# MCSE Project Report

GO-MOKU — A WEB GAME IN JAVA

**Guifan Li**

Submitted to Committee Members:

Dr. Kai-Hsiung Chang (Chair)
Dr. W. Homer Carlisle
Dr. Alvin S. Lim

Department of Computer Science & Engineering
Auburn University
Auburn, Alabama 36849

February 12, 1998

# 1. INTRODUCTION

## 1.1 World Wide Web (WWW) and HTML

The World Wide Web is a hypertext information system on the Internet. It is based on the Client/Server model [1].

A Client-Server model for computer network system involves three components: the client, the server and the network. A client is a software program running on the end user's computer host. A server is a software application that runs on the information provider's computer host. The user sends a request for information through the client software. This request travels over the network to the server. The server interprets the request and takes some desired actions. The results of the request are sent back to the client for display on the user's host. A web client is the Web browser, such as Netscape or Internet Explorer. The Web server is a machine running the Web server software.

The World Wide Web has grown very fast since its introduction in the early 1990s. It has gained the attention of millions of people around the world. Its

applications range from individual homepage to large information delivery system.

The programming language for the World Wide Web is the HyperText Markup Language (HTML). The Web actually contains a huge amount of interconnected hypertext documents on the Internet written in HTML. A hypertext document is a document that contains hot links to other documents. Hypertext links are usually visible as highlighted words in the text, or graphics. When a hot link is clicked with a mouse, the linked document or image will be displayed [5]. The following is an example of HTML document named index.html. It implements the starting page of the game Go-Moku.

```
<HTML>
<HEAD>
  <TITLE>Guifan Li's Home Page--Go-Moku</TITLE>
</HEAD>
<BODY bgcolor="FFFFFF" TEXT = "FF0000">
<BR><BR>
<CENTER>
<BR><BR><BR>
<h2>Welcome to My Game Page!</h2>
<BR>
<a href="help.html">About the Game</a>
<BR><BR>
<A HREF="guifan.html">Play against computer</a>
<BR><BR>
<A HREF=
"http://roadrunner.eng.auburn.edu/projects/cse532/guifan/c.html">
Play with another person</a>
<BR><BR>
<A href="http://www.eng.auburn.edu/cgi-bin/mailto.pl?22722">Suggestions</A>
<BR><BR>
<A HREF="http://www.auburn.edu/~liguifa/index.html">
```

```
Go to my homepage
</A>
</CENTER>
<BR><BR>
</BODY>
</HTML>
```

However, the HTML programming capability is very limited. If a fancy page or a page with more functionality is desired, the assistance from other programming languages is necessary.

## 1.2 Java

Java is a programming language that can extend the Web capability. One of the advantages of Java is a runtime library that gives it platform independence. Java language has changed the passive nature of the Internet and the World Wide Web by allowing architecturally neutral code to be dynamically loaded and run on a heterogeneous network of machines [4].

The traditional compiler produces code that can be executed only on a specific kind of hardware. For instance, the code generated on a Sun workstation cannot be run on a PC. In contrast, the Java compiler generates architecture-independent bytecodes. The bytecodes can be executed only on a Java virtual machine (VM), which is an idealized Java processor chip usually implemented in software other than

in hardware. Java bytecode files are called class files. To execute Java bytecodes, the VM uses a class loader to fetch the bytecodes from a disk or the network.

Another advantage for many programmers is that Java has a similar syntax to C++ [4]. It is easy to learn if you are a C++ programmer. Java is supposed to be simple, object-oriented, distributed, robust, secure, architecture neutral, portable, high performance, multiple threaded, and dynamic. Many painful tasks, such as network socket, are easy to implement in Java.

Java programs that work on Web page are called applets. The following example is a Java applet to print Hello world.

```
//Hello world applet
import java.applet.*;
import java.awt.*;
public class HelloWorld extends Applet {
    public void paint(Graphics g){
        g.drawString("Hello World");
    }
}
```

HTML interfaces with Java through tags. For example:

```
<APPLET code="HelloWorld.class" width=200 height=100>
</APPLET>
```

An applet is embedded in a Web document. Applets can be integrated seamlessly with the rest of the page written in HTML. Java applets can use a modern graphical user

interface (GUI). This includes text boxes, buttons, list boxes and so on. Java applets can also trap user events like key strokes, mouse movements and the like. With the applet running on the Web, it is much easier to make a Web page to accomplish a complex task.

Java's features make it the most promising contender for being the major development tool for the Internet and the World Wide Web.

WWW has made Java a popular programming language. In turn, Java has extended the capability of WWW. The Internet game is one example of their applications.

### 1.3 Go-Moku

Computer Game is always an attractive feature in the computer world. Some games are simple and easy; others may require certain skills and intelligence.

Go-Moku is a popular game in many parts of the world. It is easy to play but requires certain level of reasoning to be a good player. The game is like a Tic-Tac-Toe game playing on the GO board.

The game board of Go-Moku has a dimension of 19 by 19. Two players are required, black and white. The player holding the black stone will take the first move. Each player will try to form five consecutive stones

horizontally, vertically, or diagonally. The first player to achieve this goal, wins the game. For example, in Figure 1, the white player wins the game.

