TRIBHUVAN UNIVERSITY
Institute Of Engineering
Pulchowk Campus
Department Of Electronics and Computer Engineering

A
Minor Project Report
on
"Electronic Whiteboard"

Submitted
By
Asim Shrestha
Prabesh Shrestha
Subodh Raj Satyal

Submitted
To

Department of Electronics and Computer Engineering
Pulchowk Campus
27th Feb, 2007

TABLE OF CONTENTS

Acknowledgement

We are thankful to the Department of Computer Engineering ,IOE(Institute of Engineering) pulchowk campus for providing us the opportunity to perform a project on Whiteboard for the minor project III/II .We are also thankful to our respected lecturer Mr. Bikash Shrestha and JayaRam Timsina for their invaluable suggestions and support.

In addition, our thank also goes to all the colleagues who have offered their insightful suggestions and comments.

Asim Shrestha
Prabesh Shrestha
Subodh Raj Satyal

Abstract

In the world of programming languages the new programming techniques of visual programming is gaining priority day by day due to their efficiency over procedural techniques of programming. The course ‘Minor Project’ is aimed with giving us the knowledge of current trend visual programming and implement it in the project work. Java and its counter parts are proving to be the good programming language for the fulfillment of the project’s basic objectives and its extensions so the same language has been used as the core language for this project. Different APIs provided by Java has been used to accomplish the project.

Whiteboard is an interactive computer application within a LAN (Local Area Network) that provides an interactive screen for two or more people to be connected through a common server via a network. We as far as possible have tried to make sure that two or more people working with this application connected to the same server feel like working on the same screen. Change in one of the screen connected to the network is automatically detected by other screens connected to the network. Along with this feature, it can also be used as a standalone application to draw simple 2D graphics and write text. The users can then save the completed work for future use and later load that and continue their incomplete works.

2D APIs have made the drawing of the objects in the whiteboard screen. And the connection between the server and the clients have been accomplished with the APIs of RMI(Remote Method Invocation) .Other APIs of Java SE have been utilized to apply the concept in this project.

Introduction

The common trends of visual programming language have greatly replaced the older structured language. OOP (Object Oriented Programming) have made programming easier and more effective. The object model of java is simple and easy to extend while simple types such as integers are kept as high-performance non objects. So the need of knowledge about visual programming is essential for any field related to programming and is a must for the computer engineering students.

With the above motive in mind we have implemented the object oriented programming for the course of “Minor Project” which requires us to accomplish a small project on the current trend of visual programming. One of the objective if this course is to help us get familiarized with different aspects of programming which will provide us good knowledge and experience for the successful completion of the final year project .We have used Java as the programming language which is a complete OOP Language.

Current trend of using the computer in every field as far as possible, Whiteboard is a network based application that works on a client/server mode. It can find its application in different fields as teaching, meetings and seminars. It can be also used for distant learning as the users need not be physically present in the same location, they can interact through the network residing at their respective places. So it can change the conventional style of teaching to the emerging trend of using the computers. Instead of using the marker to write on the conventional white or black boards, it uses keyboard and mouse to write or draw the objects.

In the client window, users draw simple geometrical objects such as circles, rectangles, freehand drawing and write texts .First of all, the server is started and different clients are connected to the server. All the information sharing is done through the server via the server. The client passes the objects to the server and the server then passes those objects to all the clients thus updating the corresponding client’s window (screen).We have used file handling techniques provided by java API for saving and opening of files. It saves two files for each window one image file and the other “.out” file. The image file contains the image of the shapes we had made in the window. The image file can be viewed simply using the normal image viewer program. File with “.out” extension is whiteboard application specific file which can be opened only with our application. The

“.out” file contains the saved vector shape list so every operation possible in our application can be continued with the file with “.out” extension.

Objectives

The main objectives of choosing electronic whiteboard for our minor project are listed below:

- To implement the current trend of visual programming
- To formulate project documentation for our final year project
- To learn OOP (Object Oriented Programming) concept of java programming.
- To have the concept of Different APIs of Java (core java, RMI , 2-D ,file handling ,event handling etc)
- To learn to work in group
- To implement the knowledge gained by carrying out a small project
- To provide an easy platform for distant learning
- To complete the III/II minor project.

Methodology

After knowing to carry out a small project for the course of “Minor Project”, we thought of choosing to develop an application that will implement most of the feature of programming as well as can be implemented in the present context in day to day life. Whiteboard uses many of the programming features (drawing objects, event handling, remote computer access, file handling etc).So we chose whiteboard project.

Then we performed the requirement analysis for the project. As the requirement of the project was to choose a visual programming tool, we chose java as our programming tool because it is platform independent and powerful. As latest version of Jdk is 1.6 we used it which is backward compatible.

