WARFRUIT™

Presented By

Crazy Monkeys

Team C1:

Shaan Bushra

Steven Cheung

Vlad Rotea

Jacek Stefanowski

Andy Tsang

Final Report

Table of Contents

1. INTRODUCTION

1.1 Motivation

This project required us to produce software, particularly an interactive game which is fun to play. When deciding on the type of game to make, we drew ideas for WarFruit from games such as Command and Conquer and other such strategy games. Our game was a result of the integration of our unique ideas along with some concepts from these other games. Throughout the development of the software, we have learnt the necessary skills to produce it, for example, increasing our abilities in Java Swing, learning about the Lightweight Java GL library and other libraries. We have also learnt how they can be applied to creating a real time, interactive strategy game.

1.2 Context

WarFruit™ is a real time multiplayer strategy game which allows up to 4 teams to play in a single game, where the 2 members in each team can take the role of either an “attacker” or a “defender”. The ultimate aim of the game is to fight and destroy other teams. A secondary objective of the game is capturing and occupying the golden apple, located at the centre of the map. Once all other teams are destroyed, you win the game. This is done by building and training different monkey units, building structures such as hospitals, resource training centres and monkey hospitals, collecting resources such as bamboo, stone and fruit and attacking the opponents along with the hostile AI monkeys. The game handles user input through use of a mouse and keyboard. There is a GUI displaying a scrollable area of the map, and several control options are presented to the player. Upon starting the game, each team is shown a top-down view over their base, located in a preselected corner of the map. The team is in control of a “Monkey Headquarters” building, one “Builder Monkey”, used by the builder to construct structures, one “Banana Throwing Monkey”, used by the attacker to attack hostile monkey teams and one “Collector Monkey”, used by the attacker to scout for and collect resources found in random places on the map. Teams can also train special units as a bonus by occupying the golden apple. In the open world map, through the use of different structures such as “Monkey Hospitals”, “Banana Warehouses”, and defensive structures such as “Pineapple Throwing Towers” and “Flamethrower Towers”, the team will unlock capabilities such as training new units, improving unit and structure stats and building other structures.

1.3 Outline

This report aims to highlight the features and other important aspects of WarFruit™ such as real-time networking and HCI aspects, for example user interaction and satisfaction. It will also cover the overall journey through the process of the game's development. It includes the design of the software, showing both high and low level class structures along with the various software engineering and risk management techniques implemented in the making of this game. The essence of this report is the organisation of the team and the evaluation of the success of this project as a whole.

2. SOFTWARE DESIGN

As mentioned in the above section, WarFruit™ is a real time strategy game with a massive number of troops and buildings. Due to the large amount of data, it is impossible to implement the modern client/server model where the server runs all the game code and the client is a view of the state held by the server (Fiedler, 2010). Therefore, WarFruit™ is designed as a client-based multiplayer game in which the game logic is run on each client. Only user commands are exchanged between the client and the server to alter the state of the game (Terran & Bettner, 2001). Instead of running the game code, the server in WarFruit™ works as a repeater of user commands and redirects them to all clients. This kind of game model requires very accurate time and state control in the game, since the synchronization of each client is easily disrupted if the user commands are run in the incorrect order. The mechanism of maintaining the synchronization will be discussed in a later section, while this section will focus on the class structure and design pattern of the game.

2.1 Software Components

WarFruit™ is designed and built in three major components: the game engine, networking and the game lobby. These components are subsequently divided into several sub-parts in order to differentiate the separate logical sections, for example, the map and team logic is contained within the game engine. This separation allows the team to focus on different components simultaneously and minimize conflict when integrating the system. Figure 1 illustrates a simplified high-level class diagram, which describes the features of the three components.

2.1.1 Networking

The networking component of the software project is built to abstract away details of creating sockets and connections between hosts. It is composed of two portions: chat services and game update services. Both services use a defined set of objects representing state controls, which are sent over the network. All of these 'packet' objects contain the IP address of the sender, and a 'payload', containing the actual data being sent. In the case of the chat services, this is the string message the user has typed, and an optional control character, which allows for the server to broadcast updates such as username changes for display in the chat window. The game packets mirror all inputs that the user can make, such as moving and training units, building and attacking and healing, amongst others. Extra control packets can be sent, such as team and ready selection, secondary server selections and disconnect requests. Each of these objects contains a member of an enumerated type, allowing for differentiation of command types and logical decisions to be made on the course of action to take on each command.

Server contains the high level server logic, deciding whether to act upon a request or simply repeat game updates and chat messages to clients.

NetworkManager is a general class subclassed by ***ServerNetworkManager*** and ***ClientNetworkManager***, which are used by the server and clients, respectively, to communicate over the network.

Broadcast creates a Multicast Socket to send broadcasts to a specified multicast subnet.

ServerFinder is the client's compliment to ***Broadcast*** which receives the broadcast messages from servers and updates the client of those that have been found.

Listener is a general class that is subclassed by ***ChatListener*** and ***GameListener***, which provide a framework to listen for incoming updates and inform the parent client or server.

ConnectionHandler provides a framework for the server to listen for connections made by clients.

2.1.2 Game Lobby

ClientGUI is instantiated once the user launches the application. The lobby consists of several components: an options page, a server list and a game room. The options page allows users to change their preferences, such as their username. The server list shows the available game rooms that have been opened for users to join. Users also have the option to open a game room. The game room allows the user to choose their team and role.

2.1.3 Game Engine

GameControl is the controller class for the whole game. It runs the game loop and receives user input from *GameGUI.* It directs this input to the respective controller class. It also contains a queue of user commands that are received from the network and executes these commands by iterating over the queue.

MapControl is the controller class of the map in the game. It receives user input from *GameControl* and creates the desired command object, for example, moving units across the map. It is also responsible for running the game logic by calling the *doAction()* method of every *MapObject*.

MapObject is a generic entity class of all objects that will appear on the map. It contains the basic parameters of such objects, for example, the position on the map, visibility, and performing action. It also contains various important methods that are vital for the game engine, for instance, collision detection rendering. Its descendants are as follows: *Tile*, *Weapon*, *GatherFlag*, *Unit*, *Building*. It also contains a synchronized *mapObjectID* that is used to recognize the same object in a different client.

NavigationControl is a controller class that is used for path finding. It contains a reference to the *TileGridControl* in order to access the tiles, which provide the necessary information such as the walkability of a tile. It also contains an inner class *PathFinder* that implements the interface Runnable. *PathFinder* implements the A* algorithm over the tile grid to find the shortest path to a position. This operation is run on another thread since it usually takes more than 15ms to complete which will affect the fps of the game.

NavigationNode is an entity class that wraps up a tile and calculates the costs and heuristics of that particular tile from the starting pointing in the process of path finding.

PanelControl is the controller class of panel and the panel buttons. It receives the user input from GameControl, creates respective user commands and draws the respective object on the screen, for instance, to control units and build buildings.

MiniMapControl receives user input from *GameControl* and creates the desired command object that is related to the *MiniMap*.

TechnologyControl controls and manage all *Technology* objects. It initiates the *Technology* objects at the beginning of the game and unlock them upon valid user command is received.

TeamControl controls and manage all *Team* and *Player* objects including *AITeam* and *NeutralTeam*. It initiates the Team and Player objects at the beginning of the game upon receiving the required parameters from *GameControl*. It is also responsible for running the AI players logic.

AITeam is the entity class of AI players. It has a *runAI()* method to run the pre-set commands.

Player is the entity class representing a client in the game. It contains a *playerID* that is used while sending the user commands.

2.2 MVC Model

The overall design of the classes follows the MVC model to separate the game logic from the GUI. For example, the *GameGUI* is used to receive the user inputs via mouse and keyboard, and then send the inputs to *GameControl* which receives and processes the inputs in order to decide what action should be taken. This ensures the changes in the GUI do not affect the game logic ensuring the fewest amount of conflicts in code when the team are working on other parts of the code base.

2.3 Command Pattern

As mentioned above, the online multiplayer feature is done by exchanging user commands across the network. These commands such as moving units or attacking enemies' objects will be frequently created. A command pattern allows the command to be efficiently sent across the network and be performed in each client. Figure 2 shows a snippet of the command classes. When the system receives user input, it will create the respective *Command* object and call its *sendCommand()* method. This method will wrap the *Command* into a serializable *NetGameCmd* object and send it across the network. When the *NetGameCmd* reaches a client, it will create the respective *Command* object and perform itself by calling the *perform()* method. This design pattern allows more features or commands to be implemented very efficiently in the future using the same technique.

