5-6-2001

Babylon 5 Collectable Game Network Interface

Jeremy Tillman

2001 2001-5.0

Jeremy Tillman

Honors Thesis/Senior Project

Babylon 5 Collectable Card Game Network Interface

Project Advisor: Brad Vinz

Jeremy Tillman, CS499 & Honors Senior Project

ABSTRACT

Babylon 5 Collectable Card Game Network Interface

Babylon 5 is a space station located in neutral territory in the year 2258. It is a United Nations of sorts for the different alien races in the universe (Narn, Centauri, Human, Minbari, and the League of Non-Aligned Races). Babylon 5 is a place for Ambassadors from the different races to come to work out their differences peacefully.

A collectable card game is a game where there is a collection of many cards and a player chooses a few out of those cards to go into their deck. Each player then uses his or her customized deck to play a game. Two popular examples of this are Magic: The Gathering, and Pokemon. Usually these games are played at a table with all of the players present. A network interface will allow people separated by long distances to be able to play together.

In the B5CCG, each player starts with an Ambassador character and 4 influences in the universe. Each player uses characters, fleets, conflicts, and other things to try to increase their influence using the abilities of Diplomacy, Intrigue, Telepathy, and Military. The first player to reach 20 influences wins the game.

The network interface will first need a database to keep track of all the cards that a player has to choose from. A player needs to be able to select a number of cards from the database to go into their deck, so the deck can be saved and loaded upon playing. The database needs to be expandable so that new cards can be added to it at any time.

The game portion of the interface needs to provide a play space (like a table or a desktop) in which one can put their cards during play. Each player can toggle between their own game area and that of any other player. Also available in the play area is a deck that can be drawn from or shuffled, a hand of cards which only the person playing that deck can see, a stats area which shows the tensions between the races and the amount of influence that each race has, and several counters that a player can use to his discretion.

Cards in the play area have the ability to rotate, to flip over, to stack upon one another, to discard, and to move to another player's play area. The player must be able to create any number of counters, name the different counters, and to control when the counter goes up or down (it should look a little like a rod with beads that can move back and forth on it). There are also small marks that need to be added to cards as the need arises. Finally, there needs to be some method of communication between the players (such as ICQ). Not only can the players talk over this, but when any player makes a move, the details of the move will automatically go out to the other players so that everyone can "see" what everyone else is doing.

The network interface will be the backbone of the project. The network interface will have to interact with the play area, database components, and all user interactivity. The chat room, present on the main play area, will be the main networking component. It will allow each player to communicate in the game. Several counters and the login page will also be heavily dependant on the networking interface.

Introduction

This goal of this project is to provide a game interface for multiple players to play over the Internet at the same time. Our team is responsible for building this game from the ground up including, the system requirements, design documentation, user manual and the final product. In order for this to happen there must be a network interface powering the game. My main task was to provide this network interface while overseeing the rest of the project's requirements and contributing to the design and coding. The problem is to find the best way to construct a networking API (Application Programmable Interface) that could be easily plugged into the main game architecture. The SRS (System Requirements Specification) portion of this thesis lists the methods and requirements for this project. The SDD (System Design Document) section incorporates the way we went about trying to accomplish our goals. The later part includes the results and findings.

Table of Contents

Purpose

This document describes the requirements of the Babylon 5 Collectable Card Game Network Interface to the customer.

Scope

The scope of this document is confined to the software requirements for the Babylon 5 Collectable Card Game Network Interface. This document provides a detailed definition of the requirements for the play area interface, deck, cards, marks, tracker, and chat.