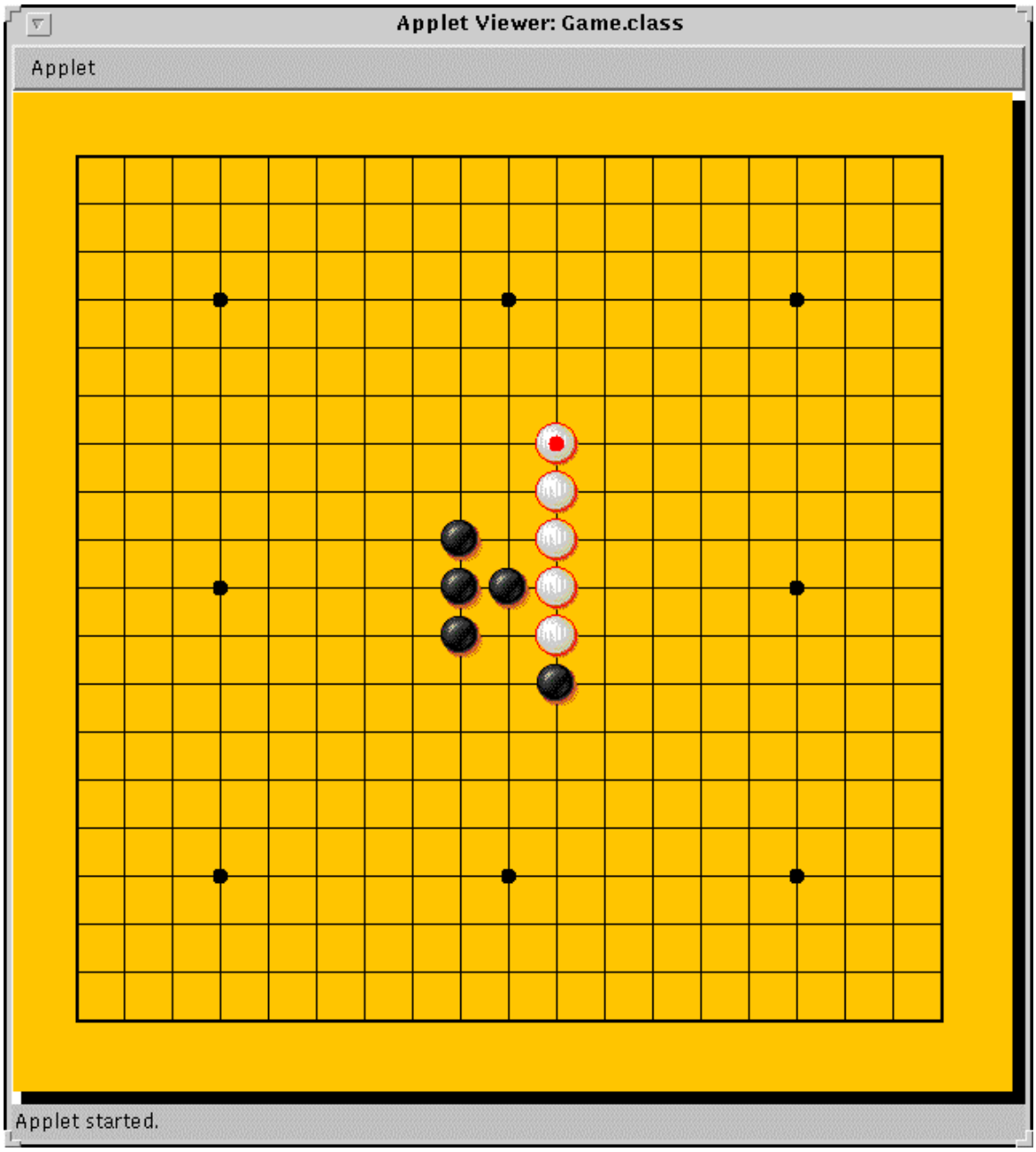


Figure 1. A winning example

This game has been played for years. There are many computer implementations of this game. However, if two old friends want to play Go-Moku in different cities or different countries, they may not be able to do so. With the development of the World Wide Web technology and Java programming language, this has been made possible.

The purpose of this project is to implement an Internet gaming environment for Go-Moku to demonstrate the capability of the World Wide Web and the power of the Java language. Two programs have been developed in this project. One provides services for two players to enjoy the game in different parts of the world. The other allows for a player to play with the computer.

# 2. DESIGN AND IMPLEMENTATION OF GO-MOKU

Since the World Wide Web is the most popular distributed information system and the Java language is the most effective language for Web programming, the game is written in Java and run on the World Wide Web. The Web location for the game is at

http://www.eng.auburn.edu/~liguifa/

## 2.1 Framework of Design

The game is composed of two major programs. One is for a human player to play with the computer. This requires the implementation of computer intelligence with heuristic functions.

The second program is to provide services to human players on different hosts. This program requires the implementation of a client-server model and network socket. The client is an applet. The server is a standalone Java application.

The starting of the game is the main page (index.html) with a choice menu (see Figure 2). It links to “human against computer” and “human against human”. “About the game” links to a help file for the game introduction.

“Suggestions” allows for users to send e-mails to the author.


Figure 2. The main game page with a choice menu

## 2.2 Human against Computer

This program provides a platform for a person to play with a computer. The interface is a Web frame page (guifan.html). A Java applet program is embedded in an HTML document (game.html) in the right window of the frame. The

control panel is another HTML document (sites.html) in the left window of the frame. See Figure 3 for the interface.