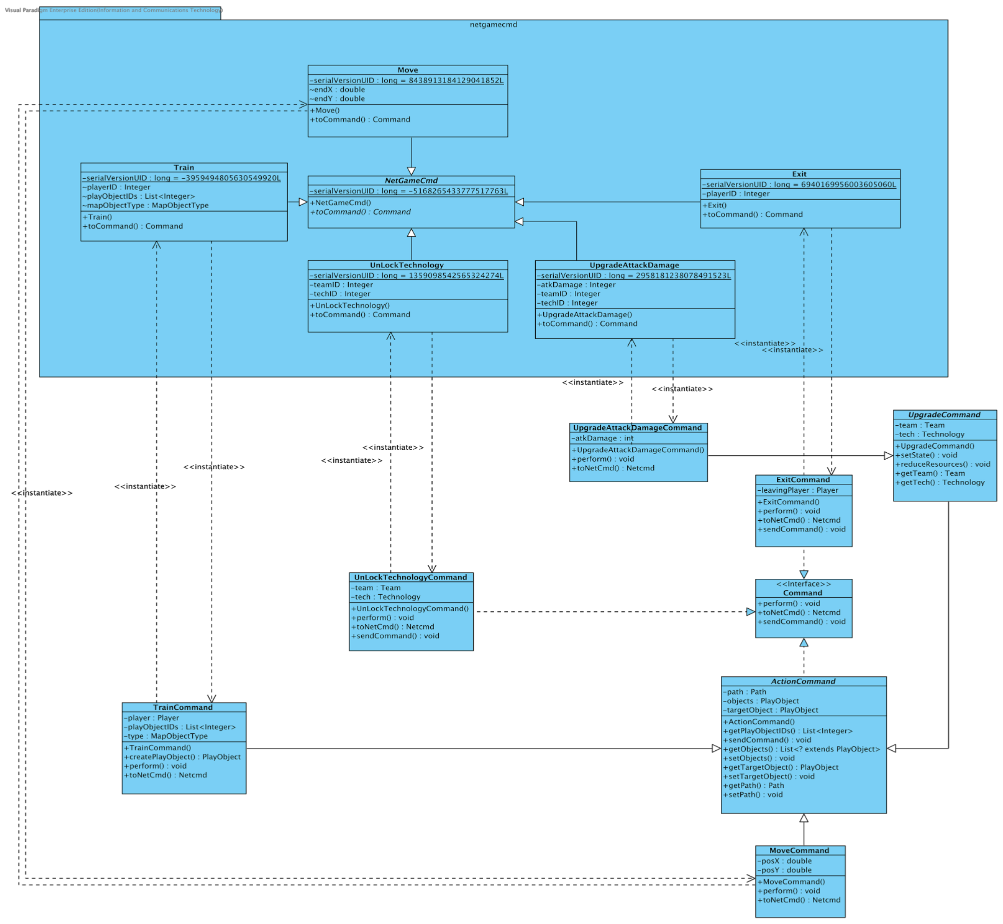


Figure 2: Command Pattern

2.4 Factory Method

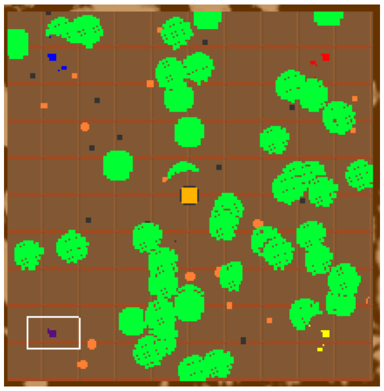
Factory methods are widely applied in the this software. For instance, both the *toCommand()* method in *NetGameCmd* and *toNetCmd()* method in *Command* are factory methods to create objects. Furthermore, there is a *createWeapon()* method in the *AttackingUnit* class which creates a *Weapon* object. The factory methods allow the creation of an object to be deferred to the concrete creator class such that the subclasses of *AttackingUnit* can override the *createWeapon()* method to create different kinds of weapons without significant duplication of code.

3. GAME DESIGN & HCI

The game design and HCI are always the most important aspects of a game, which directly affect the gaming experience of the players. Thus, our company have researched similar games on the market, for instance, "Age of Empires" (Ensemble Studios, Microsoft) and "Command & Conquer" (Westwood Studios, EA Games). This section will focus on how Crazy Monkeys developed WarFruit™ as a competitive game based on world design, gameplay mechanics and HCI.

3.1 World Design

WarFruit consists of a freely walkable map which has a golden apple at the center. This is one of the objectives, inspired by the top view of a MacBook. This world is full of variety, including randomly generated resources: stone, bamboo and fruit. This allows the user to have a different playing experience every time they play the game. The 4 teams in the world are easily identified by the color bars (Red, Blue, Yellow, Purple) attached to the units. The attacker has the duty to collect resources and attack the others while the builder has to build and defend the base.



These player roles is one of the main features which sets WarFruit™ apart from other games in the genre.

3.2 GamePlay Mechanics

WarFruit™ is a real-time strategy game, which players can enjoy in different ways; single or multiple player against AI, single player against each other or even battling between teams. In each case, the player still has the same goal; destroy the other teams and capture the golden apple. In order to achieve this goal, especially since the players in the same team share resources with their allies, a high degree of cooperation is required when collecting resources and creating buildings and units.

3.2.1 Real Time Strategy Gaming

The real-time features of the game require players to think carefully when commanding their units. Unlike turn based games, users will not be provided a significant amount of time to think about their next move. This feature could be difficult for new players who have never played a real-time strategy game before, however, once they know how to control and command their units, they would experience much greater flexibility then a turn based game.

3.2.2 Resource Management

Resource management in WarFruit™ is one of the greatest challenges to players, since attackers and builders share the same resource system, unlike "Age of Empires" or other games, where separate resource systems exist for each player. Moreover, each unit or building in the game has different resource requirements, hence players have to communicate with each other to clarify their priorities and targets. For instance, at the beginning players only have limited resources, if the builder uses all of them to train builder monkeys while not helping to expand their collecting team, they would have a hard early game. Mid-game, players would have to co-operate in training an army, unlocking technologies, or building more defensive buildings. Every small decision made by the team might lead them to defeat the opponents, or lose the game.

In addition, different resources are spawned in different amounts on the map, bamboo being the most abundant, stone being the rarest. Therefore protecting and collecting resources would be another issue players have to consider.

3.2.3 Commanding Units (Move, Attack, Heal, Repair)

Each unit or building has their own commands which can be performed by right clicking on the map or the control panel. Attacking units have different damage, range and attack speed. this variety promotes interesting gameplay. For instance, 'bananacine-gun monkeys' have high attack speed but low damage and range, whereas 'fist fighting monkeys' have high mobility and low cost but melee attack and less health. Players may decide to train lots of cheap and fast, but short range, units to counter units like artillery. However, another player might use 'pineapple throwing monkeys', which have range area damage, useful to counter against 'fist fighting monkeys'. Other than attacking units, healing and repairing units could also contribute to the methods of game play. By upgrading healing and repairing powers, such units would be able to recover more health than opponents would anticipate, which could lead to winning a battle with the advantage of surprise. This variety create many possibilities for players in creating their own strategy to play the game.

This variety create tons of possibility to players on creating their own strategy to play the game.

3.2.4 Victory

The game is won when all of the opponent team bases and units are destroyed. Victory can be achieved using different techniques, for example, as mentioned above, by building massive armies to defeat all the opponents. Another possible way, would be to occupy the golden apple, which will unlock the mysterious and unique unit the 'golden apple protector', enabling players to dominate the game. Destroying the bases of others, by using any of these methods, emphasizes the importance of the role of the attacker and builder, each having to do their best and stick to their role so as to win the game.

3.3 HCI

The user interaction of WarFruit™ comes from a wide range of user tests, including computer science students and users who had no previous experience of games.

3.3.1 Consistency

User interaction within the game has been developed through consultation of other games of this genre, so as to allow users to pick it up easily. For example, "left click" lets the user select their units and buildings. When they are selected, the panel will change simultaneously, showing the units' status and control buttons. "Left click" also allows users to interact with the buttons in the menu and panel to perform various actions like detraining from a build or even stopping a unit's action. "Right click" lets user perform different actions based on the unit selected, for instance, if a user has selected a building, it will change the assembly point. Conversely, if a user has selected an attacking unit, it will command the unit to move to the targeted position or attack the targeted enemy. "Double click" is similar to "left click", but will select all units with the same type. Several keyboard hotkeys are available as well, for instance, users could enter the chat mode by pressing "enter", or enter the game menu by pressing "esc". This type of interaction is very similar to "Age of Empires" or "Command & Conquer", where left and right click both have clear functional usages.

3.3.2 Informative Feedback

Feedback to the user has been implemented using both sound and graphics.For example, when a user tries to build a new building, an overlay will appear on the cursor. This tells the user which area is valid for building (in green) and which is not (in red). (See right). If the user continues to try to build on an invalid area using left click, the user will be notified by an error sound effect and an alert message below the top bar. Similarly, if the user clicks on unavailable buttons or technology branches, the system will notify them in the same way, using an appropriate error message.

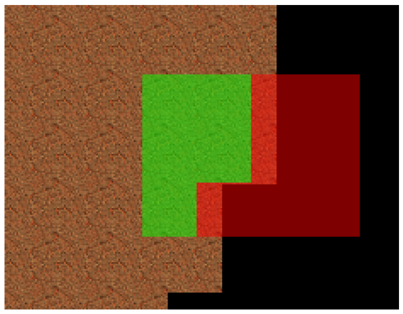
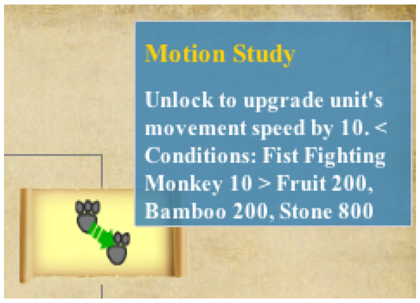


figure 1

To allow the user to interact with the system more accurately, hover windows have been implemented. When a user moves their cursor onto a button, initially the button background will change to notify of the user's cursor position. A hover window will also appear providing useful information to the user such as button hotkeys, a functional description, resource requirements or even conditions for attainment. (See left).

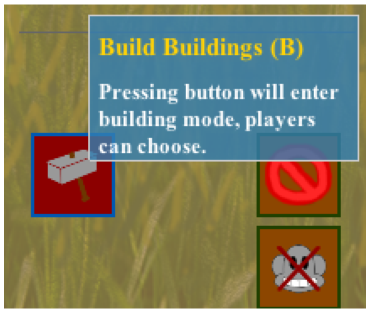
figure 2



Motion Study

Unlock to upgrade unit's movement speed by 10. < Conditions: Fist Fighting Monkey 10 > Fruit 200, Bamboo 200, Stone 800

figure 3



Build Buildings (B)

Pressing button will enter building mode, players can choose.

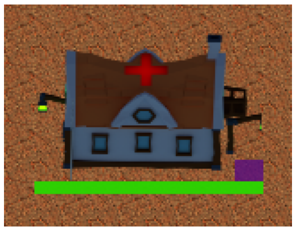
The color bar on the unit is used to show which units belong to a player. A player knows their team color, because before the game starts, the players are able to choose their team and role in the game room. The colours used are distinctive, and in addition, the neutral team and resources also have separate colors indicated on the minimap. Thus there should be no problems for a user to recognise which units belong to who.

Another graphical feedback mechanism is the team status panel. The user can view the panel by holding “tab”. It provides information of the player's unit statistics and opponents total units. This could help a user on deciding their strategy.

Sound is another important issue, one of the best examples is the error sound effect mentioned above. However, there are a lot more audio feedback mechanisms implemented in the game. For instance, when a new chat message is sent by other players, the user will be notify by a "pop" sound. In addition to this, users are able to get audio feedback in the game room as well. There are two different music tracks used, one in the game room and one after the game has launched, which can help user to realise that the game has started.

3.3.3 Aesthetics

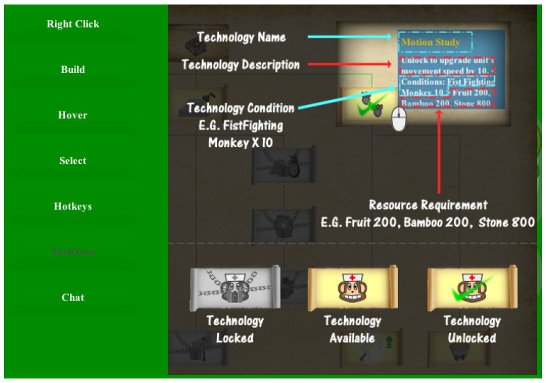
The chosen aesthetic of the game is close to real world objects, so as to effectively representing the unit's / building's function. For example, the building shown on the left is obviously a hospital.