Collections of necessary tutorials were collected from the internet by browsing through various websites. Also we downloaded various applicable e-books.

Then after collecting all the required tools we started to design the application. Then the GUI for the application was prepared with Netbeans IDE. All the coding needed was then inserted in the application. While in problems; we took the ideas from friends and various websites. Then the simple running prototype was ready. After that we kept on debugging and enhancing the prototype. This can be extended for a long time as there is no limit of adding extra features in the whiteboard.

Literature Review

RMI (Remote Method Invocation)

Remote Method Invocation (RMI) allows a java object that executes on one machine to invoke a method of a Java object that executes on another machine. It allows us to build distributed applications that a whiteboard application is.

All remote objects must extend UnicastRemoteObject which provides functionality that is needed to make objects available from remote machines and implement the remote interfaces that should extend the remote interface, which is a part of java.rmi.Remote. Its purpose is simply to indicate that an interface uses remote methods.

```
public interface ServerBoardInterface extends Remote{

……

}

public class BoardServer extends UnicastRemoteObject implements
ServerBoardInterface{

}
```

Server (BoardServer in our case) must update the RMI registry on that machine .This is done by using the rebind () method of the Naming class(found in java.rmi).That method associates a name with an object reference. Also it can specify the port through which the clients can invoke the methods .Specifying zero(0) will allow clients to connect through ramdom port. This is done using Locateregistry.createRegistry(port).

```
Registry registry = LocateRegistry.createRegistry(port);

BoardServer server = new BoardServer(port);

//binds the server under the name Boardserver in the registry

registry.rebind("Boardserver", server);
```

The client side of the application requires two command line arguments.The first is the IP address or the name of the server machine .The second arguments is the port number as set by the server application.

Registry = LocateRegistry.getRegistry(server_addr, (new Integer(port_no)).intValue());

After getting the connection and locating the registry client must look up the remote object.

```
Server = (ServerBoardInterface) (registry.lookup("Boardserver"));
```

Because whiteboard must have two way connection client must also extend UnicastRemoteObject.We can specify the port number as zero(0) as several clients might be connect at the same instance and it will not be a good idea to use various port numbers for each individual clients.

```
UnicastRemoteObject.exportObject((WhiteBoardInterface) (this.paintArea1), 0);
```

where WhiteBoardInterface is the interface implemented by the client(i.e paintArea in our case).

Java 2-D

To draw shapes in the Java 2D library, we need to obtain an object of the Graphics2D class. This class is a subclass of the Graphics class. Since we have used java version 1.6, methods such as paintComponent() automatically receive an bject of the Graphics2D class.

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g; //typecasting graphics to graphics2d
```

……

}

The Java 2D library organizes geometric shapes in an object-oriented fashion. In particular, there are classes to represent lines, rectangles, and ellipses:

To draw a shape, you first create an object of a class that implements the Shape interface and then call the draw method of the Graphics2D class. For example,

```
Rectangle2D rect = new Rectangle2D.Double(x_1, y_1, width, height);

g2.draw(rect);
```

We have constructed strokes of arbitrary thickness. For example, here is how we drew lines that are 10 pixels wide.

```
g2.setStroke(new BasicStroke(10.0F));

