ABSTRACT

VISUALIZATION TECHNIQUES FOR ROUTING PROTOCOLS AND ROUTER CONFIGURATIONS

By
Vandana Pursnani

An autonomous system (AS) is a group of routers managed by a particular organization. Exterior gateway protocols (EGP) are used between AS's.Internal Gateway Protocols (IGP) is used within an AS. The most common protocols used with TCP/IP are RIP , OSPF (Open Shortest Path First), IGRP / Enhanced IGRP .

The thesis revolves around OSPF protocol OSPF uses flooding to exchange link -state updates between routers. Any change in routing information is flooded to all routers in the network. Areas are introduced to put a boundary on the explosion of link-state updates. Flooding and calculation of the Dijkstra algorithm on a router is limited to changes within an area. Routers that belong to multiple areas, called area border routers (ABR), have the duty of disseminating routing information or routing changes between areas. Once information about routers is gathered there is no way to clearly visualize and manipulate it visually.

The thesis was aimed at visualizing this kind of Router configuration information Visually using powerful tools and to be able to manipulate the figure generated. It also aimed visualizing bottleneck paths in the router configurations. The Powerful features of Java 3D were utilized for Visualization. We utilized the GMatrix class in the Java 3D API to store the router information. This was mapped onto a 3D Cylinder. Also due to the platform independence, robustness, scalability Java was the choice for such a development since routers would be cross platform.

VISUALIZATION TECHNIQUES FOR ROUTING PROTOCOLS AND ROUTER CONFIGURATIONS

By
Vandana Pursnani

**A Thesis
Submitted to the faculty of
New Jersey Institute of Technology
In Partial Fulfillment of the requirements for the Degree of
Master of Science in Computer Science**

Department of Computer and Information Science

January 2001

APPROVAL PAGE

VISUALIZATION TECHNIQUES FOR ROUTING PROTOCOLS AND ROUTER CONFIGURATIONS

Vandana Pursnani

Dr. Constantine. N. Manikopoulos — **Date**
Associate Professor of Electrical and Computer Engineering, Computer and Information Science, NJIT

Dr. Jay Jorgenson — **Date**
Associate Professor of Department of Mathematics, The City College of New York (CUNY)

Dr. Sotirios G Ziavras — **Date**
Associate Professor of Electrical and Computer Engineering, Computer and Information Science, NJIT

Dr. Frank Y Shih — **Date**
Associate Professor of Computer and Information Science, NJIT

BIOGRAPHICAL SKETCH

Author:	Vandana Pursnani
Degree:	Master of Science
Date:	December 2000

Undergraduate and Graduate Education:

- *Master of Science, Computer Sciences*
  New Jersey Institute of Technology (NJIT) Newark, New Jersey,2000
- *MS, Computers Management*
  Devi Ahilya Vishwa Vidhyalaya (DAVV) University, Indore, India, 1999
- *Bachelor Of Science*
  Devi Ahilya Vishwa Vidhyalaya (DAVV) University Indore, India, 1997

Major : Computer Science

Presentations and Publications:

Vandana Pursnani and Manish Joshi
"Introduction to Internet and Web Technologies,"
CMC ATC, Indore, India,1998

Vandana Pursnani and Abhijeet Phatak
"Impact of World Wide Web on HR Practices"
Devi Ahilya VishwaVidhyalaya, Indore, India, 1999

**To my beloved mother and
the supreme Mystic Law
in the Gohonzon**

ACKNOWELDGEMENT

I would like to express my deepest appreciation to Dr. Manikopoulos, my research Advisor and Dr.Jay Jorgenson for providing valuable and countless resources, insight, invaluable support, encouragement and reassurance during the entire period of the research work. I would like to forward my special thanks to Dr. Ziavaras and Dr. Frank Shih for actively participating in the committee.

I would also like to take the opportunity to forward my thanks to many of my colleagues for their constant support and encouragement. I would especially like to thank Mr. Jatin Doshi, Miss. Sonal Panchal and Mr. Mukul Janeja.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the thesis is to create visualization tools for routers in a network where information about router connectivity with other routers can be determined by routing protocols like Open shortest path first (OSPF) and flooding algorithms. The router connectivity was read from a simple graph representation of the network area of routers for simulation purposes.

