CS 463 - Spring 2017 - Final Report

Project DevAI

Jacob Broderick, Kristen Patterson, Brandon Chatham

Abstract

This document details the progress made by the Starcraft AI Project. It includes a brief recap of the project, where the project currently is, and what stretch goals remain unfinished. It also talks about any problems that have occurred. This document was made with a video that details similar information, but demonstrates what the working project looks like.

Contents

I. Introduction

This project was requested by our client Dr. Fern. Dr. Fern's primary intent for this project was for us to reduce the learning curve of artificial intelligence and machine learning implementations for an AI club that will soon be created at OSU. Learning AI can be difficult but the basics can be picked up quickly. Some of the trouble spots for new developers who are not well-versed in the process of starteing projects might get hung up on configuring API's and setting up their development environment. Our project handles the configuration step for them. The environment is ready to use as soon as they open the Visual Studios project. Furthermore, if anything gets changed in a developer's settings, we have thorough documentation explaining how to fix it. Beyond this, we have included tools to help more skilled developers start data mining for machine learning algorithms with a python script we wrote and a replay analyzer we researched and chose from an open source developer. There are three team members: Brandon Chatham, Kristen Patterson, and Jacob Broderick. Brandon primarily focused on code design, implementations, and overall direction of the project to best fit the desires of the client. Kristen focused on documentation. Nearly all of the documentation, even a small bit of the comments you will find in the code, are written by Kristen. Kristen worked closely with Brandon to keep up-to-date on what the newest implementations were in order to keep documentation relevant to the current state of the project. Jacob focused mostly on choosing our development tools in Visual Studios and BWAPI as well as researched other tools that may be valuable to us. BWAPI is the standard API used for Starcraft AI and was an easy choice. Determining what other tools were well-suited to the specifications of the project were slightly more difficult but fairly simple decisions. Our client for the most part gave us autonomy on the project. While we did meet with him for required check-in's for the project, he allowed us to make our own decisions whether this was for better or for worse. We did consult him on a few occasions but usually made decisions primarily based on what the team felt was best based on what we knew Dr. Fern was most interested in.

II. Original Requirements Document

CS 461 - Fall 2016 - Requirements Document
Project DevAI

Jacob Broderick, Kristen Patterson, Brandon Chatham

Abstract

This project is to create an agent to play the game Starcraft Brood War, a real time strategy game created by Blizzard Entertainment. After this project is complete, it could be expanded upon by future students in a club setting. This project will be a template so that future students can strive to develop better solutions than those developed within this project. This document will provide the details of the project as well as provide a list of tasks the development team must complete. The goal of this document is to specify the requirements for the system to be developed.

III. Introduction

This section covers the purpose and the overview of this document. It also includes a list of abbreviations and definitions as well as describes the scope of the project. Any sources referenced throughout the document are also included.

A. Purpose

The purpose of this document is to describe the requirements of ”Project DevAI”. This document will also describe the purpose of the project as well as name some of the constraints that will be involved in the process of development. The intended audience for this document is the client and the development team. The client can make sure that their specific concerns and constraints have been met, while the development team can use this document as a reference during the development process.

B. Scope

”Project DevAI” is an AI that plays Starcraft Brood War and provides a user with a well-documented AI that can be used for research purposes. The AI should implement at least one strategy to play Starcraft Brood War. A user can interact with the AI by opening the source code and viewing it. The source code must be highly documented and modular that way users can pull any section of code from the source code and use it for their own program.

C. Definitions, acronyms, and abbreviations

AI - Artificial intelligence

Agent - The program that will play Starcraft Brood War by itself

Client - Person or organization that has requested the project to be developed

User - Any future Oregon State University club member that wishes to interact with the system

Starcraft Brood War - A real time strategy game created by Blizzard Entertainment

API - Application Programming Interface

BWAPI - API used to interact with Brood War’s code

Script - Procedure in a C program that calls a variety of generic real time strategy procedures

Micro - A term used in Starcraft to describe actions pertaining to controlling individual units, mostly related to attacking

Macro - A term used in Starcraft to describe actions pertaining to managing resources and constructing buildings, mostly related to economy management

Unit - A controllable object in Starcraft that can move, attack, and gather resources

Resource - An item in Starcraft used to construct units and buildings

D. References

[1] IEEE Software Engineering Standards Committee, IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, October 20, 1998.

E. Overview

This document contains two more sections. The second section contains information about the product perspective and functions. It also details the user characteristics and discusses the various constraints and dependencies the program has. The third section provides the specific and functional requirements of the system.

IV. Overall description

A. Product perspective

This system consists of a single application that will be ran via the game Starcraft Brood War. The application will communicate with Starcraft via an API that will inject itself directly into the game code. The software has no external interfaces with which it can be modified in real time, it must be completely autonomous and run as long as the game itself is running. The agent is allowed to save files to a folder that will be removed once the game is done. There are no constraints to how much memory is accepted by the system, but it must be able to handle the data without slowing down the frame rate of the game. The agent can create a log of what errors occur as it is running so that developers have the required information to fix any bugs. It will be limited to the operations defined in the Brood War API that will be used to create the project.

B. User interfaces

There are two kinds of users that could interact with the machine, developers and players. There shall be a way for the system to provide log files so developers can see the results of individual runs of the game. Users can play against the the agent as well, but it would require no changes from normal use. All interaction occurs within the game parameters. Players would be able to try different strategies against the AI to see how it reacts to different situations. Other AIs could also be players in our system. They would be pit against our AI to play against each other. There will be no other interaction with the users, as the AI should be able to work completely autonomously.

C. Constraints

The system must be able to run within Starcraft Brood War on most modern computers. It needs to be able to operate within the memory constraints to not affect the runtime of the environment. The system must not slow down the game by more than 1 frame per 10 seconds.

D. Apportioning of requirements

Other strategies are not included in the initial plan. If the system is completed early, then other strategies will be implemented. The AI's ability to change strategies will be delayed until future versions. The initial version shall only have one script that can be executed efficiently. The project will be opened to future developers to add more features to in a club setting. These are undocumented for now, but may be written in the future. The library will be separated into classes focused on the two generalized areas of game-play categories: micro and macro.

V. Specific Requirements

The agent should be comprised of modular components that are well-documented. Functionality should be broken into stand-alone pieces that could nearly directly translate into another developer's agent. Also, these pieces should be documented well enough such that a developer can understand and make adjustments to them per their needs. Our stretch goal is to give the agent the ability to switch between scripts based on game circumstances. This would mean adjusting to a new strategy based on the agent's resources and potentially the opponents estimated resources.

A. Functional requirements

Our agent will use the BWAPI to interact with the Starcraft Brood War game. Our agent will follow a scripted strategy. Since the intent is for the agent to play the game on its own, there is no user interface beyond running the program. People can potentially play against the agent, however this requires no additional interaction with our agent. The scripted strategies will be based on how we can engineer the agent to gather resources and expend them while also eliminating the opponent's ability to do these things.

1) Micro library objects: Objects focused on micro management decisions will often be more specific to the current state of the game and micro management decisions of the opponent.

a) Forming groups: An object for unit positioning will encapsulate the functionality of unit grouping. Units are generally grouped for combat when facing an opponent and this object will focus on ensuring combat units are moving in a group as much as possible based on input from the combat decisions object.

b) Movement: This is an object for individual unit movements and group movements. Our base strategy will focus primarily on moving aggressive units into advantageous positions for attack and defense and doing so in groups. Other strategies that may be developed in following versions may be more heavily focused on individual unit movements for specific strategies. These actions will be made based on input from the combat decisions object.

c) Attacking: This object will handle the commands of aggressive tactical decisions. The combat decisions object will extensively interact with this object in order to attack enemy units or structures.

2) Macro library objects: Objects focused on macro management will be more closely tied to basic functionality of the game rather than tactics while keeping track of the resource collection and expenditures.

a) Resource gathering: This object will handle resource collection while working closely with the state of the game in order to keep track of what resources will be needed most immediately.

b) Building production: As resources are gathered, the building production object will establish a foundation for production of necessary units for the scripted strategy. This object will work closely with the construction decision object in order to produce necessary buildings as soon as the resources are available.

c) Unit production: The unit production object will focus on production of units when resources are made available based on input from the unit production decision making object.

d) Expansions: Expansions will come about through an interaction of all three of these Macro subsections' decision object. A new gathering location will be established, new units will be built to populate the resource deposits, and resources will be gathered.

3) Decision making objects: Each object requires decision making logic to decide what to do under certain circumstances. The decision making objects will decipher what input to give the objects given a set of circumstances.

a) Combat decisions: This object will handle decisions based on combat information such as opponent positioning or unit types. Also, tactical objectives must be considered, for instance, whether a unit is defending or attacking a specific area.

b) Resource gathering decisions: Resource gathering requires decision making when deciding whether to harvest from the home base resource deposits or expansion resource deposits while also considering opponent positions. Sometimes, resource deposits are at maximum capacity of resource collecting units and therefore, the unit should collect from a less populated expansion base if one exists.

c) Construction decisions: Structures are allocated a specific amount of area when being built and cannot overlap. This object will handle the decision making with regards to where structures should be built.

d) Unit production decisions: Units will be produced based on the scripted strategy, available resources, and sometimes, opponent unit production. In later versions, this logic will more extensively respond to opponent actions. For now, this object will follow the scripted strategy very closely and execute commands for units that fit the strategy.