g2.draw(new Line2D.Double(. . .));
```

Similarly other shapes are drawn using the java 2D library.Ellipse2D describes a circle defined by a framing rectangle. Generalpath represented a geometric path constructed from st lines quadratic and cubic curves. lineTo is used to draw freehand which adds a point by drawing a straight line from the current coordinates to specified coordinates. To make a line we used setLineParm() function with parameters initial point (start_x,start_y) and final point(end_x,end_y).Filled rectangle and ellipse are generated using fill() function.

File Handling

We have used fChooser (instance of Jfilechooser which provides a simple mechanism to choose a file). The file name obtained from the dialogue box and it's location is passed to save file () function. In this function we create an object output stream and write the shapelist to this stream to save it. Here for saving the file the content of the whiteboard is stored in vector. This function saves two copies of the file one image file and the other ".out" file. The image file can be opened using any normal image viewing program and is not necessary to open the whiteboard application. This file can be used only for viewing and any modifications cannot be done. For this purpose the ".out" file is required. To open the ".out" file whiteboard application is required where the necessary modifications can be done. We also have mechanism for overwriting the previously saved files and also prompt the user if the name given by the user already exists.

We can open the previously saved file in ".out" extension in our whiteboard. Similar to the saving we obtain the file name using fChooser and dialogue box, then we create an objectinputstream and then read the elements of the previously saved vector shapelist and then repaint it to get the desired file.

Event Handling

Any operating environment that supports GUIs constantly monitors events such as keystrokes or mouse clicks. The operating environment reports these events to the programs that are running. Each program then decides what, if anything, to do in response to these events. Event sources have methods that allow you to register event listeners with them. When an event happens to the source, the source sends a notification of that event to all the listener objects that were registered for that event. We have used the features of Netbeans IDE to implement the event handling such as mouse pressed, mouse released ,mouse dragged etc .Drawing of the objects ,sending the drawn object to the server while the mouse is pressed ,invoking methods after pressing the menu bar ,changing the labels after the occurrence of events are all possible because of event

handling. These can be helpful to monitor the mouse actions and mouse movements over our whiteboard screen. Different functions have been assigned for each type of events. Event handling is also used in the selection of drawing tools and menu selections.

GUI

Swing and AWT classes of the JFC (Java Foundation Classes) have been used to complete the GUI portion of the application .We basically used the drag and drop features of Netbeans to implement the GUI building process.

Swing-based user interface elements are somewhat slower to appear on the user's screen than the peer-based components used by the AWT. But in any reasonably modern machine, the speed difference shouldn't be a problem. On the other hand, the reasons to choose Swing are overwhelming:

- Swing has a rich and convenient set of user interface elements.
- Swing has few dependencies on the underlying platform; it is therefore less prone to platform-specific bugs.
- Swing gives a consistent user experience across platforms.

This entire means Swing has the potential of fulfilling the promise of Sun's "Write Once, Run Anywhere" slogan.

Jframe, Jmenu, Jpanel, Jlabel, Jbutton, JmenuItem are some of the swing component used in this application. JfileChooser, JdialogBox are other swing components used to integrate event handling and GUI. Also Java look and feel” has been used to modify the default look of the java application.

Exception handling

Suppose an error occurs while a Java program is running. The error might be caused by a file containing wrong information, a flaky network connection, or use of an invalid array index or an attempt to use an object reference that hasn't yet been assigned to an object. The mission of exception handling is to transfer control from where the error occurred to an error handler that can deal with the situation. To handle exceptional situations in your program, you must take into account the errors and problems that may occur. there may be different types of error as User input errors, Device errors, Physical limitations, Code errors etc. Exceptions have their own syntax and are part of a special inheritance hierarchy. In the Java programming language, an exception object is always an instance of a class derived from Throwable. If an exception occurs that is not caught anywhere, the program will terminate and print a message to the console, giving the type of the exception and a stack trace.

To catch an exception, you set up a try/catch block. The simplest form of the try block is as follows:

```
try
{
  code
  more code
  more code
}
catch (ExceptionType e)
{
  handler for this type
}
```

Multiple exceptions can be caught by using a Try block and multiple catch blocks to handle each type differently. The separate catch clause for each type can be as :

