Project Eris
Inception

Team Adrastos

June 11, 2009

1 Elevator Summary

Project Eris is an online-capable, free open-source computer game based on a popular strategy board game called Blokus^{TM}. The simple, yet strategical, board game involves in placing block pieces on a grid, with the rule that they must connect only through corners. By making the game internationalized, we open the game to more than just English speakers. Attracting newcomers through a clean, simple, and customizable interface, and satisfying longtime players and expert strategists to compete worldwide with rankings, we believe this can be a successful product that has wide attraction and support.

2 Stakeholders

- Ben Boeckel
- Artem Kochnev
- Abhishek Mukherjee
- Taro Omiya
- Roy Wellington
- Anita Kuchera
- John Sturman

3 Rules

3.1 Description

Project Eris plays on either a square (with square units) or triangular (with triangular units) grids. Players holds a set number of pieces—units connected in various shapes—to place on each board.

3.2 Goal

The goal for the player is to use place as much units on the board possible before he or she runs out of moves.

3.3 Process

The game begins with each player placing a piece of their choice on the board's corners. The player can place a piece on the board only if it connects to a corner of another piece the player has already place. In addition, the piece may not overlap with any other player's pieces. The game ends when each player runs out of moves.

3.4 Conclusion

Scoring takes place in the end as follows:

1. If the player places all of his/her pieces on the board, the player is awarded 15 points
2. If the player places all of his/her pieces on the board, with the one-unit piece placed in the very end, the player wins another 5 points. This gives the player a grand total of 20 points.
3. If the player could not place all of his/her pieces, every unit on each piece the player failed to place counts 1 points against him/her.

The player with the most points wins.

4 Feature List

4.1 Online games

1. Player ID system
2. Creating your own custom game
3. Invitation to your own game
4. Spectator system, for those who want to see competitions on the sideline
5. Online rankings
6. Different rule sets
    - Fog of War
    - Different sized boards and piece sets

4.2 Customizable theme

1. Easy theme installation

4.3 Plug-in system

1. Tournament plug-ins
2. AI plug-ins

5 Business Case

The only competition to Project Eris is an online Flash®-based version made by the creators of the official version of Blokus™, Sekkoïa. The Flash®-based game, though, does not represent the game of Blokus™ well for several reason. For example, the Flash®-based game requires the user to be online to play. Those with a computer, but no online connection will not be able to enjoy the game electronically. Furthermore, has a crude and clumsy match-system between players, allowing one to play against others of inconsistent experience levels. As such, Project Eris aims to use a free, cross-platform version with a client-server interface for online matchmaking will be a better experience that preserves the vision of Blokus™. The program will be an executeable, meaning internet connection is not required to play this game. As a free product, Project Eris will not generate revenue.

6 Description of Milestones

6.1 Inception

Deadline:

October 6th

Description:

Document the basic ideas for Project Eris. Decisions will be made on our schedule, use cases, terminology, and budget. This document should give a thorough description of our project's purpose and strategies to our stakeholders.

Features:

- None

Tasks:

- State the project's vision
- Document the schedule
- List use cases
- Estimate budget
- Develop a glossary

Use Cases:

- None

6.2 Game Data Structures and Algorithms

Deadline:

October 9th

Description:

Game data structures and algorithms are completed. The game will be very bare-bones, and may not contain any graphics. Testing will be done through terminal commands and test programs.

Features:

- Square and triangular grid data structure
- Square and triangular pieces representation
- Plug-in API **(Moved to milestone 4)**
- Algorithms for game rules
- Board evaluation (used for AI) **(Moved to milestone 4)**
- Control algorithms

Tasks:

- Code data structures
- Formulate algorithms
- Set game rules
- Verify structure's stability

Use Cases:

- None

6.3 Elaboration

Deadline:

October 20th

Description:

Clarify the inception by adding system specification, internal framework, and deployment structure. More details will be added to use cases schedule.

Features:

- None

Tasks:

- Record system specifications
- Draw out a domain model
- Compose a system sequence diagram
- Design the deployment diagram
- Detail the schedule
- Elaborate use cases

Use Cases:

- None

6.4 Four Player and Two Player Game Play

Deadline:

October 23rd

Description:

The square grid board and both two- and four-player gameplay are completed. The game should implement a number of common graphics, such as the board and pieces, and demonstrate the controls and rules for the game.

Features:

- Square grid and piece graphics
- Mouse and keyboard controls
- Four-player game play
- Two-player game play
- Customizable game rules (incomplete)
- Sound handling (Phonon)

Tasks:

- Design basic user interface
- Draw square board and pieces
- Render graphics
- Code game controls

Use Cases:

- Use Case UC1 Select Piece
- Use Case UC2 Place Piece on Board
- Use Case UC3 Start an offline Game
- Use Case UC12 View Record
- Use Case UC18 Quit Game
- Use Case UC21 View Pieces

6.5 AI and Odd-Number Player Gameplay

Deadline:

November 6th

Description:

The basic AI for computer players in three- and one-player mode are completed. Graphics for multi-shaped grids are completed as well. Finally, game replay will be implemented.