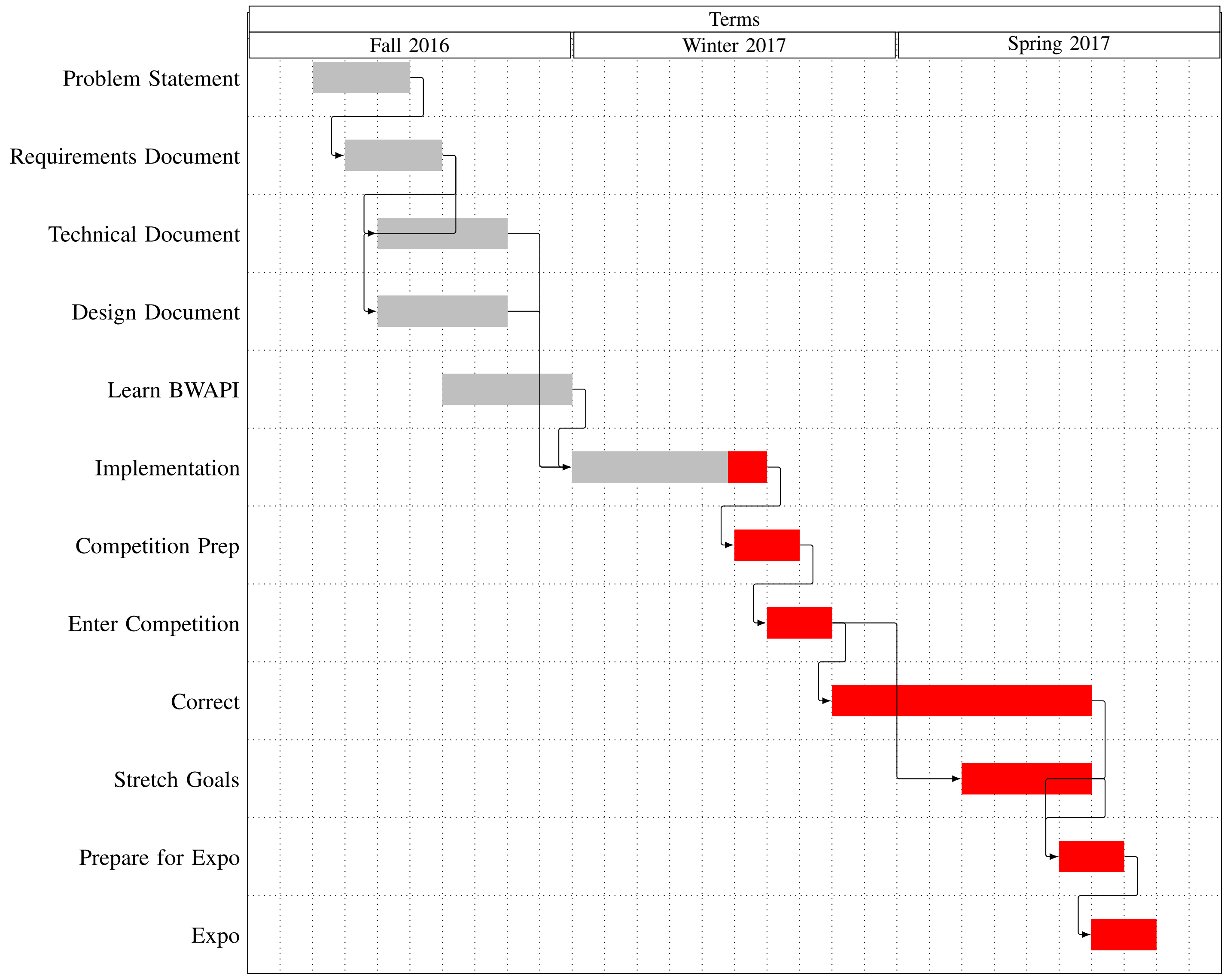

Fig. 1. Gantt Chart

VI. Changes from Original Requirements Document

A. Final Gantt Chart

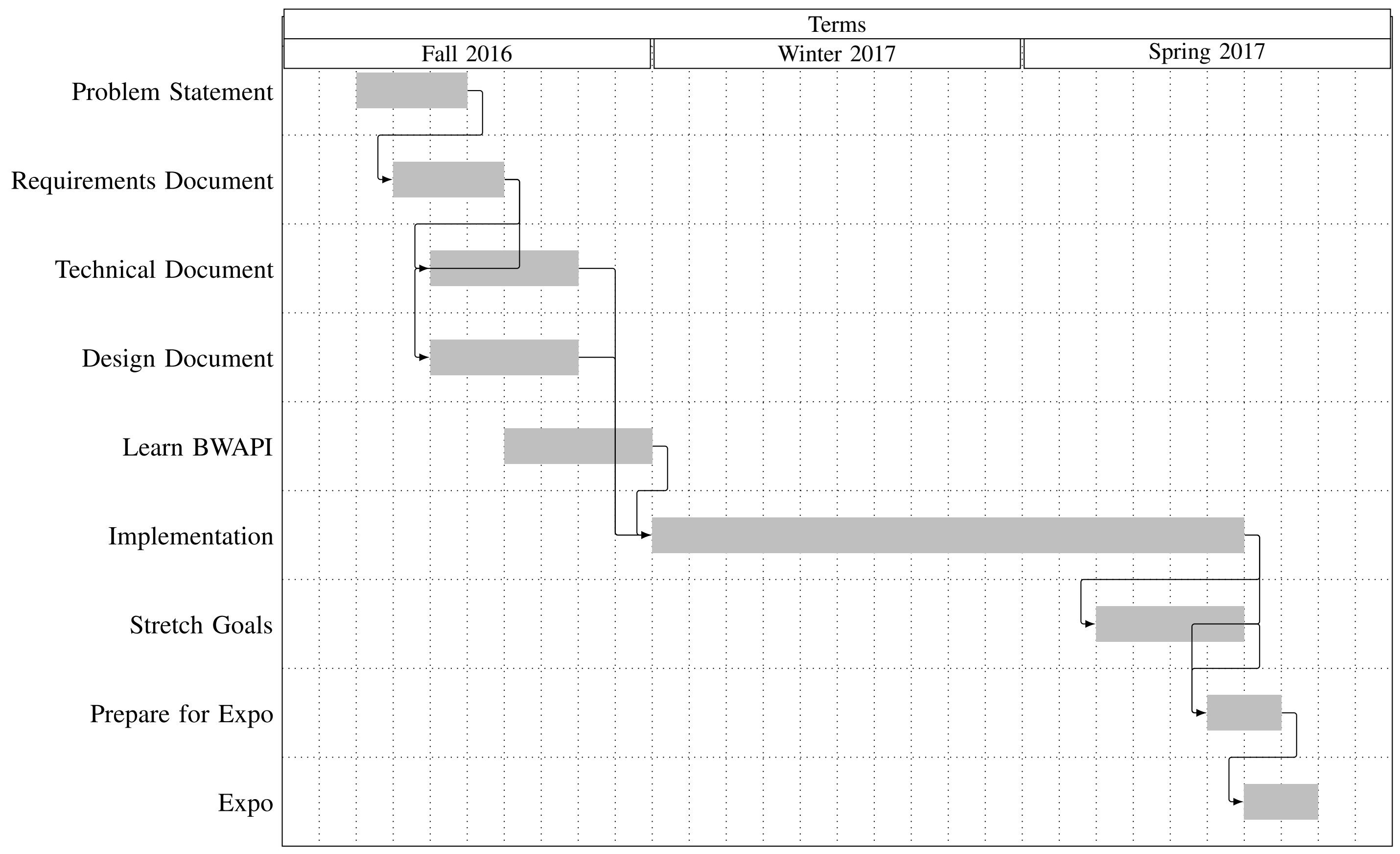

Fig. 2. Final Gantt Chart

VII. Design Document

CS 461 - Fall 2016 - Design Document

Project DevAI

Jacob Broderick, Kristen Patterson, Brandon Chatham

Abstract

The goal of this project is to create an agent to play the game Starcraft Brood War, a real time strategy game created by Blizzard Entertainment. The solution to the project will be a template that future students can use to develop better solutions than those provided in this project. In this document, we will discuss how we will design and implement our project. The design choices we make will help as a road map and checklist to follow.

VIII. Introduction

A. Date of issue and status

12/4/2016

B. Scope

This document will provide details on the design of the documentation, modularity, and the strategy implementation of the Starcraft AI project.

C. Authorship

Brandon Chatham - Modularity

Kristen Patterson - Documentation

Jacob Broderick - Decision Making

D. Design languages

The design language used will be a UML diagram.

E. Glossary

AI - Artificial intelligence

Agent - The program that will play Starcraft Brood War by itself

Client - Person or organization that has requested the project to be developed

User - Any future Oregon State University club member that wishes to interact with the system

Starcraft Brood War - A real time strategy game created by Blizzard Entertainment

API - Application Programming Interface

BWAPI - API used to interact with Brood War's code

Script - Procedure in a C program that calls a variety of generic real time strategy procedures

Micro - A term used in Starcraft to describe actions pertaining to controlling individual units, mostly related to attacking

Macro - A term used in Starcraft to describe actions pertaining to managing resources and constructing buildings, mostly related to economy management

Protoss - Alien race, very futuristic technology, builds advanced units

Unit - A controllable object in Starcraft that can move, attack, and gather resources

Race - In Starcraft there are three different playable races Terran, Zerg, and Protoss.

Resource - An item in Starcraft used to construct units and buildings

Terran - Human race, very military based

Zerg - Zerg is an alien race, focuses on building lots of units, swarm and hive mind type of race

F. Change history

IX. Design Stakeholders

A major stakeholder in the design process is future students involved in an Oregon State University club. These students will be reading and using the code included in the project.

X. Design concerns

A. Documentation - Kristen Patterson

The design concerns of the future students are that the code is well documented and organized.

B. Modularity - Brandon Chatham

In order for the code to be transferable, it should be broken into modular pieces focused on specific functionality accessible through the BWAPI.

C. Decision Making - Jacob Broderick

The best path to a successful bot that plays Starcraft is to have proper decision making. The decision making must be good enough that there is a chance for success.

XI. Design Views

A. Documentation View - Kristen Patterson

The code is well documented if it has concise comments and includes functional header comments. In order for the code to be organized, it must be modularized into simple functions and further separated into header files containing functions with similar purposes. The functions also need to be listed in order of complexity with the most complex functions requiring the highest amount of dependencies is located at the bottom.

B. Modularity - Brandon Chatham

The code will be modular with respect to the functionality of the BWAPI. Creating stand-alone modules that perform necessary Starcraft game actions will improve the learning process for student developers who use the code, and will enable them to produce AI agents more quickly even with limited or no experience with AI software design. Furthermore, the modules should have limited or no dependency on other modules. This allows for simple transferability into a developer's code.

C. Decision Making - Jacob Broderick

The AI is able to decide what strategy to implement based on the state of the game. Multiple strategies will exist that determine the behavior of the AI. Certain buildings and units will be created depending on the strategy chosen. What the units do will be determined by the strategy, but the way they move will be executed by a movement module. The strategy will work for a specific race within the game, and would be catered to fit that specific strategy. Many different strategies can be implemented and tested, and should just require specific calls from the system.