```
try
{
  code that might throw exceptions
}
catch (MalformedURLException e1)
{
  emergency action for malformed URLs
}
catch (UnknownHostException e2)
{
  emergency action for unknown hosts
}
catch (IOException e3)
{
  emergency action for all other I/O problems
}
```

Working mechanism

For the whiteboard application to be run on the network the server must be run initially. Without connection to the server the application can run on stand alone as a simple MS

Paint application. The client sends a request to the server for connection by filling a connect request form specifying the server address, port number of the server and client name. The server then checks the input parameters and then decides to connect or not.

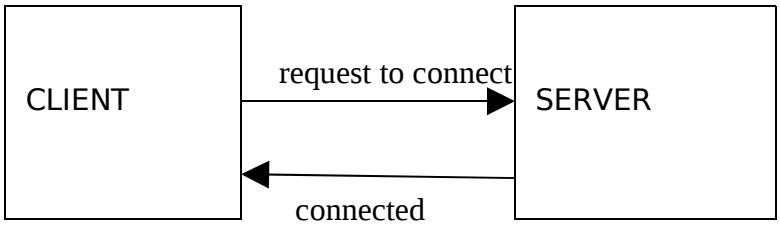

fig: server – client connection.

client side

registry= LocateRegistry.getRegistry(server_addr, (new Integer(port_no)).intValue());

this.server = (ServerBoardInterface) (registry.lookup("Boardserver"));

UnicastRemoteObject.exportObject((WhiteBoardInterface) (this.paintArea1), 0);

.

Server side

thisAddress = Inet4Address.getLocalHost().toString();

Registry registry = LocateRegistry.createRegistry(port);

registry.rebind("Boardserver", server);

Similarly other clients also connect with the server in the same manner. So many clients are connected to a single server. The client now draws shapes in its window which is stored in a shapeInfo. The shapeInfo stores all the informations about the particular shape such as the shapetype, stroke (if present), color of the shape etc. This shapeInfo is then saved to a vector shapelist. This shapelist is passed to the server. On the server a new vector called globalshapelist is present where the shapelist passed from the server is shaped. This globalshapelist stores the content of the latest shapelist so that when any

