Web/Mobile Platform and AI for Four-Player Chess
— Report One —

Naß Florian, Riviere Jérémy, Song Ji, Yang Xiaoxing
{fn112, jmr12, js212, xy212}@imperial.ac.uk

Supervisor: Dr. Rowe Reuben
Course: CO533, Imperial College London

23rd January, 2013

1 Background

The aim of our project is to develop a web and/or mobile platform that allows up to 4 players to play Chaturaji[1], either against other human players or against AI players that we are required to develop. The rules of the game will be described later on, in section 2.1.

Another aspect of the project involves working in close collaboration with another group that is developing a similar piece of software. Indeed we would like, eventually, to have our two implementations of AI players to play one another. To achieve this, we need to define a common interface for both groups, for the AI components to communicate with the software in charge of checking that the rules defined are respected.

We also intend to work closely with some people both from IT and non-IT background, in order to get feedbacks on bugs, as well as features to be integrated throughout the project.

2 Project specifications

2.1 Rules

Chess is one of the most popular two-player strategy board games played worldwide by millions of people, from the very beginner to the finest strategist. The aim of the game is to checkmate ones opponent by not allowing any route for his king to escape a threat.
In addition to the regular game of chess which is played on a square board, more than 2000 variants are known nowadays, where the differences can come from the boards shape, the number of players, the rules etc. Amongst the most known variants, are chinese Xiangqi and japanese Shogi.
As for our project, we are going to implement a chess-like game called Chaturaji which was most probably invented in India around the beginning of the second millenium, and is thus considered as one of modern chess's ancestor. The game is played on a 8x8 board as in regular chess and involves 4 players. At the beginning of the game, the pieces are disposed as shown on figure 3 below:

Investigating multiple resources regarding this particular 4-player chess game, inconsistency concerning rules and certain aspects of the game play has become evident. Hence, following set of rules is the result of an agreement among both groups dealing with this project:

1. The game is meant to be played one man for himself, and red begins. After that, play turns pass clockwise around the board.

2. The different pieces available and their associated points are:

[1]One of modern chess's ancestor

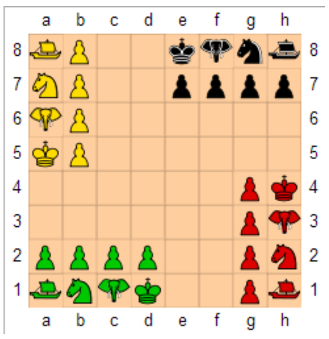

Figure 1: Initial position of the pieces on the board

- **King - 5 points**: The king moves in any direction but only one square at a time and is not allowed to jump over other pieces.

- **Elephant - 4 points**: The elephant moves in any direction, in straight lines, for any number of squares and is not allowed to jump over other pieces.

- **Knight - 3 points**: The knight moves in any direction, in a L-shape not wider than two squares and is allowed to jump over other pieces.

- **Boat - 2 points**: The boat moves diagonally in any direction, not more than two squares and is allowed to jump over other pieces

- **Pawn - 1 point**: The pawn moves forward only one square at a time. It is allowed to go diagonally only to take one opponents piece.

3. Special moves:

- **Triumph of the boat**: When a boat moves to make a square of four boats, then it takes the other three, as shown on figure 2 below:

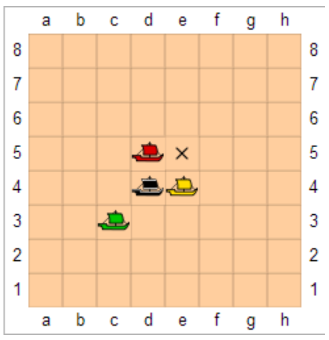

Figure 2: Green boat can take three others by going to e5

- **Pawn promotion**: A pawn is promoted once it reaches the last rank of the board, according to what piece was at this position in the initial position. It can only be promoted to a piece that the player does not have anymore on the board.
- **Conditions for pawn promotion**: When a player owns less than 3 pawns, its pawns may be promoted to a knight or an elephant only. If only one pawn is left and at most the boat and the king, then the remaining pawn may promote to any type of piece. A pawn that reaches the final rank and is not able to promote stays there as a pawn until it is taken or conditions are met for it to promote.