For generating the three dimensional representation a generalized matrix was proposed which would contain the connectivity data of each router to the other starting from one base router. For simulation purposes a matrix was generated from the graph representation of the network area with the set of routers. Once the matrix is generated it is sent to the 3D tool as a file and is read to create the figure. The matrix is a generalized matrix so that it can be expanded dynamically. The 3D figure itself is object oriented, where each element of the figure is a class so that various elements of the figure can be manipulated. The generic figure used for representation is a cylinder. It is a wire frame cylinder where the lines represent the connectivity and the router information.In the Cylinder the top and bottom, points on the circumference are connected with vertical lines representing the routers. The horizontal lines at different levels originating from these lines are the connectivity from each router to the other.

1.2 Background

Routers include an address for the network, which aids routers in finding the destination.Routing tables contain information about each of the networks in the fixed area. So when a packet arrives with an address, it compares the network addresses and forwards the packet to the destination. But when a section of the network fails the router has the responsibility to reroute the traffic. For this Routing Protocols are used. These are protocols that routers use to communicate to each other. These protocols are used to exchange the information contained in the routing table. By this a router can add a network and all other networks will know the route to that destination.

An autonomous system (AS) is a group of routers managed by a particular organization. Exterior gateway protocols (EGP) are used between AS's. Internal Gateway Protocols (IGP) is used within an AS. The most common protocols used with TCP/IP are RIP (Routing Information protocol), OSPF (Open Shortest Path First), IGRP / Enhanced IGRP. The thesis is based on the theoretical background of OSPF protocol. OSPF uses flooding to exchange link -state updates between routers. These protocols are discussed in details in Section 2.x. Any change in routing information is flooded to all routers in the network. Areas are introduced to put a boundary on the explosion of link-state updates. Flooding and calculation of the Dijkstra algorithm on a router is limited to changes within an area. Routers that belong to multiple areas, called area border routers (ABR), have the duty of disseminating routing information or routing changes between areas.

Once such information is gathered it would be very difficult to clearly visualize and manipulate this router information visually since the output of such protocols is text based or in 2D figures. The aim was visualizing this kind of Router configuration information Visually using powerful tools and to be able to manipulate the figure generated and visualizing bottleneck paths in the router configurations. The Powerful features of Java 3D were utilized for Visualization. Also due to the platform independence, robustness, scalability Java was the choice for such a development since routers would be cross platform.

CHAPTER 2

ROUTERS CONFIGURATION AND PROTOCOLS

2.1 Problem Statement

The Internet grew from a small-interconnected set of systems to a huge explosion of points over the entire globe. These systems are connected to their network. The networks are connected to routers. The thesis generating a mathematical model for this connectivity and Visualize the same. The Visualization tool is capable of manipulating the structure and generating different connectivity thus working as a simulation tool for manipulating router connectivity. The Autonomous system is visualized as a Cylinder. The vertical lines represented the routers and the horizontal lines at each level originating at the vertical lines represented the connections between routers.

2.2 Routers and Routing Protocols

At the network layer, the internet can be viewed as a collection of subnet works or Autonomous Systems (AS) that are connected together. There is no fixed structure but several backbones exist. These are constructed from high-bandwidth lines and fast routers. The local area networks (LANs) at many universities, companies and ISPs are connected to this backbone. The Internet is held together by Internet Protocol (IP)

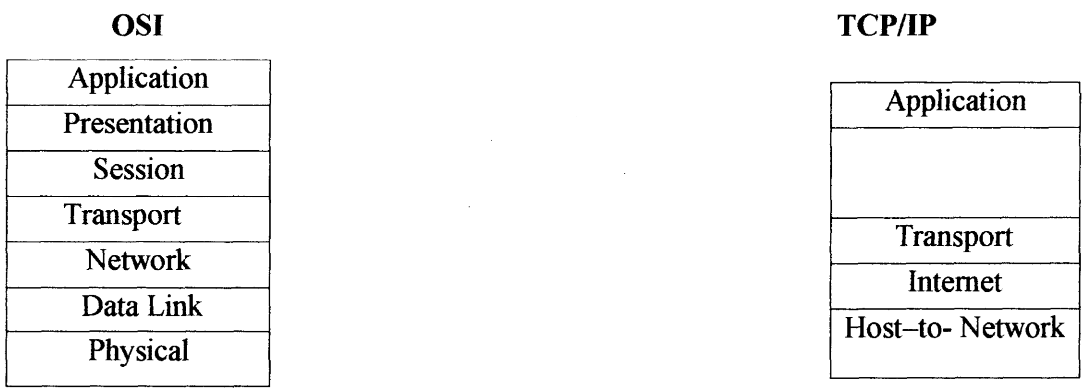


Figure 2.2.1 The OSI and TCP/IP Reference Model

The router is an intelligent element in the network because it analyzes the electric signals on a wire and translates them into meaningful data router's Network interface card can determine whether the packet is to be sent to the host. Routing is the act of moving information across a network from a source to a destination. Routing involves two basic activities: determining optimal routing paths and transporting information groups or packets through a network.