Definitions, Acronyms, Abbreviations

The following is a list of commonly used definitions used throughout this document:

Definitions

- Destiny – a small mark used in game play that can be put on a card or taken off; is used as a reward; other cards provide benefits for those with destiny marks
- Doom – another mark used as a punishment; other cards provide consequences for those with doom marks
- Vorlon – another mark used to show Vorlon influences in your faction; certain Vorlon cards may be played by those who have Vorlon marks
- Shadow – another mark used to show Shadow influences in your faction; certain Shadow cards may be played by those who have Shadow marks
- Conspiracy – another mark used to show the character is a member of a conspiracy; other cards concerning conspiracies may be played by those who have conspiracy marks
- Strife – another mark used to show extra aggression in attacking; a character with a strife mark attacks at a higher rating than usual
- Turn Order – the order in which players take actions; this order changes every round depending on the amount of player's influence; those with the least amount of influence go first
- Bead Counter – a counter used in the game; kind of like a string with beads on it so the beads can be moved back and forth on the string to show how much of a fixed resource has been used; ex. Influence counter
- Number Counter – a counter used in the game with the ability to go up and down to show how much of something someone has, ex. How much a conflict is winning by so far
- Influence – a measure of your faction's influence in the universe; the more influence you have, the more resources you can bring into play for your side; the first player to get to 20 influence and have more influence than any other player wins the game

1.2.2. Acronyms

- Rotate 90 degrees – this is where you rotate a card 90 degrees to the left to show that the card has taken an action this round and cannot take another one until the next round
- Flip Over – this is where you flip over a card to show that it is dead or neutralized
- Stack – putting on card on top of another card to either show that the card on the bottom is enhancing the card on the top, or to conserve space in the playing area
- Discard – move a card to the discard pile to show that it is no longer in play
- Move – to drag a card or another item to a different area in the play area

Overview

This Babylon 5 networked card game will allow multiple users to interactively compete against players over the Internet. They will be able to chat back and forth while playing a game they all enjoy. This project is in the prototype stage.

Overall Description

Product Perspective

The Babylon 5 Card Game is the future of this company. We are striving to be the number one provider of networked fantasy card games. This product will allow lovers of the Babylon 5 game to have a place they can go to play all from the convince of their own home. They will no longer have to travel long distances to play with one another; they can just logon and begin to play against others, near or far. Also, a player will not have to spend all of their money purchasing three copies of every card in order to play with the cards they want.

Product Architecture

Babylon 5's main component will be the play area. The play area will be a graphically rich user interface displaying the deck, cards, marks, tracker, and the chat area. The player will make all of their major moves from the play area. They also will be able to toggle between all the other players' screens to see what cards are on their play area. Another major component will be the deck builder, which allows the user to choose cards from the game database to put in their deck for a particular game.

Product Functionality/Features

Registration/Login – users will be required to give certain information on their first attempt to play the game. On subsequent attempts they will be prompted for there login info.

Play area – facilitates all major moves such as rotate, flip over, stack, discard, and move.

Deck Builder – allows the user to choose which cards they want to play with

Deck Manager – this will allow the user to update their deck during the course of play or create a new deck after the game has ended.

Toggler – allows the user to toggle between the other users' screens.

Counter wizard – allows the user to create as many game counters needed in order to keep track of things such as influence.

Marks – a bin to hold marks like Destiny, Doom, Vorlon, Shadow, Conspiracy, Strife that you can drag and drop onto or off of a card.

Tracker – displays every user's current number of influence.

Chat – allows the users to send messages back and forth anytime during the game.

Help – gives the user a summary of the rules.

User Characteristics

This game will be created specifically for users with prior knowledge of the game. Users will need to have a general knowledge of the game in order to play this game effectively. Each game feature is a specialized feature that users will already need to understand to effectively use.

Constraints

The Babylon 5 game that is currently played in the real world will be much more extensive than the version we are implementing in this project. For our project, the deck size will be limited as well as the number of users.

Assumptions and Dependencies

As stated earlier, each user will be expected to have knowledge of how the game is played and how it functions before playing our version.

Functional Requirements

User Registration/Login

Users will be required to fill out a registration form upon their first game interaction. When the same user comes back to play the game again they will be prompted for their username and password. After logging in, any saved decks will appear on the deck builder screen. At the point of login users will have a set time for other players to join the game they have started. When the time expires everyone logged in will join the game. No users will be able to join a game after it has started.

Play Area

The play area should have a space to put your cards in play. It also has a "hand" area and a discard pile. Cards can be dragged to any of these three areas. Cards may be dragged on and off of one another (to stack).