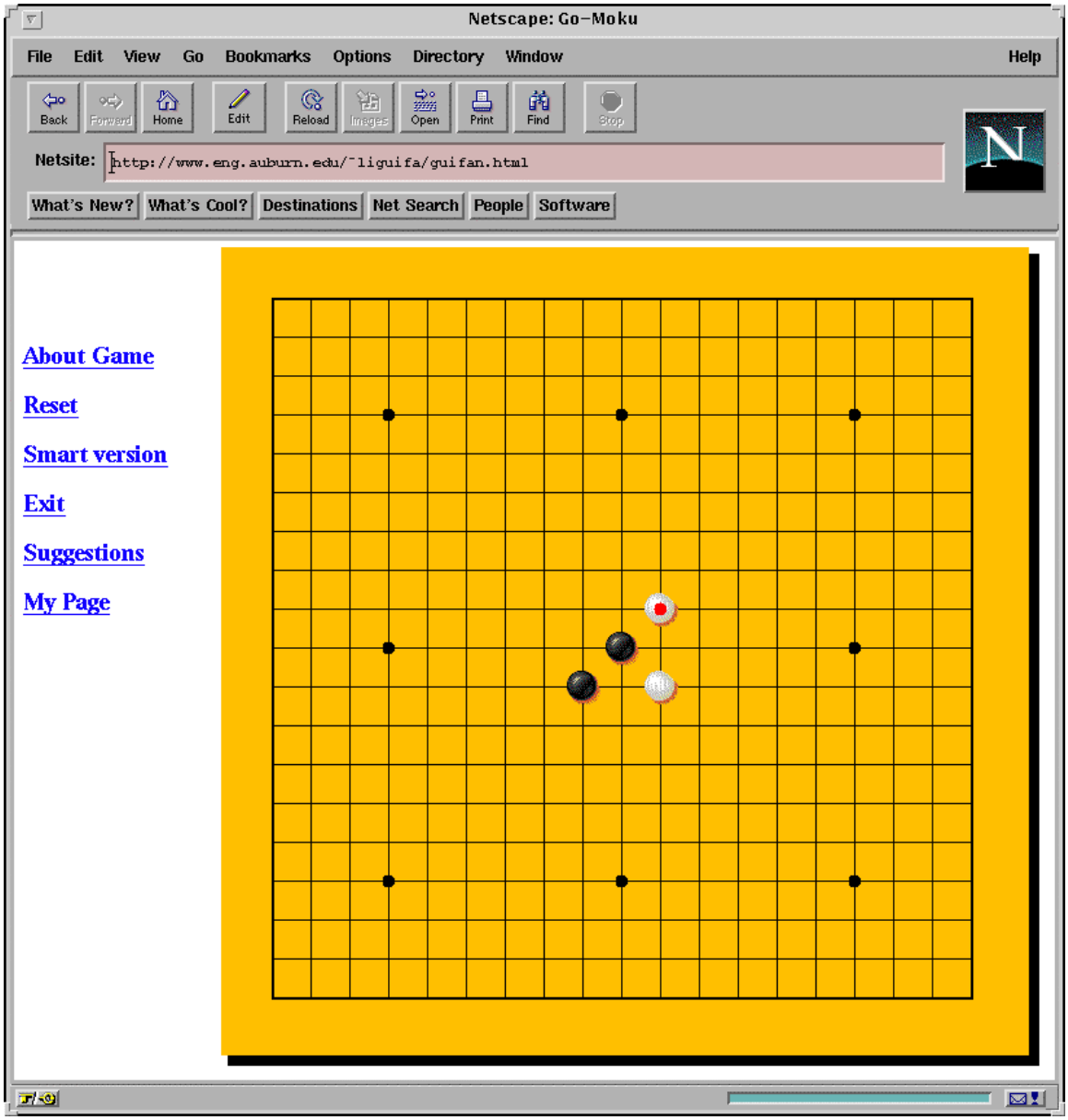


Figure 3. The GUI design for human against computer

The applet program has a few classes. One of them is responsible for drawing the game board and making stone

moves; others are for heuristic function, pattern search and payoff value calculation.

### 2.2.1. The Control Panel

In the control panel, there are six links. The first one – “About game” is a link to “help.html”. It is an introduction of the game. A user can learn what the game is all about and how to play it.

The second item is “Reset”. It allows a player to reset the game board and start a new game.

The third one “Smart version” allows users to download a more intelligent version of the same game and play on his own PC. It is much smarter and more difficult to defeat. It has 9 intelligence levels. So far, no one has beaten this version straight to the 9th level. This version has adopted breadth-first search. Its code is ten times longer, and is not efficient.

The other links are self-explanatory.

### 2.2.2. The Game Board

The game board is drawn using Java AWT methods. They are called from the paint method in the Game class which extends the Applet class. The background color of the Game applet is set to white using

```
This.setBackground(Color.white);
```

The board is a filled rectangle with orange color using

```
g.setColor(Color.orange);
g.fillRect(1,1,624,624);
```

The black lines are drawn using

```
g.drawLine(x1,y1,x2,y2);
```

The black dots are drawn using

```
g.fillOval(x,y,radius1,radius2).
```

On the board, there are 19 × 19 positions for the player to put game pieces or stones. The stones are stored as image files “wstone.gif” and “bstone.gif”. The g.drawImage() method is used to display stones on the board. The player uses mouse to make a move. When the player presses the mouse, this activates the mouseDown event to draw a black stone image at the corresponding position on the board. When the player releases the mouse, it activates the mouseUp event to call heuristic functions to make a computer move. Each time the mouse is press on the board, the computer will calculate the nearest position for this click and draw an image at the nearest position. Clicking mouse outside the board will do nothing. If you press mouse outside the board and release it inside the board, it still will not invoke the heuristic functions. There is a

mechanism to prevent a player from making two consecutive moves.

