Name: Jean-Erick Voigt

Date: 20 September 2017

Current Module: Python OOP Phase project

Project Name: zerg.py

Project Goals:

Project is designed to evaluate our ability to use OOP, classes and API. We are expected to use class objects and allow all functionality to be accessible from the Overlord class. The user will import the overlord class from outside the directory. In order to successfully execute the import, the mining directory needs to be a package. This package will contain all the modules that need to be imported in an __init__.py file. In this case, the zerg module and overlord class our placed in the __init__.py file for import.

The program must follow the syntax of the API to function properly because. This must all happen through the overlord class. It also must complete all the steps within the ticks and not exceed time to make a decision. A basic requirement for this project is to be able to use a more sophisticated algorithm and return at least more minerals than were previously returned using a random function.

Considerations:

<A bulleted list of considerations and conditions which affected the design of your project>

- Classes
    - Inheritance
    - super_init
    - passing objects back and forth between classes
- API
    - API syntax
    - Understanding how the front end functions and methods work
    - staying within the time constraints
    -
- DRY
- SOLID

Initial Design:

This program will be split into multiple classes. The Map class is designed to hold the different types of maps that will be utilized for the project. For example, a dictionary to store the map id's the map coordinates and corresponding symbols and a dictionary to store just the map id and terrain information. The purpose for this is so the map can be utilized from the overlord class and also the Drone classes. The Overlord class also currently contains some dictionaries, such as a dictionary to store the status of the map, whether it has been scouted or not; a dictionary to store the status of each zerg (deployed, home or redeployed); and a dictionary to store which map that a zerg is currently on. The overlord needs these dictionaries, so it can verify whether the zerg is ready to and which maps need which types of zergs. The overlord utilizes the Map class's dictionaries by instantiating a Map object in itself.

The scout algorithm is designed to find minerals by looking first North, East, South, then West for a clear passageway. The map is automatically updated with its neighbors after each time he looks. If he finds a passageway he will go that way. If not and he has checked all directions then he will back pedal. During this back pedal, the scout is designed so as to look for minerals to mine. If he can mine anything then he will do so until his capacity is reached. If his capacity is reached while back pedaling or his capacity is reached then he will continue to scout as much as possible. The threshold for scouting will be set between 20 and 50, depending on what can provide the highest yield of minerals. Once the threshold in action calls for the scout have been met then he will return home. At this point, the "D" will turn to an "R" for status and the and "CENTER" will be returned for the action. When the overlord this he will pick up the scout and redeploy him if possible.

Once all the maps have been scouted, the overlord will then be able to send out a miner drone. This type of drone will have the ability to go directly to the farthest mineral using the shortest path. The mineral list will be stored in a top level dictionary. The method that I will use to find the shortest path is similar to the "a star" method. This drone will have similar abilities as the scout drone except his primary focus will be to go directly to the unmined minerals and mine until capacity is reached and go back home.

Data Flow:

Everything in this program will start with the overlord and the overlords action. The overlord will return the command "DEPLOY", "RETURN" or "NONE". If the command is deploy, the drone that is up for deployment will go to the designated map, where his action will determine which way to go. This determination will be returned to the overlord in order to process the movement. If a mineral is in the neighboring direction then this command will cause the drone to mine that direction until clear.

If the overlord's command is "RETURN", then he will return the drone for future deployment. Any other commands will not produce an action for the overlord. However, the Drone does not need an overlord action in order to execute its actions. It lands on each map with its own action.

Communication Protocol:

<Discuss any custom communication protocol you have used including the various

request and responses that may be sent and received.>

Potential Pitfalls:

Potential pitfalls will be understanding how the API will interface with the user's program. As the developer of the backend for the API, you must fully understand how the front end will expect the API to work. If this cannot be done then the program will give unintended results. Understanding the action method for the drone class and overlord class is crucial. The program will need to stay within the constraints of the timeout. If not then it will not be able to mine correctly.
The path finding will be somewhat tricky because the map that stores the recorded location will need to be accessed by the drone and the refreshed for each map. However, that drone will need updated status of each map it goes on as well so as not to go over the same paths. Going back to the landing strip from the destination may be tricky because if you use a list that has multiple coordinates, that list may not work efficiently unless you pop off indices.

Test Plan:

User Test:

Test Cases:

The instructor provided test cases to run the code against.

Conclusion:

This project will be very challenging because of the logic involved with it. While developing for this project the challenges have more to do with the logic of making sure the classes are able to share information and the logic is working well with the API then the actual coding. The lines of code for this project will not be the most intense. In order to be successful in this project, you must understand how classes work to the extent that you can use them with each other and inherit information from different classes.