XII. Design Viewpoints

A. Documentation - Kristen Patterson

1) Pattern: The pattern viewpoint covers the design for the commenting of the code. Commenting in the program will require reuse of function header comments to describe the function's purpose. Functional header comments will follow a specified template of describing the name, process, inputs, outputs, and requirements of a function. In-line comments must be included for any variable declarations, loops, and function calls as well as any ambiguous declarations. The parts of the functional header comment must directly associate with code in the function. For example, the inputs are the parameters of a function and the outputs are the returned values. This viewpoint will be supported using the UML composite structure diagram.

2) Composition: The composition viewpoint is used for the design of modular functions and files. Composition will help to localize and simplify blocks of code into functions. Functions will then be localized into separate files based on their purpose. Most functions will be decomposed into modules with a singular purpose while their will be a few functions that control the system and call upon the simpler modules. The HIPO diagram will be used to supplement this viewpoint.

3) Dependency: The dependency viewpoint designs the order of the functions in each file. The design of each file will follow the order of least complex function with least dependencies on top while more complex functions that use other functions are on the bottom of the file. A UML component diagram will be used to show the dependencies.

B. Modularity - Brandon Chatham

1) Structure: The design concerns of modularity are code transferability, and understandability. Transferability refers to a student developer's ability to move code from one project to another with little or no required code edits. Understandability refers to the level of effort a student developer would have to invest in order to understand what a module is doing. Modularity of the library will be categorized with respect to individual classes or small groups of classes relevant to a desired functionality. The BWAPI classes are already broken into categories for interacting with the AI agent and game. This library will access functions relevant to specific actions based on functionality of the module. This design convention makes learning easier by providing concise examples for student developers to use and study. An example would be pulling game state information with regards to in-game resources. This will be important for many of the logic modules created. These modules will organize the functions that pull the appropriate data and can send it to other modules for further algorithm-based actions. In this example of resource collection, modularity makes it easier for students to separate the logic of data gathering from the other categories of functionality. Additionally, it makes modules more transferable to other AI agents without disrupting other systems within the overall body of code.

C. Decision Making - Jacob Broderick

1) Strategies: The strategy viewpoint covers a scripted strategy that will accomplish a single thing. For the race Terran, this could be a strategy that involves spending all available resources on the marine unit. These strategies will also determine how and when the AI will take certain actions such as attack/defend. These strategies will be based around timings that are used to determine when it is the best possible time to do certain actions.

2) Strategy Choice: The choice of strategy will depend on the state of the game. If the opponent is playing aggressively, a defensive strategy can be chosen. It also can determine when to upgrade units, or choose different units to build. If the enemy has numerous air units, switching to an anti-air option would be more successful than continuing with a non anti-air unit based strategy.

XIII. Design Overlays

A. *Modularity - Brandon Chatham*

As mentioned in the Requirements Document, categories will be broken into micro and macro game actions. Micro actions are tactical and more unit-specific while macro actions are resource gathering and base construction focused. Also, these categories are broken into decision-making modules that pull game data and algorithmically decide what should be done and communicate with action modules that act on the decision making. This is applicable to the modularity design concerns as it provides developers easier code sharing and transferability when developing competitive AI's with other developers.

1) Modularity of Strategies: These strategies should be modular, so that they can be tested and removed as needed. This will allow for optimization of the AI when playing many games with other AIs. This will also allow us to test our bot against itself, with different configurations.

XIV. Design Rationale

A. *Documentation - Kristen Patterson*

1) Commenting: Adding function header comments and using a template will help future students to quickly and easily find the information they need about the function. In-line comments will also help future students follow the process in the function.

2) Modular functions and files: Keeping functions modular and organized into separate files will help future students to easily navigate the code. It will also help them use the code as the can take separate modules or files easily.

3) Ordered functions: Ordering functions by complexity will help the future students navigate as they can quickly find simple functions to use. They will also be able to physically see the use of simple functions to support larger more complex functions.

B. *Modularity - Brandon Chatham*

1) Categorization of Modules: Categorizing the library modules first based on the functionality they serve with respect to the game, then micro and macro game actions, and finally AI decision-making or action will make library components concise, easier to understand, and more transferable across AI agents.

C. *Decision Making - Jacob Broderick*

1) Choosing Strategies: Multiple strategies will be required in order to ensure that the AI has the highest chance of success. It will gather data from data modules that will assist in discovering the best option on how to proceed.

XV. Revisions

A. *Design Views*

1) Documentation View - Kristen Patterson: Modular functions have been changed to object oriented classes for better organization. The order of functions has changed as well because most class based functions have the same complexity and the dependencies don't matter when they are located in a class.

B. Design Viewpoints

1) Documentation - Kristen Patterson:

a) Composition: Composition will still be used for organization functions except functions will be further organized into classes.

C. Design Rationale

1) Documentation - Kristen Patterson:

a) Modular functions and files: The focus now is to section classes by functionality and keep dependencies low. Each class will have its own header and cpp file in order to clearly seperate them and reduce dependencies.

b) Ordered functions: With functions being sectioned into classes and files of their own there will be very little need to order functions based on complexity as most will be simple.

D. Design Revisions

1) Modular Hierarchy:

a) Reducing Abstraction: As development has progressed, the team has realized an error in the design. Previously, the plan was to abstract the objectives of the game into categories - micro and macro while separating the logic from the modules. Im hindsight, this was foolish. This is a poor design choice for several reasons. First, abstracting the design into more layers might make each layer easier to understand, but as student developers, it is overly ambitious to try and organize a code base like that as it makes it more likely that we will code ourselves into design conflicts. Second and more importantly, this library for students who understand the game best by the strategies and actions the game is already based on. Designing around these concepts makes the library intuitive to users. Lastly, there is no reason to separate logic modules from the API function calls. This might make code slightly cleaner, but it adds more classes than are necessary.

b) New Approach: Now, development is focused on common actions needed in every AI module that can be easily understood, used, and developed further. Additionally, the team is writing algorithms to optimize these actions. The new design will help developers hit the ground running with a robust foundation for their AI modules. Most importantly, the design should be consistent to allow the code base to be passed on to other students for further development and learning opportunities while already having a design paradigm to continue using.

XVI. Tech Review

CS 461 - Fall 2016 - Technology Review

Project DevAI

Jacob Broderick, Kristen Patterson, Brandon Chatham

Abstract

The goal of this project is to create an agent to play the game Starcraft Brood War, a real time strategy game created by Blizzard Entertainment. The solution to the project will be a template that future students can use to develop better solutions than those provided in this project. In this document, we will discuss the technologies we will utilize to complete our project. The technologies we choose will help to narrow down ambiguities in our approach as well as inform the client of the environment and supplemental tools we will be using.

XVII. Introduction

There are many technologies that are required in order to make an artificial intelligence that is capable of interfacing with a game that is closed source like Starcraft Brood War. There is the program itself that would determine how the system is ran, an API to connect to an outside system, and the code that determines how the program actually works. There are many different technologies for each category. This document will outline which technologies have been chosen to complete the cast, and which criteria were used to choose them over other technologies. Kristen Patterson is in charge of the formatting and code used in the project, Brandon Chatham is in charge of the API and how it interfaces with the system, and Jacob Broderick is in charge of how the program is compiled along with how it interacts with external tools. These are all important aspects of determining how an AI can be made for a game that is closed source. Some APIs are not suited for making a scripted program to actually run the game. There are also limitations on which programming languages can even be used for such a task. The game also can only accept certain file types to interface with the program. All of these things are weighed in this document.

XVIII. Technologies

A. Programming paradigm for functions - Kristen Patterson

1) Options:

Paradigm	Description	Reason for Selection
Modular	Paradigm focused on separating functions into independent modules.	Modular has a high focus on independence of functions.
Object Oriented	Paradigm focused on objects such as data and attributes rather than actions.	Objects can help to section off parts of the program.
Recursion	Paradigm focused on solving a larger problem using solution from multiple smaller problems.	The program requires the solution of many smaller sections.

2) Goals: The goal of comparing the chosen programming paradigms for functionalization is to clarify which method of coding is best for our project. It will also help to remind us of how functions need to be set up and whether a section of code needs to be broken down into separate blocks.

3) Criteria:

a) Independent: Functions must be stand alone and should not rely on other functions.

b) Flexible: The functions must be easily changed and modifiable so other programmers can use it.

c) Easily tested: Each function can be tested individually and the results can be easily identified.

d) Portable: A function is portable if it can be taken from our code and applied to another code and work with little to no change.

e) Simple: A function is simple if it has only one purpose.

4) Table:

Paradigm	Independent	Flexible	Easily tested	Portable	Simple
Modular	Highly independent	Can be changed without major issues in the code	Individual tests can be completed with specific results	Mainly interacts with the API rather than other functions	Only serves one purpose
Object Oriented	Independent classes of functions and attributes	Can change code in objects easily but may cause problems in other code	Requires some handling of the object to test a specific function	Objects protected code can be easily ported	An object can serve a wide variety of purposes
Recursion	Requires dependence on other functions or previous occurrences of itself	Difficult to change as it can require changes to other functions	Errors can be difficult to find	Difficult to port to other code without including other functions	Has one goal but multiple steps to get to the goal

5) Discussion: After analysis, modular fits all of the criteria. Object oriented comes close but there is some problems it has. Modular also fits the client's specifications and will make coding the project a lot simpler. Recursion is the opposite of what we want in our system so we will not be using it in anyway.

6) Selection: The client specifically requested that we have our code be modular enough that a coder can grab a function and use it in their program with little to no problem. Keeping the system modular will also help with documenting the code and keeping it well-commented.

B. Programming language - Kristen Patterson

1) Options:

Language	Description	Reason for Selection
C++	Commonly used programming language with many functionalities derived from C.	It is commonly taught and used and is the recommended language from BWAPI.
Java	Class-based programming language that promotes few implementation dependencies.	It can be run on many different systems.
C Sharp	Programming language derived from C mainly used for object oriented code.	It has a high focus on objects while keeping a lot of the functions of C.

2) Goals: Comparing languages will help to settle the environment as well as the core of our assignment. Choosing a language will also help decide the programming practices and format we will decide on.