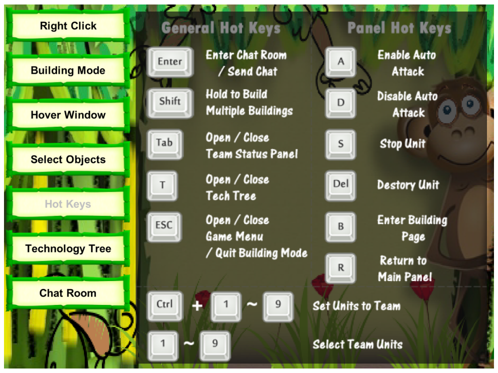


3.3.4 Learnability and Memorability

A well designed game is easy for a user to learn and memorise. In WarFruit™, various tools have been provided to the user, such as hover windows mentioned above, which help a user to learn different conditions for different technology, units or buildings.

Another main feature is the help window in both the game menu and inside the game. Sufficient instructions are provided by the help window for a user to learn how to control their units and conduct different operations.


Right Click
Build
Hover
Select
Hotkeys
TechTree
Chat
Technology Name
Technology Description
Technology Condition E.G. FistFighting Monkey X 10
Motion Study
Unlock to upgrade unit's movement speed by 10.
Conditions: Fist Fighting Monkey 10 > Fruit 200, Bamboo 200, Stone 800
Resource Requirement E.G. Fruit 200, Bamboo 200, Stone 800
Technology Locked
Technology Available
Technology Unlocked

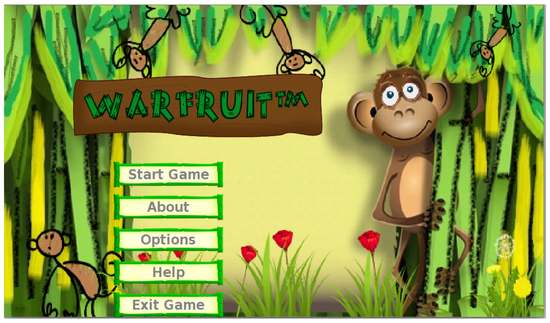


General Hot Keys		Panel Hot Keys	
Enter	Enter Chat Room / Send Chat	A	Enable Auto Attack
Shift	Hold to Build Multiple Buildings	D	Disable Auto Attack
Tab	Open / Close Team Status Panel	S	Stop Unit
T	Open / Close Tech Tree	Del	Destory Unit
ESC	Open / Close Game Menu / Quit Building Mode	B	Enter Building Page
		R	Return to Main Panel
Ctrl + 1 ~ 9	Set Units to Team		
1 ~ 9	Select Team Units		

4. GUI / AI / Networking

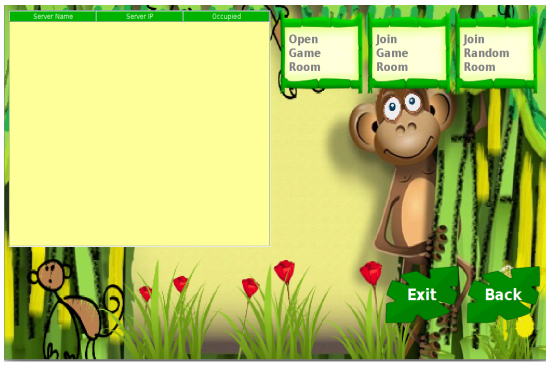
4.1 GUI

4.1.1 Welcome Screen



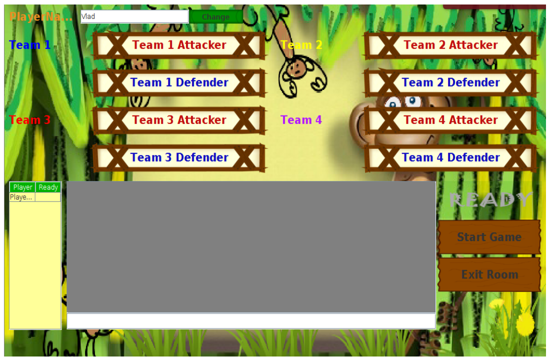
At launch, WarFruit™ displays the welcome screen, from which a user can choose between different options. Start game will open a game lobby which is described in section 4.1.2. The About menu contains information about the developers. From the Options menu, a user can change different features in the game, like set a default username, music volume or scroll speed. The Help window contains tutorials about gameplay and different in-game features.

4.1.2 Server List



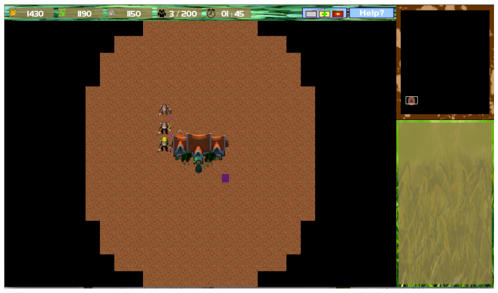
The game lobby is reached via the Start game button from welcome screen. From here, users have different options, opening a new game room, joining an existing one or joining a random game room. The left table depicts information about existing servers; the server name, server IP and number of players who have already joined. The buttons situated in bottom right corner will let you to go back to the welcome screen or exit the game.

4.1.3. Gameroom



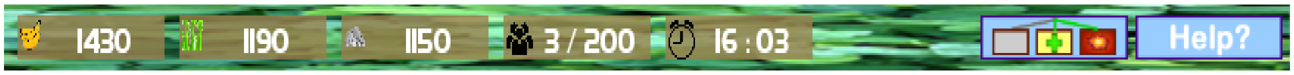
After a user joins an existing server or creates a new one, the game will redirect them to the game room window. From there, the user can choose a team and which role they want to play. Also, the game room includes options for changing username, in-game chat and a list of other players and their status (ready or not). After users select teams, they must set their status as ready. When everyone is ready they can start the game.

4.1.4 Main Game



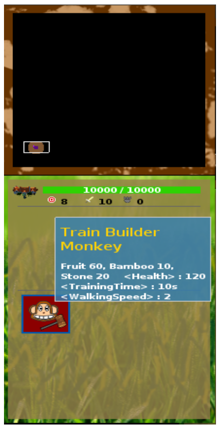
After the game is started, the user will be redirected to the main game window. This window is full screen and consists of 3 main components: a playable map, top bar (4.1.4.1) and control panel (4.1.4.2).

4.1.4.1 Top Bar



The top bar is situated at the top of the main game window and shows information about game resources. WarFruit™ has 3 types of resources which are described by pictures in top bar. The top bar shows the user's monkey population size out of the maximum population along with a timer and buttons for the technologies menu and the help menu.

4.1.4.2 Control panel



The control panel is drawn on the right of the main game window and it contains two main objects: the minimap and control features for the selected object or group of objects. The minimap depicts an overview of the map and also offers functionality for camera control over the map. When an object is selected, the control panel shows it's lifebar, attack and defense power and moving speed. When a group of objects are selected, the panel will show the selection with representative icons for each object selected. Also, when a building is selected, the panel will show the queue of monkeys that are currently in creation/training. In the bottom of the panel, buttons for an objects functionality are placed. These include functionality for creating new buildings, new monkeys or destroying currently selected objects. Hovering over buttons will open a tag which describes button functionality and resources needed in order to perform the action in question.

4.1.4.3 In Game menu

The game options menu is reached by pressing "Esc". It provides functionality for changing scroll speed or sound volume. From this menu the user can exit the game, or resume the game after they have made changes.

4.2 Artificial Intelligence

This section will describe the artificial intelligence techniques that are applied in this software. There are three major AI components: Path Finding, Auto-action and AI Player.

4.2.1 Path Finding

The path finding technique is applied to all moveable units to get to a position on the map with the shortest path, whilst avoiding blockages. An A* search algorithm over the tile grid is used as the basis of such a technique. The search requires two sets of input parameters: a start position and a target position (Lester, 2005). It will perform the shortest path algorithm by calculating the costs and heuristics of each route. However, the A* algorithm will return nothing if the target position is unreachable (which is a common scenario in this game). Therefore, the Path Finding technique is enhanced to satisfy this situation. If no direct route is found, the system will calculate and set the closest reachable tile to the original destination as the new target position and return the route.

4.2.2 Auto-action

In order to simulate a real-world experience, units and buildings are able to start actions without user control. For example, a building will automatically attack an enemy's unit which passes by its vision range and collector units will continue to collect nearby resources after the previous one is destroyed. This allows users to manage objects effectively and efficiently.

4.2.3 AI Player

As mentioned in the previous section, this game is played with four teams. If there are not enough players, AI Players will automatically fill the remaining teams. The AI Player is similar to a human player. They have to send out commands in order to control their troops or buildings. They also have to collect resources so as to be able to train more units or build more buildings. The priority of the AI Player is to collect resources as much as possible, because resources are the key to victory. Concurrently, the AI Player will train attacking and healing units to prepare an attack on other teams. After 30 attacking units have been trained, the AI Player will randomly attack an enemy team.

4.3 Networking

4.3.1 Before Game

When the user is in the server list GUI, they can start a new server, or join any servers that are available on the subnet they are connected to. If the user chooses to start a new server, a new server object is started on their computer, yet from that point on their experience is that of any other player, connecting to their server in the same way as any other computer on the network. The server will create a list of hosts, initially only containing itself, which will be sent to each client when changes occur to the state of any host connected or if a new host connects. In addition to this, server sockets for chat and game commands are opened, with corresponding connection handler threads. On connection of a client, a Host object is created, containing the sockets to that host and it's state, which is added to the hosts list. The server also starts a broadcast, using the Broadcast class, to send a multicast message to all clients containing the current number of players connected to the server's game room. This enables updating of the server list GUI by the client, through use of the ServerFinder class.

4.3.2 Pre-Game

When a player in the game room selects a role, a team selection command is sent to the server. The server checks through the hosts list and if that role ID is free, sends the updated hosts list out to all clients. It is based on this that the client's GUI then updates to show the change. Similarly, when the ready checkbox is selected or deselected, a ready command containing a boolean representing the selection is sent to the server, which again, sends out the updated hosts list. The start game button sends a start game packet to the server. The server then checks through each host in the list, checking whether everyone is ready to play. If so, a start game packet is sent back to all hosts, containing a list of the currently filled roles by ID, an identification of the host computer, and whether the computer receiving the start game command is hosting the server or not. This is the mechanism by which the host computer knows to run AI teams. These teams use the same sockets as the host client. If a player is found not to be ready, the server sends out a chat notification, informing the players of who wants to start the game, and which players are still unready.