Features:

- Draw triangular board and pieces
- One-player game play
- Three-player game play
- Computer controlled players
- Game replay

Tasks:

- Formulate AI
- Add more rules
- Control recording
- Draw triangular board and pieces

Use Cases:

- Use Case UC20 Watch Replay

6.6 Online Game Hosted

Deadline:

November 20th

Description:

Basic online multiplayer are completed. The host server for hosting custom online games will be up and working. Database for games and player information will be implemented.

Features:

- Customizable game rules (full)
- Hosting custom online games
- Online account
- Ranking algorithms
- Online communication

Tasks:

- Build and host server
- Add more rules
- Test connection
- Code ranking algorithms
- Add online user interface
- Develop ranking website

Use Cases:

- Use Case UC4 Start an Online Game
- Use Case UC5 Join an Online Game
- Use Case UC7 Change Password
- Use Case UC8 Compare Rankings
- Use Case UC9 Create Account

- Use Case UC10 Login
- Use Case UC13 Change Auto-connect Settings
- Use Case UC17 Play Ranked Game
- Use Case UC19 Spectator Chat
- Use Case UC23 Change Tournament Type

6.7 Finishing Touches

Deadline:

December 1st

Description:

The game will be polished to better depict the game. Game presentation will be cleaned, and menu controls will be tweaked to be more user friendly. Plug-ins and themes will be installable. Test cases and programs will be used to find possible bugs and mistakes we have made. Finally, translations for the game menus will be completed.

Features:

- Bug fixes
- GUI tweaks

Tasks:

- Bug fixes
- Review code
- Review user interface
- Review graphics
- Review sounds
- Consult translators
- Implement translation
- Configure plug-in directory

Use Cases:

- Use Case UC6 Install a Plug-in
- Use Case UC14 Change Language
- Use Case UC15 Install a Theme
- Use Case UC16 Change Theme
- Use Case UC22 Load Plug-in

6.8 Transition

Deadline:

December 4th

Description:

The documentation for how to play the game will be created. Records will review on how we've tested the product, and how the testers reacted to it. The document will further mention how our development process went, and our suggestions on the best ways to enjoy this game.

Features:

- Gameplay documentation

Tasks:

- Create installation file
- Type the user guide
- Record test results
- Note best practices
- State personal peer reviews (to be done individually)

Use Cases:

- None

7 Work Breakdown Structure

7.1 Eris Project

1. Concept
   (a) State vision
   (b) Determine use and function
   (c) Define rules and requirements
      i. Determine libraries to be used
      ii. Decide on valid connection speed
   (d) List features
   (e) Consider risks
   (f) Create schedule
2. Art design
3. Programming
   (a) Design class structure
   (b) Program core data
   (c) Define common classes
   (d) Program graphical user interface

(e) Construct network

4. Testing
5. Documentation

8 Effort Required to Code Features

The total effort required to code all the features, according to table 2, is 88 hours.

9 Budget

9.1 Computers

The computers for coding this project costs exactly $2,000 each. Every group member will need a computer for better communication and production for this project. As such, we will be purchasing 5 computers for a total of $10,000.

9.2 Server

A server will be used to host a wiki, subversion, and Bugzilla. Fortunately, this relatively cheap computer will only cost around $500. Only one is necessary.

9.3 Salary

Our salary will be $20 an hour. We will work roughly 300 hours this semester, coming to a total expenditure of $6,000.

9.4 Free Software

Our project takes advantage of several free, open-source, pieces of software such as the Linux operating system, Qt GUI design toolkit and numerous others. Since the total for all of these pieces of software is $0, we will not list them all.

9.5 Grand Total

With 5 computers, one server, and salary, our sum for this project totals to $16,500.

10 Use Case

Use Case UC1 Select Piece

Actor: Player.
Goal: The player chooses a piece to use.
Pre-condition: The system recognizes that the player still has at least one piece to play.
Description:

1. Player selects a piece on the tray of pieces.

Post-condition: The system allows the player to move the chosen piece.

Use Case UC2 Place Piece on Board

Actor: Player.
Goal: The player places a selected piece on the board.
Pre-condition: The system allows the player to move a piece.
Description:

1. Player drags at a spot on the board to place the piece.
2. Player selects that spot.
    (a) If the selected position on the board isn't valid, the player must select another spot on the board.

Post-condition: The system places the piece on the board, and lets the next player play.