3) Criteria:

a) Performance: The program created by the language cannot slow down a game in Starcraft Brood War by 1 frame per 10 seconds.

b) Easy to comment: The language must provide a simple system to accurately and efficiently comment our code.

c) Large project support: The language needs an environment that can increase efficiency and organization to help in the development of a large project.

d) Commonly used: A language is commonly used if it is widely taught, understood, and other coders prefer to use it in projects.

e) Simple documentation: There must be a resource that can adequately describe the functions included in the language and how to interact with them.

4) Table:

Language	Performance	Easy to comment	Large project support	Commonly used	Simple documentation
C++	Highest performing language of the three	Provides a way to comment a line or a whole section of code	Allows the use of headers and other source code to further organize projects	Yes	Yes[1]
Java	Has a slight reputation for being slow and having memory leaks	Includes normal commenting plus a way to convert comments to documentation	Uses intricate class and file hierarchy in large projects	Yes	Yes[2]
C Sharp	Like Java is an intermediate language which reduces performance	Includes single line and delimited comments, also includes hotkeys to change lines of code to comments	Searches the whole project for any files specified in the namespace	Yes	Yes[3]

5) Discussion: C++ is an exact fit for what we require from the language. Both Java and C Sharp slow down the program, and while that may be avoided, it is easier to just use a language where that won't be a problem. C++ also is well-known for most computer science majors in Oregon State University which will make it more convenient for future club members that want to use our code.

6) Selection: Due to the client's request, we will be choosing to work in C++. This language will also help in modularizing our system as well as staying within the tournament's guidelines for performance.

C. Environment and IDEs - Kristen Patterson

1) Options:

IDE	Description	Reason for Selection
Visual Studio[4]	IDE created by Microsoft that can create Windows programs.	Highly popular IDE for C++ and C Sharp and recommended by BWAPI.
Eclipse[5]	Cross-platform IDE mainly used for Java. Eclipse requires plugins in order to customize the environment.	Popular IDE for both Java and C++ projects.
IntelliJ IDEA[6]	Java integrated IDE that supports cross-platform and was created by JetBrains.	Thorough tools and development centered around Java.

2) Goals: The goal for the comparison of different IDEs is to distinguish a sleek efficient environment for the developers. The IDE must also provide tools to increase the speed of debugging and must work correctly with the language we select to work with.

3) Criteria:

a) Compatible languages: A language is compatible if an IDE can successfully implement and compile that language.

b) Debugging tools: An IDE must have tools that can adequately help a programmer find, assess, and fix any bugs in the program.

c) Compiling tool: An IDE must include a tool to compile the system in a way that the programmer specifies.

d) Format: The format of an IDE must be readable and well organized for any programmer to be able to digest lines of code easily.

e) Navigation window: A navigation window should be included for programmers to be able to access any other source files and header files quickly and seamlessly.

4) Table:

IDE	Compatible languages	Debugging tools	Compiling tool	Format	Navigation window
Visual Studio[4]	It currently supports C++ and C Sharp. It used to support Java	Has breakpoints, watch windows, and is able to edit source code while program is running	Includes C and C++ compiler and uses a project build system	Can shrink and expand functions and blocks of code for easier reading	Includes navigation window on the side and tabs on top to easily change between files
Eclipse[5]	Supports C++ and Java	Includes breakpoints for line and exceptions and can watch expressions	Built-in compiler, compiles when it is saved	Highly customizable format through extensions	Depends on what extensions are added to customize
IntelliJ IDEA[6]	Only supports Java	Can watch variables and expressions and can use breakpoints	Includes a Java compiler	Can shrink functions	Includes a list of files in a window on the left hand side

5) Discussion: Visual Studio is a convenient environment for working on large projects. All the members are also more comfortable with using Visual Studio than the other IDEs. Since IntelliJ's and Eclipse's main focus is Java we may come across problems since we are working with C++.

6) Selection: Since we are choosing to program in C++, and it is the recommended environment for the BWAPI, we will be working within Visual Studio.

D. APIs - Brandon Chatham

1) Options:

API	Description	Reason for Selection
BWAPI	BWAPI designed to build AI agents.	Includes functionality to automate necessary game mechanics and use information on the current state of the game.
Brood Data API	API focused on data mining for future use in machine-learning algorithms for AI or competitive player training.	Useful for gathering data for design algorithms.
BW Spectator API	API designed for spectator-mode interface customization with real-time game stats and interaction with units.	Useful for observing performance of AI.

2) Goals: The goal for the selected API is to fit the appropriate needs of the team in development of an AI. Additionally, it should be designed such that in the future, the API will support the desire for machine learning implementations using current game data. Finally, it should be well-documented and ideally have a community that has provided further learning tools such as videos.

3) Criteria:

a) AI Functionality: Allows for interaction with the AI agent in order to automate the game mechanics.

b) Current Game Data: Allows for gathering of current game stats in order to make immediate decisions based on that information.

c) Documentation: Thorough explanations of how the API works and instructional information such as tutorials on how to use it.

4) Table:

API	AI Functionality	Current Game Data	Documentation
BWAPI	BWAPI supports interfacing with game mechanics for AI agents.	Provides current game data for data mining or machine learning strategies.	Thorough documentation and extensive learning tools created by the community.
Brood Data API	N/A	Excellent data collection functionality.	Provides minimal documentation and learning tools.
BW Spectator API	N/A	Interface for collecting current data for presenting important game stats to viewers.	Well-documented but minimal additional learning tools.

5) Discussion: Because BWAPI is built to support AI agent functionality, and the others do not, BWAPI is the obvious choice. Furthermore, while Brood Data API is more powerful for data mining for data-driven machine learning algorithms, BWAPI is more well-rounded with learning tools and enough game-data functionality. Lastly, BW Spectator API allows for displaying relevant game stats to viewers but does not provide any functionality to interact directly with an AI agent.

6) Selection: Since BWAPI offers the most versatility of the three options and is the most useful for AI development, BWAPI is the API we will be using.

E. Documentation for API - Brandon Chatham

1) Options:

Documentation Source	Description	Reason for Selection
BWAPI Homepage[7]	The homepage for the BWAPI that provides organized explanations of the libraries included in the API.	Very thorough and well-organized into specific libraries making it easier to find the certain tool you may be looking for.
BWAPI Wiki[8]	Web page with links to BWAPI source code documentations as well as other links to other BWAPI related information such as extensions, tutorials and game fundamentals.	Useful for identifying and explaining all of the tools available to the project.
Source Code Comments[9]	Explanations of code functionality directly within the code.	Concise explanation of code located inside of the code for easy access.

2) Goals: The goal for the selected Documentation source is to find a source that is thorough, well-organized, and authoritative.

3) Criteria:

a) Thorough Information: The documentation should provide information on all of the built-in functionality provided by the API.

b) Organization: The documentation would ideally be organized into subsections for efficiently sorting through material.

c) Authoritative: The documentation should be definitively true and current.

4) Table:

Documentation Source	Thorough Information	Organization	Authoritative
BWAPI Homepage	The BWAPI homepage provides documentation on the entire API.	The information is organized into subcategories to organize functionality.	It is the most genuine source of information for the BWAPI as it was created by the developer who made the BWAPI.
BWAPI Wiki	Extensive links to resources like the BWAPI Homepage, information on BWAPI extensions, and other learning tools such as instructional videos.	Organized into categories however it does not hose the information, it redirects users to other sources.	Some of the resources may not be good for development consider they are outside developer's work thus, is not authoritative.
Source Code Comments	Thorough and applicable information for small, modular pieces of API source code.	Paired with the code the comments refer to.	Authoritative because they were written by the developer who created the BWAPI.

5) Discussion: While the BWAPI Wiki does have a broad reach of resources, it may not be the best immediate source for development as it merely redirects us to the more applicable BWAPI Homepage. Source code comments are extremely useful, but they are not as descriptive as an entire cite can be like the BWAPI Homepage. The comments sometimes tell users how to use a function, but does not provide a holistic explanation of what it is doing.

6) Selection: The BWAPI Homepage is well-organized and provides specific enough information to explain how to use the API while also being the authoritative source and thus, is our choice.

F. Choice algorithm - Brandon Chatham

1) Options:

Type	Description	Reason for Selection
Scripted Tree Selection	Creating a tree with scripted strategies to alternate between in response to game data.	Significantly easier implementation that still provides flexibility between strategies.
Machine Learning - Supervised	AI would base decisions on outcomes of previous instances of decisions and those outcomes.	Identifies patterns in outcomes and can eventually identify the appropriate response based on previous successes or mistakes.
Machine Learning - Reinforcement Learning	Creates a reward system for which the program will attempt to maximize reward by responding to each point of data given.	Powerful when making decisions and evaluating the effectiveness of that decision given the current state of the game.

2) Goals: The goal in choosing a choice algorithm is to select something that is moderately flexible while not requiring an extensive amount of time implementing.

3) Criteria:

a) Flexibility: The choice algorithm should be able to choose between several scripted strategies.

b) Efficacy of Decisions: The choice algorithm should choose the best option it is aware of based on certain criteria regarding current game and/or previous game data.

c) Implementation Time: The choice algorithm ideally will be able to be implemented without extensive knowledge in machine learning or mining a significant amount of data from previous games.

4) Table:

Decision Algorithm	Flexibility	Efficacy of Decisions	Implementation Time
Scripted Tree Selection	Fairly flexible with an unlimited number of possible strategies.	Decisions cannot be evaluated for effectiveness, only if they were the correct decision based on the pre-defined decision algorithm criteria.	Relatively short implementation time.
Machine Learning - Supervised	Very flexible because strategies are fluid, considering all options at all times.	Efficacy is reinforced as the AI attempts to maximize efficacy with each decision based on previous outcomes.	Long implementation time, especially when considering time taken while creating a data pool to use.
Machine Learning Reinforced	Very flexible but slightly constrained to a predefined reward system that determines what a successful decision was.	Efficacy of decisions is reinforced but entirely based on the definition of success is programmed within the reward system.	Long implementation time required to create logic that constantly considers all options with regards to the reward system.

5) Discussion: While both machine learning algorithms would be fantastic to implement and would make for an impressive AI, they are not possible to implement within our projected development schedule.

6) Selection: We will be implementing an algorithm that picks from a tree of scripted strategies. It will decide what the best strategy is based on the current game data.

