Requirements: Web-Based Strategy Game With Complex AIs

Josh Polkinghorn
BACS

Alana Weber
BSCS

Faculty Mentor: Laurie Murphy

CSCE 499, Fall 2012

Contents

1 Introduction **4**

2 Project Description **5**
2.1 Functional Objectives 5
2.2 Learning Objectives 6

3 Development resources **7**

4 Requirements **8**
4.1 Performance Requirements 8
4.2 Design Constraints 8
4.3 User Characteristics 9
4.4 Assumptions 9
4.5 Security 9
4.6 Reliability 10
4.7 Portability 10
4.8 Maintainability 10
4.9 External Interfaces 11
4.9.1 Client-Side Interface with the User 11
4.9.2 Game Start Screen 11
4.9.3 Game Play Screen 11
4.9.4 Win / Loss Screen 11
4.9.5 Server Interface with Client 12
4.9.6 Server Interface with the Virtual Players 12
4.10 Use Case Models 12
4.10.1 Start 12
4.10.2 Take Turn 13
4.10.3 Win or Lose 13

5 Task Breakdown **14**

6 Preliminary Timetable **15**

7 Budget **15**

8 Development Documentation **15**

9 Glossary **17**

10 Appendix: Interface Figures **18**

List of Figures

1 Gantt Chart 15
2 Game Start Screen 18
3 Game Play Screen 19
4 Game Play Screen With Armies Pop-up 19
5 Game Over Pop-up on Loss 20
6 Architecture 20

List of Tables

1 Start Use Case 12
2 Take Turn Use Case 13
3 Win Or Lose Use Case 13

1 Introduction

The infusion of artificial intelligence in the everyday lives of people during the past few decades is staggering, and the effect has been positive. One need not look very far to find examples of its influence, from constructing cars, to performing delicate microsurgery, to assisting the elderly and disabled, to predicting the money market. Machines can process much more data than a person at a faster speed, and many times appear to be more intelligent than the person operating them. However, there is a characteristically human activity that the computer has not yet been able to reproduce; that is, thinking. The computer can make decisions logically, but can it make decisions based on the actions of a human being? Can it strategize? Can it sense and feel the intentions of person and respond appropriately? The field of artificial intelligence explores these very questions.

One of the fastest-growing applications of artificial intelligence is in the gaming industry [1] because in this application, the computer is actually competing against a person rather than working with him or her. In this case the computer needs to be able to match and even outwit the thinking capabilities of the person. The exploration of artificial intelligence in the gaming industry is booming because it offers an excellent springboard for AI research in other areas. Games are fascinating toy examples that capture the complexity of real-world situations, from driving a car to strategizing a battlefield attack. Computers have already shown their potential in this arena. Two famous examples of artificial intelligence in games are the Jeopardy whiz Watson, the supercomputer created by IBM, which outperformed its human counterparts in the popular game show; and the chess-playing computer Deep Blue, also created by IBM. This machine won against the chess world champion in 1997.

In this project, we seek to explore the field of artificial intelligence in greater depth by creating a web-based strategy game in which the user plays against the computer. We will develop at least two different levels of AI difficulty for the user to interact with. The game that we intend to build will be similar to the board game Risk, in which players compete to gain control of the most world territory while destroying the armies of the other players. In order to reasonably play against a human, the AI will need to make strategizing decisions based on the actions of the human players. We intend to research and implement the most effective techniques and algorithms for AI strategizing.

Game AIs must be deliberate and careful in order to present a challenge to the player, but must also contain some purposeful random flaws to keep the game entertaining [2]. This being said, the flaws must be purposeful and not accidental, so as to make it possible to win against the AI in the same manner every time the game is played. The game AI should be able to process the same information about the game as a human player, but not to have an unfair advantage over the person playing. It must have the ability to "think" and make decisions based on this information, and act accordingly. It might have a way of remembering and learning which moves are successful and which are not. There are many techniques that give the AI the ability to perform these functions, such as command hierarchies, behaviour trees, filtered randomness, genetic algorithms, neural networks, and finite state machines, which are currently the most popular architecture for game AIs. The artificial intelligence component of the project will be the exploratory, creative portion that we intend to work out as we progress.

This document will present the required functionality and constraints of our strategy game.

2 Project Description

2.1 Functional Objectives

Throughout this document, we will differentiate between the core functionality of the game and additional features that will be added as time permits. Core functionality includes the basic graphical user interface, the two basic AIs, and the client-server architecture. Additional features vary in scope and complexity. For this section, we will discuss purely the core functionality..