### 2.2.3 Game Tree and Search Strategy

Search is an important part in implementing the computer move. There is a need to find all possible moves on the board. The minimax search (Figure 4) is adopted for the game [5]. The search tree considers two levels. The computer will pick the largest value (see next section for value calculation) while the human player picks the smallest.

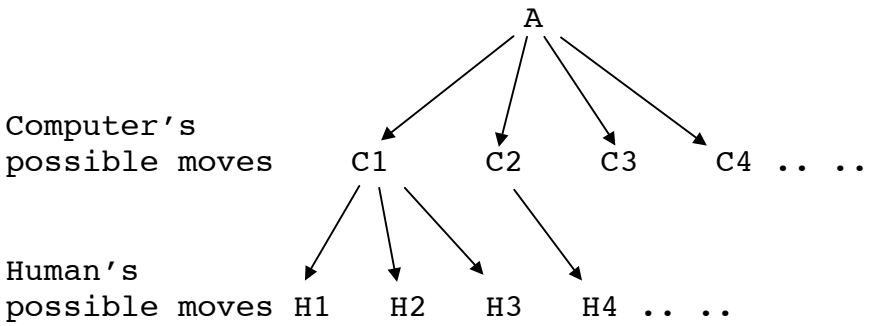


Figure 4. Minimax tree

Since both players are trying to get five stones in one line, the interesting area on the board is where the stones congregate. The whole board is divided into small squares or rectangles. For example, when searching for a consecutive five stones on horizontal line, we can start from the first row and first column. Check the first five positions of the first row and see if they are all occupied

by stones of the same color consecutively. Then move one column to the right, and check again. Follow the same steps until reaching the right end. Then search the second row the same way until finishing the whole board. If a five is found, the program declares a win or loss status.

When searching consecutive five stones diagonally, a square area of 5 by 5 is checked.

Checking stones for 4, 3, 2 and 1 all uses the same strategy. The computer will search black and white stones separately. When a pattern is found, its value will be added to the total value. The value for each move is the sum of the values of all patterns on the board after the move.

The search starts to work after the human player has made a move. The computer is supposed to put a white stone at an empty position. Then human would choose another empty position and put a black stone there. The computer will search all patterns on the board and calculate the total heuristic values for its possible moves. In a similar way, the human player chooses his next move that yields the smallest value from all the possible moves. This will continue until a win or loss is reached or the board is full. This is illustrated in Figure 5.

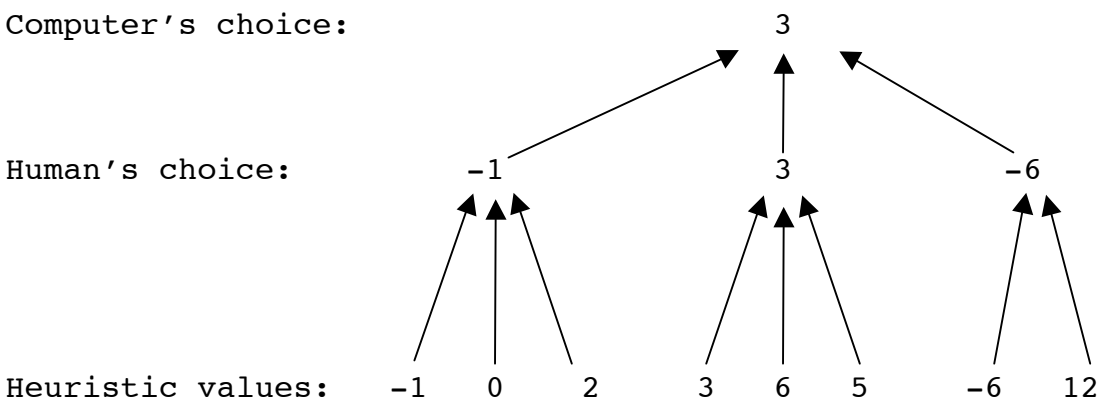


Figure 5. Example of the minimax tree search

In the example, for the first possible computer move, the human player has three possible moves with heuristic values of –1, 0, and 2. The human choose the smallest value, –1, and passes it to the computer. For the second and third computer moves, the human player gives the computer 3 and –6, respectively. Then the computer picks the largest value, i.e., 3, from all the values the human provides and makes the move.

### 2.2.4 Heuristic Functions

The computer move is calculated through a set of heuristic functions. The white stone represents the computer. Its heuristic value is positive. The black stone stands for the human player and its heuristic value is negative. Since the game is to try to form five consecutive stones, a pattern of this nature would have the highest value. Four white stones in one line would be the next

highest. The same rules apply to the black stones. Five black stones in one line has the largest negative value (or the least value). Four stones in one line would be the next largest negative value. See Table 1 for value assignments. In the table, we notice that there are three values for the 3-stone patterns. Some are higher, and some are lower. If we consider the patterns in Figure 6, it is easy to understand. The three blacks on the right are dangerous to the white. It has a value of –20, while the three blacks on the left is worthless. It all depends on the specific situation around those stones.