client connects in between of the process the window of the newest client contains all the informations that were happening before. the server then sends the contents of the globalshapelist to all the clients which enables in the communication between the clients.

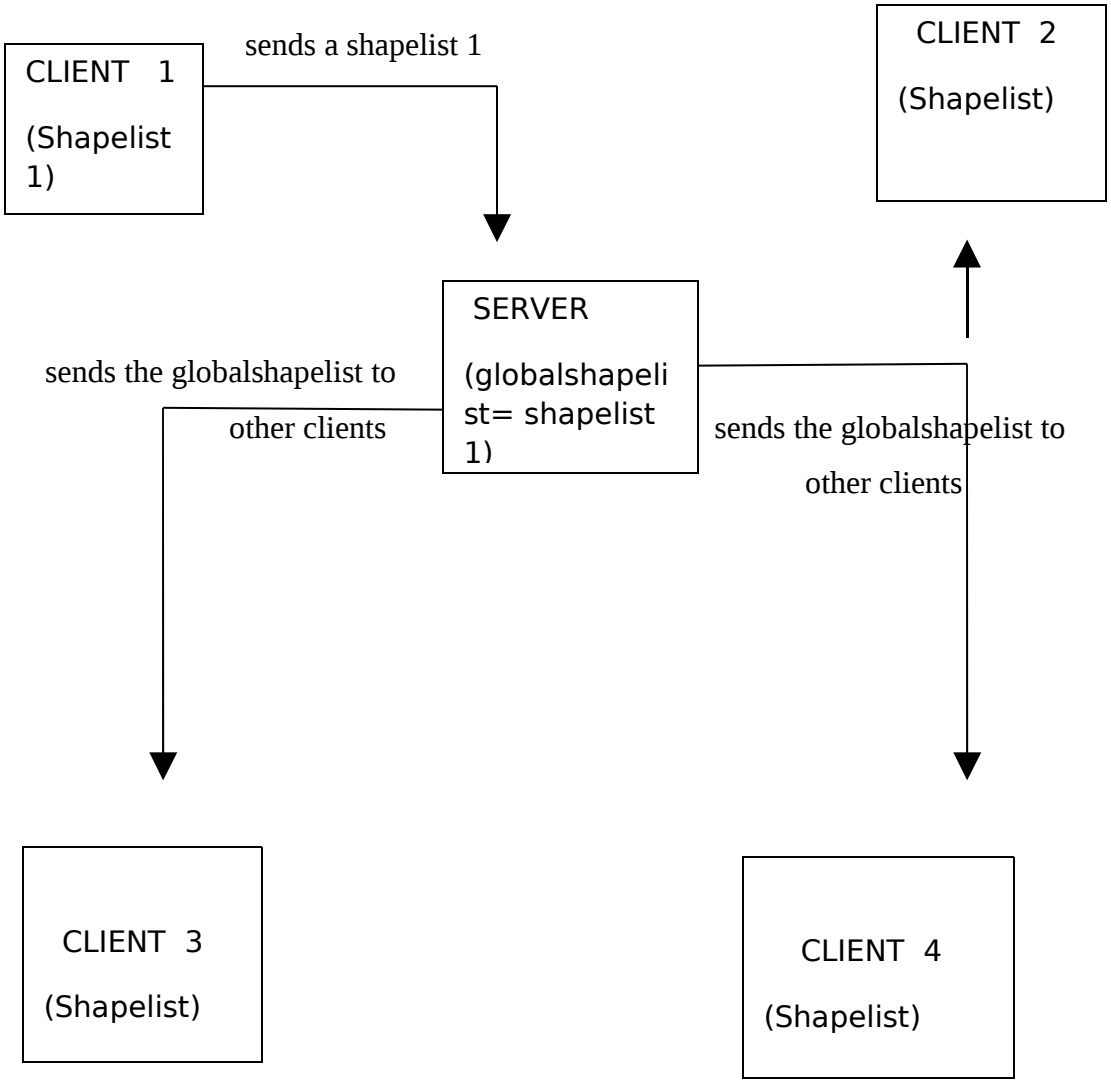

Fig-Server broadcasting the shapelist it receives

If new user connects to the server after the intercommunication between the clients i.e. if some shape is already made then the server casts the shapelist of all shapes already made

by other user to the new user so that he/she can view what has already happened earlier in his/her absence and now he can contribute.

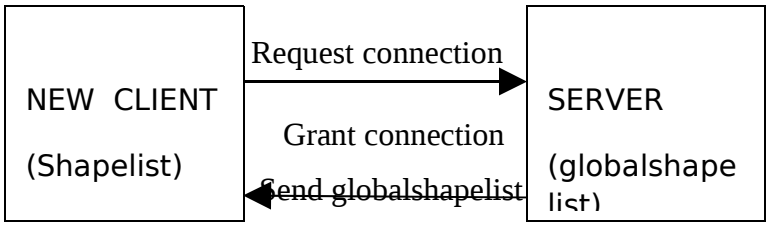

Fig-server sending it's globalshapelist to newly connected client

In the window the position of the mouse in the window is also shown. Also it is also shown whether the user is connected to the server or not.

Packages and classes

In this application we have made four packages namely images, mainboard, remote and whiteboard. The packages consist of a number of classes that have been used for the

development of the application. In the image package we have used image files to create icons for the drawing tool buttons and icons for the menu items.