4.3.2 In Game

There is no change in the server when the game starts, as all the necessary connections are already set up. On start of a game, the server ceases the running broadcast. From this point on, the server effectively becomes a simple repeater, forwarding game updates to clients.

4.3.3 End of Game

When a server exits, it sends out a closing broadcast. On disconnection of a client, the client is simply dropped from the hosts list. When the server disconnects, however, assuming it is not the end of the game, all clients automatically pause the game. A predetermined secondary server starts a new server object, which goes through the normal start-up process and listens for connections. However, once everyone has been reconnected, the server sends a resume command to all clients, and the game continues.

5. SOFTWARE ENGINEERING AND RISK MANAGEMENT

5.1 Project management techniques used

Initial WarFruit game design included a lot of technical aspects which we had to cover by an adaptive project management technique. Agile provided us all we needed in terms of project management.

Agile features

- Adaptive planning - where changes were necessary, they could be continuously incorporated into the project, in order to bring improvements in the software.
- Evolutionary development - we added features on the fly, based on different ideas that we considered an important improvement or an interesting feature.
- Rapid development - we had coding session where all team members were present in order to discus quick changes or bug fixing. All members being present changes have been solved in short time with maximum efficiency.
- Efficient and face-to-face communication - 2 weekly meeting in which we discussed what we have done from the last meeting and what we need to get done until next meeting
- Quality focus - test driven development, with automated unit testing and integration testing.

5.2 Risk management

5.2.1 Identifying risk

Warfruit is ambitious large project, and as in any project things may go wrong. It was important for us to have an idea about functionality that has a high failure risk and have a backup plan in case of something failing. Apart from software/hardware failure, we had to consider personal related problems and handle them with minimal impact on project development.

5.2.2 Managing risk

- Computer malfunction - working code on the SVN repository - Impact Reduction
- Personal issues - having more than one team member up to date with functionality of a particular piece of code - Impact Reduction
- Integration failure - unit and integration testing - Probability reduction
- New functionality that might fail - branching project on SVN, in order to have a working copy at any time - Risk Avoidance

The change management process of this project can be expected to be efficient, due to the adoption of the popular agile development process. Where changes are necessary, they can be continuously incorporated into the project, in order to bring about improvements in the software. Changes made to the game which are different from the original specification should be properly documented. This can be done via unit testing, black and white-box testing and analysis of use case scenarios, allowing the developer to easily monitor any changes at a later date. All the team members can make changes to their particular component in the software and this should be approved by the project manager. However, it should be remembered that any changes will need to be reported and estimations should be done before implementation, so that the project can be completed within the project timeframe. This document may be revised by the team if it is realised that the requirements for a particular component of the software system have changed. The team will discuss potential alterations and agree to what needs to change, via meetings or group discussion. Section 3.4 Classes and Objects and Appendix 6.1 Gantt Chart, may be particularly liable to change, since whilst developing the game certain aspects of the designed hierarchy or timeframe constraints may be cause for alteration of the development strategy

6. Evaluation

6.1 Evaluation of Objectives and Aims

The success of this project can be measured against the requirements and goals stated in the Software Requirements Specification (SRS) document for our game.

6.1.1 Functional Requirements

The functional requirements have been successfully incorporated into the game, for example, when networking is considered our game provides the user the ability to create servers, so as to host local area network games. If a player is unable to connect to the server or loses a connection during a game, they will be informed via an appropriate error message, such as a BugReport. When display and units are considered, the system generates an open world map to represent the game world, and show user controls through a GUI. The system also generates a new unit, with associated attributes such as health, type, and usage. These were stated in the specification and our game successfully matches the criteria.

6.1.2 Non-Functional Requirements

The non-functional requirements of the game have also been successfully implemented. This can be seen from evaluating various aspects of the game such as performance. Our game operates in a robust manner, and is fast enough to support at least 8 players connected via a network.

6.1.3 Reliability

The aspect of reliability has also been met as our game has been extensively tested and bugs are rectified.

6.1.4 Availability

In terms of availability, the game allows one player to act as the host server while everyone else is connected.

6.1.5 Security

When it comes to security, there are no major security issues in our game hence network security may be treated as low priority.

6.1.6 Maintainability

In terms of maintainability, our game is clearly commented and is accompanied by JavaDoc. Our game is also portable, allowing it to work on all major operating systems. These were stated in the specification and once again our game successfully matches the criteria.

6.1.7 Usability

Usability testing has revealed that our game is fun to play with, which was one of our topics of focus during implementation. The game has a clear outlook, with clearly labelled buttons.

6.2 Evaluation of Project Management

The Gantt chart in this report shows the initial project plan set at the beginning of the project. Since it shows the expected start and approximate completion date, it is crucial to know that work has been completed by each team member accordingly. It has greatly assisted us in implementing plans and making changes to our game within our set deadlines each week. Our project management has been quite successful due to the fact that we all placed ourselves under this working deadline. Furthermore, we also had regular meetings with our tutor, which has allowed us to set our targets and ensuring that work-rate remained consistent.

6.3 Further Work

Certain elements in this project leave scope for further development. For example, along with team chat and global chat functions, there could be a facility for point to point private chat. If extended into a full enterprise environment, the game could become a massive multiplayer online game, with a persistent game world.

7. TEAMWORK

7.1 Organisation

The essence of this team project is the organisation of our team and how we communicated with each other in coming up with our game. We did this by using various communication channels, such as Trello, Canvas, Facebook Chat, Skype and emails. We mostly used Trello to keep each other up to date with our work. We also engaged in face to face meetings twice a week to ensure that every team member is aware of their own tasks and is progressing according to their targets. We discussed in person what type of game to make and how we would implement it. We mostly did coding at home in our own time and kept each other updated about our tasks completed on a regular basis, including committing codes on SVN. This team project has greatly assisted us in getting to know each other's strengths and weaknesses and allocating particular areas of the game to each other. The project also encouraged pair programming where two team members worked together at times to code quickly and point out mistakes during coding. It also enabled us to brainstorm ideas to add new features to the game. Our team had an autocratic/democratic structure where our team leader would delegate responsibly to us for particular aspects of the game but it was also democratic in a way because we also were able to choose what tasks to do and within our set deadlines each week.

7.2 Methods of Communication

7.2.1 Team Meetings

The team conducted bi-weekly SCRUM meetings, where we discussed the status of the game. During these meetings, we discussed and assigned tasks for members to undertake over the next week or weekend, and provided each other with status updates of how our previous week's work was going.

7.2.2 Trello

The team used Trello to organise tasks that were to be done, as well as post meeting times and minutes. Using the features available, we classed tasks as important, or less important. This allowed us to get tasks done in suitable amounts of time.

7.2.3 Facebook Chat

For less important immediate discussion, such as when bugs were found and general discussion how the game should work, we used Facebook's group chat feature. We also used this facility to iron out misunderstandings that had occurred in meetings.

7.2.4 Team Coding Sessions

The team congregated to work on multiple parts of the project concurrently. A useful aspect of doing this was that when anyone had ideas or found integration errors, it could be implemented or solved almost instantly.

7.3 SVN activity

Regular code commitments were made on SVN after a piece of code was written by a team member. So, this team project enabled extensive use of SVN. We made a total of 582 commitments. We made commitments on SVN every time we updated the code or added a new feature or if a bug was fixed. Communication was vital during updating a code on SVN because we had to make sure same every team member was aware of the latest version of the game and if any changes were made which could have created conflicts in another person’s code classes.

7.4 Problems

Problems arose when some members of the team found it a little difficult to keep up with their tasks on time as they also had other responsibilities on top of this project. To make sure things were still done on time, we called meetings to set new deadlines for team members and agreed to share the workload in case they were unable to complete their assigned tasks set within the new deadline. The main objective was to work collaboratively and finish the game within the final deadline. Meetings over Skype were also carried out to ensure a particular task was finished by a set time limit.

8. Summary

This report gives an overall, in-depth view of the various aspects involved in the development of this game and provides an insight into those aspects, such as teamwork and the game's design. It also covered how these features were integrated together to come up with a successful game.

8.1 Software Design

The creation of our software depends greatly on the fundamental principles that were set at the beginning of this project. This section shows detailed information about the assembly and integration of the different software modules such as networking, game lobby, game engine which then outlines the inside features such as *MapObject, NavigationControl, MiniMapControl* and more.

8.2 Game Design & HCI

This section focuses on the user and how they are going to interact with the game. Features of the user interface have been outlined in great detail such as the game world design, HCI and the game play mechanics. Further exploration of game features such as, resource management, moving, attacking, healing, repairing and winning the game were then explored.

8.3 GUI / AI / Networking

This is probably one of the largest sections in this report that outlines in depth the graphical user interface design for our game. The GUI section shows in detail the various components that comprise the interface, such as the game lobby and game room. The AI section features how the artificial intelligence is implemented and the techniques that are applied in this software, such as path finding, auto action and AI players. The networking section shows how the game implements the automatic client connection and restarting server features, which are some of the main highlights of our game.

8.4 Software Engineering and Risk Management

This section demonstrates the software engineering techniques that were used in making this game, also covering how the various techniques are incorporated into the game, given the nature of the game and time constraints. Risk management looks at the identification of risks involved in this game and how the techniques were applied in order to minimise their effects.

9. Individual Reflection

9.1 Steven Cheung Wai Tak (1465388)

9.1.1 Overview

This module has been very challenging and rewarding. At the beginning, our team had no experience on game development and we had to deal with unfamiliar problems within a short period of time. This required very good time management and self-learning ability. Although I have had two years of experience with Java before, I found building a game without using an existing game engine to be another level of difficulty. This module has also been a remarkable experience for me to exercise the software engineering practice and programming skills that I have learnt. Since the time frame for this project was pretty short, project management became very important. This module has also given me an insight of how to produce large scale software by separating it into smaller chunks and integrating them at the end, which is important for my future career. After all these months of good work, I am really proud of what our team has achieved. We have developed software that is far more complete than what I expected at the beginning.

9.1.2 Working as a Team

As a team member in C1, I considered myself being a positive contributor and responsible team member. I was active in discussing the game details in team meetings, throwing in ideas on how to make the game even better and communicating with other team members. I was willing to follow the team decision rather than insisting my own opinions. Apart from that, my main duty was to analyze the game model and to design the related class structure and data flow. I was also responsible for implementing the game engine and AI components. The team worked pretty well during the whole period. Most of us were clear about our team goal and the nature of the game. However, at some point, some teammates were confused about the materials I used in the project and I should have taken more initiative to explain and help others to be familiar with the design.