The product will be a game similar to the board game Risk, in which the player seeks to dominate the most territory by strategically moving armies and attacking opponents' territories. Prior to beginning the game, the user will be able to select options including a color for his or her territories as well as the number and difficulty of AI opponents. Once in the game, the user will be able to take turns by selecting one of his or her own territories in which to place a certain number of armies, and then selecting territories adjacent to her or her own territories to attack with a specified number of armies. The attacks are simulated using a formula and a random number generator. If all defending armies are defeated, the territory changes control

to the attacker. If the user attempts to give bad input (e.g. by clicking in an invalid area), the game will notify the user of the mistake and not process the input.

The system will be able to handle input from the user in the form of gathering the (x, y) coordinates of mouse clicks. It will use the location of the click as well as the turn sequence to create a record of the move that the user wishes to make. It will then process this move according to the rules of the game. After the human player has taken a turn, the system will gather moves from the AI players based on continually updated game states. After accepting and processing the moves from the AI players, the system will update the game state, and display the updates to the user through the graphical user interface.

The back-end of the product will be structured in a client-server architecture in which the game logic, as well as programs for running the AI scripts, will be stored on the server machine, and input from the user will be handled via a website displayed on the client machine.

The success of the game will be measured by the difficulty and complexity of the AI, as well as the presentability of the user interface. The AI on the most difficult setting should win against an average human player in at least 50% of the games that it plays. The user interface should be aesthetically pleasing, intuitive, and fully functional according to the game specifications.

Some additional features that we would be pleased to add as time permits are as follows: the capability for multiplayer over the network, additional levels of AI difficulty, an enhanced user interface with the ability to specify the gameplay map, the addition of luck cards and other random game events.

2.2 Learning Objectives

Through this project, we hope to gain experience in three main areas. First, we will work with web interface development, gaining experience with HTML, CSS, and Javascript, including how the three of them work together. Second, we will explore a client-server architecture fitting with our project. We hope to learn how to handle multiple synchronized requests to a server and how to deal with standard networking issues such as dropped connections and concurrency issues. Finally, our main objective will be to explore game AI algorithms and techniques in depth. The latter part of our project will center around experimentation and prototyping of different AI implementations.

3 Development resources

The following are the books we intend to use to educate ourselves on techniques for AI development.

Introduction to Game Development [2] contains useful information for techniques in developing AIs. In particular, sections 5.3 and 5.4 talk about structure for the AI scripts as well as popular methods for common AI functionality, for example, using finite state machines as a framework.

Artificial Intelligence for Games [3] is a useful in-depth resource on AI strategies, such as decision making, and learning. It also includes applications for board games specifically.

Introduction to the Design and Analysis of Algorithms [4] has information on graph theory and algorithms for dealing with graph problems. This will be useful in handling the board state because the board will be stored as a graph. Graph theory will also be important when dealing with graphs for the AI to find appropriate moves on the board.

In addition, we will require some online resources for learning HTML5, CSS, Javascript, and XML. The following are useful websites we intend to use.

- www.w3schools.com [5]– This website has thorough tutorials on all those subjects, as well as extensive references for them. We will start our coding by familiarizing ourselves with their resources.

- javascript.info [6]– This website contains extensive tutorials on the use of Javascript; this will be useful for coding the client side user interface of our product, which will be done entirely in Javascript.

The remaining resources that will be required for the product will be a versioning control system, a code repository, web hosting, and code editing facilitators.

- Github – The repository and web hosting will all be handled via Github. Our project repository is currently hosted on Github; in addition, our client-side website will be hosted via gh-pages. Github is free and open source.

- Git – Version control will be handled via Git, also open source.

- Eclipse IDE – We will be using the Eclipse IDE for coding the bulk of the project. Eclipse has plug-ins for managing all the types of code we plan to work with. Eclipse is open source as well.
- IDLE – We will use the IDLE editor that is built into the standard Python distribution when coding the AIs.

4 Requirements

4.1 Performance Requirements

Due to the client-server architecture of the game, the performance of the game from the user's perspective will be dependent upon the status of the network. Assuming that the network is functioning optimally, the game engine itself should provide the user with near-immediate feedback. In the event that the multiplayer capability is added, the first version of the game will handle a relatively small number of users, no more than 100 online users at a time.