Use Case UC3 Start an Offline Game

Actors: Player.
Goal: The player starts an offline game with set rules.
Pre-condition: The system is not running a game.
Description:

1. Player chooses to start a new offline game.
2. Player chooses options for game in the game configuration.

Post-condition: The system starts a new offline game.

Use Case UC4 Start an Online Game

Actors: Host, Players, and Time.
Goal: The host creates and plays an online game with set rules.
Pre-condition: Network host have not started an online game.
Description:

1. Host chooses to start a new game.
2. Host provides the system game rules.
3. Players join the newly created table.
    (a) If the time runs out, and there are vacant spots, Host fills them with AI-controlled players.

Post-condition: Network host successfully starts a new online game.

Use Case UC5 Join an Online Game

Actors: Players and Table.
Goal: The players finds and joins an online game.
Pre-condition: Network host have an online game available for the player.
Description:

1. Player requests to connect to a game.
2. Player does one of two things:
    (a) Player already knows the required information to connect to the host, and enters it.
    (b) Player does not know the information, and selects a host from a list of available game hosts.

3. Player chooses a table to join.

4. Table waits until enough players join.

Post-condition: Network host connects the player to the selected online game. If the connection fails, player is notified.

Use Case UC6 Install a Plug-in

Actor: Player.
Goal: Player successfully installs a plug-in.
Pre-condition: System have not installed the plug-in yet.
Description:

1. Player requests to install a plug-in.

2. Player navigates to the location of his or her downloaded plug-in, and selects it.

Post-condition: If the system installs the plug-in correctly, its features are now available to the player. If the plug-in installation failed, the player is notified and given a reason.

Use Case UC7 Change Password

Actor: Player.
Goal: Player changes his or her account's password.
Pre-condition: System recognizes the player logged into the online game table.
Description:

1. Player requests to change its password.

2. Player provides a new password.

Post-condition: The system updates the account's password. If update fails, the player is notified and given a reason.

Use Case UC8 Compare Rankings

Actor: Player.
Goal: Player views the online rankings.
Pre-condition: System recognizes the player logged into the online game table.
Description:

1. Player requests for the online rankings.

Post-condition: The system displays the online rankings.

Use Case UC9 Create Account

Actor: Player.
Goal: Player creates a new online account.
Pre-condition: System does not have the player's account information yet.
Description:

1. Player requests for a new account.

2. Player provides personal information.

Post-condition: The system creates a new account. If the creation fails, the player is notified and given a reason.

Use Case UC10 Login

Actor: Player.
Goal: Player logs in to the online game table.
Pre-condition: System has the player's account information.
Description:

1. Player requests for login menu.
2. Player provides his or her username and password.

Post-condition: The system allows the player in the online game table with the information provided. If the login fails, the player is notified.

Use Case UC11 Change Rules

Actor: Player.
Goal: Player changes rules for a game.
Pre-condition: System is preparing for a new game.
Description:

1. Player requests for a new game.
2. Player provides the board type.
3. Player provides the number of players playing.
4. Player provides the AI difficulty.
5. Player provides the preferred time limit.

Post-condition: The system will configure the new game's settings to the one provided.

Use Case UC12 View Record

Actor: Player.
Goal: Player views his or her own record.
Pre-condition: System is running.
Description:

1. Player requests for his or her record.

Post-condition: The system displays the player's records.

Use Case UC13 Change Auto-connect Settings

Actor: Player.
Goal: Player changes his or her own auto-connect settings.
Pre-condition: System is running.
Description:

1. Player requests for his or her auto-connect settings.
2. Player provides his or her preferred settings.
3. Player selects "OK".

Post-condition: The system changes the auto-connect settings, and attempts to re-connect with those settings.

Use Case UC14 Change Language

Actor: Player.
Goal: Player changes the language.
Pre-condition: System is running.
Description:

1. Player requests for the list of languages.
2. Player selects on his or her preferred language.
3. Player selects "OK".

Post-condition: The system changes the language and updates the menu text.

Use Case UC15 Install a Theme

Actor: Player.
Goal: Player successfully installs a theme.
Pre-condition: System have not installed the theme yet.
Description:

1. Player requests to install a theme.
2. Player navigates to the location of his or her downloaded theme, and selects it.

Post-condition: If the system installs the theme correctly, its features are now available to the player. If the theme installation failed, the player is notified and given a reason.

Use Case UC16 Change Theme

Actor: Player.
Goal: Player changes the theme.
Pre-condition: System has a theme installed.
Description:

1. Player requests for the list of installed themes.
2. Player selects on his or her preferred theme.
3. Player selects "OK".

Post-condition: The system changes the theme and updates the graphics.

Use Case UC17 Play Ranked Game

Actor: Player.
Goal: Player plays a ranked game.
Pre-condition: System is connected online, and is displaying the online game table.
Description:

1. Player selects a ranked game.

Post-condition: The system lets the player enter the ranked game.