9.1.3 Summary

Our project has been a success with more than the required features being implemented in only 11 weeks. I have benefitted from this module and it will help me to become a good leader in the future.

9.2 Andy Tsang Chun Yin (1467193)

9.2.1 Overview

After all the weeks our team has spent on the project, I strongly believe our project exceeded our expectation. We have formed a very strong team with almost everyone knowing their duties. We all have the same aim and target of developing an awesome game for the user. Although we have faced obstacles during many stages, we have learnt much on how to solve them as a team. Meanwhile, I have learnt more about teamwork, and how to lead a team in the future, and these experiences were my greatest reward from this module.

9.2.2 Working as a Team

Having had experience of leading a team, I decided not contend for leadership this time. Being a member of the team, besides sticking to my duty, I have learnt to observe others, providing help and assistance to them if necessary. Unfortunately, due to tests and other assignments, we were a bit behind the schedule. However, our group was capable of regrouping and reassigning our priorities, dropping unnecessary features and highlighting the important ones. This lead our team finish the project on time and still have extra-time to develop some bonus features, like reallocating servers and 3D transformation.

The time we have spent together has helped us to get to know each other better, irrespective of particular team member's strong points or weaknesses, helping us a lot in compensating each other's shortcomings. This is key to leading our group to the goal.

9.2.3 Personal Learnings

Though I have over 3 years experience with Java, I have never tried to build a game. Because of this, the project was really a great challenge for me. Throughout the development process, I have learn a lot about the importance of system design, which is an issue I wasn't so much aware of before. I have had a chance to enhance my skills on graphic design during the development of some of the textures, which was one of my weaknesses before. Last but not least, the learning of lwjgl give me a surprising result. At first, I wondered why we had to develop a game without using a popular game engine, but it turns out the knowledge I have gained will benefit me for my final year project.

9.2.4 Summary

Our team has created a piece of work exceeding our expectations. Without this team, I would never have had such an experience on creating such a massive system. These experience will benefit me a lot when I am in the industry in the future.

9.3 Shaan Bushra (1365687)

9.3.1 Overview

This module of team project has brought so many challenges and new difficulties, along with its rewards. Personally, I did not have any past experience of making a game, especially in Java and I think the same applies for the team as a whole. This module has taught us how to manage time and collaborate with other people in making software. The biggest hurdle was integrating our ideas together to make such an interactive real-time strategy game where our programming skills were put on test within a very strict time constraint. This module has really given me the opportunity to apply my software engineering skills into practice. This software was modular, allowing each of us team members to create separate classes for various parts of the game. I was in charge of designing the Graphical User Interface (GUI). I drew the initial design first on paper then the actual GUI was designed. I also made the main menu, it was initially written using JavaFx but then it was later on written by my team member in Swing to keep a consistency in code. I was also in charge of creating textures for the game, like buttons and other GUI textures. I had problems with committing my code on SVN, so I sent my menu and the textures to my team members using Facebook Chat. They then incorporated them into the game. Most of the coding was involved in game logic and networking. In this report I have done the introduction, evaluation, teamwork and summary. Overall, my personal experience of working in a team has been really challenging but yet very enjoyable, it has taught me how to work under peer pressure which actually helped me become more productive.

9.3.2 Working as a Team

I was an active member of the team, discussing ideas and listening to the ideas my other team members came up with. I am more of a listener, so most of the times I used to listen to others ideas then expressing my own views about it. We also made constructive criticisms about the ideas of other team members to help ourselves brainstorm better ideas together. Every team meeting has greatly aided this team project in achieving the final software we aimed for. Regular meetings ensured all of us were always clear of our individual goals. There were hardly any conflicts in ideas during the project because we knew our tasks from day one. I am proud of all my team members and it has been a pleasurable journey and I would love to work with them again if given an opportunity in the future.

9.3.4 Summary

This project has resulted in great teamwork which in turn has led to the successful creation of our game WarFruit. I am so glad to have done this module as I have acquired important skills and I look forward to developing my skills further in software development in the future.

9.4 Vlad Bogdan Rotea (1503752)

9.4.1 Overview

This team project came as a new challenge for me, pushing me to do things that I never thought I would do. My programming background is in C/C++ and I have never developed any kind of game. So, I chose to work on the graphics part of the game, having in mind that my previous experience in C/C++ might be useful in OpenGL development. None of our team members have had any previous knowledge about OpenGL, so I started learning. Isometric graphics (2.5D) had the highest failure rating of the entire project, therefore, as a backup, we were developing using two graphics versions: a top-down 2D and an isometric 2.5D. Even with all the problems we faced during our development, the outcome was awesome: a realtime strategy game done by a small team in 10 weeks.

9.4.2 Working as a team

The team was well balanced, with everyone working on different aspects of the game. Ideas were integrated well together, and everyone had a chance to express their thoughts. The two weekly meetings provided a strong basis for our good collaboration and overall teamwork. I have been working on game models and the isometric view. This turned out to be a lot of work in a tight timeframe, also including a steep learning curve. LWJGL doesn’t provide as much documentation and support as OpenGL C libraries do, and there are quite a lot of differences between them, as C/C++ libraries as mostly written based on pointers. There is a lot of support for OpenGL, but not that much for LWJGL so converting between C++ to Java was time consuming stuff. I tried different approaches on isometric view, but only the third iteration met my requirements.

9.4.3 Personal learnings

My biggest outcome from this team project was learning LWJGL, which is simply OpenGL for Java. Working in a team on such a large piece of software gave me an invaluable experience of team work, new friends and an interesting game to play.

9.4.4 Summary

Facing such a big task to design and develop an isometric game engine was too much for me in the given timeframe, so we decided to leave it for a further development iteration. Even if the engine was completed, the lack of time to test all the functionality forced me in the end to focus all my attention to the 2D version of the game. Models compiled by me and the critical ideas that I brought to game development helped form the fully working game at the end of development iteration.

9.5 Jacek Stefanowski (1483653)

9.5.1 Overview

Having not had much experience with coding in Java previous to this academic year, I have valued the opportunities presented in this module to improve my skills with this language. I have written a few small games previously, and work on projects of varying size, but this is the first time I have had to integrate my skills on a distributed project. As the elected leader of the team, I feel that I have allowed for a democratic environment to flourish, yet able to make key decisions for the team at the right moments. This has allowed us to produce an excellent game, which reflects the skills of the team well. I think that I have been approachable and helped the team to overcome issues that we have had, for example in resolving minor disagreements that may have come up during the development of the game idea.

9.5.2 Working as a Team

I feel that the team has worked incredibly well throughout the course of the module. There have been certain issues, but as a team we have pulled together to get things done. Most members have carried out the tasks allocated with passion and to a good standard. We have been able to integrate our skills well, with each contributing member producing a certain section of the game, whilst happily helping each other on different aspects. Over the course of spending many hours together working on the project, we have bonded well, and were able to give constructive criticism. This meant reflected on the game, as each time it meant the game became a better product.On a few occasions the team slipped slightly behind schedule, but the core of the team were able to work at a good pace to remedy the situation.

9.5.3 Personal Learnings

Having completed the project, I know that my skills in Java have massively improved. I have also improved my skills of communication, becoming wary of issues such as language barriers and cultural differences that may cause slight misunderstandings. The project was an enjoyable challenge, which has allowed me to improve my skills in more areas than I thought, such as communication, leadership and responsibility, rather than just coding.

9.5.4 Summary

I really enjoy playing the game, and I will even more enjoy continuing to work on it after the end of the project. I think this shows that the module has been enjoyable, and that the team has worked well to produce a game that I feel stands out from the crowd as one of the best.

10. REFERENCE

ThinMatrix. (2014). *3D OpenGL RPG Game.* Available: https://www.youtube.com/user/ThinMatrix. Last accessed 22nd Mar 2015.

Gamedev. (2012). *Dynamic objects in Isometric Map Drawing algoryth.*Available: http://www.gamedev.net/topic/629496-dynamic-objects-in-isometric-map-drawing-algorythms/. Last accessed 22nd Mar 2015.

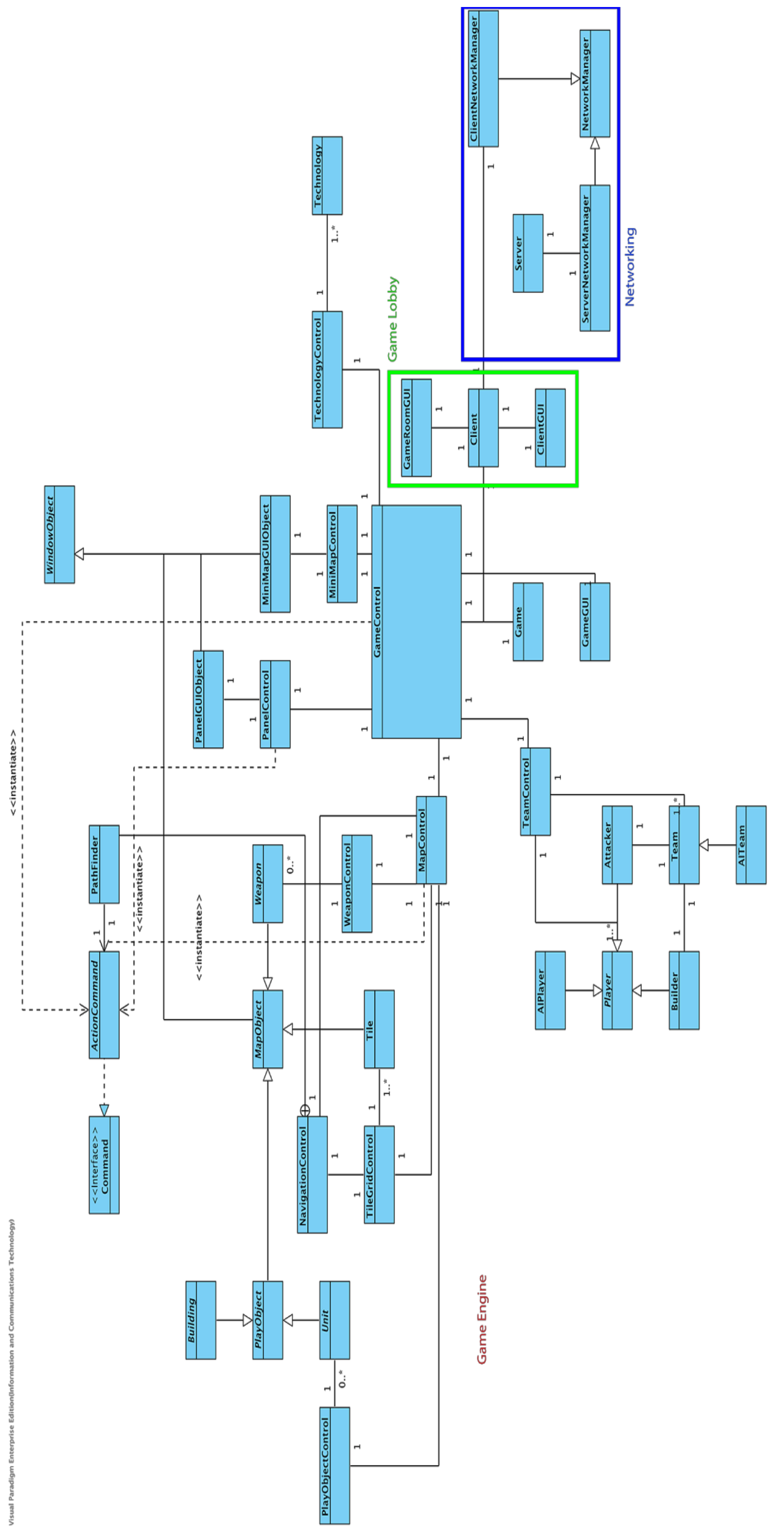
Fiedler, G. (2010). *What every programmer needs to know about game networking.* Available: http://gafferongames.com/networking-for-game-programmers/what-every-programmer-needs-to-know-about-game-networking/. Last accessed 22nd Mar 2015.