4.2 Design Constraints

The structure of the game will follow a Client-Server architecture. The game logic and server itself will be stored on the server side, while the client side will display the UI, receive input from the user, and transport it to the server for validation and manipulation according to the game rules. Whereas the architectural plan will allow for the AI scripts to be run from any location, in this version they will be run on the server machine for ease of use. The HTML/Javascript code necessary to display the UI will be stored on the server and sent to the client when the client loads the game page. The paradigm used to build the game will be Model-View-Controller, in which the model will hold the game logic and the AI scripts; the view will be visible by the user; and the controller will handle the transport of input from the user to the model and the transport of the game state back to the client view.

Different components of our project will be done using different tools and programming languages. The user interface will be scripted using the HTML5 Canvas, CSS, and Javascript. The game engine will be coded using Java 1.7, as will the server itself. The AI scripts will be coded in Python, but our architecture will allow us to write them in any language if we so choose.

4.3 User Characteristics

Our expected users are strategy gamers of any skill level who might or might not have any experience with the game framework. As such, having a clean and intuitive user interface will be important, as well as readable explanations of different UI elements, gameplay mechanics, and AI types. In keeping with the traditional Risk board game, we will not expect any users under the age of 10.

The users operating the server will not be expected to have advanced knowledge of the game's inner workings. AI interactions should be handled on the server side, and he server itself should be self-contained, such that one only needs to run an executable to start it (or possibly enter some basic commands).

4.4 Assumptions

We assume that the user has a basic knowledge of computer operations, has a reliable TCP/IP connection, and is operating the game on an updated internet browser. We will assume the server machine is running the specified versions of Java and Python.

4.5 Security

No user information will be collected by our program. If we decide to implement multiplayer capabilities, we may need to store usernames and passwords, in which case we would need to consider the security of our storage. Additionally, both the client side and the server side will be designed to handle malformed requests that might otherwise cause problems. In the event that a feature is added to the game in which information in the form of text is collected from a user, length checks will be added to the string parsing so that attacks such as SQL injection and buffer overflows may be prevented.

The possibility of a user manipulating the game through means other than the presented UI exists; this possibility will be dealt with by a check on the server side of the requests it receives. It will check the length as well as the content of the requests in order to verify their source.

4.6 Reliability

Since our project will involve networking, reliability will be a major concern. HTML and Javascript, being largely reliable and stable, will handle many of the major concerns. However, more effort will be needed to ensure that the server is stable and error-proof. The server will guard against bad input by doing input validation before processing requests from the client. The networking issues of concurrency and dropped connections will also be addressed in the server. The game should not fail more than once for every 100 games played. We will ensure reliability by ample testing after each step in the implementation process.

4.7 Portability

The game should be portable to the latest versions of Google Chrome (v 22.0) and Mozilla Firefox (v 16.0). As a web-based game, it should be playable on all three operating platforms, Windows, Mac, and Linux. Backwards compatibility in browsers, however, will not be guaranteed. A version of the aforementioned browsers that is not updated will not be guaranteed to run the game. Since our project is web-based, our focus will be on supporting multiple browsers rather than specific operating systems. At a minimum, our project should work on the latest versions of both Google Chrome and Mozilla Firefox; if time allows, we may also ensure functionality on other browsers. The server half of the project will be done in Java, making it portable to any platform that supports Java. The AIs will be written using Python; the platform operating the server side must have Python.

4.8 Maintainability

The code of the final product will be easily maintainable by another programmer, and will be adequately abstracted so as to facilitate the addition of features in the future. It will be straightforward to alter the style of the user interface, as well as to update the logic of the game and add additional capabilities to the AIs. We will need to ensure that the code is clear and well-documented. Generally speaking, the only major maintenance would be updating the browser and possibly the scripting language version on the client-side machine.

4.9 External Interfaces

4.9.1 Client-Side Interface with the User

The players will interact directly with the game primarily through three interfaces, the game start screen, the gameplay screen, and the win / loss screen.

4.9.2 Game Start Screen

Upon visiting the URL at which the game will be hosted, the user will first encounter the Game Start Screen (see Figure 2). He or she may choose the options of which color his or her territories will be on the map, the number of AI opponents he or she would like to face, and the difficulty level of said opponents. Upon the event that music is added into the game, the user will have the option of turning the volume up or down. When finished, the user clicks "Start."