In mainboard package we have three classes, connectform.java, whiteboardhelp.java and mainGUI.java. The connectform.java class is used to create the connect form for the client to connect to the server asking for the server address, server port number and the user name. Here we have specified the server port in the code to have a constant value.

The whiteboardhelp.java creates a new window displaying some information about the whiteboard, what it does shortcuts for different functions of the whiteboard. Overall it provides basic help for the users on how to use this application.

mainGUI.java is the main class of our application. It displays all the GUI information. The screen we get in the application is designed in this class. It consists of a Jframe containing panels. The white area in the mainGUI (a panel) is the paintarea used for the drawings. The icons for the drawing tools are displayed in the left hand side of the screen. It also consists of drop down menus containing different functions applicable. The functions of the drawing tools and the menu items are controlled by the mouse action listeners. We also have a provision to display the connect status of the client at the bottom left corner. An indication for mouse position on the window has also been displayed. The selected label displays the current drawing tool selected (such as line, eraser, selector etc) .above that the color information chosen at that instant is also displayed. The drawing tools used are the general geometrical shapes. Provisions for selecting a specific shape are provided by the icon selector. User can also draw freehand drawings using the freehand icon. Color chooser is also provided to select the color. Jcolorchooser class of Netbeans API has been used for color choosing dialogue box and for the selection of color.