Use Case UC18 Quit Game

Actor: Player.
Goal: Player quits a running game.
Pre-condition: System is running a game.
Description:

1. Player requests the termination of the current game.
2. Player selects "OK".

Post-condition: The system quits the game.

Use Case UC19 Spectator Chat

Actor: Player.
Goal: Player chats on a game he or she is spectating.
Pre-condition: System is spectating an online game.
Description:

1. Player provides his or her message.
2. Player presses the enter key.

Post-condition: The system sends the message to host.

Use Case UC20 Watch Replay

Actor: Player.
Goal: Player watches a reply of the game he or she just played.
Pre-condition: System is finished running a game.
Description:

1. Player selects "Replay".

Post-condition: The system plays the replay.

Use Case UC21 View Pieces

Actor: Player.
Goal: Player views his or her available pieces.
Pre-condition: System is running a game, and the player has at least one piece available.
Description:

1. Player selects the tray of pieces.

Post-condition: The tray displays the player's available pieces.

Use Case UC22 Load Plug-in

Actor: Player.
Goal: Run an installed plugin.
Pre-condition: System has a plugin installed.
Description:

1. Player selects the plug-in menu.
2. Player selects a plug-in.

Post-condition: The system loads the selected plugin.

Use Case UC23 Change Tournament Type

Actor: Player.
Goal: Player changes the tournament type of his or her preferance.
Pre-condition: System recognizes the player logged into the online game table.
Description:

1. Player selects the tournament menu.
2. Player selects a tournament type.

Post-condition: The system changes to the selected tournament type.

11 Supplemental Specifications

11.1 Functionality

	Must	Should	Could	Won't
Starting an Offline Game	X			
Starting an Online				X
Joining an Online Game				X
Install a Plug-in				X
Compare Rankings				X
Change Language			X	

Capabilities

The game will have capabilities for extended artificial intelligence and tournament modes through the use of plugins. Users with the technical backing who wish to extend the game to include new AI modes or tournament modes. Project Eris' capabilities includes the ability to run more than one instances. There is a limit, however: only one instance of the game may be logged in online.

Security

Challenge-based logins - Network games that require a password, or a user name / password combination will use a hash-based challenge-response based exchange to authenticate with the server. This method prevents the password from being sent directly, keeping it secure.

Most security will reside in the server, which must verify the user account information. Both the account information stored in the server and the login information passed to the server will be encrypted. A possible security hazard will be when one creates a new account, or when one creates a new password.

Human Factors

Since Project Eris' aim is to spread the game to as many people as possible, it will attempt to localize to as many languages as possible. Localized languages includes:

- English
- Spanish
- Russian
- Japanese

Human Factors

The game is fairly simple to understand - there are only a few base rules to comprehend before someone can play a game. A young child, with some help, should be able to grasp the rules of the game. Additionally, there are no obvious cultural barriars in the game - it is merely placing pieces on the board.

11.2 Usability

Aesthetics

Project Eris is designed to be built on the Qt toolkit. A side effect of this is that the entire application will look like all other Qt applications on the clients computer. In addition, there will be the option to replace the built-in theme with a user given style sheet to further customize the look and feel of the game.

Consistency

The dialogs and graphical user interface will be composed under several well-known consistency guidelines, such as the KDE and GNOME Human Interface Guidelines (HIG).

Documentation

Documentation for Project Eris will come in two varieties, one for standard users and one for users who wish to develop for the program. Standard users will have access to PDF documentation describing the rules of the game and the description of normal gameplay.

Users who wish to develop will have the above resources, as well as access to a few other resources: the application programming interface and sample code which can be used with our application. The goal of this additional documentation would be to ease the process for these users to create plug-ins.

Frequency and severity of errors

Users may encounter very minor errors on a regular basis depending on their input. For example, the user placing a piece in an improper place will be prompted with an error clarifying why this move is not allowed.

Recoverability

Minor errors encountered by users will be easily fixable by the user. Major errors will have to be handled by a developer or a well informed and willing user. By having the project open sourced, we drastically improve the response time for major errors.

11.3 Reliability

Predictability

Project Eris is most stable during an offline, fully manual game. Reliability lowers with AI, online play, and installed plug-ins. While we cannot take responsibility for plug-ins and online play, we can take numerous measures to alleviate the problems through error handling and automatic backups. In plug-ins especially, Project Eris will run an internal test to make sure the plug-in is both compatible with the product, and stable.

Accuracy

Thanks to simple rules and clear limits, Project Eris can accurately execute the game limits and controls. Many inaccuracies will root from the user's system, connection speed, and installed plug-ins. The only apparent internal inaccuracy are the occasional skips in the AI calculations. Many game players, though, are both accustomed and forgiving to such blunder.

