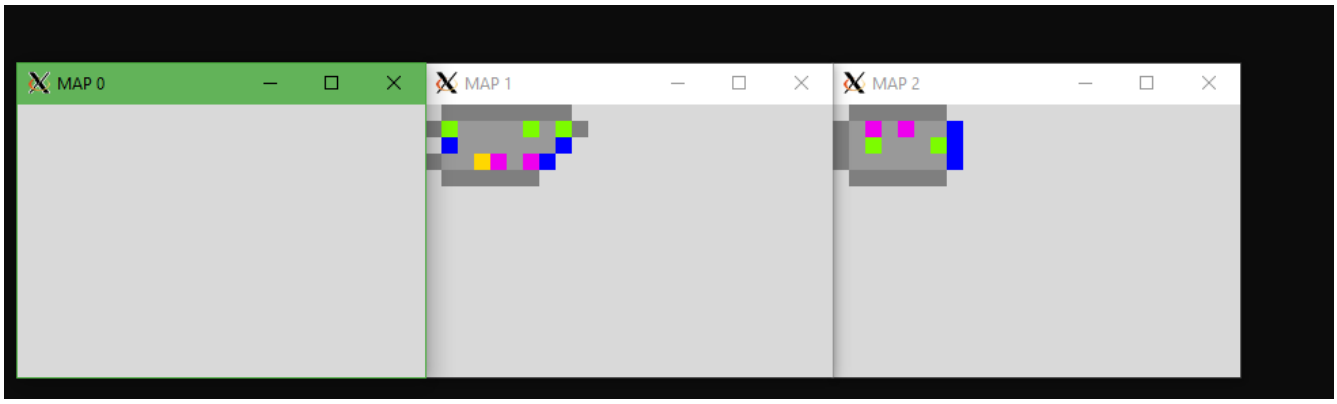
<u>100 ticks/100 resources</u>

```
Total mined: 18
skydev@skydev:~/TDQC-Projects/mining$ python3 begin_mining_expedition.py maps/map06.txt maps/map06.txt maps/map06.txt
```

```
Total mined: 83
skydev@skydev:~/TDQC-Projects/mining$ python3 begin_mining_expedition.py
```



Test with blank map (maps/map06.txt is map 0 in the below pic)




```
######################
#   ~~    ~ #         #
# _         #  ##     #
#             #       #
######################
###########
#   _Z   **#
#*    Z   *#
#~      ~ ~#
###########
###########
#         * #
#Z~     ~*~#
#     _   * #
###########
Total mined: 57
skydev@skydev:~/TDQC-Projects/mining$ python3 begin_mining_expedition.py maps/map05.txt
```

2500 ticks/55 resources