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.

CONTENTS

I. Introduction

II. Technologies

A. Programming paradigm for functions

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) Discusion:

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

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) Discusion:

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

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) Discusion:

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

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

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

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

H. Compiling environment

I. Compiling type

III. Conclusion

IV. 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