Failure

Most system errors can be quickly corrected due to easy access on the source of the problem. Thus, system errors should be dealt within 10 seconds. Errors online will require a higher time limit, though. Depending on the connection, online errors may not be encountered until 10 minutes later. Fatal errors by definition will terminate the program and will do so within 10 seconds as well.

Speed

Due to online play and plug-in capabilities, the speed of our product will change frequently. Default configurations will be simple, and therefore, very quick. We'll attempt to optimize loading plug-ins and sending information online to keep the game speed consistent. However, the connection speed and the number of plug-ins installed will be left to the user's responsibility.

Efficiency

Efficiency of this product will be largely consistent and capable. With careful calculations, we can easily limit the resources used for plug-ins by using them only when necessary. Thus, even with large amounts of plug-ins, the game performance will be relatively consistent and predictable. Similarly, only a small amount of information has to be sent for online gameplay. Hence, the the online gameplay will be very quick, to a point where only connection speed will be relevant.

11.4 Performance

Resources

Resource consumption should be as minimal as possible. However, any loaded plug-ins or AI will negatively impact on performance by taking up more memory and processor power.

Speed

The application should have a very fast start-up, shutdown and response time on any relatively new computers due to the relatively small nature of the application. Network performance will be limited by the bandwidth of the user's internet connection. The game will only officially support broadband connections as the reduced performance and reliability of dial-up will have a negative impact on the user's experience with the game.

11.5 Supportability

Testability

Unit tests are built for major system components, ensuring that these components work as desired.

Extensibility

The game has a plug-in architecture for extending AI capabilities and tournament modes.

Adaptability

The game runs on a variety of systems, including Linux and Windows. Since the GUI will be based on the Qt framework, extending the game to other systems is limited to the sytem's ability to support Qt framework.

Maintainability

If an update is needed to be released, the hardware hosting the player rankings can also host a download section for updates. Additionally, some operating systems contain convenient ways to push an update to the user. Changes in the program will not matter for local games, only for network games. Network games should detect if the client is using an incompatible version of the software, and notify the user appropriately.

Configurability

The game will be able to be configured in several ways. On creating a new game, the player will be able to choose the number of players, whether the players are human or AI, and what kind of AI should be used for that player.

Ease of Installation

On Windows, an installation program will be provided to install the game on a system. On Linux, the user can either build the game directly from source, or choose from a binary package, if available for his distribution.

Localization

Localization capabilities are planned, including support for multiple languages, and the ability to easily translate the game to other languages.

11.6 Design Constraints

Internal

The game is concise in the limits for computation and memory. For example, only up to 4 player can play at a time. Similarly, only 4 AIs can be running at the same time. Thus, game algorithms will be limited only by what's set by the game rules. Memory management, on the other hand, will involve in file manipulation, and so its limits are only constrained to the user's system.

User Interface

Since our project uses the Qt GUI toolkit for user interface, every limits from that toolkit inherits to this project.

Sound

While sound is a low priority for this product, its constraints should be considered. The system will only play up to 4 sound effects at once. Adding more will only decrease the performance of the system, and furthermore, disturb the player due to horrible sound mixing. Similarly, only one background music will be playing at a time, to avoid conflicting sounds.

Graphics

Project Eris will largely rely on simple, yet informative graphics to give quick feedback to the user. For this reason, there's a necessity to limit the amount of special effects for the game. All effects should be very simple–such as flashing–unless to indicate victory. In doing so, the user can remain focused to the goal of the game, rather than consistently bothered by the annoying effects.

Online

Online gameplay is only limited by the user's connection and server's capabilities. The number of online games available, for example, are limited by the memory the server can hold. The program itself will put no constraints in online play.

11.7 Implementation Requirements

Due to our design decisions, Project Eris will only require the user to have Qt 4.2 or greater. With the Windows operating system, this dependency will be included with our product. For the Linux implementation, it will be the responsibility of the user or the Package Manager to install this dependency.

11.8 Interface Requirements

Project Eris requires two input devices: a keyboard and mouse with two buttons. The output screen for the system must be capable of displaying millions of colors and a minimum resolution of 1024 x 768 pixels. Only Linux and Windows operating systems will be supported. Any user intending to play online must have a broadband connection, as a dial-up modem will have an adverse affect on network gameplay performance.

11.9 Physical Requirements

Single System

A game using a single system only requires a modern computer built within eight years. This computer must be running either Linux or Windows operating system. Additionally, if the user wants to play online, he or she must have an internet connection.

Online Multiplayer Hosting

Hosting online games for other players requires an internet connection capable of handling traffic from each of the players on his or her server. Thus, a broadband connection is required for traffic handling.