G. Extensions to the API - Jacob Broderick

1) Options:

Type	Description	Reason for Selection
BWTA	The BW terrain analyzer. Analyzes maps in broodwar to give information about possible spawn locations.	Would be useful for determining map based strategies without having to implement it from scratch.
SparCraft	A tool that simulates battles between specified units. Would assist in scripting how the AI would battle other AIs.	It would be useful for testing battle modules in the AI.
Map Editor	A map editor that allows users to put their AIs in unique situations to see how it handles them.	Extremely useful if one would need to test very specific situations over and over again.

2) Goals: The goal in choosing an extension is to pick something that is useful, multipurpose, hard to implement from scratch.

3) Criteria:

a) Useful: Above all the extension has to be useful for the project to require it.

b) Multipurpose: The chosen extension should allow for multiple different uses so that the project can benefit the most with as little overhead.

c) Hard to implement otherwise: The extension should be something that would not otherwise be able to be implemented in the time frame.

4) Table:

Decision Algorithm	Useful	Multipurpose	Hard to implement otherwise
BWTA	API would allow for the AI to analyze maps which is very useful in determining winning moves.	This would allow us to better determine attack and defense around the map.	This would be something that would be relatively difficult to develop for every single map in Brood War.
SparCraft	This API would be a good way to test the battle effectiveness at different unit values with very little effort	This algorithm would only be useful in testing combat effectiveness of the AI.	This would be something that would be very hard to develop within the time given for this project.
Map Editor	This extension is useful because it would allow for testing in many different situations.	Extremely multipurpose since it would allow for any map/situation to be created.	This would take a very long time to implement from scratch because of the fine tuning for use in the Starcraft system.

5) Discussion: All of these extensions would be very useful if there was unlimited time to learn and explore with them, unfortunately this project has a very short time line compared to the amount of work that can be done.

6) Selection: The Extension that could be utilized best given the time line would be the Terrain analyzer. It would save us time because there is a lot of work involved with map analysis. It also would not add too much overhead to the project to try to learn and apply it.

H. Compiling environment - Jacob Broderick

1) Options:

Type	Description	Reason for Selection
Visual Studio C++ (G++)	GNU compiler for C++ plugged in to visual studio	Convenient compiler because of the chosen IDE (Visual Studio)
Intel Compiler	Compiler released by intel which has a lot of useful optimizations for things like kernel development.	It could be more optimal than other compilers if the project made use of features that it has to offer.
Java Compiler	Compiler used by java to create programs runnable in the java runtime environment	If the project is written in Java this would be the optimal choice

2) Goals: The goals that the proper compiler should meet are: Fast, Compatible, Convenient.

3) Criteria:

a) Fast: The chosen compiler should not take too long in order to generate a usable program for the system.

b) Compatible: The chosen compiler should be compatible with the requirements of the system.

c) Convenient: Ideally the compiler should not add any overhead to the work environment while still being able to perform.

4) Table:

Decision Algorithm	Fast	Compatible	Convenient
G++	Very fast and optimized compiler.	This compiler is compatible with the system.	It is available for use within the IDE that has been chosen
Intel Compiler	The intel compiler is the most optimized compiler of the choices, and should outperform them.	The compiler is compatible with the system that has been chosen.	The compiler would have to be configured in addition to working out the system, it would not be very convenient.
Java Compiler	The slowest of the options, this compiler has a lot of overhead because it has to prepare code for the runtime environment.	The compiler is not compatible with our environment.	The compiler would not be convenient because it would require a change to the entire project

5) Discussion: The c++ compilers would both be useful within the system. Most compilers can perform the functions that are required for this project, some are just more convenient and compatible with what is required by the project.

6) Selection: The visual studio G++ compiler will be the choice for this technology. It is already built in to the tools that will be used to develop this project.

I. Compiling type - Jacob Broderick

1) Options:

Type	Description	Reason for Selection
DLL	A windows library file that can contain information about how to fun a program. It would be a library with how the ai would run	Small compact file that is accepted by the tournament. Recommended by the chosen API
EXE	Windows executable file type. A program output as an EXE can be ran directly	EXE provides flexibility in how the program is ran.
JAR	Java executable program file. Works well for projects using the Java environment	Would be useful with a custom environment that would allow for java files to be ran directly.

2) *Goals:* The goal in choosing the right output is to choose one that is flexible, while also being usable by the API.

3) *Criteria:*

a) *Compatibility:* The chosen output should be compatible with the system that our API chooses.

b) *Size:* The chosen output should be compact to allow for transfer and portability.

c) *Implementation Time:*

4) *Compiler:* The chosen output should work with the compiler that our environment uses.

Decision Algorithm	Compatibility	Size	Compiler
DLL	Completely compatible and recommended by the API	Small enough that it can be sent to tournament sponsors	It can be compiled by the chosen platform
EXE	Compatible with tournament rules and can be used by the API	Larger than dll on average, still okay for sending	It can be compiled by the chosen platform
JAR	Not compatible with tournament rules or the API (Java should output to dll)	Not compatible with the system at all	It can be compiled by the chosen platform with configuration

5) Discussion: The program should compile down to the DLL format. It is the recommended format by the chosen API, and has the least problems with all other rulings. An exe could be used, but there already exists a standard used by many with BWAPI.

6) Selection: We will be compiling down to a DLL for use by the game.

XIX. Conclusion

A. General Findings

The technologies we will be using are what we consider to be best suited for our goals for this project. While most of these technologies were feasible, a few were not within scope.

B. Portability and speed

Our client has expressed a desire for this software to be designed in a manner that is flexible and mobile. This is in large-part why we chose the modular programming paradigm. Additionally, we wanted a low-level language that was compatible with our API which is how we arrived at C++. Furthermore, it is widely suggested development with the BWAPI be done in Visual Studio and we have familiarity with this IDE.

C. API

The BWAPI is the premier API for AI programming with regards to Starcraft Brood War. Not only is it extensive, but the documentation is useful and includes functionality that will prove useful for future developers who plan to develop machine learning algorithms.

D. Extensions and compilation

There are several extensions to the BWAPI which should be considered and while we have chosen one that is most applicable, extensions are more likely to be used once our project is passed onto future developers. Nevertheless, we have chosen BWTA2 for map analysis because of strategic positioning advantages in fights and small learning curve. Furthermore, in choosing Visual Studio as our IDE, we have chosen G++ because it is built-in. Lastly, we will be delivering our project as a DLL for our AI competition because that is the suggested format by BWAPI.

XX. Bibliography

[1]"Reference - C++ Reference". Cplusplus.com. N.p., 2016. Web. 14 Nov. 2016. http://www.cplusplus.com/reference/

[2]"Java SE Documentation - Tutorials". Oracle.com. N.p., 2016. Web. 14 Nov. 2016.
http://www.oracle.com/technetwork/java/javase/documentation/tutorials-jsp-138802.html

[3]"CSharp Reference". N.p., 2016. Web. 14 Nov. 2016. https://msdn.microsoft.com/en-us/library/618ayhy6.aspx

[4]Microsoft, "Visual Studio IDE". Visual Studio. N.p., 2016. Web. 14 Nov. 2016. https://www.visualstudio.com/vs/

[5]Eclipse, N.p., 2016. Web. 14 Nov. 2016. https://www.eclipse.org/ide/

[6]JetBrains, "Intellij IDEA The Java IDE". JetBrains. N.p., 2016. Web. 14 Nov. 2016. http://www.jetbrains.com/idea/

[7]"Brood War API." BWAPI: Main Page. N.p., n.d. Web. 14 Nov. 2016. http://bwapi.github.io/index.html

[8]"Main Page." StarCraft AI, the Resource for Custom StarCraft Brood War AIs. N.p., n.d. Web. 14 Nov. 2016. http://www.starcraftai.com/wiki/Main_Page

[9]Bwapi. "Bwapi/bwapi." GitHub. N.p., 2015. Web. 14 Nov. 2016. https://github.com/bwapi/bwapi

XXI. Revisions

A. *Programming paradigm for functions - Kristen Patterson*

1) Discussion: Upon further inspection, object oriented programming is much simpler to organize and is easier to grab and pick up for new users. The biggest problem with object oriented programming is the dependencies that classes can have. In order to make it simpler for users to select specific sections of our code to use in their code, classes are kept as simple as possible.

2) Selection: The choice of paradigm has switched from modular to object oriented classes making the projects well organized. Classes also it is similar coding style as BWAPI, our chosen API to interact with the game.

B. *Environment and IDEs - Kristen Patterson*

1) Discussion: With further investigation, BWAPI requires Visual Studio Community version 2013 in order to build and compile the AI.

2) Selection: The team can use either Visual Studio Community version 2013 or version 2015 with v120 compiler tools.

C. *API Documentation - Brandon Chatham*

1) Discussion: Shortly after beginning development, we realized that there are already well-known AI's built and available on Github. This has been a supplemental tool for me especially as I use it for architectural insight into how these modules can be designed. Additionally, it provides countless examples of how to use the BWAPI while other BWAPI resources mostly focus on functionality without examples. Lastly, these established AI modules are developed by at least two professional software developers. Bug reports, code changes/fixes and version releases for these modules are all documented. With that said, it is safe to assume these developers and their code are an authoritative, and up-to-date source, much like our criteria required.

2) Selection: The team has begun using these other AI module implementations as learning tools to supplement other documentation.

XXII. Weekly Blog Posts

A. Kristen Patterson

Fall 3

Last week I had clarified the exact details of the project with my group mates and also contributed to the problem statement by revising the abstract, writing the problem definition, and writing both the introduction and some of the conclusion.
Planning with teammates about the requirements document and discussing problem statement with client.
Had some clarification issues with some details in the problem statement. Client was also on vacation so could not verify whether the document fit his criteria early enough.

Fall 4

Last week our group had met with our client and had him look over our problem statement and provide feedback. I also corrected some grammatical and formatting errors with my section of the document. I also corrected the document based on the client's feedback.
Next week we need to get our client to sign our final draft of the problem statement. We also are going to have a rough draft of a requirements document by Wednesday to have ample time to set up a meeting to discuss with the client.
We had a problem when scheduling our meeting with the client as not all the group members were able to attend. I also had trouble figuring out why apostrophes would not show up properly on our tex document.

Fall 5

Last week I had completed the introduction, formatting, gantt chart, and abstract of the requirements document.
Next week we will meet with our client and discuss the requirements document as well as revise any issues we have with our document.
We had trouble meeting with our client to get a signature as he was very busy. I also had some time management problems as it was a very busy week.