4.9.3 Game Play Screen

Upon clicking "Start", the user will be directed to the Game Play Screen (see Figure 3). This is where all game interaction will take place. The user will be notified in the Player Controls box when it is his or her turn to play. When it is, he or she may click on the map to choose one of his or her territories to manipulate. If the user clicks on a territory that he or she does not own, the web page will display an error in the Alerts box on the bottom of the screen and do nothing with the click. Upon clicking in a valid territory, the user may select the number of armies he or she wishes to place there in the form of a pop-up (see Figure 4). The Player Controls box will summarize this decision. The user may then decide which territories he or she would like to attack. Again in the form of a pop-up, the user will be asked with how many armies he or she would like to attack. This decision will be added to the summary in the Player Controls box. When satisfied with the turn, the user may click "Done."

4.9.4 Win / Loss Screen

When the game is over, the user will see a pop-up displaying who won, and asking if he or she would like to play again (see Figure 5). Upon clicking yes, the user will be directed back to the Game Start screen to choose options.

4.9.5 Server Interface with Client

The server will interact with the client by means of an HTTP connection; that is, the client will send requests to the server of moves in the form of POST and GET requests that the user wishes to make, and the server will respond with an updated game state after all players have taken turns. The corresponding game state sent from the server will be in the form of an XML (or possibly JSON) file. See Figure 6.

4.9.6 Server Interface with the Virtual Players

The server should be able to receive a move from the virtual players, that is, the programs running the AI scripts, using the same protocol as the moves received from the human user. It should also be able to send an updated game state to the virtual players in the same manner as the one which it sends to the human user, that is, in the form of XML. See Figure 6. The precise implementation of this interface has yet to be decided. The virtual player programs may constantly poll the server with requests to know if it is their turn yet or not, or they may reside in the server itself in a kind of plug-in architecture.

4.10 Use Case Models

4.10.1 Start

Table 1: Start Use Case

Player	System
1. Chooses player color.	
2. Chooses number of opponents.	
3. Chooses difficulty.	
4. Clicks "Start"	
On successful connection:	
	5. Opens connection
	6. Begins game engine
	7. Displays the game screen to the user
On unsuccessful connection:	
	5. Displays a timeout message to the user

4.10.2 Take Turn

Table 2: Take Turn Use Case

Player	System
1. Clicks a country on the map	
	2. Displays a pop-up allowing the user to choose the number of armies to move
3. Inputs the number of armies; clicks "Go"	
	4. Adds the army moves to the turn summary
5. Chooses territories to attack by clicking on an opponent's territory	
	6. Adds the attack to the turn summary.
7. Clicks "Done"	
	8. Carries out the turn according to the game logic.
	9. Fetches a move from the other virtual players
	10. Displays the resulting game board state to the user.

4.10.3 Win or Lose

Table 3: Win Or Lose Use Case

Player	System
	1. Displays a win or loss message to the user in the form of a pop-up. Asks if the user wishes to play again.
*T*o play again:	
2. Clicks "Yes"	
	3. Displays the start screen with options again
*N*ot to play again:	
2. Clicks "No"	
	3. Exits

5 Task Breakdown

The following outline gives a breakdown of tasks for the entire design and implementation of the game.

1. Design Phase
   (a) Make changes to Requirements Document as needed
   (b) Prototype Client GUI interactions
   (c) Prototype Client-Server interactions
   (d) Write Design Document
2. Game Engine
   (a) Code basic game engine
   (b) Test game engine logic
3. Server
   (a) Code server
   (b) Test server-engine interactions
   (c) Code client protocol
   (d) Test server-client interaction
4. Client GUI
   (a) Code HTML/Javascript UI
   (b) Test user input and validation functions
   (c) Test client-server interactions with GUI
   (d) Test user functionality with game engine
5. AIs
   (a) Code Virtual Player program with client functionality
   (b) Script a test AI
   (c) Test Virtual Player program with test AI, including full network functionality
   (d) Write other AI scripts
6. Testing
   (a) Test AI scripts against other AI scripts
   (b) Final testing of AI vs Human
7. Extras (if time)
   (a) Multiplayer

(b) Cards
(c) Additional AI difficulty levels
(d) Enhanced user interface
(e) Random game events