Master Server

In addition to the above, there must be hardware for a "master server". The master server will need an internet connection capable of handling the traffic of players reporting their scores and looking up their scores.

12 Risks

There are several risks that we will face during the course of this project. For example, we will be using Qt toolkit to create the graphics for this game. However, many of us are unfamiliar with the Qt library, and will have to learn it to complete the project. Similarly, a computer-controlled player is necessary to make certain game rules work. Many of us are not familiar with programming AI, though, so we will have to experiment creating a α-β pruning tree.

13 Glossary

13.1 General Terms

Player: A user of the software that engages in playing our game.
Host: A special type of player to whom other players can connect by IP to play our game.
Spectator: A user of the software who watches a game without participating.

13.2 Game Constructs

Host: A computer that is setup with a copy of the game to which other players can connect to play our game together. A Host computer runs one or more tables.
Table: A place where the host, players and spectators can congregate to play our game. A table is a game in progress or waiting to start.

13.3 Game Flow

Game: An in-progress game, possibly with multiple humans or AI playing against each other.
Retire: What a player must do when he or she can no longer make any moves.
End of game: When all players are retired or when a player uses all of his or her pieces.

13.4 Game Elements

Board: A grid of variable height and width that is made up of square tiles onto which pieces can be placed.
Piece: A shape made up of a variable number of blocks that a player (human or computer) places onto the game board to play our game.
Tray: The location in the GUI that holds a player's pieces to be dragged onto the board.
Flip: The action of taking a piece and flipping it either vertically or horizontally.
Rotate: The action of taking a piece and rotating it 90 degrees either clockwise or counterclockwise.
Select: The action of clicking a piece, board spot, GUI menu, etc. Also includes for click-and-hold and let-go action.
Drag: The action of moving a game element with your mouse.

13.5 Technical Terms

α-β **Tree:** An optimized search algorithm that searches for the works towards maximizing the worst-case scenario.
Package Manager: An application included with most Linux Distributions which handles installation and uninstallation of a program.

14 Status Report

Some data structures and algorithms has been coded for Milestone 2. To prevent coding conflicts, a faster communication process is required to keep everyone up to date. As such, we've set up a new IRC chat room via freenode.

The team has also went through a brief Qt framework tutorial. The group should be ready to add a graphical user interface soon.

15 Who dun' it?

1.	Elevator Summary	All team members
2.	Stakeholders	Ben
3.	Rules	Taro
4.	Feature List	All team members
5.	Business Case	Artem, Ben
6.	Description of Milestones	All team members
7.	Effort Required to Code Features	Abhishek
7.	Budget	Artem, Taro
8.	Use Cases	All team members
9.	Supplemental Specifications	All team members
10.	Risks	Artem, Taro
11.	Glossary	Artem, Ben, Roy, Taro
12.	Status Report	Roy

Number	Date	Scheduled Milestone
1	October 6th	Inception
2	October 9th	Game Data Structures and Algorithms
3	October 20th	Elaboration
4	October 23rd	Four Player and Two Player Gameplay
5	November 6th	AI and Odd-Number Player Gameplay
6	November 20th	Online Game Hosted
7	December 1st	Finishing Touches
8	December 4th	Transition