Fall 6

Reviewed SRS document with client and made changes based on input.
Discuss with team about following documents and clarify materials needed with client.
Our team was fairly busy this week so there was a decent amount of time management problems.

Fall 7

Got the final draft of the SRS document signed and submitted as well as clarified the technology review with our teachers and TA. Also broke our project down into 9 parts and split up the project.
Finish my portion of the tech review then make correction and meet with client to make sure it is acceptable.
I had time management problems as I had other assignments due. There was also some clarification issues on what our 9 parts of the tech review should be.

Fall 8

Finished our Tech Review and submitted it. Discussed course of action for design document and progress reports.
Discuss format of final progress report and make any revisions to the tech review.
Mainly just time management problems again as this has been a busy term. Also need to clarify some questions for upcoming documents.

Fall 9
Received revisions for Tech Review and discussed changes to them as well as assigned roles for the design document.
Complete my section of design document as well as my progress report and presentation.
No problems this week.

Fall 10
Completed my section for the Design Document and Progress Report.
N/A
I have encountered some family issues that produced some problems in my schedule.

Winter 1
Set up the environment to work on the project.
Start gathering modules for and practice using BWAPI.
Trouble getting a new schedule to meet with our team.

Winter 2
Discussed timeline for project and started economy portion of code.
Continue working on economy portion.
Had a lot of other assignments that made it difficult to have more time for this project.

Winter 3
Economy and Construction portion of code is finished.
Work on unit production and control.
More time issues and sickness as well.

Winter 4
Started functions for training and selecting marines.
Start function for moving and attacking.
Just more time issues.

Winter 5
Finished my portion of revisions and progress report and started code for alpha video.
Finish alpha video code and presentation.

I had 3 midterms so I had less time to focus on the project than I wanted.

Winter 6

Finished all the revisions and progress report

Participate in elevator speech practice and start documentation of code

Time management

Winter 7

Practiced elevator pitch for expo.

Start Documentation

Didn't get anything done as I was busy on other assignments.

Spring 1

Wrote up documentation for scraper and scouting class.

Come up with visuals for the second draft of our poster.

Set up of new schedule with team.

Spring 2

Discussed poster draft 2 with Kirsten for feedback and corrections and made revisions. Also found good visuals to use in poster.

Meet with group for group photo and meet and discuss poster with our client.

Was unsure of the best type of visuals to use for the poster.

Spring 3

Completed poster draft 2 and discussed meeting time with client. Also completed flowchart for visual on poster.

Meet with client to discuss final poster and make corrections.

Had trouble trying to come up with visuals for the poster and the general structure but I ended up going to Kirsten and she helped.

Spring 4

Discussed poster with client and made final corrections to code as well as discussed deliverables with our TA.

Make final corrections to the poster and submit it before the deadline.

We had some trouble trying to set up a time to meet with our client but because we had contacted him early we had plenty of leeway before the deadline.

Spring 5

Completed Wired interview.

Prepare for Expo.

Had trouble trying to set up an appointment for the interview.

Spring 6

Completed What we have so far portion of the Progress Report and the project details portion of the presentation.

Prepare for Expo.

Had trouble splitting up the work for the group.

Spring 7

Completed attacking function in program and prepared video for expo as well as participated in Expo.

Class sweep up.

Tried to finish up everything quickly for expo.

Final Post

If you were to redo the project from Fall term, what would you tell yourself?

I believe one of the biggest issues my team faced was that we got caught up in some of the more non-essential parts of the project and we didn't focus on completing the project before working on stretch goals and interesting mechanics.

What's the biggest skill you've learned?

The biggest skill I have learned during this process is how to be able to talk about a project on multiple different levels of complexity.

What skills do you see yourself using in the future?

The skills I have learned on how to explain my project and how to properly work in a team and manage my time appropriately.

What did you like about the project, and what did you not?

I liked the concept and the coding of the project. It just felt like the schedule was catered more towards larger and more complex projects and I would have like to have more time focusing on development rather than extraneous planning.

What did you learn from your teammates?

My teammates taught me a lot of technical skills that they were used to using and I believe I also helped them with it as well.

If you were the client for this project, would you be satisfied with the work done?

I believe we provided a good starting point for students to use to supplement their research and learning.

If your project were to be continued next year, what do you think needs to be working on?

The actual artificial intelligence portion of the code would be the focus as our project's main purpose was to supplement the development of that process.

Speak a little about your expo experience.

I spoke with many individuals ranging from industry reps to people who did not really know what a video game was.

B. Brandon Chatham

Fall 3

Next week we plan on meeting with Dr. Fern to talk over our drafted problem statement. We did our best to write it this week with what we had gathered from our first meeting with him about what he imagines the project being. Hopefully, what we have is in-line with what he wants and he will sign-off on it.

This week we did some research on what is required for our competing in the Starcraft AI competition. We also started looking at some other agents created by other universities to see what other people have done. I familiarized myself with the BWAPI and did some research on what the benefits of our creating a sandbox for our project will be and why it was necessary. Sandbox development is a pretty basic and fundamental component to our project that I really had no familiarity with so I wanted to make sure I caught myself up to speed in that respect.

Our primary problem was Dr. Fern being out of the office this week, but that was not a big issue. Also, I still need to learn LaTeX which is not exactly a problem, but something I need to focus on. I had an issue trying to mess around with the BWAPI this week because in order to use it, I need to install a version of Stacraft Brood War which costs 15 dollars. I figured Dr. Fern might have a solution to that so I did not buy it, yet.

Fall 4

This week we have been working on revising our problem statement and making sure it is up to par. Also, we met with all of our advising sources to sort out what those revisions might be. Tomorrow, they will be completed.

Next week we will start working on our next document. I need to start looking into the IEEE standard and familiarize myself with those standards.

We had to meet with Prof. Fern a bit later than we had hoped because he was out, but we should be up to speed.

Fall 5

We worked on revising our problem statement. Additionally, Kristen made a gantt chart for us and we pulled together our requirements document rough draft.

Next week we will meet with our client to talk over what our requirements document should have.

I had trouble balancing my coursework with a large project in Operating Systems due and trying to get myself organized for the career fair. Also, we had some issues getting our document signed by our client but we sorted that out.

Fall 6

We received our feedback on the problem statement and will hopefully talk over how we can improve it soon. Kristen worked more on refining the requirements document after getting feedback from Prof. Fern on what he felt needed to be changed.

Next week we will meet on Tuesday, begin working on other documents, and potentially fine-tune our problem statement.

I had some trouble getting adjusted to the formatting expectations of the IEEE document specifications and am still getting used to LaTeX.

Fall 7
Started planning our work for the Tech Review document and talking about what technologies we will be talking about.
Writing our Tech Review for Monday and beginning to familiarize ourselves with the resources we will be working with for the project.
Not many problems other than two of our team members being sick but we have been able to keep up with the workload.

Fall 8
We finished our work on the Tech Review.
Meet with group to start talking about our final project presentation and the design doc.
Trying to wrap my head around the IEEE format for the design doc and how I'll apply what I am focusing on for the project.

Fall 9
Got feedback on our work for the Tech Review. We were not exactly thrilled with our overall grade but we plan to make some good revisions.
Meet with team to talk about our upcoming work for final presentation, design doc, and the tech review revisions.
Still trying to make sure I'm correctly understanding what is exactly expected for this next doc.

Fall 10
Worked on and completed my design document portion.
Trying to find a way to work around Kristen's unfortunate family situation to get our final presentation done.

Winter 1
I set up my machine with Starcraft Broodwar and BWAPI. I started fiddling around with some of the example modules and familiarized myself with how to start running the AI modules on our local machines.
Start writing code for the resource module which will allow us to start expanding into other modules once we have resources to spend.
Having some difficulties finding times for our group to meet and struggling to balance school with many job applications.

Winter 2
Met with the team and started talking about our individual development schedules.
Writing code and pushing to our repository.
Same as last week

Winter 3
Started getting development organized and got to the ”starting line” for writing code and wrote code this week which we

pushed.

Continue working on modules and utilizing them in our AI module. The resource module is nearly complete with basic functions to get the AI started. We will need to develop more sophisticated logic behind optimizing resource gathering in order to best suit our strategy and really, and economic desires. (The more resources the better in all scenarios as any improvements should improve every possible strategy.)

None

Spring 1

Worked on the scraper implementation.

Talk with Kristen about the documentation for the scraper.

None.

Spring 2

Worked on the scouting class a bit.

Work with the team on deciding our poster layout.

Finding time to meet with our client.

Spring 3

Working on finalizing our project and making sure our expectations are met.

Meet with client and fix any minor bugs.

One persistent bug in our resource gathering class.

Spring 4

Fixing the expansion bug.

Bring in a replay processor for our data mining.

None.

Spring 5

Still need to fix the expansion bug.

Get our midterm report done and our pitch ready.

None.

Spring 6

Tried to fix the expansion bug but were not able to.

Get our midterm report done and our pitch ready.

Couldn’t fix the bug.

Spring 7

If you were to redo the project from Fall term, what would you tell yourself?
To pay more attention to machine learning neural networks and how to fit that into the project.

What's the biggest skill you've learned?
I have developed my skill to teach myself new technologies outside of the classroom setting which will be invaluable in the workplace setting.

What skills do you see yourself using in the future?
My ability to convey my technical ideas and skills as well as my further-developed self learning skills.

What did you like about the project, and what did you not?
I did not love how little guidance we received from our client but I did enjoy being able to learn through trial and error. Also finding ways to fix our mistakes once we realized we had made them.

What did you learn from your teammates?
A lot about technical skills and how to best work together on such a large project.
If you were the client for this project, would you be satisfied with the work done?
The package we developed is pretty decent and I think it could be a good starting point for new developers in the AI club he plans to start.

If your project were to be continued next year, what do you think needs to be working on?
Definitely implementing neural networks that make our AI more dynamic. As is, we have created an interface to interact with it, but the neural network would take our project to the next level.

Speak a little about your expo experience.
Expo was probably my least favorite part of the capstone project. While it was fun explaining the project to interested people, it was a bit of an elongated event in my opinion.

C. Jacob Broderick

Fall 3
Work for last week Last week Brandon and I met with Dr. Fern. We figured out exactly what the project should look like and what the expectations were. We will be building towards a tournament viable agent that a club could build on.

