Project Report - SC2 Robot Feeder

Yuqi Zhang, Wang Dong, Tianqi Wang

1.Summary:

Our goal is to produce a robot that is able to defeat a Veryhard built-in AI. The ability to defeat the Veryhard robot is the border between casual gamer and a core gamer. Our robot mainly focuses on the early and middle game scouting, building and attacking to achieve our goal. It has 60% win rate against HardVeryHard AI right now.

2.Motivation:

There are plenty of unofficial AIs online, some of them can defeat pro gamers. But most of the strong robots take a long time to program or trainning. And All of our programmers are unfamiliar with Starcraft II. It took us a lot of time to learn to defeat built-in AI. So, we decided to build a fundamental robot that was stronger than official VeryHard AI as a start.

However, building the robot is not the only goal of us. We are also learning to use third party libraries.

3.Approach:

To achieve our goals, we started by reading the library and online examples to start. The c++ documentation gave us a human-friendly overview of sc2api.

After doing some research, four significant aspects in defeating an AI are scouting enemy base, building structures, managing units and game events handling.
These components are the basic functionality of a robot. With all of them we can build a strong robot that can scout, expand, attack and produce units.

To achieve the four subgoals, we have created a huge number of functions to help us.

3.1.Events handling:

Event handling is the most important aspect of the four. Events are interfaces of the running game. We do preprocessing in OnGameStart event, getting game results from OnGameEnd event and implement our robot in OnStep events. There are other game events that are helpful. But these three are most significant.

3.2.Scouting:

Most scouting functions that can be used in game events are placed in FeederScout.cpp. And some helper functions that game policy maker does not need to know are placed in Utility.cpp. We calculate potential enemy base locations via the player birth location and the playable map size. Since the map is symmetric, we only need one point to calculate the rest three points. All these calculations are finished before the game starts. After the game starts, a random scv will be sent to scout and this scv will not be assigned other tasks until finishing scouting. When the scv finds enemy structures or gets killed by enemy units. The event handler will mark the closest potential enemy birth location as the true one.

3.3.Building Structures

To build a strong strike force, we must have a stable economy and productive training facilities. Our bot can renewably collect resources by both building a new command center or moving the old base to a new location to gather minerals. Our force is largely made of infantries, which can be produced early in the game. By having several barracks, a rapid training process can be executed. To stay competitive in middle of the game, our bot will research higher upgrade and train medivacs in starports, improving the survival chance of our brave soldiers. Even we have such management, we still focus on a early stage of the game, since our units are too primitive. It is not our advantage to face a large number of advanced foes.

3.4.Managing Units

First, it is essential for our bot to know when to attack and when to retreat. It is hard coded and tested that 40 units are a strong force against the built-in AI. In most cases, it can defeat the enemies within one or two waves. Our strikeforce is largely composed of marines and some marauders. Often, we have the number advantage over our foes. And with the help of stimpacks, our infantries are strong against zerglings and protos. However, it shows a terrible disadvantage to against terran mechanical forces. If our army got beaten, they would retreat to the rally point to save our number.

Microcontrol is also a part of units management. But it should not be talked about separately. We will discuss the reason in conclusion part why we make it a subtopic. Currently, we are doing hit and run when there are only a few units to control. When the army grows up, we use medivac to save dying units and deliver them to a safe place. To figure out which units we should save, units health info are cached. We compare current health with the cached health of the same unit, when the health falls we should save this unit.

4.Modules:

We are going to talk about some important cpp files and some huge functions here.

ManageArmy():

This is the function where we implement microcontrol and strategy. For different types of units we would assign assign different tasks for them here. And some simple policy parameters are calculated and stored here.

ManageUpgrades():

This is the function where we choose which upgrades to take and when to take.

BuildArmy():

This is the function where we build army in game directly. And the number of each kind of units is decided here.

BuildStructre:

These are functions about deciding what structures to build, where to build and how many to build. Besides, building expansion is also implemented here.

FeederDemo:

All unfinished functions or unreliable functions are placed here.