Table 1. Heuristic value assignments

| No. of stones | Value for white | Value for black |
|---|---|---|
| 5 | 100 | -100 |
| 4 | 50 | -50 |
| 3 | 20 | -20 |
| | 17 | -17 |
| | 13 | -13 |
| 2 | 10 | -10 |
| 1 | 1 | -1 |

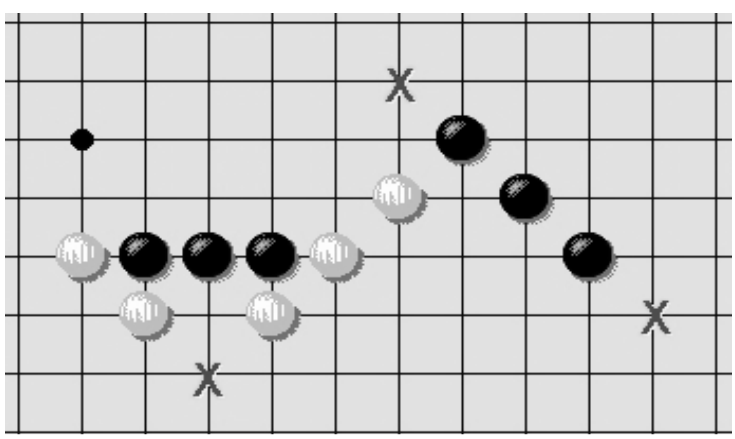

Figure 6. Example 1 for value assignment

The example in Figure 6 will give you some ideas about the value assignments. The calculated value for the example should promote the white to win. X's are the possible moves for white. In Figure 7, the black already has four consecutive stones in a line. The calculated value should guarantee the white move to block the black from forming five (losing for computer). It is especially important when the look ahead level of a minimax game tree is only 1.

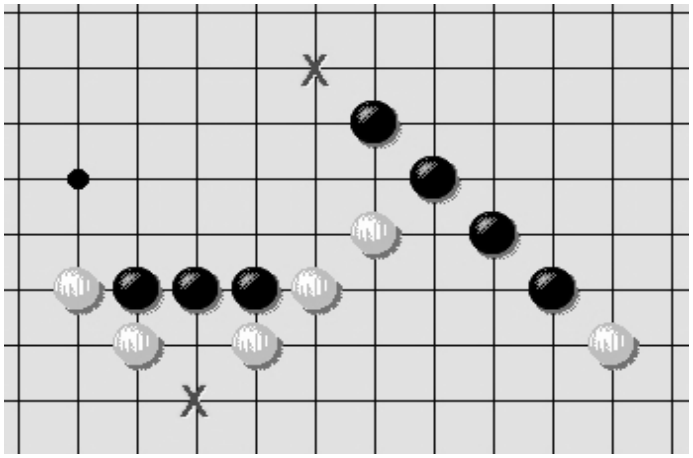

Figure 7. Example 2 of value assignment

The example code for searching a horizontal 3-stones pattern on the board follows:

```
//search horizontal 3-stones pattern from left to right
for(i=up_border;i<low_border;i++){
        if(i<0) i=0; //Prevent array from out of boundary
        if(i>18) break;
        for(j=left_border;j<right_border;j++){
                if(j<0) j=0;
                if(j>18) break;
                if(pos[i][j]==0  //Empty position
                  &&pos[i][j+1]==c    //c could be white or black
                  &&pos[i][j+2]==c   // Stands for 0 c c c 0
```

```
                &&pos[i][j+3]==c
                &&pos[i][j+4]==0)
                        value = 20 * c;
            if((pos[i][j]==0
                &&pos[i][j+1]==c
                &&pos[i][j+2]==0   //0 c 0 c c 0
                &&pos[i][j+3]==c
                &&pos[i][j+4]==c
                &&pos[i][j+4]==0)
                ||(pos[i][j]==0
                &&pos[i][j+1]==c
                &&pos[i][j+2]==c   //0 c c 0 c 0
                &&pos[i][j+3]==0
                &&pos[i][j+4]==c
                &&pos[i][j+4]==0))
                        value = 17 * c;
            }
        }
}
```

The computer move involves huge amount of searching and calculation. For a complicated board status, it may take a PC or Sun workstation several minutes to do this.

### 2.2.5. Position Control

A two dimensional array, pos[19][19], is declared to manage the 361 positions on the game board. In the beginning, the array is initialized with 0's. Zero represents a blank position; one (1) stands for a white stone; minus one (-1) for a black stone. The reason to use 1 and –1 is that it is convenient to find the opponent color. If the value in color variable "c" is 1 (white), then the opponent color would equal to "0 – c" (black).

When the human player makes a move, the corresponding position in the array will be assigned value minus 1. When the computer makes a move, the corresponding position in the array is assigned 1.

If the value of a position in the array is not 0, which means the position is not empty, a move to this position is prohibited.

### 2.2.6. Performance Improvement

The described minimax search tree approach yields a very slow operation speed. In order to improve the game performance, several techniques have been tried.

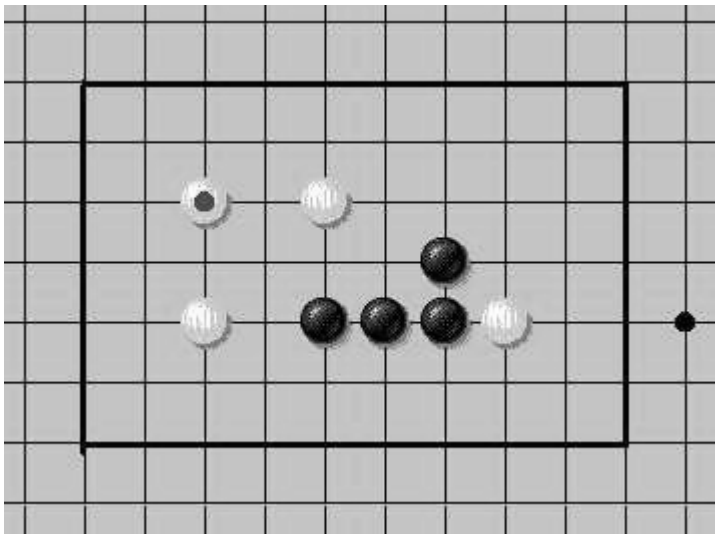

Figure 8. Set borders

The first method we tried is to set borders (Figure 8). Before starting the search, find the scope of the

stones already on the board. We search the area with the stones only. This method greatly reduces the search work. Each computer move on a fast machine takes less than 1 minute, which makes game playing possible. The borders are set two positions outward, which give the human player and computer more space for their possible moves.