Deck

There is a deck (if a card is dragged here, it is placed face-down on top of the deck.) If the deck is double clicked, one card is put into the "hand" area. If the deck is right clicked, it shows a pull down menu with the features "Draw Card" (as if the deck were double-clicked) and "Shuffle".

Cards

If any card is double clicked, it rotates 90 degrees to the left as shown. If it is right clicked, it has the options:

- Rotate 90 degrees
- Flip Over
- Move
- Put on Top of Deck
- Put on Bottom of Deck
- If the "move" option is chosen, another pull down menu offers where to move it to:
    - Move
    - Player 1 Area
    - Player 2 Area
    - Player 3 Area

Marks

There is a bin with all the different marks (Destiny, Doom, Vorlon, Shadow, Conspiracy, Strife). These marks can be dragged onto and off of a card.

Tracker

The tracker on the bottom of the screen keeps track of influence level, turn order, and tensions between the races.

Chat

On the right is the chat between the players. Players can talk on the chat; additionally, there will be an automatic notification of moves made. The moves that will be reported are:

- A card is brought into play (put in play area)
- A card is rotated
- Someone gains influence
- Tensions change

Menu Bar

The menu bar will have six main menus.

- File
    - Start New Game
    - Order
    - Quit Game
- Toggle
    - Your View
    - Player 1 View
    - Player 2 View
    - Player 3 View
- Counters
    - Bead Counter
    - Number Counter
- Turn Order
    - Change Turn
- Deck Manager
    Edit a saved deck
    Create a new deck
- Help
    Start a game
    Build a deck
    Help on interface
    Summery of game rules
    About the creators

Menu File

The user will click on the start game button to officially start the game. No other players can join the game once it has started. It will also contain the order that the players need to move in and a quit game option. All the players will have to agree to quit the game for the game to end.

Menu Toggle

This will toggle you between your screen and any other player's screen. You can see everything on all the other players' screens except for their hand, which will be blacked out. You can't, however, move anything when you are on another player's view.

Menu Counters

When creating a new counter, you get a choice between a bead counter or a number counter. (See screen shot, the "influence" counter is the bead counter and the "conflict" counter is the number counter.) You can make as many as you want and put them wherever you want on the screen and you can close them whenever you want. When you make one, a box pops up for you to name the counter. For a bead counter, you can right-click on a bead and you get a pull-down menu of colors (red, blue, green) so you can change the color of a single bead. Each player automatically gets a bead "influence" counter that starts at four (color red) and can't be closed. This one is monitored for the influence counter (only the red ones count) for the status bar at the bottom of the screen.

Menu Turn Order

This will give you a "Change Turn Order" option which provides the ability to change the order of the turns. The status bar showing turn order is updated for everybody's screen. Any player may do this.

Menu Deck Manager

Another menu option on the main interface (play area) will be the deck manager. Here a user can change their deck in the middle of the game, or create a new deck of cards to be used in future games.

Menu Help

The help section will contain basic game rules for players that might need a little help. It will also include instructions on how to start a game, build a deck, and about the makers of the program.

External Interface Requirements

This program will have to interact with an Access database. This database will be designed to hold:

- User Information
  - Full Name
  - Username
  - Password
  - Saved Decks
- Game Card Information
  - Name of the card
  - Type of the card
  - Abilities
  - Marks
  - Cost
  - Game text
- Game Data

Internal Interface Requirements

A networking interface will be needed in order for the game to function as a multiplayer game. Game data will need to be shared along with other essential data. The areas the network interface will have to provide support for are

- User Registration/Login – connect all of the users together and notify the server upon entry.
- Play Area – after each user moves their action needs to be recorded on the server.
- Cards – card information will need to be shared across the network.
- Marks/Counters – server counters will need to be reflected across each players screen to inform all the players of each other's influences and tensions.
- Chat – a chat area needs to be established so that players can communicate with each other.
- Toggle – each player needs to be able to view every other player's screen at any time during the game.

Each of these network requests need to be bundled in one network interface package so that integration with the main game programming will be simplified. The networking package needs to work in such a way that it is scalable to allow more than four players to play in the future.

