# Battlecode
— Report One —

Karlson Lee, Timothy Lim, Robin Marle, Peter Efstathiou, Mohammed Al-Hakim, William McGehee
{kl909, ma1015, rm1715, wm2315, tl5515, etc.}@doc.ic.ac.uk

Supervisor: Dr. Stephen Muggleton
Course: CO530/533, Imperial College London

5th February, 2016

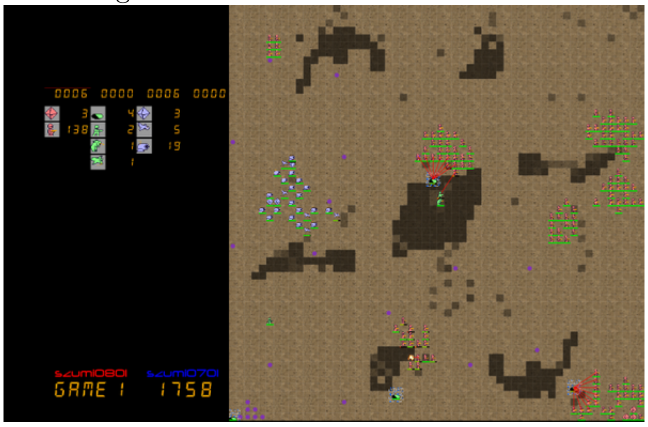
## 1 Introduction

Battlecode is an MIT programming competition held every year that combines battle strategy, software engineering and artificial intelligence. It is a Real Time Strategy (RTS) game, pitching two teams of virtual autonomous robots against each other.

The goal of the game is to defeat your opponent by destroying all of their Archons, their mobile head quarters. This simple objective can be achieved in a very wide variety of ways. Our bot will not only need to form effective plans but it must also anticipate, react, and disrupt the opponent's plan. The game features imperfect information of the state of the game, the state of our opponent, but also the state of our teammates. Finally, there are multiple resource constraints to manage, including both financial and computational resources.[2]

The plot this year is "Zombie Apocalypse", the two teams must now struggle for resources and attempt to survive amidst the ruins. Zombie robots will spawn on the map and attack the players. Each player will begin with several Archons, and the player who loses all Archons first loses the game. Players must scavenge the map for resources and units as they evade or fight the zombies and opposing player faction.[2]

Figure 1: Battlecode Client Interface



## 2 Requirements

### 2.1 Capturing Requirements

The goal of this project is to produce a bot that plays this game. As the final goal of the project is well defined, capturing the minimum, medium and super requirements is essential to the success of the project. If the requirements are unrealistic the project will inevitably fail, but if the requirements are easily met then the project is also too simplistic and ultimately irrelevant. [4]

## 2.2 Requirements Overview

1. Minimum Requirements
   - Working Bot
   - Defensive Strategy (Turtle)
   - Offensive Strategy (Napoleon)
2. Medium Requirements
   - Automated Test Suite
   - Beat MIT Reference Bot
3. Super Requirements
   - Compute success indicators
   - Literature review of relevant machine learning techniques
   - Implement and evaluate machine learning techniques such as Neural Nets and Reinforcement Learning
   - Implement and evaluate self Program Induction Technique

## 2.3 Requirements Outline

### 2.3.1 Minimum Requirement: Working Bot

The team aims to submit at least one working AI to the MIT Battlecode competition. A fully functional bot consists of basic functionality such that it will be able to survive and succeed in an environment with a trivial enemy. Additionally a requirement will be to develop two main strategies, one defensive and one offensive.

1. Basic Functionality
   - Produce units
   - Engage enemy units
   - Implement a defense strategy
   - Implement an offensive strategy
2. Defensive Strategy (Turtle)
   - Build turrets around the Archon (home base)
   - Turrets distribute themselves autonomously to maximise coverage
   - Build soldiers
   - Soldiers distribute themselves in the proximity of the turrets to protect them
   - Repair nearby units
3. Offensive Strategy (Napoleon)
   - Engage a defensive strategy until the enemy has been scouted and enough units are available
   - Pack up turrets and move in formation to the enemy location
   - Eliminate the enemy

### 2.3.2 Medium Requirement: Automated Test Suite

An automated test suite is critical for having a quick turnaround time for iterating through different versions of the bot. MIT provides a basic test framework called The Battlecode Headless Client, the goal is then to extend this to be able to play two bots against each other across a selection of maps.

- Automate games between two players across a number of maps (at least 20)
- Output results in XML format
- Randomized starting locations for players

### 2.3.3 Medium Requirement: Beat MIT Reference Bot

As part of the competition, MIT releases a Reference Bot that students are encouraged to beat. A medium requirement of this project will be to best the Reference Bot in at least 50 percent of the available maps.