8. Wrap-up
   (a) Prepare the final document
   (b) Prepare for the presentation at the Academic Festival

6 Preliminary Timetable

Task	Start Date	End date
Design Phase	10/25/12	12/18/12
Changes to Requirements Document	10/25/12	11/1/12
Client GUI Prototype	10/25/12	11/8/12
Client-Server Prototype	11/8/12	11/22/12
Design Document (Diagrams, Use Cases)	11/22/12	12/18/12
Design Presentation	12/4/12	12/11/12
Implementation	1/2/13	5/1/13
Game Engine (Code/Test)	1/2/13	1/16/13
Server	1/16/13	2/16/13
Client	2/16/13	3/16/13
Code/Test GUI offline	2/16/13	3/9/13
Test client-server interactions with GUI	3/9/13	3/16/13
AIs	3/16/13	5/1/13
Code/Test Virtual Player	3/16/13	4/1/13
AI Scripting Creativity	4/1/13	5/1/13
Final Testing	4/15/13	5/1/13
Wrap-up	4/15/13	5/26/12
Prepare final document	4/15/13	5/26/13
Prepare academic festival presentation	4/15/13	5/3/13

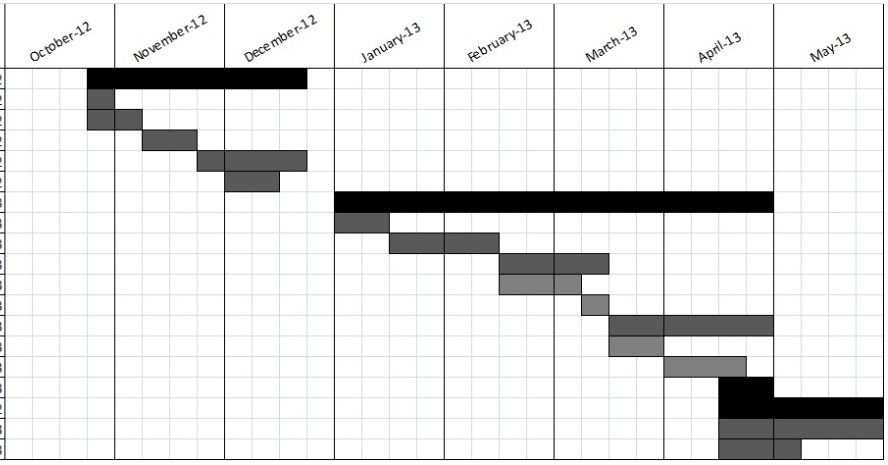

Figure 1: Gantt Chart

7 Budget

We do not anticipate any costs for our project. All resources required for development are either currently in our possession or open source.

8 Development Documentation

Our project webpage is hosted on Github and can be found at the following address: `http://wds-project.github.com/Capstone/index.html`. From this page our code repository can be accessed on Github. This page also

contains all project documentation up to date. The repository is under the menu header "Github project", and all submitted documents can be viewed from the "Documents" link. Our team blog can be found at `http://wdscapstone.blogspot.com/`, including posts from both team members.

References

[1] K. Hilpern, "Artificial intelligence: Transforming the world we live in," *The Independent*, 2007.

[2] S. Rabin, *Introduction to Game Development*, ch. 5. Course Technology PTR, 2 ed., 2009.

[3] Levitin, *Introduction to the Design and Analysis of Algorithms*. Addison-Wesley, 3 ed., 2012.

[4] I. Millington, *Artificial Intelligence for Games*. Morgan Kaufman Publishers, 1 ed., 2006.

[5] "W3schools." Webpage, 2012.

[6] I. Kantor, "The javascript tutorial." Webpage, 2011.

9 Glossary

Our project does not currently require a glossary; however, once we get further into implementation, we will have more terms to define, especially pertaining to game strategies and AI concepts.