A metric is a standard of measurement, such as path length, that is used by routing algorithms to determine the optimal path to a destination. Routing algorithms initialize and maintain routing tables, which contain route information. Route information varies depending on the routing algorithm used. Routing algorithm in an AS is called as Interior Gateway Protocol (IGP), an algorithm for routing between ASes is called as Exterior Gateway Protocol (EGP).

2.3 Interior Gateway Routing Protocol OSPF

Open shortest Path First became a standard in 1990It has open literature .It supports a variety of distance metrics, including physical distance, delay and so on. I is a dynamic algorithm, which adapts to changes in the topology automatically and quickly. It is capable of routing real time traffic one way and other traffic a different way. I is also capable of doing load balancing, splitting the load over multiple lines by utilizing the second best route also. It also has a strong security protocol. OSPF works by abstracting the collection of actual networks, routers and lines into a directed graph in which each arc is assigned a cost (distance, delay, etc.). It computes the shortest path based on the weights of the arcs. A serial connection between two routers is represented by a pair of arcs, one in each direction. The weights may be different. A node for the networkitself plus a node for each router represent a multi-access network. The arcs from the network node to the routers have weight 0 and are omitted from the graph. What OSPF does is represent the actual network as a graph and then compute the shortest path from every router to every other router. OSPF allows large networks to be divided into numbered areas. These numbered area could be a network or a set of contiguous networks. Some routers may belong to no area and the areas do not overlap. Outside the area its details are not visible. Every AS is connected to a backbone called as Area 0 by tunnels. The data can travel from one Area to another through these tunnels. The tunnels are represented with arcs in the graph and have a cost. Each router connected to two or more areas is a part of the Backbone. Within an area each router has the same Link state Database and runs the same shortest path algorithm. The main task for each router is to find the shortest path from itself to each other router, including the router connected to the backbone. Separate metrics can be provided with the graphs for optimizing routes in terms of delay, throughput and reliability.