Table 1: Milestone schedule

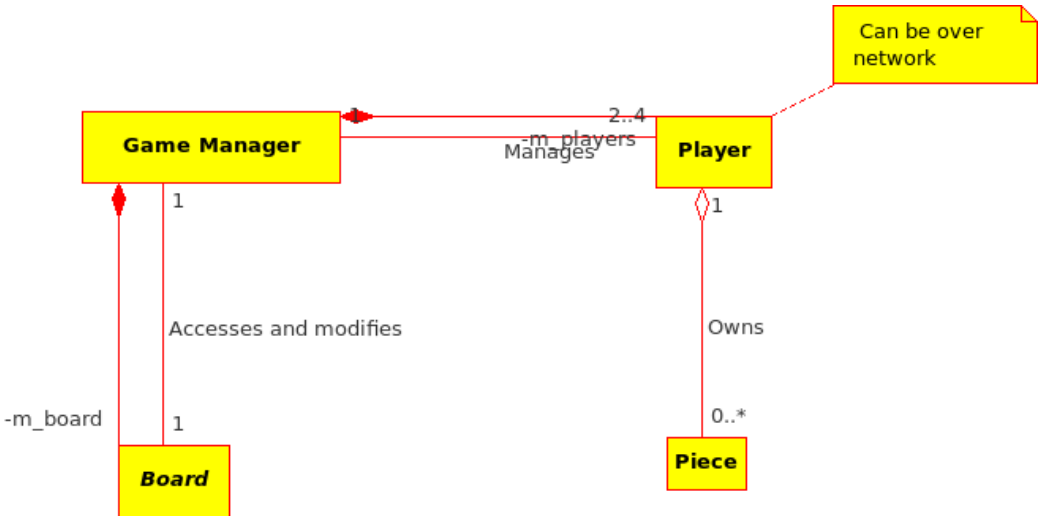

Figure 1: Domain Model Diagram

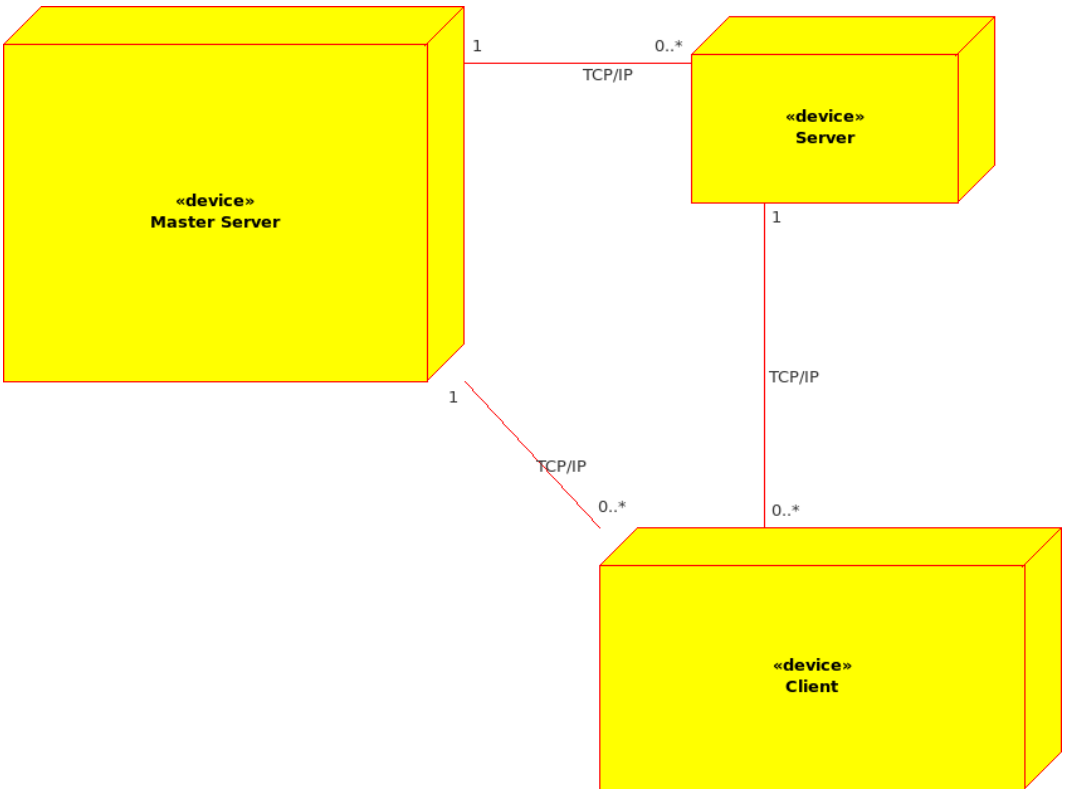

Figure 2: Deployment Diagram

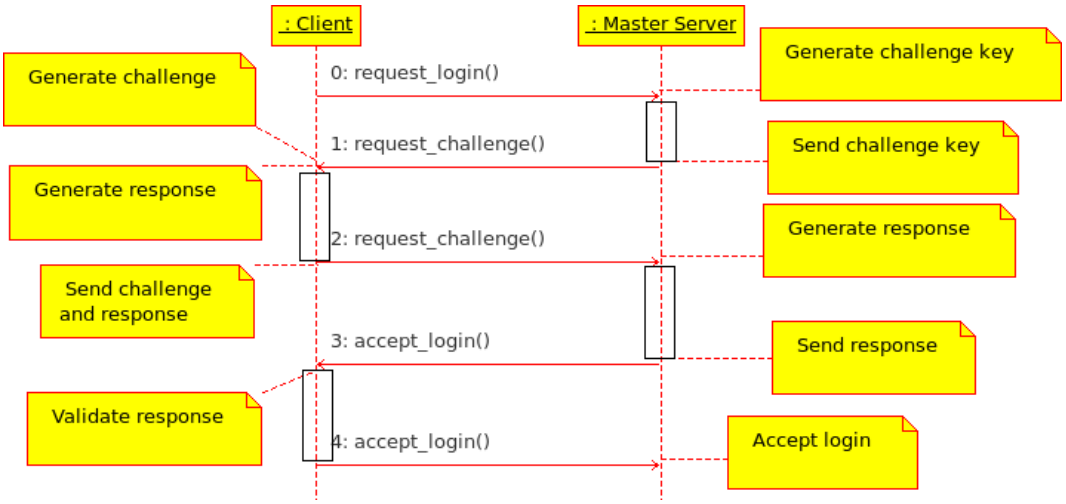

Figure 3: System Sequence Diagram: Client and Master Server

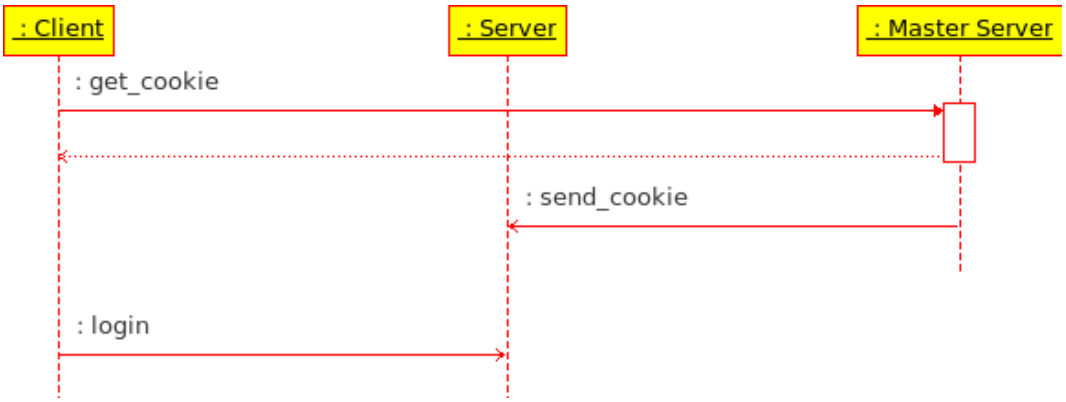