In the remote package we have three classes boardserver.java, serverboardjavainterface.java; whiteboardinterface.java. The boardserver.java is the main server class. It contains the code for the server. The registry function is done in this class. Serverboardjavainterface.java is the interface for the server and contains the definition for

the functions of the boardserver.java. whiteboardinterface.java is the interface for the client(i.e. mainGUI class) and contains the definitions for the functions of the mainGUI class.

The whiteboard package contains the classes paintArea.java and shapeInfo.java. piantArea.java contains the main code for the paintarea (i.e main white area working part of the application). The drawings are done on this part of the window. The main paint function (i.e. drawing of the shapes and the texts from the corresponding shapelist) is in this class.

shapeInfo.java class contains information about the shapes to be drawn. The attributes and information of the shapes selected and drawn by the user are stores in the instance of this class and then added to the shapelist of the client.

Screenshots

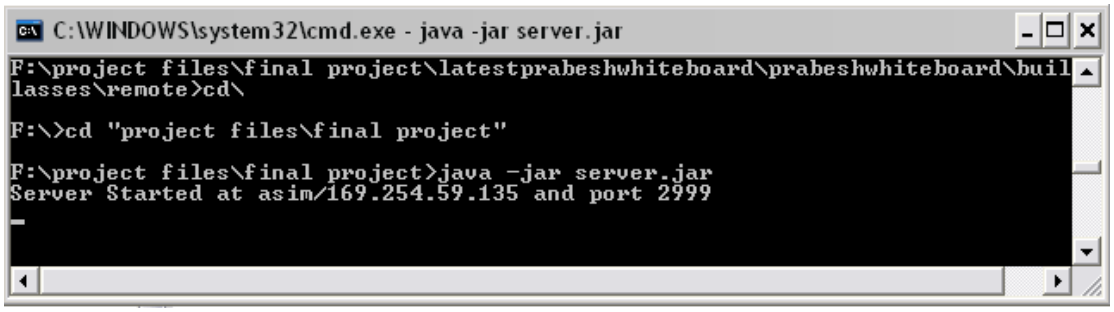

Fig: server started at respective port

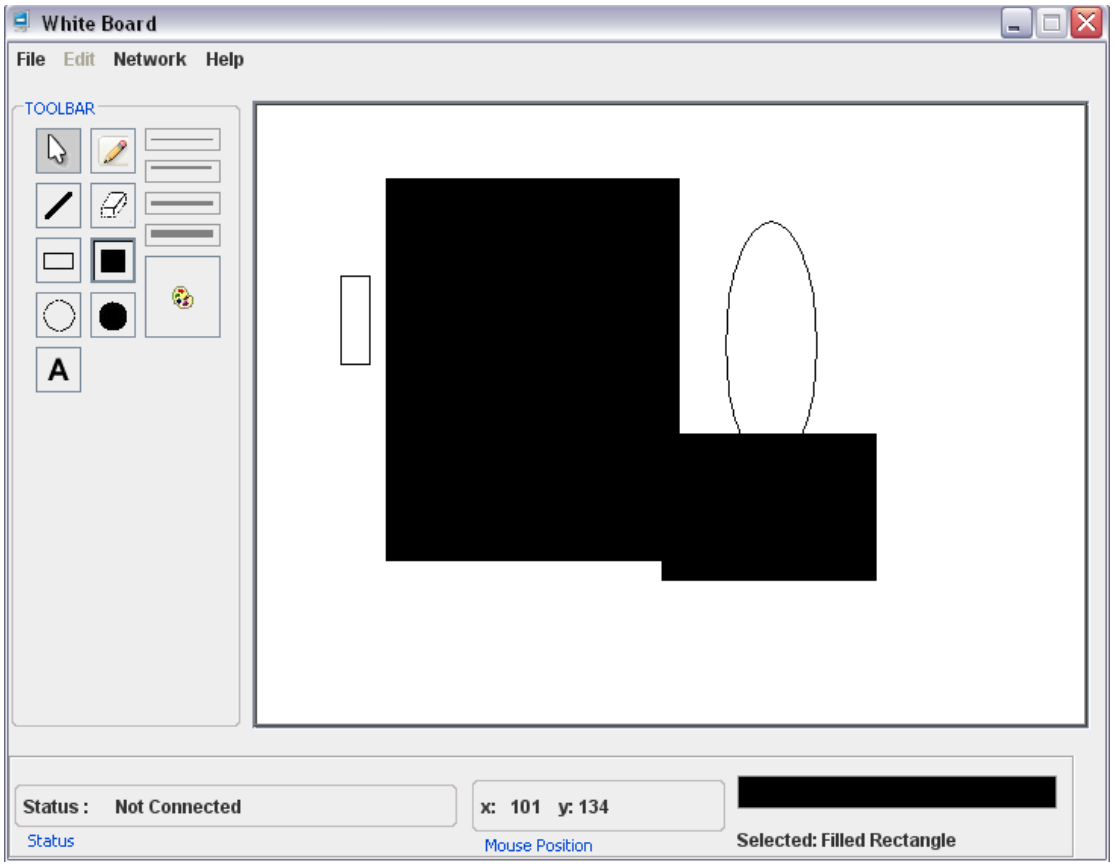

Fig : stand alone client with some drawings on the paintarea

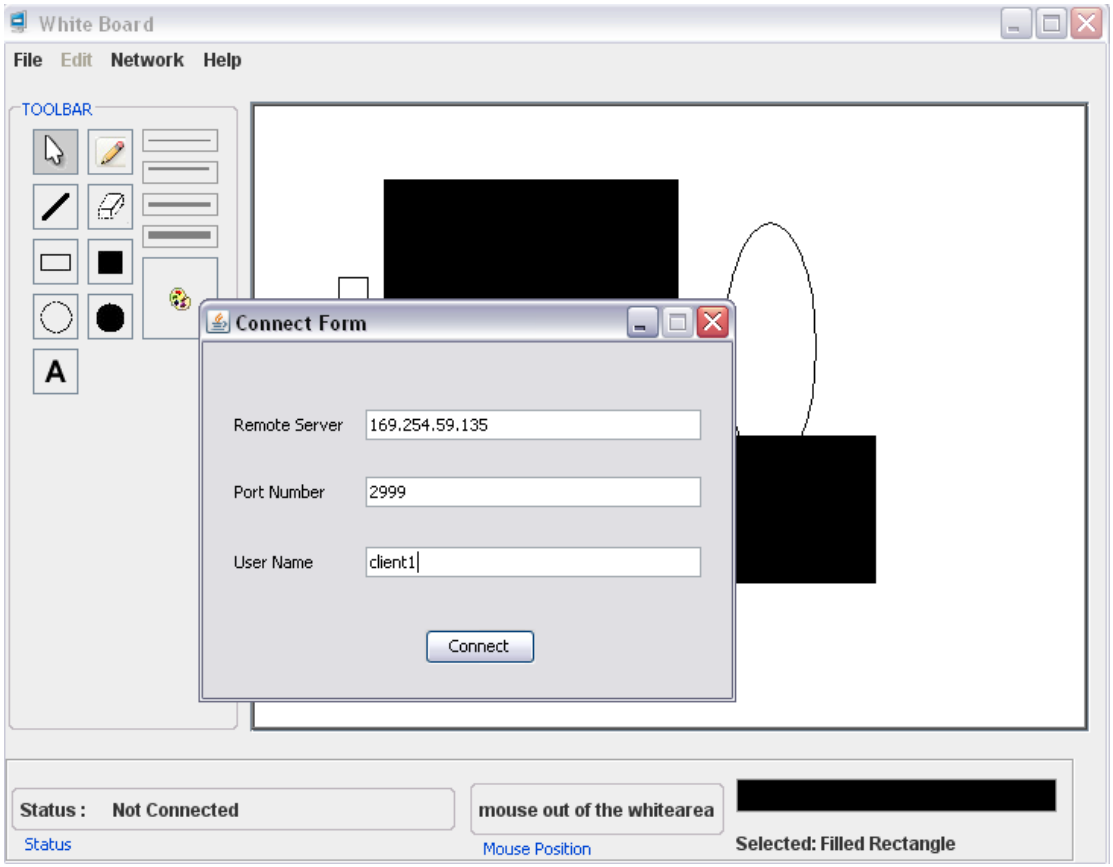

Fig : client requesting connection to server

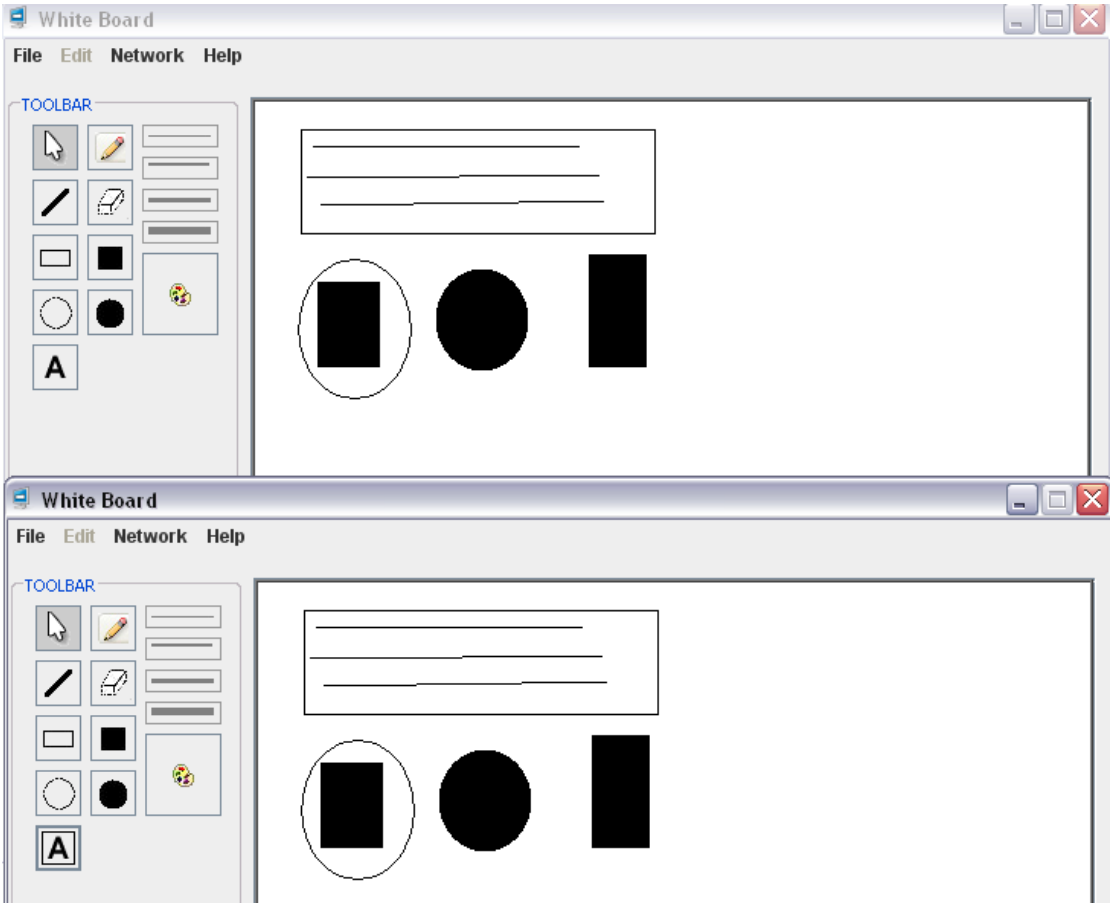

Fig: two client connected to the server and sharing the objects

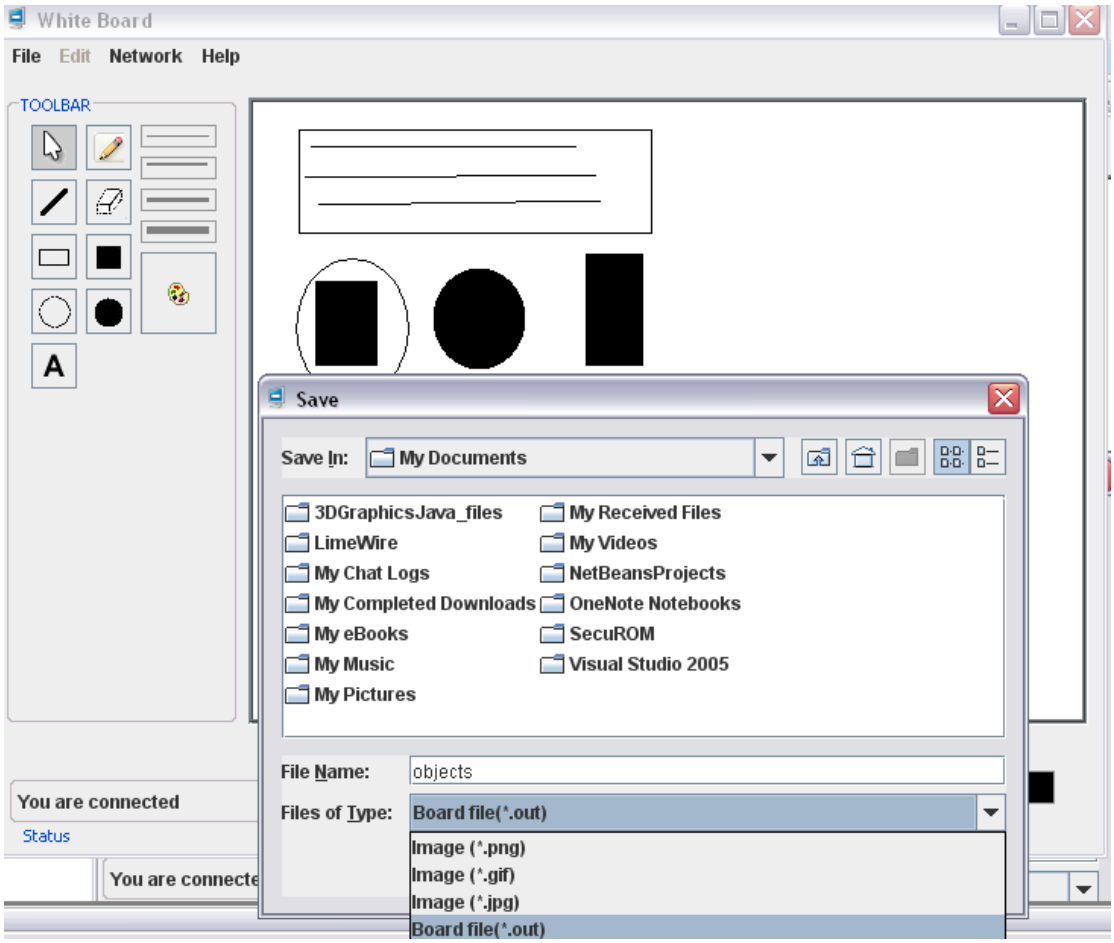

Fig: save operation

Scope of the project