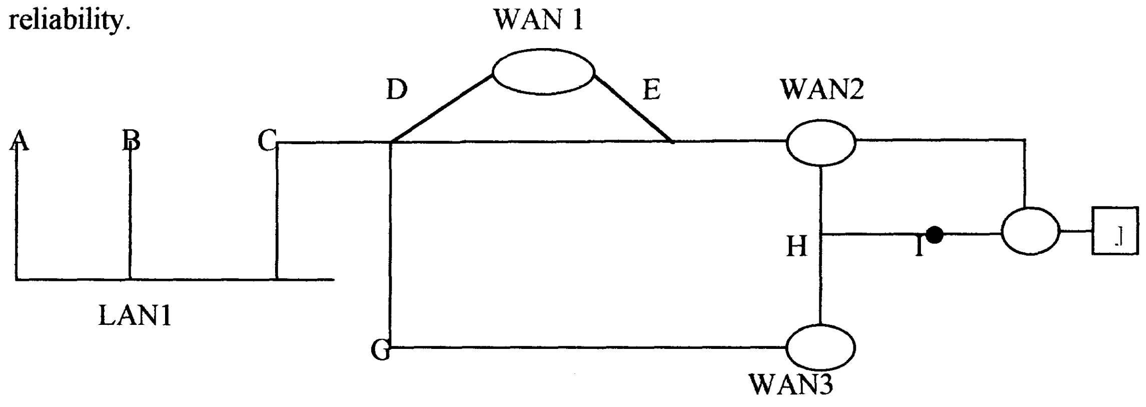


Figure 2.3.1 An Autonomous System

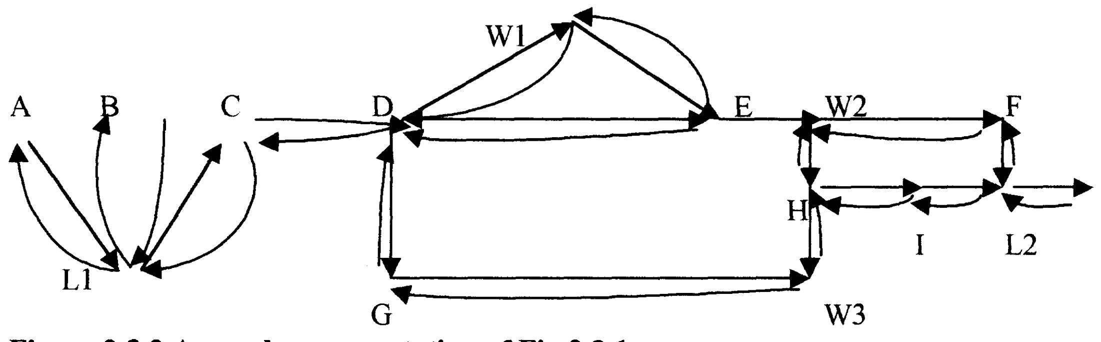


Figure 2.3.2 A graph representation of Fig 2.3.1

2.4 Graph Theory and Applications to Networks

Graph theory finds a lot of application in the real world. It has been widely used in the theory of electrical networks and several other applications. A graph is a series of vertices connected by segments called edges. One mathematical representation of a graph is through an adjacency matrix. If there are N edges, then an NxN matrix would be used to represent it. Each element of the matrix is either a 1 or a 0. 0 in entry (i, j) of the matrix means there is no direct edge connecting vertex i to vertex j. An entry of 1 means an edge exists between vertex i and j. In a directed graph an edge between vertex i and j does not necessarily mean that there is an edge between j and i .The adjacency matrix is not necessarily symmetric in such a case. The Peterson graph in section 2.5 is one such example.

The Gmatrix class of Java 3D API is a class to hold the adjacency matrix of the router network graph. The matrix is mapped onto a 3D figure of a Cylinder.The vertical lines represent the routers. N routers correspond to N Vertical lines. The horizontal lines are spaced across N levels on each vertical line representing that routers connectivity to other routers which is represented as a 1 in i^{th} row and j^{th} column.

2.5 Deriving Generalized Matrix for Graph Visualization

Due to the expanding nature of networks and the Internet in general it is important to have a dynamic structure representing the router configuration. Matrix Algebra has applications in many areas of mathematics, and especially in Graph Theory. A graph is a series of vertices connected by segments called edges. A matrix whose entries are the number of edges that connect one vertex to another may represent the term "graph". Such matrix is also known as an Adjacency Matrix.

The adjacency matrix is a representation of a directed graph with n vertices using an n × n matrix, where the entry at (i,j) is 1 if there is an edge from vertex i to vertex j; otherwise the entry is 0. A Weighted graph may be represented using the weight as the entry. An Undirected graph may be represented using the same entry in both (i,j) and (j,i) or using an Upper Triangular Matrix. Example (Graph figure Taken From http://www.cs.oberlin.edu/classes/dragn/labs/graphs/graphs33.html on 11/21/00 at 12:49 PM)

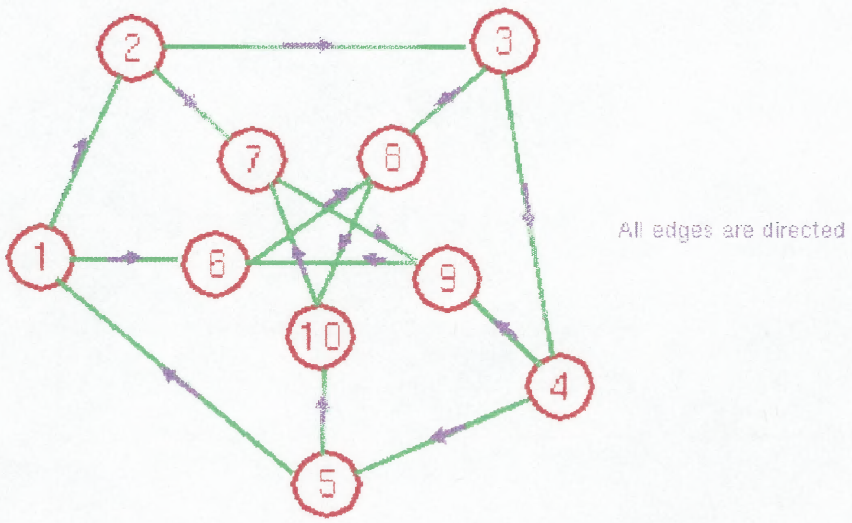


Figure 2.5.1 Peterson Graph

Adjacency Matrix:

1 : [2,6]
2 : [3,7]
3 : [4,8]
4 : [5,9]
5 : [1,10]
6 : [8]
7 : [9]
8 : [10]
9 : [6]
10: [7]

Generating the N x N GMatrix:

	1	2	3	4	5	6	7	8	9	10
1	0	1	0	0	0	1	0	0	0	0
2	0	0	1	0	0	0	1	0	0	0
3	0	0	0	1	0	0	0	1	0	0
4	0	0	0	0	1	0	0	0	1	0
5	1	0	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	1
9	0	0	0	0	0	1	0	0	0	0
10	0	0	0	0	0	0	1	0	0	0

Figure 2.5.2 Generating the N x N GMatrix

A Generalized matrix is created from Java 3D class GMatrix. This GMatrix is used to generate the 3 dimensional cylinder. This Matrix maps on to a 3D Cylinder. The vertical lines represent the routers. N routers correspond to N Vertical lines. The horizontal lines are spaced across N levels on each vertical line representing that routers connectivity to other routers.

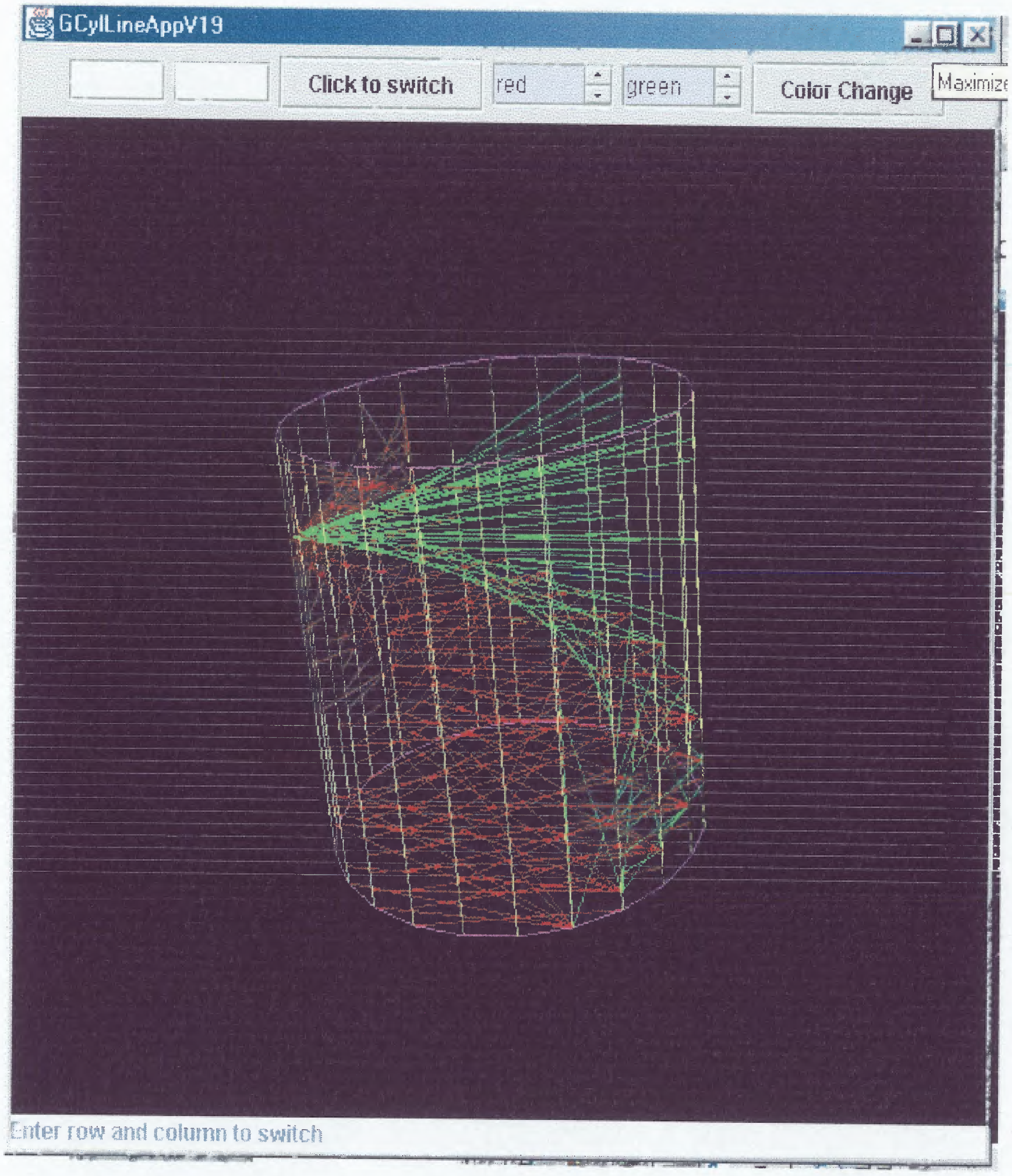


Figure 2.5.3 3D Representation as a cylinder

CHAPTER 3

PROGRAMMING ASPECTS FOR VISUALIZATION

3.1 Programming Aspects and Logic Flow

The program involves the following:

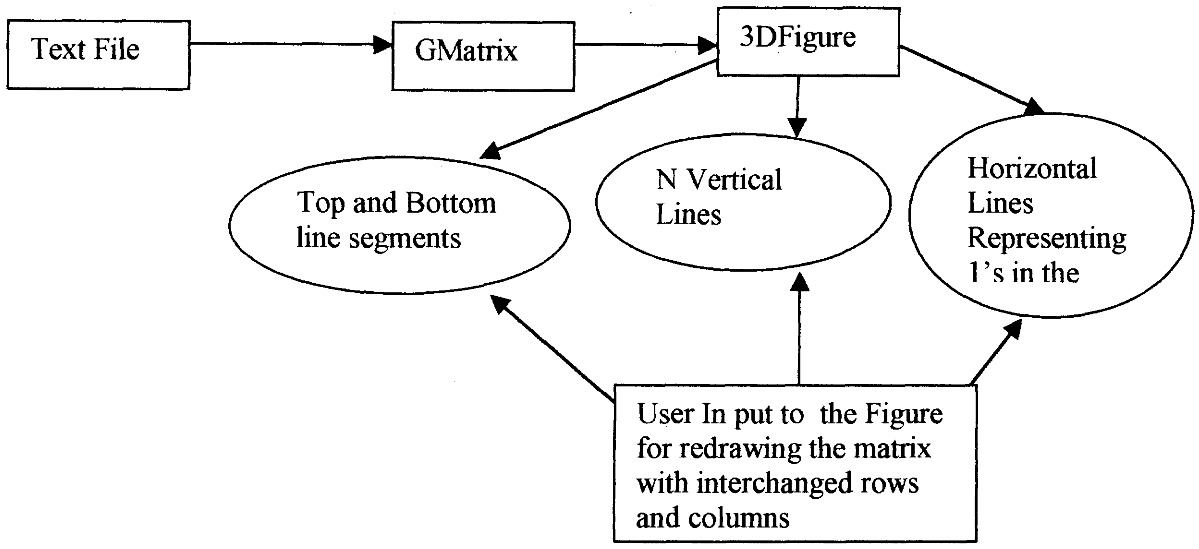

Text File
GMatrix
3DFigure
Top and Bottom line segments
N Vertical Lines
Horizontal Lines Representing 1's in the
User In put to the Figure for redrawing the matrix with interchanged rows and columns

Figure 3.2.1 Logic Flow of the Program

Reading the matrix from a text file .The format of the file is such that the first line of the file has a number which determines the dimension of the matrix which is a generalized matrix – which means the dimensions of the matrix are variable and can be decided at runtime. Generating a three dimensional figure for the matrix. The 3 D figure is broken into different parts Which are:

- Top and Bottom of the figure Which is made of N Lines
- N lines joining the top and Bottom vertices
- Horizontal lines at different levels on the vertical lines representing the 1's in the matrix.
- Colors to represent the incoming lines from various points at various levels on the vertical lines.