Lester, P. (2005). *A* Pathfinding for Beginners.* Available: http://www.policyalmanac.org/games/aStarTutorial.htm. Last accessed 22nd Mar 2015.

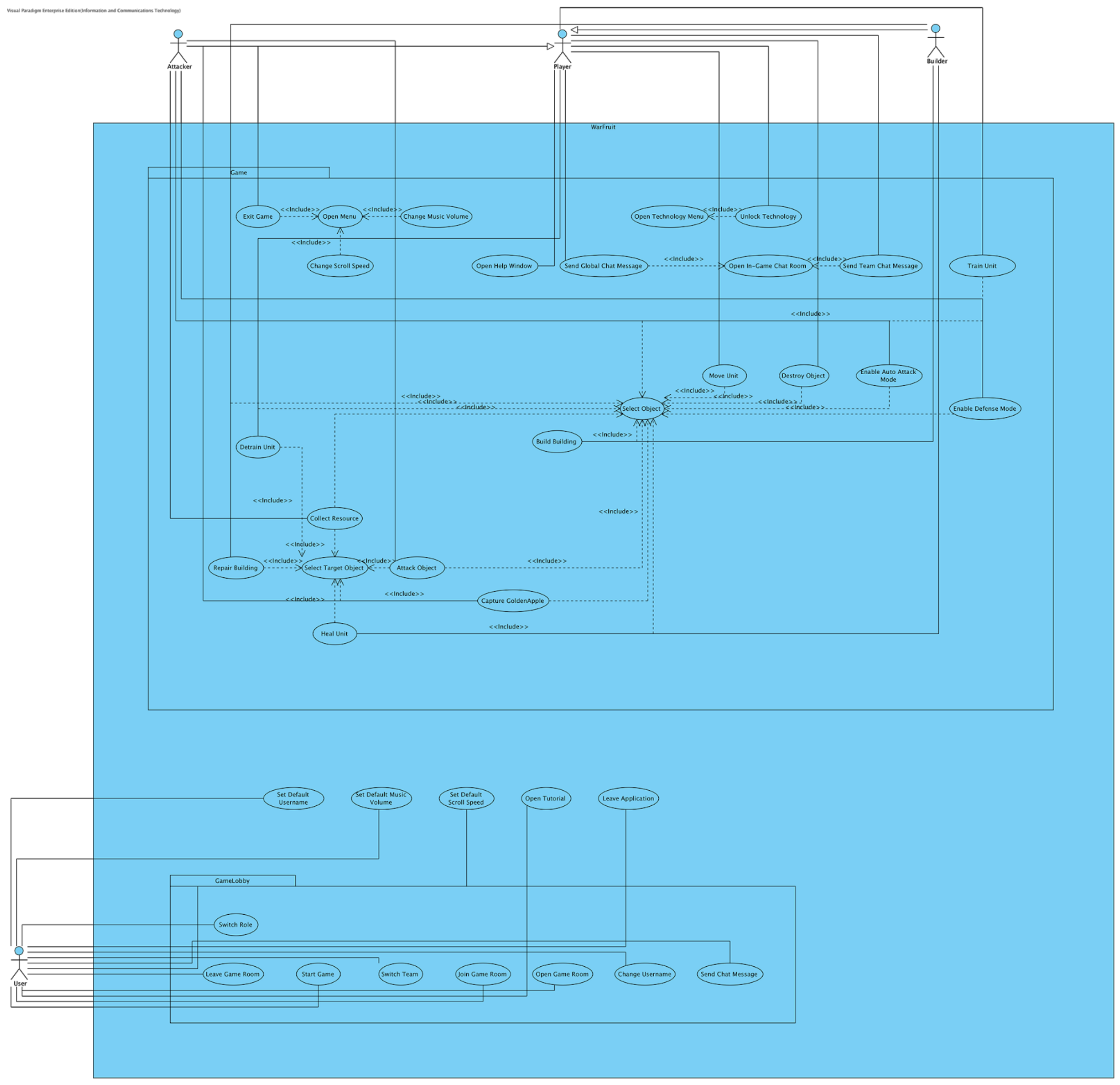
Terrano, M & Bettner, P. (2001). *1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond.* Available: http://www.gamasutra.com/view/feature/3094/1500_archers_on_a_288_network_.php. Last accessed 22nd Mar 2015.

APPENDIX

Figure 1 - High level class diagram.