Internal Data Requirements

We will be responsible for keeping track of all the counters across all the different user's play areas. A timer will be set for each player at 5 minutes for each move. If their timer expirers they will forfeit the game.

Design and Implementation Constraints

The main obstacle that we will have to overcome in this project will be networking issues. No one on the team is an expert on what we need to do to make a multi-user game. We plan on using the TCP/IP protocols associated with Java to broadcast game information to all the users. If this method fails we will store all of the game activity in a database, which will be perpetually accessed by each user.

Other Requirements

Non-Functional Requirements

Safety Requirements

None

Security and Privacy Requirements

For a user to access a saved deck their username and password will be required.

Environmental Requirements

None

Computer Resource Requirements

Computer Hardware Requirements

At least Pentium Pro along with 32mb of ram

Computer Software Requirements

Internet Explorer 4 or above or Netscape Communication 4.0+

Computer Communication Requirements

Internet connection – preferably 56k or better
If a users connection dies or they have been inactive for over 5 minutes that users will be automatically forfeited from the game.

Software Quality Factors

None

Packaging Requirements

We will package this project on a CD. After successful installation a user should be able to start or join a game with another user.

Precedence and Criticality of Requirements

The precedence is the same as listed in the features and functionality in section 2.3 with the first feature being the most important to the last being the least.

Qualification Provisions

This section defines a set of qualification methods. For each requirement in Section 3, specify the methods to be used to ensure that the requirement has been met. A table may be used to present this information, or each requirement may be annotated with the method(s) to be used. Qualification methods may include: Demonstration, Test, Analysis, Inspection, Special.

Requirement	Qualification
Registration	Demonstration, Test
Deck Builder	Demonstration, Test
Deck	Demonstration, Test
Play Area	Demonstration, Test
Toggler	Demonstration, Test
Counter Wizard	Demonstration, Test
Marks	Demonstration, Test
Tracker	Demonstration, Test
Chat	Demonstration, Test
Help	Demonstration, Test

Requirements Traceability

Upward Traceability

Traceability will be added in the next revision.

Downward Traceability

Traceability will be added in the next revision.

Table of Contents

Introduction

1.1 Purpose

This document gives design descriptions of the Babylon 5 Collectable Card Game Network Interface to the development team.

1.2 Scope

The scope of this document includes the definition of program modules, data entities and interface definitions for Babylon 5 Collectable Card Game Network Interface.

1.3 Definitions and Acronyms

- Babylon 5 Collectable Card Game Network Interface can be abbreviated to **B5CCGNI** in the text.
- Program module means a separate package containing related objects in Java programming language.
- **RMI** – Remote Method Invocation. A java utility package that allows computers to interact over the Internet.

3. Decomposition Description

3.1 Module Decomposition

3.1.1 GUI Module Description

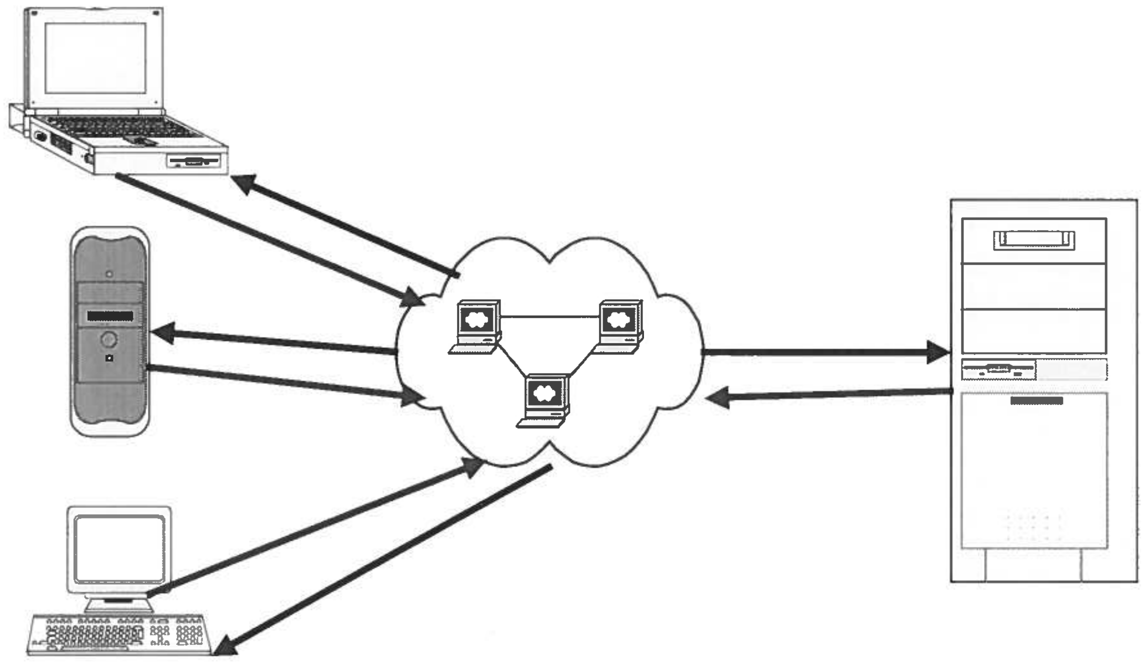
This module contains classes that are responsible for building Graphic User Interface, accepting and processing user inputs, and representing system events. The classes are:

- abstract class Mark, for representing the marks (Destiny, Vorlon) used during the game;
- class Counter that is used as a generic counter for events;
- class Card to represent the cards in the game.

3.1.2 Networking Module Description

Networking module is responsible for client-server interactions of the game. It includes interfaces and classes for updating the information on players' screens, such as chat area in the main playing view. The API of Networking module is used by GUI module.

Figure 1. Client Server Network Setup



The backbone of the networking module will be the java RMI package. RMI is the tool used to power the client/server layout, as depicted above. Each of the networking requirements listed in the SRS will be met as detailed below.

- *User Registration/Login* – when a user starts the game they will be prompted with a login screen. Their username and password will be sent through the networking interface for validation. Upon validation the RMI based connection function will be called and the server will add the player to its action list.

- *Play Area* – there will be several methods set aside in the client and server RMI based code that will perform any game actions that involve network activity.

- *Cards* – card information will be stored on the server and a retrieve card information method will be called from the client RMI based code to take advantage of this functionality.

- *Marks/Counters* – there will be several textboxes on each player's screen designated for the other players. These textboxes will automatically be updated across the network when the originating player changes the value of the textboxes throughout the game.

- *Chat* – a textbox for the player to type what they want to send over the chat room as well as a send button and a main chat room textbox will be on the main game screen. The sender will be able to send a message to everyone or an individual player. The chat message will be sent from the client to the server and then the server will decide if it needs to go to a specific player or everyone. The server will then send the message to the appropriate recipients including the player that originally sent the message.

- *Toggle* – the status of each player's screen will be kept in a status array on the server. When another player makes a toggle request to see another player's screen the server will grab the status array of the player's screen to be displayed and send it to the requesting client. In the main game code there should be a switch view function that uses the status array received from the server to manipulate that player's screen to be identical to the other player.

3.1.3 Database Connectivity Module Description

Database Connectivity module is used to connect to database, to read the card names, and to load or save the current game. It will give its interfaces to GUI module to let its utilization.

3.1.4 Utility Classes Module Description

Utility Classes module contains auxiliary entities used by all other modules in the project. These classes are:

- class Card, encapsulating the current state of any card in the game;
- class Deck to keep the current deck composition of some given player.

