University of Maryland Baltimore Campus

Tool Developer Qualification Course

Josh Carroll, James Viner, Raymone Miller

Zerg Mining Expedition

Contents

1 Project Summary

This program is a package that will be imported by a user to implement the functionality of Zerg drones scouting for, mining, and retrieving minerals from several given tile maps in a certain amount of time and given a certain amount of starting minerals with which to create the drones. The Overlord will spend its time routing drones to their destination and ensuring collisions are resolved. The drones themselves are responsible for the physical actions of scouting, harvesting minerals, and being retrieved by returning to the drop zone.

2 Features Targeted

2.1 Map

The program will be able to read maps, that will have certain mineral densities. These mineral densities will detail to the Zerg Overlord how many minerals there are on the map. The maps will also have hazards such as walls and pools of acid.

2.2 Drones

Drones in this program will explore the maps and collect minerals on the behalf of the overlord. While exploring the map, after being deployed, they will endure hazards such as pools of acid and will return minerals to the Zerg Overlord as well as update the Overlord's internal maps with the discovered terrain data.

2.3 Overlord

The Overlord will handle the movement of drones from deployment onto the map, movement around it, and eventually retrieval. Based off context relayed by the drones, the overlord will make pathfinding decisions. If the Overlord does not make their decision within one second, (they are on a tight schedule), their move will be skipped.

2.4 Refined Mineral

Refined minerals are what the entire program will revolve around. The Overlord starts with a number of minerals to use to construct drones and then must harvest as many minerals as possible from each map. Every 10 points of health will cost 1 mineral, every 5 points of mineral capacity will cost 1 mineral, and every move per tick will cost 3 minerals. These three statistics make up a drone, where no stat can be zero.

2.5 Dashboard

The dashboard will be what the user uses to see the full context of the map, It will display the map with specific items representing hazards among other things that may prove beneficial for the overlord. A "#" symbol represents a wall, a "_" represents a deployment zone and "*" represents minerals.

2.6 Path Finding Algorithm

A path finding algorithm will be needed to best find a way to reach specific minerals in a map. This will most likely be Dijkstra's algorithm, however if we find implementation easier then thought then we can implement A* algorithm. This will require heavy testing as it will serve as the foundation of the program.

2.7 Map Context

The map context is what the drone object will return to the overlord at the end of each tick. This will show what is at the tiles in the four cardinal directions from the drone and where the drone currently is on the map with an X and Y coordinate.

3 Architecture

3.1 Data

There will be a Zerg parent class, likely an abstract base class. The Overlord class and the Drone class will inherit from that Zerg parent class. There will also be a Dashboard class that will be responsible for displaying drone movements and their surroundings. This class will be present in the GUI.

4 Significant Functions

`Overlord._create_drone(drone_type)`

This method takes in a drone type and creates a new drone of that specified type.

`Overlord.add_map(map_id, summary)`

This method registers an identifier for a map with the Overlord, along with a summary of the map

`Overlord.action(context)`

This method takes a map context object as parameter and returns a value representing the action taken

```
Overlord.recall_drones()
```

Recalls drones back to the overlord

```
drone.action(context)
```

Checks the map context object and returns what direction a drone would prefer to move in.

```
drone.steps()
```

Returns cumulative steps a drone has taken since deployment.

```
drone.get_init_cost()
```

Returns the cost in minerals needed for a drone to be built

```
Overlord._del_minerals(coord, drone_id)
```

Remove a mineral from the set of known minerals

```
Overlord._select_map()
```

Select the map to deploy the scout to.

5 Order of features

1. Initialize a Github page for the project.
2. submit up a design plan.
3. submit up a man page

4. Build basic objects for drone and overlord
5. adding testing to ensure needed objects can be produced as needed.
6. implement a path finding algorithm
7. produce necessities for GUI implementation such as classes need
8. implement file reading with maps
9. produce maps in GUI that meet restrictions.
10. Have overlord produce drones
11. enable path finding behavior in drones.
12. Have drones draw their path as they traverse it in the GUI interface.
13. run through program several times to find any unique bugs
14. final writeup.

6 User Interface

This program uses a GUI to display information related to the moment-to-moment action, including drone statistics, a tile map for each map file, and a legend for that map.