4. A player who manages to take all three other kings is granted a bonus of 54 points.

5. When a player loses his king, he is out of the game. All the pieces he has in game remain on the board and can be taken by the other players.

6. The aim of the game is to earn more points than the other players. The game may end in one of the following ways:
   - Only one king remains on the board.
   - When two players are left, they are both granted a certain amount of moves to finish the game. If both kings remain on the board after these turns, the game ends and the winner is the one who has the more points.

2.2 General architecture

The system architecture consists of a central game server that is linked to a database in order to store information, e.g. all moves of a game or high scores. Users connect directly to the game server via a web and/or mobile user interface. The artificial intelligence component of the system connects to the game server through a distinctive interface that has been agreed on by both groups working on this project. This way, the AI components of both groups can easily be exchanged or connect to the game server simultaneously in order to play one another, for instance.

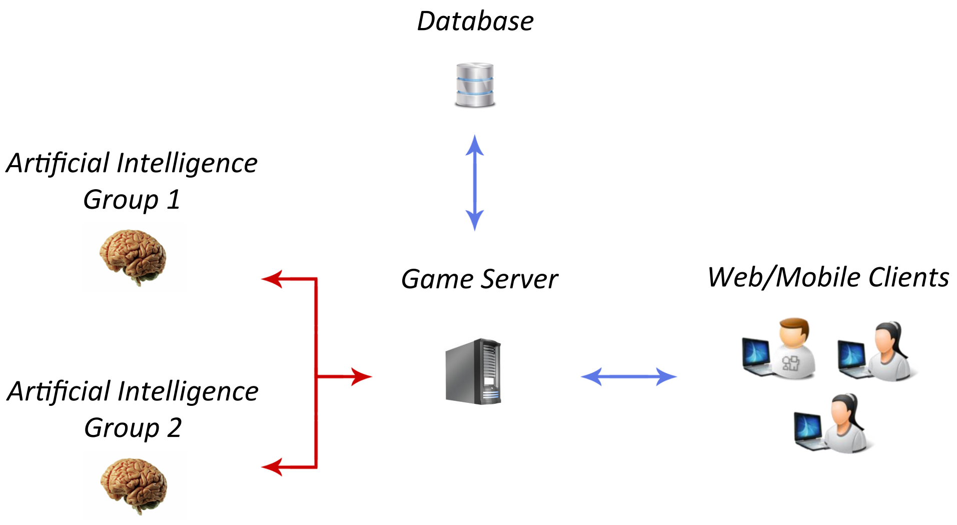
Database
Artificial Intelligence Group 1
Game Server
Web/Mobile Clients
Artificial Intelligence Group 2

Figure 3: Global architecture. Blue lines indicate independent communication paths, while red lines show the communication of components by both groups using a common interface.

Our idea is to implement our platform as a three-tier layered application. The presentation tier will be responsible for displaying the current state of the game to the players and allow them to play at their turn. To do so, we thought of using HTML5/CSS and Javascript, which are standard web technologies, supported by all major browsers on many different operating systems, including mobile OS.

The logic tier will be developed using J2EE technologies and ActiveMQ[2]. It will be responsible for checking that no rule is infriged by any player as well as keeping players aware of the current state of the game at any time.

Finally, the data tier will consist of a PostgreSQL database, provided by the DoC and will hold information about players, to allow them to login to our system. One other interesting feature would be to be able to store information about the games, to be able to draw statistics/review them, but this is a minor priority feature, that we may implement once our basic features are developed.