Figure 2. Module Decomposition

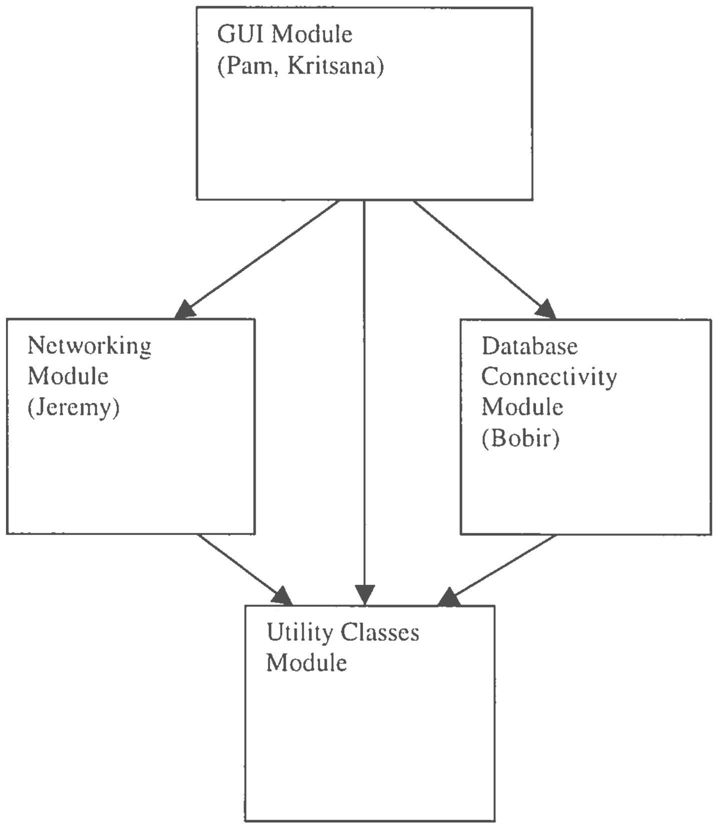


Figure 3. Package Decomposition

```
babylon
        babylon.GUI
        babylon.network
.................................................... babylon.database
        babylon.util
```

3.2 Data Decomposition

3.2.1 Database table view *Cards*

This table contains essential information about every possible card variant that can be used in the game. This table never updated during the game, but can be changed to accommodate alterations in the new types of cards. Each entity in this database table has following properties:

- cardID – the unique number to identify this particular card,
- abilities – the magic abilities of this card,
- marks – the types of marks that can be assigned to the card,
- name – the mnemonic name of the card,
- type – the game type of the card,
- cost – cost or weight of the card,
- text – textual description, used to give additional information to the player.

3.2.2 Database table view *Users*

In this particular table, the program will keep the track of users, who signed as players. Normally, to play game, one must first register in the system, and get his password.
The fields are:

- userID – some unique number for identification in the system,
- firstName
- lastName
- password – some string representing the password for entering a new game. Can be encrypted (hashed) or be kept as is,
- AboutMe – text containing personal info, interests, hobbies and etc.

3.2.3 Database table view *Decks*

One player can save more than one game. For this reason system must save information about each game that is saved by the particular user in the Decks table of database. Structurally this table is quite simple, main computation is performed on the client side. Decks has following columns:

- userID – the foreign key in the table,
- deckID – the unique key representing particular game,
- deckName – the name of the saved game by which players can identify it,
- cardString – field that holds the current names of the cards.

4. Dependency Description

4.1 Intermodule Dependencies

All modules are dependent on the Utility Classes module. The GUI module uses Networking and Database Connectivity module as well. The dependency is exhibited in using the classes and interfaces that are inside other Java packages.

5. Interface Description

5.1 Module Interface

5.1.1 Networking Module Interface

Package **`babylon.network:`**
Class Network, public functions:

- Send_Chat(strMsg as string, intTo as integer);
- Send_Tension(intTCount as integer, intPlayer as integer);
- Send_Influences(intICount as integer, intPlayer as integer);

These functions are used in communication between the server and some number of clients. The interface is given to GUI module, as a automatic object of class Network.

5.1.2 Database Connectivity Interface

Database Connectivity Interface is composed of following package:
`babylon.database`
There would be one class, which will be instantiated in the GUI part (client part) of the program and will serve as an interface to the used database.
Class Database
Interface functions:

```
getConnection(String url);
getConnection(String url, String userID, String password);
```

The above functions will override the functions of default JDK database interface and will connect to the database.

```
ExecuteQuery(String sqlString);
```

This command will execute an SQL query and will return the results to GUI part.