Work For this week This week we set up the repository, and got al of the invite sent out. We will work on hammering out details for our project, such as the starting race we would like to use.
Fall 10
Work done this week: Worked on the design document, specifically the decision making portion.

Set backs: I had the flu and was unable to attend class.

Spring 7

If you were to redo the project from Fall term, what would you tell yourself?

Spend more time focusing on strategy so that it could win more.

What's the biggest skill you've learned?

I learned how to use APIs that can transfer to real world use.

What skills do you see yourself using in the future?

Being able to talk with other members in a group to ensure everyone is on the same page.

What did you like about the project, and what did you not?

I wish that we had a little bit more information on what exactly we were supposed to do at times. I did enjoy being able to solve and learn about problems related to artificial intelligence.

What did you learn from your teammates?

I learned how to work together with other teammates better.

If you were the client for this project, would you be satisfied with the work done?

It was what the client had asked for. He would maybe be a little more happy if others could pick up the projects, but it will be good for others to work on it.

If your project were to be continued next year, what do you think needs to be working on?

The AI needs to win a game, and win more consisitently.

Speak a little about your expo experience.

I did not enjoy expo. I have a herniated disk and had trouble standing for that period of time. Very few people actually stopped by and most weren't interested in anything except for the fact that it was a project about a video game.

XXIII. Final Poster

XXIV. Project Documentation

A. Project Walkthrough

The project is sectioned off into many classes for a library described below as well as an AI module. This AI module is the main location where the interaction with the game occurs. It has a startup phase where it converts the play map into data for the AI to use. After the initial startup phase the AI starts a nested for loop where it goes through each individual unit and gives it command. The worker is in charge of gathering materials, constructing buildings, and scouting. The Command Center is in charge of training workers. The barracks are in charge of training military. The marines are in charge of moving to a choke point and attacking the enemy. It does this loop each frame until a victor is decided. A victor is decided when on of the players buildings and enemies are all destroyed.

B. Installation

In order to install this project all that is needed are the supplementary programs described below and cloning from our repo.

C. Running Project

After everything is downloaded and code has been implemented it is required to compile and import the data into the game. In order to compile the game you need to go into ExampleAIModule in the Project folder. Then you need to run ExampleAIModule.vcxproj. When the project loads up change the solution configuration to Release not Debug In the Solution Explorer window, right click ExampleAIModule and select Build and check to see it was built correctly. Go back to the Project folder and go into the Release folder. Here you will find ExampleAIModule.dll, copy it. Go to where StarCraft Brood War is installed and find or create a folder called bwapi-data and inside it find or create a folder called AI. Inside the AI folder paste the copied file ExampleAIModule.dll and start up ChaosLauncher found in BWAPI. Make sure to run ChaosLauncher as administrator. In ChaosLauncher make sure BWAPI Injector(1.16.1)RELEASE is checked and highlighted then select config. In the config menu make sure the path for ai goes to the path where you pasted the .dll file. Now go back to the ChaosLauncher and click Start then create a custom game and it should be set.

D. Requirements

In order to install the project and correctly use it it is necessary to download other supporting software as well. It is also preferable to be working on a device running at least Windows 7. Below is a list of the required programs to be installed:

StarCraft Brood War
Must be updated to version 1.16.1
Visual Studio Community 2013
Can also be Visual Studio Community 2015 with 2013 compiler tools
BWAPI 4.1.2
BWTA 1.7.1

Important! Blizzard has taken down their installer for the original game without any patches and replaced it with version

1.18 included which does NOT support this API. As of late, there is no official installer that can be used to get version 1.16.1.

E. API

1) ResourceGathering: Purpose:
The purpose of the ResourceGathering class is to start the game and have functions pertaining to the economy of the game. The class contains basic functions and variables to keep the gathering of minerals and gasses efficient.

Functions:
buildWorker
Input
Command Center
Processes
If the Command Center is idle and fails to construct a worker more supply will be built. Also, if supply is within 4 of the maximum supply, the AI has enough minerals, and it is more than 3 minutes into the game, then more supply will be built.
Output
Either a worker is made, a supply depot is made, or nothing happens.

workerGather
Input
Command Center
Processes
Uses type of Command Center to determine race. Selects a worker to perform construction of supply structure.
Output
Worker is selected to be able to perform processes.

gatherGas
Input
Worker
Processes
Commands the worker to gather gas.
Output
Returns true if the worker is correctly directed to gather gas and false otherwise.

gatherMinerals
Input
Worker
Processes

Commands the worker to gather minerals.

Output

Returns true if the worker is correctly directed to gather minerals and false otherwise.

getMineralCount

Input

None

Processes

Gets in-game amount of minerals.

Output

Count of minerals.

getGasCount

Input

None

Processes

Get in-game amount of gas.

Output

Count of gas.

Variables:

The only variables for the class is optimum gatherers for both minerals and gas and the current gatherers for minerals and gas.

2) Building Construction: Purpose:

The purpose of this class is to build any buildings necessary to play StarCraft. Any other buildings can be included in this class following this same pattern.

Functions:

buildCenter

Input

Command Center, Building location(optional), player flag

Processes

Uses type of Command Center to determine race. Selects a worker to perform construction of an expansion.

Output

Command Center is built.

buildSupply

Input

Command Center, player flag

Processes

Uses type of Command Center to determine race. Selects a worker to perform construction of supply structure.

Output

Supply depot is built

buildGas

Input

Command Center, player flag

Processes

Uses type of Command Center to determine race. Selects a worker to perform construction of gas structure.

Output

Gas structure is built.

buildBarracks

Input

Command Center, player flag

Processes

Build Terran Barracks if the input is of Terran type.

Output

Barracks is built.

buildGateway

Input

Command Center, player flag

Processes

Build Protoss Gateway if the input is of Protoss type.

Output

Gateway is built.

buildSpawningPool

Input

Command Center, player flag

Processes

Build Zerg Spawning Pool if the input is of Zerg type.

Output

Spawning Pool is built.

checkConstructionStarted

Input

Player flag

Processes

Check construction flags and switch them off accordingly.

Output

Construction status is checked.

Variables:

None

3) UnitAction: Purpose:

The purpose of this class is to be able to control units individually to perform actions. This includes buildings as well as other movable units.

Functions:

checkUnitState

Input

Unit whose state is being validated.

Processes

Checks the state of the input unit.

Output

True if unit is not dead, being constructed, disabled

trainMarines

Input

Barracks to train marines, player flag

Processes

Starts process to train marines.

Output

Marines are trained.

selectArmy

Input

None

Processes

Selects all marines.

Output

Marines are store in a queue.

Variables:

None

4) PlayerInfo: Purpose:

The purpose of this class is to store variables to keep track of the game state that would otherwise be a global variable.

Functions:

adjustMineralOffset

Input

player flag

Processes

Adjusts the mineral offset to store the process of building until it is actually built.

Output

Minerals are adjusted.

adjustGasOffset

Input

player flag

Processes

Adjusts the gas offset to store the process of building until it is actually built.

Output

Gas is adjusted.

resetMinerals

Input

player flag

Processes

Reset the mineral offset to zero.

Output

Offset is reset.

resetGas

Input

player flag

Processes

Reset the mineral offset to zero.

Output

Offset is reset.

Variables:

Flags to store what buildings are being built, counts of our current barracks and expansions, and offsets in the resource counts for planned buildings.

5) MapTools: Purpose:

The purpose of this class is to convert the map being played into data for the AI to interpret and use.

Functions:

getNextExpansion

Input

None

Processes

Finds next resource location to build an expansion.

Output

The tile position to build the base.

getAbsoluteTileDistance

Input

Tiles at starting and ending locations.

Processes

Calculates absolute distance not considering terrain.

Output

Distance

Variables:

None

6) ScoutManager: Purpose:

The purpose of this class is to send out units to scout and be have units attack enemies that are seen.

Functions:

scoutStartLocations

Input

Unit that will be sent to scout.

Processes

Checks all possible base starting locations starting with the furthest away from your starting base.

Output

Reference to the scout.

attackTarget

Input

Unit that was sent to scout and the target to attack

Processes

Attacks the target with the scouting unit.

Output

Unit attacks target.

Variables:

Flag to see if the unit is scouting and the unit that is scouting.

Below is a website providing a tutorial and documentation for BWAPI:

http://www.teamliquid.net/blogs/485544-intro-to-scbw-ai-development

https://bwapi.github.io/annotated.html

XXV. New Technology

This project helped us familiarize ourselves with new API's and the process of developing a library for other developers. Usually in an academic setting, no one else is using your code. This gave us a new perspective. We were not writing code to just solve a problem, but instead helping solve a problem in an easy to understand way as well as design it such that it may be useful to solve even greater problems. We had to think of what a developer might want to do with our API before we even considered how we would write wrappers for the other API's. The primary resource used was: http://www.starcraftai.com/wiki/Main_Page. This web page encapsulated nearly everything the team needed with links to all of the tools we used. Additionally, when choosing our development tools and language, we consulted this page because it provides information on different technologies to use when interacting with Starcraft AI. From here, the team was able to determine what best fit the needs of the project and made it easy to know what tools were available that should be considered as the project progressed. The team did not consult many books on this topic. For the most part, the team used a small amount of previous knowledge from AI coursework and leveraged online sources. The same goes for on-campus resources. Only one or two on-campus resources were used sparingly if at all.