Figure 4: System Sequence Diagram: Client and Server

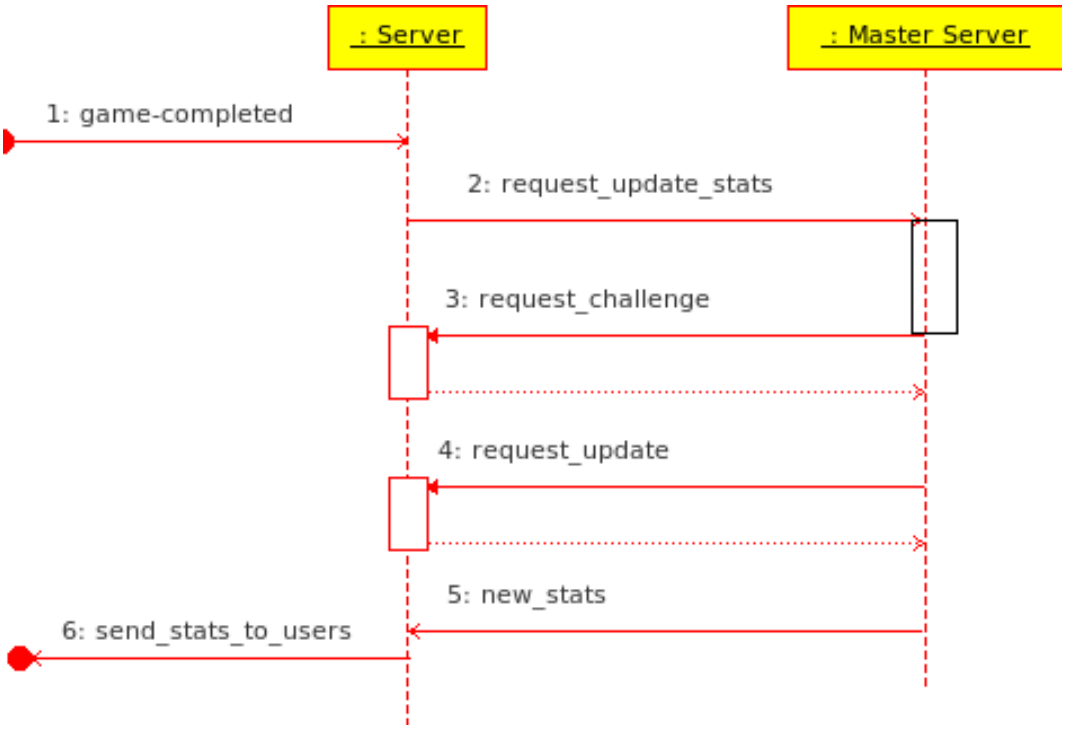

Figure 5: System Sequence Diagram: Server and Master Server

Feature	Time (man-hours)
Game Data Structures	22
Plug-in API	12
Game Rules	10
Board Evaluation	7
Control Algorithms	6
Graphics	7
Players	5
Networking	12
Sounds	7
Total	88

Table 2: Effort required to code features