[2]Messaging Service API - See activemq.apache.org

2.3 Functional requirements

2.3.1 Basic requirements

As for our basic system, we intend to have the following requirements met:

- The first requirement will be for us to set up and configure our different tools: the webserver, the database server, ActiveMQ, version control system and continuous integration system. This should be available by the end of week 3. Priority: High.
- The system will be available from a web browser and will not require any third-party software to run. Priority: High.
- The chess-board will be shown in 2D, and we already have a first version of it, to which we only need to add the implementation of the rules for moving the pieces. Priority: High.
- An interface will be available for users to register a login/password that they will be asked in order to be able to play. This feature should be available by the end of week 3, as it is a basic functionality and can be easily taken from one project we worked on previously. Priority: High.
- The clients-server communication feature will constitute our highest priority. Indeed, it involves developing the game logic to ensure that the rules are fully respected and the communication protocol from clients to server. This will probably take us up to two/three weeks and will constitue our base for all the other requirements found below. Priority: High.
- The AI will probably require most of our time. That's why its development will be started as soon as the game logic will be fully operational, so that we can experiment different techniques for the AI. Priority: Medium.
- A player will have the choice either to create a game for others to join or join a game already created. When he creates a game, a player can specify how many players will be played by the AI. A web page will display all available games and provide a way for users to choose a game to join. Priority: Medium.

2.3.2 Advanced features and improvements

Once we have met all the basic requirements, we will focus on enhancing user experience:

- Develop a 3D rendering system for the game. This is feasible with WebGL as WebGL is similar to OpenGL and powerful in 3D modeling and rendering. Priority: Low.
- Develop different levels of AI. This could be easily implemented by changing computing time and algorithm complexity of AI. Priority: Low.
- Add the feature of a rotating board for each player so that his/her pieces are facing upwards. Priority: Low.
- Troughout the project, we'll try to integrate the features pointed out by our users, by holding a list of them sorted by priority. Priority: Low.

3 Development process

Regarding development methodology, we decided to use Scrum because of its iterative and agile strategy. Once the core requirements to the software system have been identified, an initial version with basic functionalities will be implemented employing rapid prototyping methods. A test driven development (TDD) approach will be followed, which will be divided into short development cycles (sprints) in order to guarantee consistent progress. Regular communication in daily Scrum meetings will ensure that all developers are up-to-date. Conducting user studies, we aim to get feebacks from stakeholders about bugs and errors, and receive constructive proposals, which can be evaluated regarding usability of the product. Thus, once the first minimal version of the project has been developed, we would like to ensure weekly releases, with integration of the proposals made by the stakeholders.

4 Technology overview

The software will be designed to run platform-independently, regardless of operating system or browser. In order to achieve this behaviour and still put rapid prototyping into practice, the graphical user interface of the web client will be implemented using JavaScript and HTML5/CSS in the first iteration cycles. As a possible extension, the user interface may be equipped with 3D graphics by the use of WebGL.
The connection to the game server will be established using WebSockets, an up-to-date technology that is supported by all major browsers, and servlets.

The game server uses the Apache ActiveMQ technology, which supports cross language clients and protocols. This feature is essential for this project, because it allows the game server to handle both, WebSocket connections from web clients and connections from the AI component.

The artificial intelligence component will be developed in a later step and the programming language will most likely be C++ for it will allow us to fully optimise the algorithms. Because of the cross language capabilities of ActiveMQ, however, we will be able to further investigate about whether we can find a language that is more suitable for AI programming.

Last but not least, we will use a continuous integration system, TeamCity, in order to ensure that our production system is always up-to-date, and fits our bench of tests. GIT will be employed as a version-control system for this project. Indeed, it can be easily integrated to TeamCity and is the one with which all of us are the most familiar.

5 Boundaries

- Hardware environment: computers in DoC, for which we have no guarantee that they will be available 24/7.
- Software environment: any browsers supporting HTML5/WebGL
- We will have to rely on our users availability for testing, which may lead to a clash with our scrum approach.

6 Project schedule

- 23 January - 6 February: Finish requirements with high priority
- 6 February - 20 February: Finish requirements with medium priority
- 20 Febraury - 6 March: Finish requirements with low priority

7 Division of work

As we decided to go with the scrum methodology, we will meet regularily to define the next features to implement and divide the work. However, for the beginning of the project, we decided to split the work as follows:

- Client-Server communication (Jérémy, Florian)
- Web Interface (Florian, Ji)
- Artificial Intelligence (Xiaoxing, Ji)
- Collaboration with the other group regarding AI interface (Jérémy)