XXVI. What was learned

A. Brandon Chatham

I mostly familiarized myself with machine learning, data mining, and the process of developing a library. There was a great deal of trial and error getting the tools configured and ready to use. This caused me to reach out to developers of both API's we used more than once. I gained a better idea of what it actually means to create industry level code as I was comparing my implementations and ideas to those of sophisticated AI modules that were developed by professional software developers for a competitive setting. As for non-technical experience gleaned from this project, it helped me learn how to effectively talk about my skills as a developer and best explain my previous work. I have found that I do a much better job of conveying my technical ideas in interviews and I think that is largely due to the confidence this project has given me. While it was not a huge up-taking, I was constantly trying to fix something or familiarize myself with something new and this now gives me the confidence that if I need to pick something up for a job, it will be a piece of cake. Project work is difficult. I found myself going through periods of not being able to make much progress and other times making leaps in implementations. It was extremely tedious trying to configure the API tools at times but it was very important that I figured those issues out in a timely fashion considering the team needs the code to work properly if progress is going to be made. It did not teach me much about project work that I hadn't learned from playing soccer (your team relies on you, encourage them, etc.) but it was more practice to prepare me for working in the real world comparable to what my internships had given me. Project management requires you to wear many hats. You need to plan for the future while ensuring the current tasks are being taken care of. Furthermore, you need to understand the situations of your teammates who are working on the project and evaluate what is actually realistic considering their workload, personal life, etc.. Working in teams is nothing new for me. Playing 20 years of organized sports has given me one of my best skills: seamlessly coming into a team and doing my best to find my fit while applying my skill set as effectively as I can without intruding on others. My first intention is to work hard and communicate effectively. If a leadership role arises and it fits me like it did on this project, I'm happy to take it. This was another opportunity to develop my skills in this area. If I could do it all over, I would spend more time trying to figure out how to create a basic neural network for our library. That would be have been an outstanding tool for new developers to familiarize themselves with and would have been awesome to explain to employers how we did it.

B. Kristen Patterson

The technical information I learned is developing a library for a large project, how to properly organize and document a large project, and the intricacies of machine learning. It also put a great importance on time management as even setting up the environment to program was such a lengthy task it could much more time than I originally thought. Since we were building a library, I also had to frequently search through documentation and try and decipher what each function use was and check to see if there was a more efficient alternative. The practice in machine learning also helped me to realize the numerous amount of conditionals that could be found when trying to replace a human with an agent. The non-technical information I learned was the experience i have taken away for a project as big as this. I have not really worked on projects as large as this and a lot of it was easily translated into other fields. Since I had worked on this project for so long and talked about it continually for months on end I was able to describe it in any level of complexity. This translates well in being able to talk about any project or have confidence in any of my methods. In order to be able to answer quickly and show that I understand my field clearly I need to just spend time to voice my projects and actions. Group work and project work require a large amount of time and dedication. When working on projects and in a group it is important to spend time organizing and try to make a schedule and keep it. It is also important to be able to designate time and resources appropriately before any work even begins. I have also realized that projects require a level of compromise and being able to give features and functions a priority based on how necessary they are to a projects success. Managing a team and project can be fairly difficult. Not only do problems show up in the form of bugs or realizing functions will not be able to be delivered, but they can also be in the form of lack of time to complete certain functions and not being able to accurately distribute resources to parts of a project. It also requires being able to understand the group's strengths and where they are best utilized. Interaction with teammates can be slightly hectic. As an individual apart of a team it is important to be able to show empathy and understanding for any misgivings that may occur in a teammates life. It is also important to not try to burden teammates with large workloads due to issues on your end. While rough, it is also important to try and keep the group on task and motivate them or provide help with their work. If I could redo this whole experience I would have mainly tried to focus more on the core requirements of the project before focusing on the stretch goals. I would have also liked to give myself more free time to work on the project in the winter by taking a lighter class load. If I had more time I would have been able to start on the actual artificial intelligence portion of the code as that would have made our project more attractive.

C. Jacob Broderick

I learned how to take an api and convert it into functional code that furthers a goal. The BWAPI was well documented for what each things does, but didn't offer much information on how to actually use it. I spent a lot of time researching what the AI needs to do to be successful. A lot of research was put in to starcraft and what is required to actually play a game of starcraft. There are a lot of little things that are important for figuring it out. I learned a lot about how to fit someone else's code for my purposes. We needed to use tools created by other people to assist with the api. BWTA was a tool that was very helpful for moving the project forward. It caused lots of problems for the project because it was extremely hard to configure.

I learned how to write documentation about code that I have written. This is a skill that is usually not taught in the classes that I ahve taken. I learned how to express what I mean by a section of code and how to turn requirements into code into documentation. This skill helped when writing all of the individual docuents that were required to move the project forward. During internships I didn't need to write as much because there was a team member who's job was to write the parts that I didn't. Having more exposure to writing has helped me understand what goes into the other parts of the team than just coding.

I learned a lot about balancing projects with this class. I have worked on many different projects for a couple different jobs. I usually didn't have a bigger project alongside them. Juggling this project with work responsibilities actually added a lot more work than I initiallyl thought I would require. This is something I will be able to take further when I work on real world projects full time. It would be helpful to know how to juggle projects when I am given multiple projects to work on.

I learned little about project management with this project. I didn't take a role that helped me manage the other team members. I instead did the work that I was given. I know that project management can be very difficult and it is a skill that I would like to learn. I did learn that I need to be easier to manage. There were times where I didn't do as much work as quickly as needed. I learned that working in teams is pretty difficult. Sometimes tasks were completed by team members when others were working on them. It didn't happen often but it did happen. If I were to do this all over I would spend more time on this class. I struggled with depression and it made it difficult to always be available to complete tasks. I let my teammates know about this stuff, but I think it would have been better if I were a good member of the team.

XXVII. Appendix 1

A. Code Segments

Below are some segments of code that are important to our program and how it runs.

```
void ExampleAIModule::onFrame()
{
    // Called once every game frame

    // Display the game frame rate as text in the upper left area of the screen
    Broodwar->drawTextScreen(200, 0, "FPS:␣%d", Broodwar->getFPS());
    Broodwar->drawTextScreen(200, 20, "Average␣FPS:␣%f", Broodwar->getAverageFPS());

    // Return if the game is a replay or is paused
    if (Broodwar->isReplay() || Broodwar->isPaused() || !Broodwar->self())
        return;

    //BWTA draw
    if (analyzed)
    {
        drawTerrainData();
    }

    if ( analysis_just_finished )
    {
        Broodwar << "Finished␣analyzing␣map." << std::endl;;
        analysis_just_finished = false;
    }
    // Prevent spamming by only running our onFrame once every number of latency frames.
    // Latency frames are the number of frames before commands are processed.
    if (Broodwar->getFrameCount() % Broodwar->getLatencyFrames() != 0)
        return;

    //Check for building flags if minerals/gas offset flag(s) are set and we need to check if the offset needs
        to be reversed.
    if(player->mineralsOffsetFlag || player->gasOffsetFlag)
        BuildingConstruction::checkConstructionStarted(player);

    // Iterate through all the units that we own.
    for (auto &unit : Broodwar->self()->getUnits())
```

```
{
    if (!UnitAction :: checkUnitState(unit))
        continue;

    if (unit->getType().isWorker() && unit != player->scoutManager->scoutingUnit)
    {
        if (!player->scoutManager->unitScouting && player->enemyBase == BWAPI::TilePositions::None
            && analyzed)
        {
            ScoutManager:: scoutStartLocations(unit);
            player->scoutManager->unitScouting = true;
            player->scoutManager->scoutingUnit = unit;
        }
        else
            ResourceGathering :: workerGather(unit);
    }

    // Start performing actions with unit.
    // If the unit is a worker unit.
    if (unit->getType().isWorker())
    {
        ResourceGathering :: workerGather(unit);
    }
    else if (unit->getType().isResourceDepot() && !player->expansionCount &&
        ResourceGathering::getMineralCount() + player->buildingMineralsOffset >= 400)
    {
        Broodwar << "Expanding" << std::endl;
        // player->expansionCount++;
        player->adjustMineralOffset(-400);
        TilePosition newExpoTile = MapTools::getNextExpansion();
        Broodwar << newExpoTile << std::endl;
        BuildingConstruction :: buildCenter(unit, newExpoTile, player);
    }
    else if (unit->getType().isResourceDepot() && Broodwar->self()->supplyTotal() * .8 <
        Broodwar->self()->supplyUsed() && ResourceGathering::getMineralCount() +
        player->buildingMineralsOffset >= 100) {
        BuildingConstruction :: buildSupply(unit, player);
    }
```

```
        else if (unit−>getType().isResourceDepot() && player−>barracksCount <
            Broodwar−>self()−>supplyUsed()/6 && ResourceGathering::getMineralCount() +
            player−>buildingMineralsOffset >= 150)
        {
            BuildingConstruction :: buildBarracks(unit, player);
        }
        else if (unit−>getType() == UnitTypes::Terran_Barracks && Broodwar−>self()−>supplyUsed() / 2 < 100)
        {
            UnitAction :: trainMarines(unit, player);
        }
        else if (unit−>getType().isResourceDepot() && Broodwar−>self()−>supplyUsed() / 2 < 16) // A resource
            depot is a Command Center, Nexus, or Hatchery
        {
            ResourceGathering::buildWorker(unit, player);
        }



    }
}
```

This is the on frame function. It determines most of the things that happen during the normal running of the AI. The function is extremely customizable. This particular code segment determines when buildings are built. This is the heart of the program and what makes it run.

```
void BuildingConstruction :: buildBarracks(BWAPI::Unit base, PlayerInfo∗ player)
{
    UnitType terranType = UnitTypes::Terran_Barracks;
    Unit barracksBuilder = base−>getClosestUnit(GetType == terranType.whatBuilds(). first && (IsIdle ||
        IsGatheringMinerals) && IsOwned);

    if (barracksBuilder && ResourceGathering::getMineralCount() + player−>buildingMineralsOffset >= 150)
    {
        if (terranType. isBuilding () && !player−>buildingBarracks)
        {
            TilePosition targetBuildLocation = Broodwar−>getBuildLocation(terranType,
                barracksBuilder−>getTilePosition());
            if (targetBuildLocation)
            {
                // Register an event that draws the target build location
                Broodwar->registerEvent([targetBuildLocation, terranType](Game*)
                {
                    Broodwar->drawBoxMap(Position(targetBuildLocation), Position(targetBuildLocation +
                        terranType.tileSize()), Colors::Blue);
                },
                    nullptr,    // condition
                    terranType.buildTime() + 100);   // frames to run

                // Order the builder to construct the barracks structure
                barracksBuilder->build(terranType, targetBuildLocation);
                player->barracksCount++;
                player->adjustMineralOffset(-150);
                player->buildingBarracks = true;
            }
        }
    }
}
```

This is the build barracks function. It is like all of the other build functions. It gets the type of building, and it selects a unit to build the building and then checks if the conditions are right to build the building. If they are correct then it will go ahead and build the building. it will then update the player class to let it know how many barracks are created. All functions use an offset to determine if there are enough minerals to build the building after all other buildings in queue are done.