CS1006:Programming Projects
Practical 2:Starcraft
University of St.Andrews

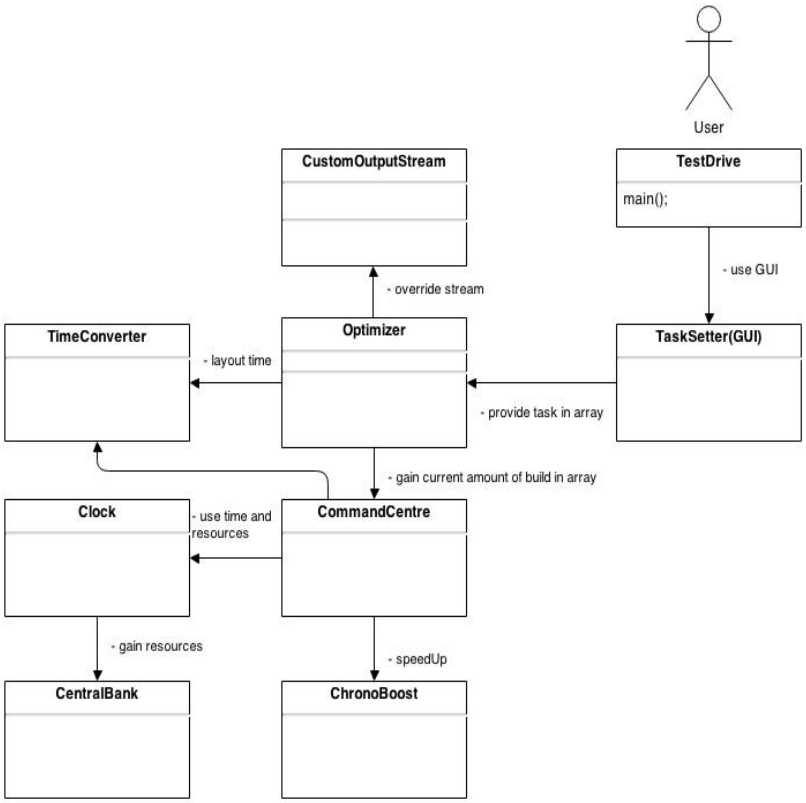
130016030 120024946

March 7, 2014

Tutor: Michael Oliver Brock

1 Introduction

StarCraft II is a military real-time strategy video game developed and designed by Blizzard in 2010. In short, it is one of the main RTS franchise and very popular in e-sports. The main objective of the game is to build an army that is capable of destroying opponent's base,while the opponent is trying to do the same thing. Furthermore, player can choose to play in one of three races, namely: Terran, Zerg or Protoss. Facilities and resources are required to build an army and buildings. It could be seen from above explanation that, gameplay requires balancing investment in resource collection, provision of production facilities and building actual army.
In this practical the requirement was to create a simulator of the game for a single player and the effective build order optimizer for Protoss race. Having considered StarCraft project explanation, it is reasonable to look at the main design decisions.



2 Design

The UML diagram above represents class structure of the StarCraft II project.(we have included a separate folder with screenshots for your convenience) Evidently, the program starts from the class TestDrive. There is a main method that calls SwingUtilities where GUI can be safely updated. After that, the main GUI form displays. This is one of extensions in this project. Rather than follow pre-written strategy and goals, it was developed and designed give advantage for user to set its own goals. This can be done by clicking at radio buttons and putting numbers to white boxes.

All these numbers are in the array that transferees to the class Optimizer by creating the object. Class Optimizer has access to three classes, namely: TimeConverter, CustomOutputStream and CommandCentre.CustomOutputStream is used to override System.out. This can benefit for user to see output in JTextArea rather than in the console. TimeConverter has in Optimizer and CommandCentre to format time. For example, instead of output 65 sec, it will print out 00:01:05 Although, Java is the Object-Oriented Programming language, it was designed to store all methods that responsible in constructing

buildings and units in CommandCentre. However, creating many individual classes that inherit methods from parent class also can do this.

