University of Maryland Baltimore Campus

Tool Developer Qualification Course

Josh Carroll, James Viner, Raymone Miller

Zerg Mining Expedition

Contents

1 Project Summary

The program is designed to be an API which will help a developer in coding an actual program. This API will consist of Zerg overlords and Zerg Drones. The overlords will send drones out upon mining expeditions to collect and bring back minerals that the Zerg race need to survive. As these drones are deployed and explore the map they will suggest a route that is the best path to the minerals according to the drone's current knowledge of the map.

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 much minerals there are overall in the map. These maps will also have hazards to the drones such as walls and puddles of acids.

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 acid and puddles and will return not only minerals to the Zerg Overlord that they work for but also the next desired position based off knowledge they currently hold of the map.

2.3 Overlord

The Overlord will handle the movement of drones, from deployment into the map, movement around the map and return from the map. Based off knowledge returned from the drones, the overlord will make the decision they seem best. 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 drones will collect refined minerals for the Overlord and refined minerals will be used in the production of not only normal drones but other classes of drones as well. For every 10 points of health a drone will have it will cost 1 refined mineral. For every 5 units of capacity the drone will have it will cost 1 mineral. For every 1 move that the drone will have it will cost 3 minerals.

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. These representations will be among others detailed in the map description page of the rubric.

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 a tick. This will show what is at either direction of the drone and where the drone currently is on the map with an X and Y coordinate.

3 Architecture

3.1 Data

The file will read in the files initially as a string. Their will be a Zerg parent 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 drone movements and their surroundings. This class will be present in the GUI.

4 Significant Functions

```
Overlord.build_drone(health, capacity, moves)
```

This method takes in a health integer, capacity integer and a moves integer as a parameter and builds a drone.

```
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.remove_drone(drone)
```

This function removes a drone from the game Overlord's inventory, usually when said drone is dead.

```
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.return_drone(drone)
```

calls the drone to be returned to the overlord

```
Menu.present()
```

presents a overall menu to the user

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 will use the CLI on the initial start up so the user can decide what maps the program will take. It will use a GUI to show what parts of the map the drones have discovered as well as what stats are being used during game play.