After the completion of the project it can be used for the purposes stated below:

- For teaching/learning in educational institutes
- For interacting between members of a conference meetings
- People now prefer using keyboard and mouse rather than chalk and duster.
- Distant learning
- Standalone Paint application

Limitations & Future Enhancement

Time Constraint played an important role for the completion of the project. Due to lack of time the project could not be extended to the limit possible.

Some of the limitations are listed below:

- All the clients must connect to a single server
- Concept of super peer connected to the main server was not implemented
- Chat was not implemented
- Audio transfer was not implemented.
- Application is not bug free

Future enhancement for this application can be:

- Several super peers can be created to reduce the load for a single server
- Voice transfer can be implemented to create a sense of real classroom
- All of the bugs in the application can be removed

Conclusion

We have successfully completed the project. In course of our project development we faced different hindrances. These difficulties were solved with the help of seniors, colleagues and different other resources. During the development of the project we learned about the Java Programming and also gained idea about the networking using Remote Method Invocation(RMI). We also gained idea about the different API's of Java.

References

1. “The Complete Reference”, Herbert Schidlt
2. Core Java vol I and vol II
3. “Java: How to Program”, Deitel and Deitel
4. www.wikipedia.com
5. www.sun.java.com
6. www.sourceforge.net
7. www.google.com