Nevertheless, to justify designed decision it can be suggested that it CommandCentre is very closely related to the Clock class. Hence, to check that required minerals and gas is enough there must be methods in CommandCentre to do this. Therefore, CommandCentre stores all information to avoid duplication and overriding the code. Also, it keeps tracking these numbers in the array, sothat Optimizer can compare array from CommandCentre and TaskSetter to see is task finished or not.

Clock class has a javax.swing.timer that allows increment time and resources gained from the CentralBank. On the other hand, any design cannot be absolutely perfect. As it was mentioned, this design is not purely object-oriented since all methods and variables are in CommandCentre. However, this design provides advantages as well as drawbacks mentioned above. Also, GUI is not designed to automatically set task if it is depended. For example, it will be impossible for the program to create a Zealot without making a task to create a Gateway. Additionally, it was required to provide some transitions methods in Clock class to CentralBank, so CommandCentre can have access to resources. Please note that the displayed timestamps for building is the time at which that build starts.

Overall, this design produces a working solution that fully meets advanced deliverable. Chrono Boost, Special Units,Warp Gates,Upgrades, Realistic Timing, Supply extensions were also attempted and work well. On top of that, GUI user-friendly interface, allowing user set its own goals also counts as an extension.

3 Search strategy

According to the specification, very sophisticated search strategy also counts as an extension. Also, it is essential to implement it to get high grade. To begin with, it was planned to implement **genetic algorithm**. However, based our research, genetic algorithm is not the most efficient way to solve this problem. Hence, although it is complicated, it would not make a reason to implement it if it is not efficient. After that, there was a plan to solve it using critical path analysis, but again it is not the fastest way. Finally, it was decided to implement **Prims Algorithm** with some changes to fit our design solution.

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

Pseudo code above represents Prims Algorithm for connected and weighted graph. However, the project involves searching in directed graph with vertices and edges G=(V, E)

On top of that, another reason to choose Prims algorithm is time complexity. Genetic algorithm takes O(GNM), where G is number of generations, N is the population and M is the size of individuals. However, Prims algorithm takes O(E log V). This means that in general case Prims algorithm works faster than genetic.

Class Optimizer has an implementation of Prims algorithm with some changes. It was achieved by using Threads. First thread creates an object of the class CommandCentre that runs a clock. Second thread has a game loop that breaks if task is completed. It also randomizes array to find the most appropriate task to do.

As it was mentioned before, theoretically it should work faster. However, on practice it was impossible to prove because other groups cannot show what their build time is. Also, it is important to note that real time extension was

attempted, so delays involves in searching. Obviously, it works slower than the solution without delays. In addition, Pylon also will be constructed if there is a little supply.

To sum up, our solution involves implementation of modified Prims algorithm. Theoretically, it should work faster than genetic algorithm approach. However, there was no chance to compare it on practical with other group solutions.

4 Testing

Testing always comes to be the most difficult part, especially for this project. According to one of the demonstrator, this project can be proved that it works by providing a screen shoots of output basic and advanced goals. These pictures are included in the project folder.
However, going beyond that it possibly can be proved by the principle of induction.

. Let P (n) be the statement where n is the number of buildings and units Evidence that P (1) holds can be found in the projects folder. So, this starts the induction P (n+1) will also hold with one notice building assimilators requires more probes to gain resources. If there are not enough probes, appropriate message will be printed out
.
. This prof is not purely mathematical, however developers believe it makes sense. If not, where is not other way to show that it works.
Overall, developers made their best to prove this solution. Additionally, core functionality can be tested straightforward because Optimizer depends on it.

5 Conclusion

To sum up, this project was very enjoyable. It improved many programming and computer science skills as well as taught new concepts. Obviously, there solution cannot be perfect and there is always will be a room to improve it.

This is a working solution that fully meets advanced deliverable. Chrono Boost, Upgrades, Special Units, Realistic Timing,Warp Gates, Supply extensions were also attempted and work well. Moreover, GUI user-friendly interface, allowing user set its own goals also counts as an extension. Believed that modified Prims algorithm also counts as extension, this project brings to the advanced solutions. Although, one extension ("Extra Base") was not attempted, our own extensions were designed and delevoped.