The second strategy is to consider some special cases. This is different from the $\alpha$-$\beta$ cut-off tree [5]. In an $\alpha$-$\beta$ cut-off tree, the computer still needs to calculate the heuristic values and compare them. In our modified search approach, the computer searches the most critical situations. If any is found, an immediate action is taken without calculating the heuristic value. These situations include four and three consecutively in one line. At the same time, the heuristic value calculation methods for four and five are removed, which also reduces the calculation time.

In the improved search, the computer is trying to identify white four and then black four, white three, and black three. When the white has a four connection, the computer can make a move to win without carrying any calculation. On the same token, if the black has a four connection, computer has no other choice but to block. When a white three-connection is found, the computer can make a

move to form a four-connection to be in a possible winning position. Similarly, if the black already has a three-connection, the computer has no choice but to block.

Another possible way to improve operation speed is to reduce the levels of the minimax search tree. In our earlier program, only two levels were adopted. This might not be a good solution. However, after some tests, we found that even one level is still very intelligent for this game.

The demo version of the game on the Web has only one search level. It still provides certain challenge to the players. We can add more search levels to the minimax tree when fast machines or better algorithms are available.

Another problem is the obvious no-so-interesting moves. Although the computer seems to play well, it often puts stones in the same positions if the human player plays the same way. To avoid repeated moves, randomness is introduced. Whenever the calculated values are equal, the computer will randomly select one. So the move is not fixed. The code is like:

```
if(value == max_value){//If equal, randomly select one
        Random random = new Random();
        int r = random.nextInt();
        int ran = r%2; //generate only 0 and 1
}
```

An additional improvement for the game performance is to use a red dot to indicate the last computer move. This will assist human player to identify the last move by the computer. After the human makes a move, the red dot will disappear.

In addition, some other measures have been taken to prevent the game from crashing. One of them is to use a move flag. If mouseDown is activated twice, only the first move is counted. If mouseDown is activated outside the board, the value of the flag would not change and no move is made.

Since the stones are images with a fixed size, the board resize is disabled. In the program, there is a check_win method. This method is called every time a move is made either by the computer or the human player. If a five connection is found by the check_win method, the program will declare a win by drawing red circles around the five stones and goes into the win status. Further move is forbidden.

### 2.3. Human against Human

The most interesting part of the Go-Moku game is the version played by two human players. Two persons can play the game through the Internet.

The design of this part follows the client-server model. The server is a standalone program handling all requests from the clients. The client is an applet dealing with the interface with the human players.