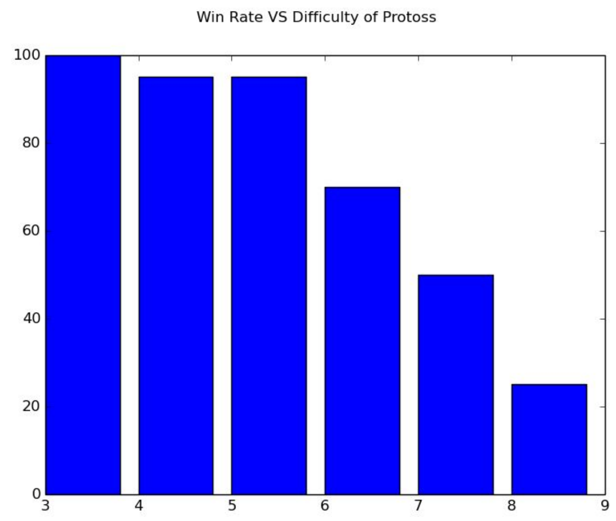
Utility:

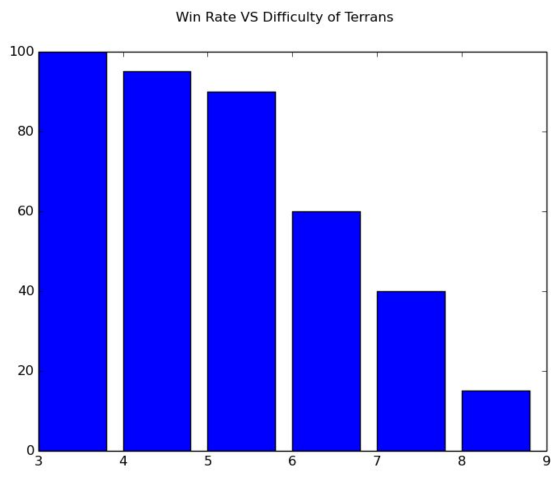
All helper functions are stored here.

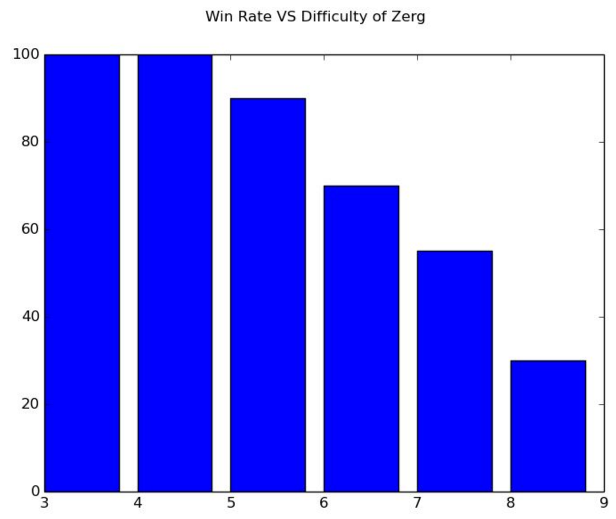
5.Conclusion:

Stats for 20 games for each difficulty and race:

Difficulty: {Medium:3, MediumHard:4, Hard:5, HardVeryhard:6, VeryHard:7, CheetVision:8}







We can say proudly that our robot has achieved our goal. It has a 60% win rate against Veryhard AI. And we noticed that simple micro control does not influence the winning rate as much as we expected. Since we found that the true reason we lose is enemy units are overwhelming our units. The enemy has units which deal area damage that hurts our fundamental army. And micro control does not help it because It can not change the nature of units.

6.Future work:

To improve the performance of the robot, we plan to upgrade the following aspects. Firstly, some advanced units may be considered to support our army in the middle-late game. They can be tough against high area damage units, such as tanks. Secondly, we can improve our micro control by gathering army before an attack. Thirdly, it will be a great advantage to let our bot learning to find high grounds to attack and defend. Also, it should learn to retreat if the enemy has great height advantage.