5.1.3 Utility Classes Module Interface

Class Card and its public functions is one of the classes in this package (**`babylon.util`**).

```
public class Card extends Object {

        public Card(){}; // default constructor
        public Card(int cardIDnew, int abilitiesNew, int marksNew,
                 String nameNew, String typeNew, int costNew, String textNew) {
               carID = cardIDNew;
               ...
        } // end constructor


        // get functions
        public int getCardID() {return (cardID); }
               ...

        // set functions
        public void setCardID(int cardIDNew) {cardID = cardIDNew;}


        // member variables
```

```
	private int cardID;
	private int abilities;
	private int marks;
	private String name;
	private String type;
	private int cost;
	private String text;


} // end class Card
```

Another class is class Deck. The class deck essentially is an array of class Card objects. In implementation it is acceptable to use either the whole object or its ID number.

Results & Discussion

Developing a network interface for a multiplayer game in Java can be very difficult if the correct networking method is not chosen. There are basically three main methods to choose each with their own strengths and weaknesses.

The first method is a database bound method. Each player establishes a connection with the game database on a specified server. There are then two main routines a push and a pull. The push method records each player's moves and comments in the database. The pull method perpetually checks the database for new or updated data and displays it to the user's screen when appropriate. The strength of this approach it is easy to setup. Setting this networked game up only takes a server containing the database, and a web page where players can join the game. The disadvantages are that you do not really have a networked game, this method only emulates a real-time network game. This method will not scale well, and even with four players the game has a substantial amount of lag time.

The second method is to use sockets. This method requires the main game code to be stored on a server and for each client to connect through sockets. The server would dedicate a socket to listen for requests made to it, as well as each client. Upon game startup each player would send the server their internet address and port number. The server would then store this information and know how to connect with each player. When an action took place in the game that required network activity a request would be made to the server to perform this activity, the request would be decoded by the server, then the server would perform the action and send an encoded message back to the client to perform it. This method is how a true client/server network game should take place. It is in real-time and is only scalability issue is the processing and memory limitations of the server. The problem with this method is that it is really difficult to implement and takes a lot of advanced coding. Socket coding is not easy because every request made the server and clients has to be encoded and decoded. Also each client machine would have to be set up to have a specified socket open for listening which can cause security leaks, and is difficult to set up.

The final method is called RMI (Remote Method Invocation) this method uses the Java programming package called java.rmi. This method is similar to the socket method in functionality but no socket programming has to be done. RMI does the work automatically for you. To implement RMI you simply write the methods that will be invoked on the client machine and place them there, write the server code to handle the requests and direct them to the client, and write the code to synchronize them. Once this is finished and the correct services are started the client makes function calls to the server to handle the tasks that need to take place. This method takes the hassle out of socket programming, yet it gives you the same functionality. It is easy to use and very powerful. The only difficulty is that a batch file will need to be written to start the rmi registry and the client files each time the program needs to run.

After months of study and sampling the different methods of developing a networked java game, the RMI method turned out the be the best fit. I implement and used RMI in half of the time I had spent on socket programming. In the end RMI was more powerful, flexible, and easy to use.

Conclusion

Working with a team to build a project from the ground up was a very beneficial experience; which helped prepare me for real word programming projects. We learned how to work as a team and accomplish a common goal. Being responsible for the main part of the project has helped me to understand what it takes to be a good project lead. The research involved in developing a network interface for this project was exciting and demanding, but the yield was very successful. I found that RMI is the best tool for building interlinked multiplayer games in java. Discovering this and implementing it was the goal of this project and although the overall group project was not completed 100 percent successfully the main networking part is. The networking fundamentals that I learned in this class will be very beneficial to me in my future career.

References and Acknowledgement

First of all I would like to thank my team that worked hand and hand with me for numerous hours on the overall project. They are Bobir Ismailov, Pam Crawley, and Kristana Sujinunkul. I would like to also note that SUN the developers of Java and their reference website was a key source for research and development of this project along with the other sites listed.

http://java.sun.com

http://www.corejavabook.com

http://users.skynet.be/rmi/

Pressman, S. R., *Software Engineering—A Practitioner's Approach,* fourth edition, McGraw Hill Inc., 1997

Pflenger, L. S., *Software Engineering—Theory and Practice,* Prentice-Hall Inc., 1998