### 2.3.4 Super Requirement: Literature Review

The team will read and review literature relevant to the machine learning techniques that they intend to use. Each member will hold a workshop on their paper, to ensure that the knowledge is effectively shared.

### 2.3.5 Super Requirement: Implement Program Induction

Program induction is a very modern technique in self writing programs. Flags enabling or disabling behaviours will be set and a learning algorithm will find optimal combinations of behaviours. [7] [6]

### 2.3.6 Super Requirement: Success Indicators

In order to demonstrate progress, we will compute metrics that describe the desired behaviours. The indicators can be divided by behaviour they measure, facilitating unit testing and allowing objective measure of marginal code improvements.

1. Macro indicators (e.g Win percentage)
2. Combat indicators (e.g Combat event win percentage)
3. Movement indicators (e.g Time taken to reach destination)
4. Scouting indicators (e.g Percentage map explored)
5. Code Quality indicators (e.g Number of exceptions thrown)

### 2.3.7 Super Requirement: Implement Neural Nets For Combat Micro

Combat micro is a subtle and complex art, where different conditions require different combat strategies. The team will train a neural net to classify different combat situations and map optimal combat strategies to them.

## 3 Development Methodology

### 3.1 Development Methodology

Each Battlecode AI can be interpreted as a collection of behavioural features. As these features can be clearly defined and logically ordered by priority, we have decided to adopt an Agile Scrum development methodology. [5] [3]

The team decided at the beginning of the project that the first and primary priority is that communication is key and therefore discussed later in the report is the collaboration and communication tools used. As well as these, the team scheduled workshops to capture requirements from the team members. These workshops allowed for the team to discuss and agree on requirements, priorities and risks.[5]

The workshops were created as goal capturing exercises where group members were allowed to question and break down the overall goals to more specific requirements that can be further broken down into sprints to follow an agile methodology.

## 3.2 Sprints

The 14 weeks between the submission of this report and the final project deadline will be divided into seven two week long sprints. Each sprint will begin with a planning session. During these planning sessions, the team and project leaders will determine the set of features to implement during that sprint. A sprint backlog will then be prepared specifying the tasks that need to be completed in order to accomplish the selected features. Tasks will be assigned to team members either by group leader assignment or self-selection. [5]

## 3.3 Scrums

During the sprint period, short meetings (scrums) will be held every other day. During each scrum, each member of the team will discuss what he has accomplished since the last scrum, what he intends to do over the next two days, and any obstacles he has/expects to experience in meeting the sprint objectives. Any obstacles, delays or other issues will be addressed by the team leaders. [3] [4]

## 3.4 Sprint Reviews

At the end of each sprint period, a review session will be held. Work that was completed during the sprint period will be reviewed and discussed. Any issues will be highlighted. Tasks that were not completed will be identified. The estimated completion time and feature backlog will be adjusted accordingly. Progress and areas of difficulty will be presented to Stephen Muggleton after each review session.

## 3.5 Documentation

Documentation will be maintained as inline comments conforming to the Javadocs standard. Javadoc allows automated creation of HTML documentation pages, and are supported by many IDE's. This will be supported with the wiki.

## 3.6 Unit Testing

To ensure that code changes both do not break existing code and represent marginal improvements, it is important to have robust and regular unit testing. Any code revision will have to demonstrate that it is error free and shows equivalent or better performance on all sections of the automated tests, using the test suite.

### 3.7 Task Scheduling

| Minimum Requirements | | |
|---|---|---|
| Task | Priority | Sprint |
| Implement Bug Nav for pathing (Offence/Defence) [1] | 5 | 1 |
| Implement Fleeing (Defensive) | 5 | 1 |
| Implement Turtling (Defensive) | 5 | 1 |
| Implement Napoleon (Offencive) | 5 | 1 |
| Attack Micro (Offensive) | 5 | 1 |
| Implement scouting and messaging system | 4 | 1 |
| Switching between strategies | 4 | 2 |
| Gathering of Archons at the beginning to build stronger base | 3 | 2 |
| Implement Archons specific micro behaviour | 4 | 2 |
| Testing whether Soldiers or Guards is more effective | 2 | 2 |

| Medium Requirements | | |
|---|---|---|
| Task | Priority | Sprint |
| Automate running of the MIT headless client | 5 | 3 |
| Change parameters in maps and player starting locations | 2 | 3 |
| Extract results from XML replay files | 4 | 3 |
| Output description of winning conditions during the game | 3 | 4 |
| Improve pathing using DFS or BFS and compare to Bug Nav | 3 | 4 |
| Implement Viper Strategy | 1 | 4 |
| Identify Zombies Dens using scouts | 4 | 4 |
| Implement kiting in attack micro | 2 | 4 |
| Pathing and formation (e.g moving turtle, etc) | 5 | 4 |

| Super Requirements | | |
|---|---|---|
| Task | Priority | Sprint |
| Literature review of relevant machine learning techniques | 3 | 5 |
| Compute success metrics | 5 | 5 |
| Implement meta-level program for offline learning | 5 | 6 |
| Implement program induction techniques | 5 | 6 |
| Implement neural nets and reinforcement learning for combat micro | 3 | 7 |