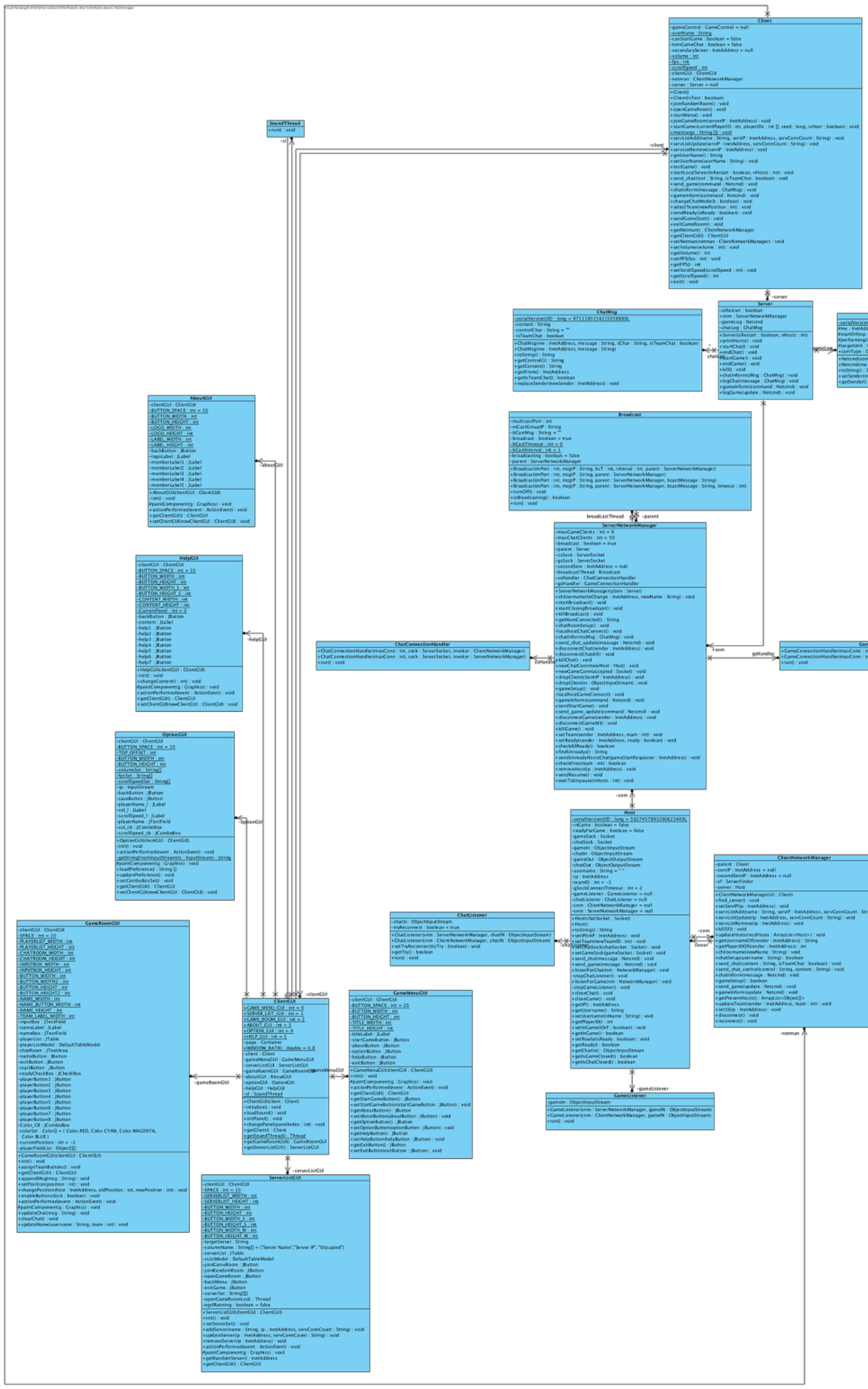


Appendix I - Refined Use Case Diagram

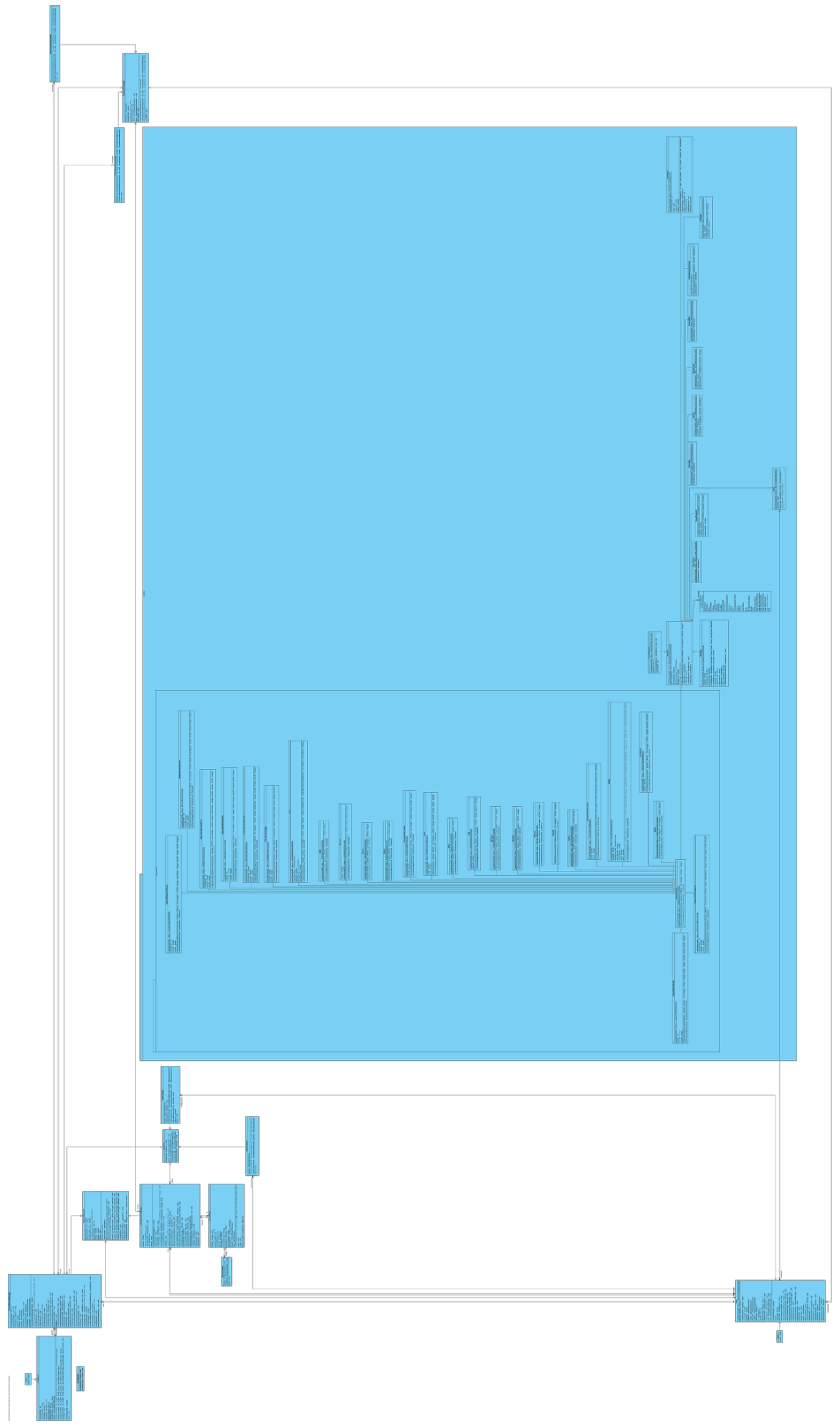


Appendix II - Refined Class Diagrams

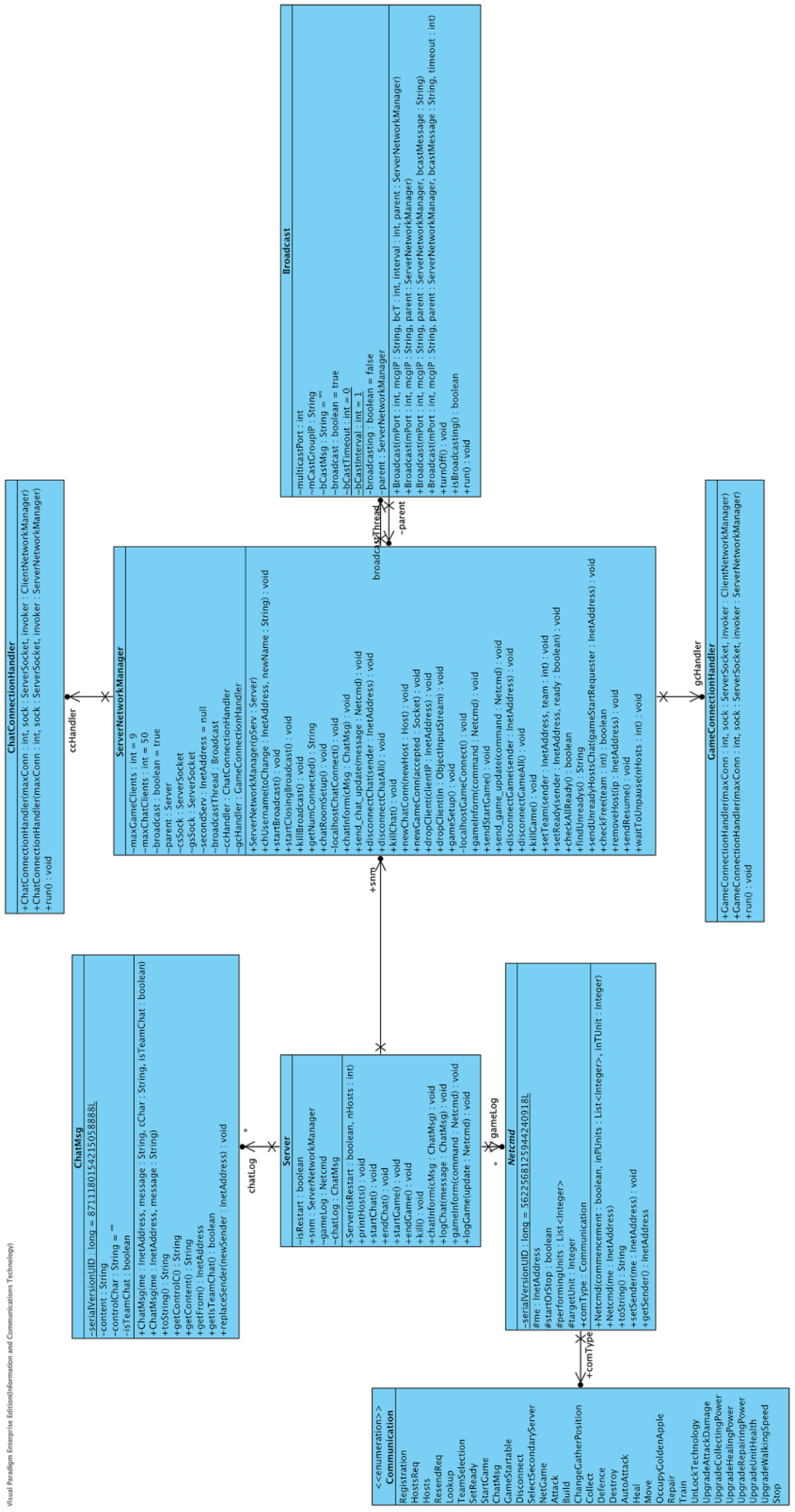
src.client



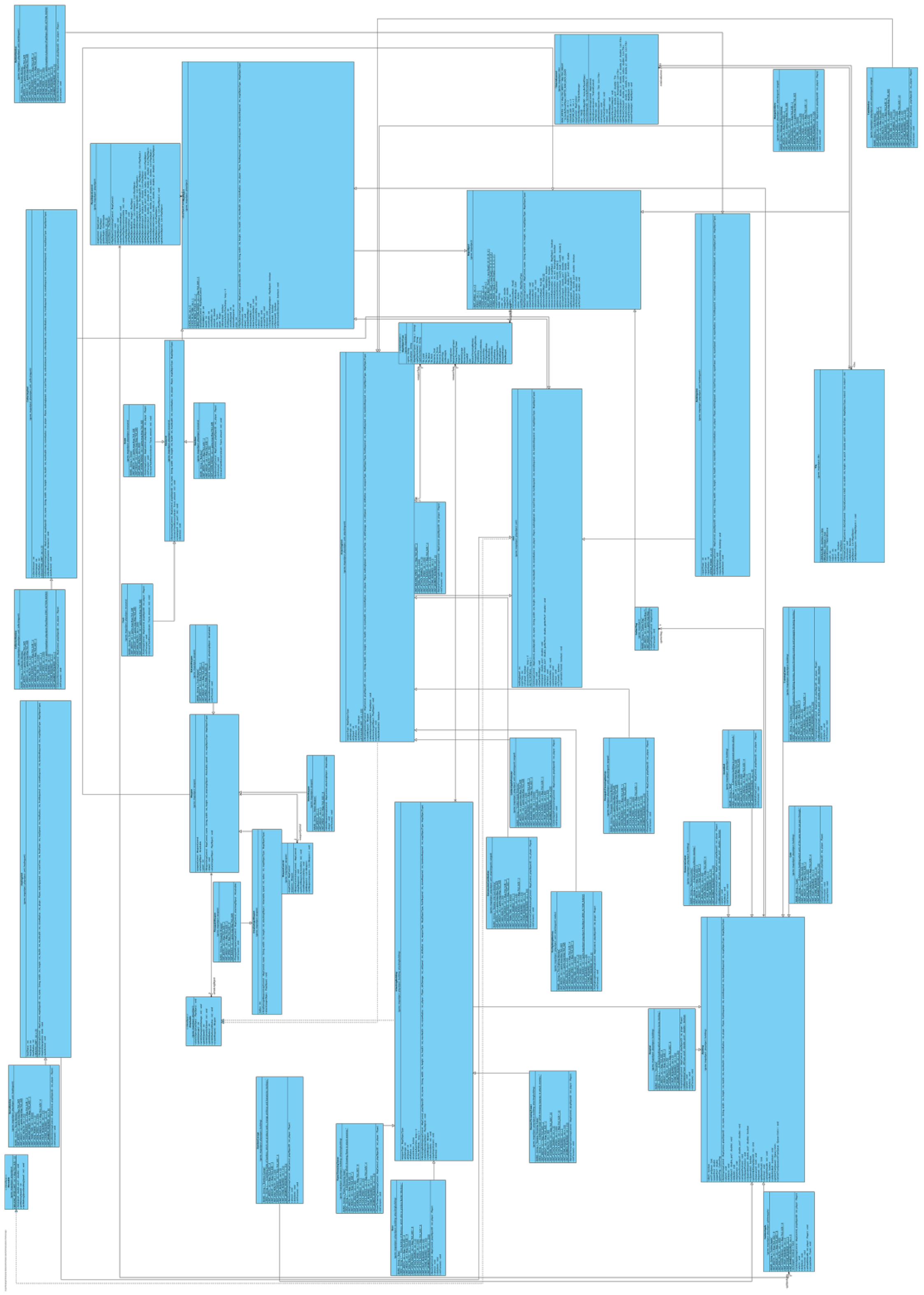
src.networkmanager



src.server


Visual Paradigm Enterprise Edition(Information and Communications Technology)
ChatConnectionHandler
+ChatConnectionHandler(maxConn : int, sock : ServerSocket, invoker : ClientNetworkManager)
+ChatConnectionHandler(maxConn : int, sock : ServerSocket, invoker : ServerNetworkManager)
+run() : void
ccHandler
ChatMsg
~serialVersionUID : long = 8711180154215058888L
~content : String
~controlChar : String = ""
~isTeamChat : boolean
+ChatMsg(me : InetAddress, message : String, cChar : String, isTeamChat : boolean)
+ChatMsg(me : InetAddress, message : String)
+toString() : String
+getControlC() : String
+getContent() : String
+getFrom() : InetAddress
+getIsTeamChat() : boolean
+replaceSender(newSender : InetAddress) : void
chatLog
*
ServerNetworkManager
~maxGameClients : int = 9
~maxChatClients : int = 50
~broadcast : boolean = true
~parent : Server
~csSock : ServerSocket
~gsSock : ServerSocket
~secondServ : InetAddress = null
~broadcastThread : Broadcast
~ccHandler : ChatConnectionHandler
~gcHandler : GameConnectionHandler
+ServerNetworkManager(pServ : Server)
+chUsername(toChange : InetAddress, newName : String) : void
+startBroadcast() : void
+startClosingBroadcast() : void
+killBroadcast() : void
+getNumConnected() : String
+chatRoomSetup() : void
~localhostChatConnect() : void
+chatInform(cMsg : ChatMsg) : void
+send_chat_update(message : Netcmd) : void
+disconnectChat(sender : InetAddress) : void
+disconnectChatAll() : void
+killChat() : void
+newChatConn(newHost : Host) : void
+newGameConn(accepted : Socket) : void
+dropClient(clientIP : InetAddress) : void
+dropClient(in : ObjectInputStream) : void
+gameSetup() : void
~localhostGameConnect() : void
+gameInform(command : Netcmd) : void
+sendStartGame() : void
+send_game_update(command : Netcmd) : void
+disconnectGame(sender : InetAddress) : void
+disconnectGameAll() : void
+killGame() : void
+setTeam(sender : InetAddress, team : int) : void
+setReady(sender : InetAddress, ready : boolean) : void
+checkAllReady() : boolean
+findUnreadys() : String
+sendUnreadyHostsChat(gameStartRequester : InetAddress) : void
+checkFree(team : int) : boolean
+removeHost(ip : InetAddress) : void
+sendResume() : void
+waitToUnpause(nHosts : int) : void
Server
~isRestart : boolean
+snm : ServerNetworkManager
~gameLog : Netcmd
~chatLog : ChatMsg
+Server(isRestart : boolean, nHosts : int)
+printHosts() : void
+startChat() : void
+endChat() : void
+startGame() : void
+endGame() : void
+kill() : void
+chatInform(cMsg : ChatMsg) : void
+logChat(message : ChatMsg) : void
+gameInform(command : Netcmd) : void
+logGame(update : Netcmd) : void
+snm
*
gameLog
Broadcast
~multicastPort : int
~mCastGroupIP : String
~bCastMsg : String = ""
~broadcast : boolean = true
~bCastTimeout : int = 0
~bCastInterval : int = 1
~broadcasting : boolean = false
~parent : ServerNetworkManager
+Broadcast(mPort : int, mcgIP : String, bcT : int, interval : int, parent : ServerNetworkManager)