10 Appendix: Interface Figures

Figure 2: Game Start Screen

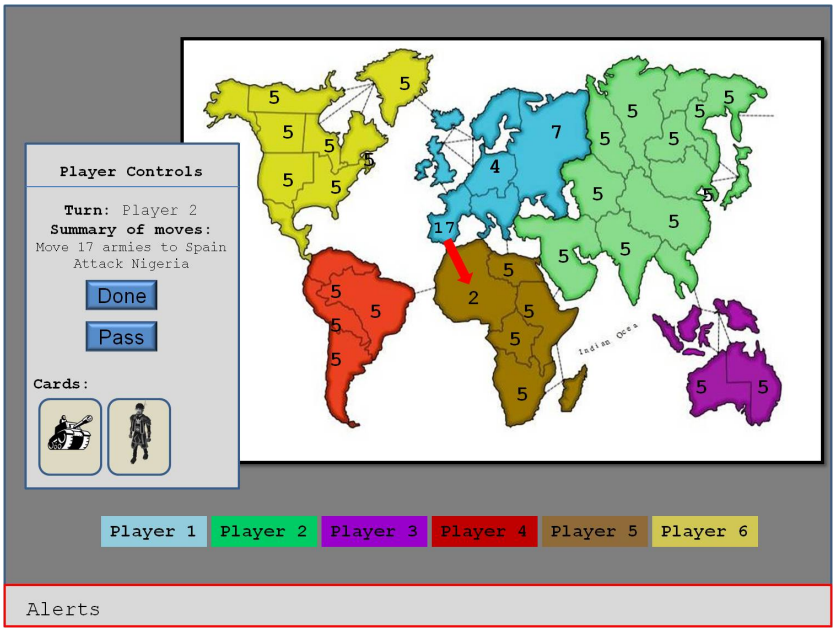

Figure 3: Game Play Screen

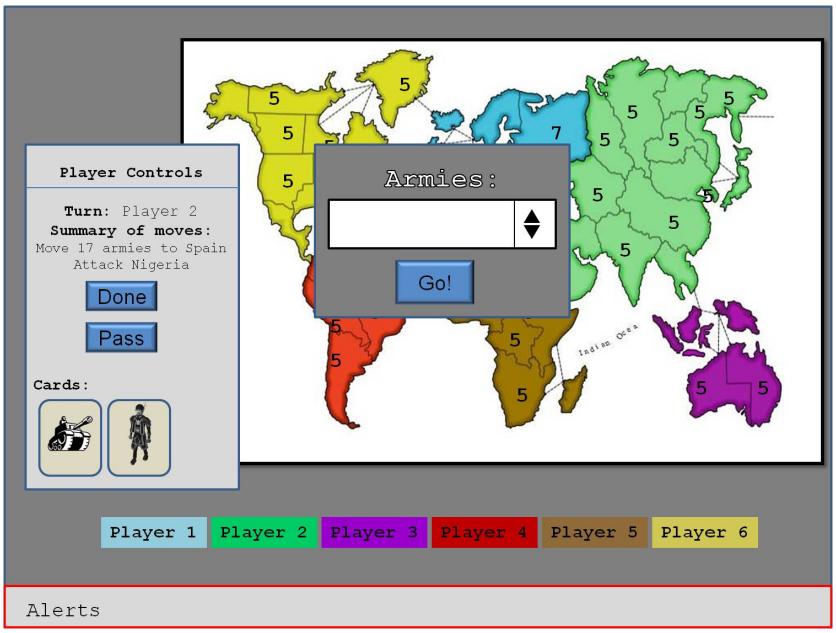

Figure 4: Game Play Screen With Armies Pop-up

Figure 5: Game Over Pop-up on Loss

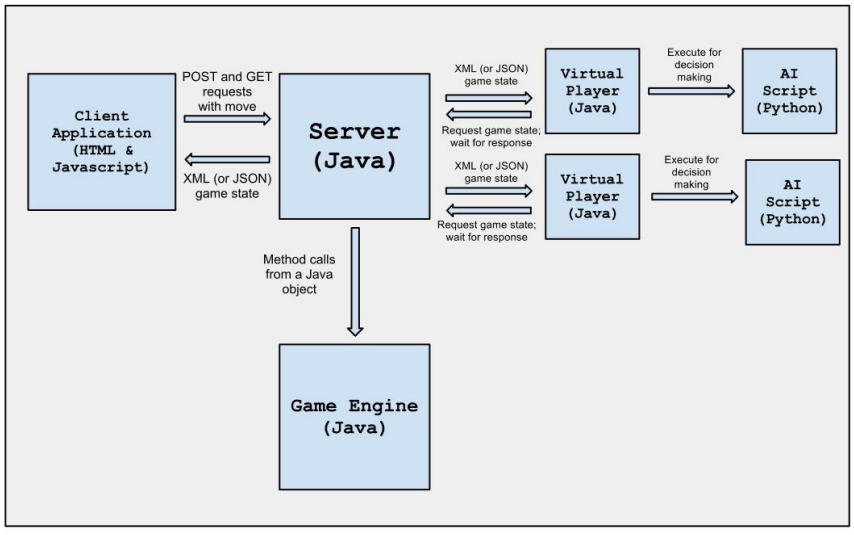

Figure 6: Architecture