## 4 Team Structure

Timothy Lim and Peter Efstathiou are selected as co-leaders of this project, responsible for code integration and project coordination. Karlson Lee will be responsible for organising meetings, taking minutes and scheduling. Our project is split into three segments - minimum, medium and super, with team members working on separate steams depending on what stage we are in.

## 5 Team Working Tools

### 5.1 Version Control

The team consists of six team members therefore version control is essential to ensure the tracking of changes, reverting changes, merging individual development branches and resolving conflicts. There

Table 1: Task Allocation

| | Minimum | | | Medium | | Super | |
|---|---|---|---|---|---|---|---|
| | Working Bot | Offensive (Napoleon) | Defensive (Turtling) | Beat MIT teaching assistant bot | Automated Testing Suite | Implement Machine Learning | Implement Program Induction |
| Timothy Lim | x | | x | x | | | x |
| Peter Lewis Efstathiou | x | x | | x | | x | |
| William McGehee | x | x | | x | | x | |
| Mohammed Al-Hakim | x | | x | | x | | x |
| Karlson Lee | x | | x | | x | x | |
| Robin Marle | x | | x | | x | | x |

are many types of version control systems, however this project requires a free, open-source, distributed and disconnected environment. It is also important that the version control system (VCS) contains adequate documentation, support and is popular amongst developers. [4] [3]

Git, is an open-source environment that meets all these requirements. It also used by many popular projects such as Googles Android, Eclipse and the Linux Kernel. All the team members have working knowledge of Git which drastically reduces the initial learning curve and enables focus to be on the requirements at hand.

Github is an online hosting service for Git. It offers a simple UI for browsing history, bug-tracking, wiki. It is distributed, allowing team members to work separately at different times and locations. It is also the most popular Git repo, and thus has a large body of resources available online and excellent support.

### 5.2 Collaboration Tools

Communication will be done using Slack, a project management platform for team communication. It provides full visibility to all aspects of the project. Relevant materials are shared, discussed, segmented using Slack channels. This is also convenient for organising sprints.

### 5.3 Workflow Control

To encourage agile development technique, Trello is used as the task scheduling tool. This tool gives access to a visual board that displays the ongoing tasks. These tasks are represented as cards with labels and priorities.

## 6 Feasibility and Risks

Although the minimum requirements are relatively feasible, risks remain as most team members are unfamiliar with Java and the proposed project management tools. However, the MIT competition has been running for the last 10 years and therefore there is a great deal of relevant material online.

The medium requirements are more challenging. However, they are still manageable risk as they can be accomplished with adequate time, planning and implementation.

There are several risks involved in the super requirements of the project. Most members have limited experience with state of the art artificial intelligence and machine learning technique. This knowledge has to be acquired alongside the requirements of the project. Secondly, there has not been an implementation of neural networks and program induction techniques in Battlecode before. Although this will demonstrate genuine innovation from the team, it will involve a process of trial and error.

We will continuously adjust the schedule to these risks by working intensively with our supervisor, team sprints, efficient project management and code development integrity as specified in this report.

## 7 Project Boundaries

### 7.1 Software

The software will be written in Java, using the IntelliJ IDEA IDE. The Battlecode Client suite will be used predominantly for testing AI features, making map, etc. Audience This project is relevant to AI for strategy game environment and for the Battlecode community around the world. The ultimate goal is to use state of the art AI/machine learning technique in this particular context.

## References

[1] Steven Arcangeli. 2012 battlecode winner writeup, November 2012.

[2] MIT Battlecode Dev. Battlecode 2016 game specs:, November 2016.

[3] Chris Sims Hillary Louise Johnson. *Scrum: a Breathtakingly Brief and Agile Introduction.* Dymaxicon, 2012.

[4] Fidelis Perkonigg. Module 530: Software engineering practice and group project - lecture 5 - capturing requirements.

[5] Ken Schwaber. *Agile Project Management with Scrum (Developer Best Practices).* Microsoft Press; 1 edition, 2004.

[6] Armando Solar-Lezama. Combinatorial sketching for finite programs. Technical report, IBM T.J. Watson Research Center, 2003.

[7] Armando Solar-Lezama. *Program Synthesis by Sketching.* PhD thesis, B.S. (Texas AM University), 2003.