+Broadcast(mPort : int, mcgIP : String, parent : ServerNetworkManager)
+Broadcast(mPort : int, mcgIP : String, parent : ServerNetworkManager, bcastMessage : String)
+Broadcast(mPort : int, mcgIP : String, parent : ServerNetworkManager, bcastMessage : String, timeout : int)
+turnOff() : void
+isBroadcasting() : boolean
+run() : void
broadcastThread
~parent
<<enumeration>>
Communication
Registration
HostsReq
Hosts
ResendReq
Lookup
TeamSelection
SetReady
StartGame
ChatMsg
GameStartable
Disconnect
SelectSecondaryServer
NetGame
Attack
Build
ChangeGatherPosition
Collect
Defence
Destroy
AutoAttack
Heal
Move
OccupyGoldenApple
Repair
Train
UnLockTechnology
UpgradeAttackDamage
UpgradeCollectingPower
UpgradeHealingPower
UpgradeRepairingPower
UpgradeUnitHealth
UpgradeWalkingSpeed
Stop
+comType
Netcmd
~serialVersionUID : long = 5622568125944240918L
#me : InetAddress
#startOrStop : boolean
#performingUnits : List<Integer>
#targetUnit : Integer
+comType : Communication
+Netcmd(commencement : boolean, inPUnits : List<Integer>, inTUnit : Integer)
+Netcmd(me : InetAddress)
+toString() : String
+setSender(me : InetAddress) : void
+getSender() : InetAddress
gcHandler
GameConnectionHandler
+GameConnectionHandler(maxConn : int, sock : ServerSocket, invoker : ClientNetworkManager)
+GameConnectionHandler(maxConn : int, sock : ServerSocket, invoker : ServerNetworkManager)
+run() : void

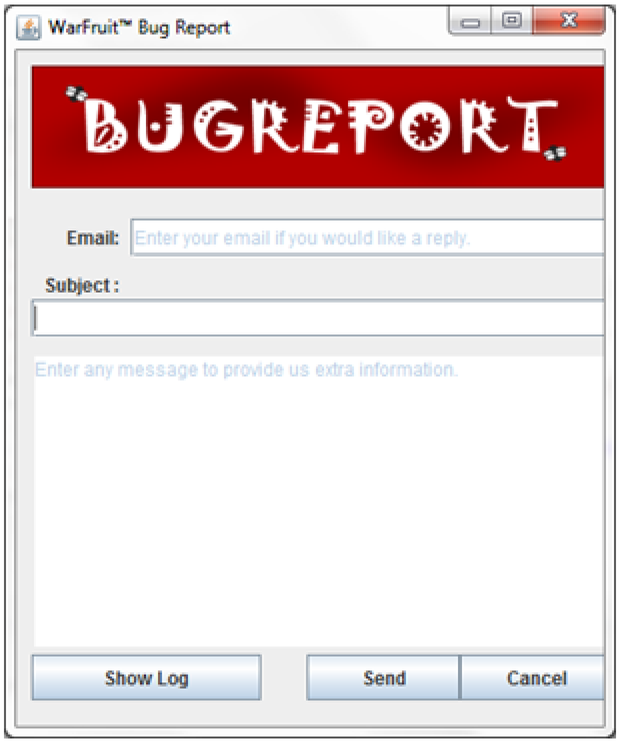
src.game



Appendix III - Maintainability

a. Bug Frame

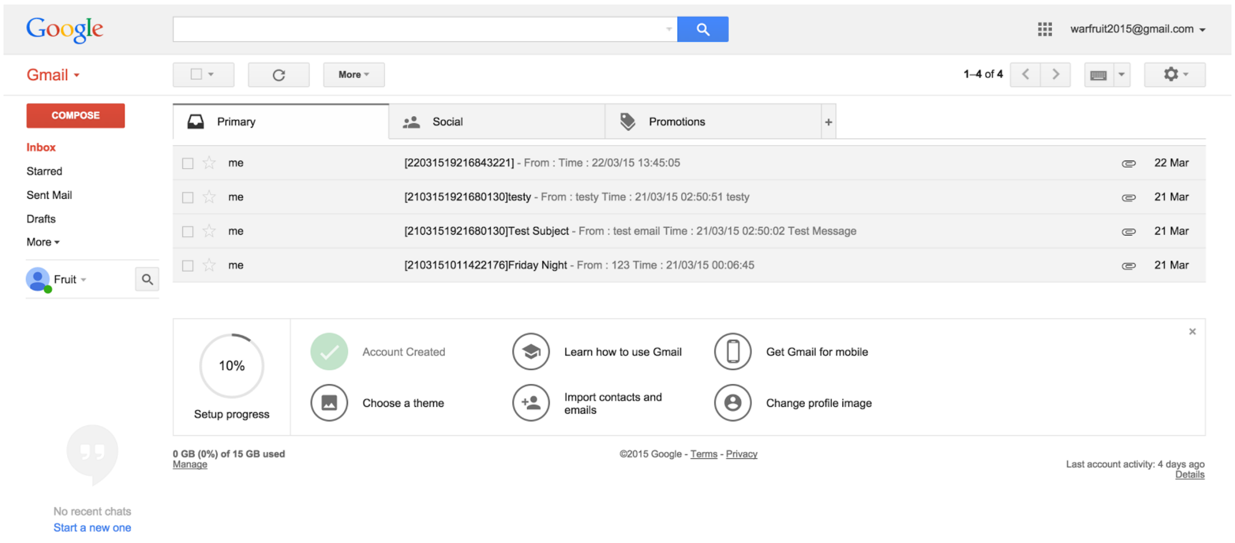
In the rare event that a user encounters an error in the code, the ExceptionCatcher class will run. This will display a bug report window as shown on the right. In this window, the user can give their email address, a subject and any supplementary information necessary to reproduce the bug. The exception can also be viewed.


WarFruit™ Bug Report
BUGREPORT
Email:
Enter your email if you would like a reply.
Subject :
Enter any message to provide us extra information.
Show Log
Send
Cancel

When the send button is pressed, an email containing the user entered data, a copy of the exception report and the user's email if given are sent to our team's email address, warfruit2015@gmail.com. This allows for the team to continue working on the game, fixing any bugs that are found if the game were to be published.

b. Log Viewing

Our company has an official email which keeps the bug report and log file from user. This allows us to solve the bug as soon as possible. Meanwhile, each report has a unique report ID and reported date and time was marked in the email, which allows



me	[22031519216843221] - From : Time : 22/03/15 13:45:05	22 Mar
me	[2103151921680130]testy - From : testy Time : 21/03/15 02:50:51 testy	21 Mar
me	[2103151921680130]Test Subject - From : test email Time : 21/03/15 02:50:02 Test Message	21 Mar
me	[2103151011422176]Friday Night - From : 123 Time : 21/03/15 00:06:45	21 Mar

0 GB (0%) of 15 GB used
Manage

©2015 Google - Terms - Privacy

Last account activity: 4 days ago
Details

developer easy to follow.

Appendix IV - Presentation Materials

a. Game Trailer

https://www.youtube.com/watch?v=9Xag16CRJIw

b. Prezis Slide

http://prezi.com/4fgz4feeusk4/?utm_campaign=share&utm_medium=copy&rc=ex0share