### 2.3.1. Interface Design

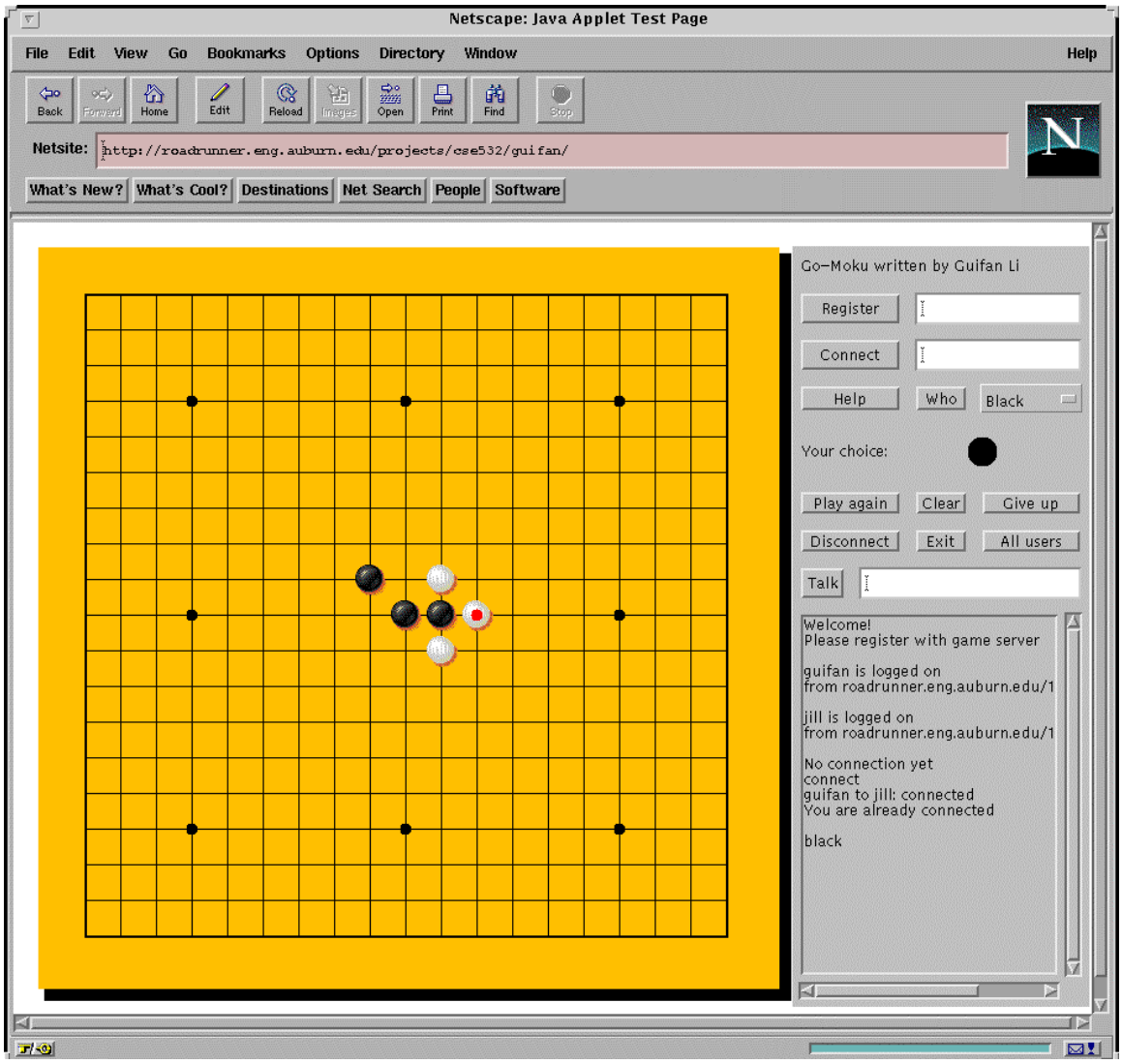

Figure 9. The interface for human against human

The interface of "human against computer" is an HTML frame page. But the "human against human" uses only the applet as the interface (Figure 9). The control panel for the game board is also embedded in the applet. The applet program contains Board, P_panel, and ClientThread classes.

The board and control panel is put on the screen using BorderLayout [6].

```
SetLayout(new BorderLayout());
add("Center",canvas); //This is the game board
add("East",mypanel);  //The control panel
```

The Board class is similar to the one in "human against computer". It is responsible for game board drawing, making moves, and a win status checking. In addition, it needs to send its stone positions to its counterpart.

The control panel is located on the right-hand side of the screen. It has buttons, labels, text fields (single line), and a text area (multiple lines). They are arranged on the panel according to the GridBagLayout [7].

```
GridBagLayout gb1 = new GridBagLayout();
SetLayout(gb1);
GridBagConstraints c = new GridBagConstraints();
c.fill = GridBagConstraints.BOTH;
c.insets = new Insets(5,5,5,5);
c.gridx = 0; c.gridy = 0;
c.gridwidth = 8;
c.gridheight = 3;
c.weightx = c.weighty = 1.0;
```

```
gb1.setConstraints(infoLabel,c); // Author information
add(infoLabel); // add to the layout
......
```

The first item in the control panel is infoLabel, which displays the author's information.

The second item is a registerButton and registerText field. The user types in his/her username here to log on to the game server.

The next item is a connectButton and connectText field. The player chooses a user from the username list to connect to and play with.

The helpButton shows you how to use the control panel. The whoButton finds the users who currently logged on to the game server and have no connection yet.

The selectLabel and the Choice menu allow players to select a stone color. The selected color will be drawn on a canvas on the right of choiceLabel "Your choice".

The play again button resets the game and starts a new game with stone color switched. The clearButton resets the text area below. The giveupButton allows a player to stop the current game when he finds that he has no hope to win this game and start another. The disconnectButton stops current connection with the partner. The allusersButton will find all users currently logged on. The exitButton

allows the players to logout from the game server unconditionally and show a new page of HTML document (for this project, it shows the main game page). When exitButton is pressed, the method "closeMe()" is invoked.

```
public void closeMe(){
        hide(); //Hide the control panel
        board.hide(); //Hide the game board
        //Display HTML document
        try{
          appletcontext.showDocument(new
          URL("http://www.eng.auburn.edu/~liguifa/index.html"));
        } catch(MalformedURLException (e) {
                System.out.println(e);
        }
}
```

Two players can communicate through talk text field.

The last is a text area, in which all messages are displayed.

There are also other commands which are hidden from the users. They are for management purpose. For instance, "/n" is used to check how many users have logged on since the running of the server. Other commands allow the server manager to talk to a specific user or to broadcast messages to all users. Some commands even allow the manager to kill a user process. Since some of the commands are not friendly, they are not accessible to the users.

A command usually starts with a "/" which can be distinguished from the common talk messages.

The stone position is also transmitted through the same line, the network socket. If a user enters a number in the talk window, it is very dangerous. It may be interpreted as a stone position. In turn, it may crash the whole game. So the program adds a '*' at the beginning of a string, if it is a number entered in the talk textfield. When the partner client program receives the string, it will remove the '*' and display it back as a number in the text area.

### 2.3.2 Network Coordination

The game for human against human is a game of networking. The game server provides all the communication services to the clients (human players).

The networking connection is implemented through sockets. The server is a program running all the time. It is in a status waiting for a client to connect to it. If a client tries to connect to the server, the server accepts the request and establishes a socket for connection. Then the server is ready for the next connection. The examples for client socket and server socket follow [7].

```
//Client socket example
import java.net.*;
import java.applet.*;
public class Client extends Applet implements Runnable{
        Socket sock;
```

```
        String servHost;
        int  servPort;
        public void init(){
            servHost = new String("roadrunner.eng.auburn.edu");
            InetAddress addr = InetAddress.getByName(servHost);
            sock = new Socket(addr, servPort); //Create a client socket
        }
    }

    // Server socket example
    import java.net.*;
    import java.io.*;
    public class Server{
        public static void main(String args[]){
            int port;
            try{
                ServerSocket serv = new ServerSocket(port);
            } catch (IOException io) {
                System.out.println("Error: creating server socket");
            }
            try{ //Accept a connection from a client
                Socket clnt = serv.accept();
            }  catch (IOException io) {
                System.out.println("Error: accepting a connection");
            }

            /* code to implement a task */
            // close the socket connection
            try{
                serv.close();
            } catch (IOException io) {
                System.out.println("Error: closing server socket");
            }
        }
    }
```

When the sockets are established, the communication between server and clients is carried out through the sockets.

This program allows multiple clients to access the server through multithread programming on the client side and a slave server (multithread) implementation on the

server side. Each time a new client is connected to the server, a new thread is created.

The users' information is stored in a class called RegisteredUsers. In this class, some of the items are not used right now. They may be used in the future. The server currently has 100 connections available. If more users request this service, we can modify the code to allow even more users. But more users might reduce the server's performance.

Networking with multiple clients causes a problem, i.e. the clients compete to communicate with the server. This could result in a server hangup. The solution is to synchronize all requests and jobs dealing with the clients. For instance:

```
//synchronize user registration
public synchronized register(String name, int thread){

}
```

To explain how the client-server model works, let's go back to the control panel. The server is supposed to be running all the time. When a client is connected, the server will send the client a message and ask it to register. The client then types his username in the register text field. The client program will add a header "/register" to the message. When the server receives this

message, it sends the message to a parsing method. The parsing method first analyzes the header and then sends it to the corresponding method for processing according to the header.

All button commands on the control panel are treated this way. When the client needs to find a game partner, he can use the “who” button to find the potential partner list. Without the connection, the client could do nothing. When connected, the client can communicate with only the partner connected.

When a new user is logged on, all clients currently on the server will receive a message of this event. The user waiting for connection could make a connection to the user newly logged on.

After a connection has been set up, only one user is allowed to select the stone color. The player holding the black stone makes the first move.

The method mouseDown is also used to make a move. This is not only for the black but also for the white. Both players use mouseDown to place a stone. When the mouse is pressed, the stone image will be drawn immediately on the board of the player who makes the move. Then the stone position message will be sent to the partner through the server. A position on the board is specified using two

numbers. One is x and the other is y. The legal value ranges from 0 to 18. The position values x and y are packed in a string. Then a flag with value 1 or 2 is appended to indicate the number of digits for y. For example, x and y have values 12 and 9, respectively, the string will be 1291. This is done by the following code.

```
//send stone position to opponent
//the position is x and y
if(y<10) //one digit
        gamePanel.socket.out.println(x + "" + y +"1");
else //y has 2 digits
        gamePanel.socket.out.println(x + "" + y +"2");
```

The empty string "" here is used to prevent the arithmetic "add" operation (x + y) of x and y.

On the partner's side, the string will be decoded (see the following code). Since all messages are sent through the same socket, first we should test if it is a number. If it is a number, the message is converted into x and y.

```
//Convert the string back to numbers
try{
        str = in.readLine(); //read msg from socket
        num = Integer.parseInt(str); //convert to number
        //if number, calculate position
        if(num%2) == 0){//check if the last digit is 2
                num = num / 10; remove the last digit
                x = num / 100; // Get x
                y = num % 100; // Get y
        }
        else {//if the last digit is 1, example 1291
                num = num / 10; remove the last digit, 129
                x = num / 10; // Get x, 12
                y = num % 10; // Get y, 9
        parent.board.drawOpponent(x,y);//draw opp. stone
        // if not a number
```

```
} catch (NumberFormatException e) {
        System.out.println("Not a number");
}
```

As mentioned before, a number message (other than string message) sent through the talk text field is automatically inserted with a header "*" to prevent confusion.

During the game, both players can communicate through the talk field and the display text area.

The game now can serve 100 players, that is, 50 games at the same time. Each game is independent of all other games. A player can stop current connection and then connect to another player who has no connection yet.

When a player stops the current connection, his partner will receive a disconnection message and the game board is reset. If the player logs out or closes his browser, his partner will also get a disconnection message and all users will get a message about the logout.

# 3. DISCUSSION

From this project, it is clear that the Web, as a distributed information system, has a great potential and Java is a powerful language for Internet programming.

After this game has been put on the Web, many feedbacks have been received from users. Some said the game is great and the algorithm is smart. Others said the server is not robust and they also reported some bugs. My colleague and I tested and debugged the program for a while. Now all major bugs have been removed. It has been working fine since January 28, the date I restarted the server.

However, to achieve a robust game status, more work is still needed. We may choose to adopt fixed username and password, when more people play this game. We may also create a record about each user's performance and list the top ten players.

For the human against computer version of the game, we need to continue to try better algorithm to improve the game performance. On the other hand, if the computer is too difficult to defeat, it may discourage people to try.

# 4 REFERENCES

[1] December, John and Mark Ginsburg. HTML & CGI Unleashed. Sams.net Publishing. 1995.

[2] Vanhelsuwé, Laurence. Mastering Java 1.1. SYBEX. 1997.

[3] Ritchey, Tim. Java!. New Riders Publishing. 1995.

[4] Cornell, Gary; Cay S. Horstmann. Core Java. SunSoft Press. 1996.

[5] Rich, Elaine. Artificial Intelligence. McGraw-Hill Book Company. 1983.

[6] Flanagan, David. Java Examples in a Nutshell. O'Reilly. 1997.

[7] Nagaratnam, Nataraj; Brian Maso; and Arvind Srinivasan. Java Networking and AWT API. Waite Group Press. 1996.