- Two text Fields to accept the seed from the user in terms of row numbers and column numbers, and a button to activate the drawing of the figure again with the interchanged rows and columns, for visualizing various configurations.
- Two input fields to accept the color combination; this is essential to bring out clarity of appearance and representation of the matrix.
- Various other features are attached to the 3D figure like:
    - Rotation of the 3D figure about an axis
    - Zoom in and Zoom out of the figure to observe the figure closely as the lines change position or to observe a particular intensity of lines at a particular spot.
- Various programming aspects are taken care of like Object Oriented programming using different classes for:
    - Class for top and Bottom lines
    - Class for Vertical Lines for N vertical lines between top and bottom vertices.
    - Class for horizontal lines to represent 1's
    - Class to generate the 3D Universe in which all these geometries are put together and various appearances and transformations and translations applied to it.
- Platform independence Using Java and Java 3D API: this program is capable of working on any platform provided it has the Java Virtual Machine for that particular platform.

CHAPTER 4

MATERIALS AND METHODS

4.1 Java 3D API and choice of Java 3D API

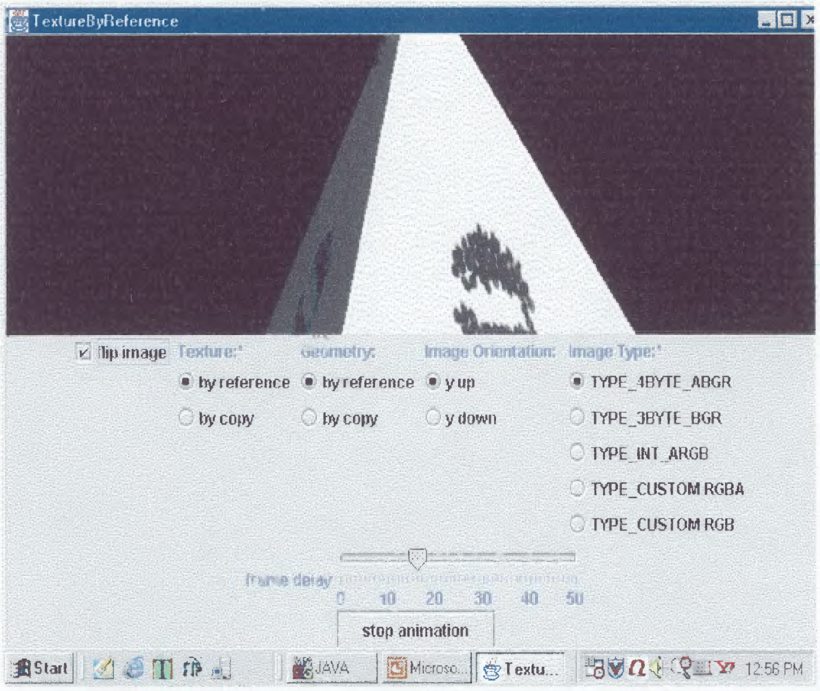
“ Java 3D is a network centric, scene-graph-based API for developing 3D Applets and Applications” source - An official Sun statement

Java 3D's scene graph architecture makes it easier than ever to write 3D programs. Java 3D is designed to work well across the network hence the relevance to this project for visualizing cross platform router configurations. Java itself has a very string scalability factor, Java 3D can run on everything from a laptop pc to a high-end workstation and even a supercomputer, and takes advantage of the hardware as it scales. It also scales across a range of viewing environments, from flat screens to fully immersive computer assisted virtual environments (CAVE) systems -- without rewriting code. Java 3D also supports the latest in virtual reality –sensor and display devices, including head mounted displays. Java 3D is considered to be a mid – to a high-level fourth generation 3D API. It is set apart from its predecessors by the Scene graph architecture for organizing graphical objects in the virtual 3D world. Scene graphs hide a lot of rendering details from the programmer along with several advanced features for more flexible and efficient rendering.

It is possible to add 3D graphics to Java applets and applications so that a whole Application can be created around this Visualization of routers and thus be very useful for real world applications utilizing the features of Java paradigm itself. The application would be platform independent provided the Java Virtual Machine is present on the system. Another reason for selecting Java 3D API for development is because of the level of detail, flexibility and Integration with other Java Applications.

This can be easily seen in the following examples taken from the Java web site at http://www.java.sun.com/ and http://www.java.sun.com/products/java-media/3D/collateral/

Examples Demonstrating Integration with other Java Features



[✓ flip image]

Texture: [✓ by reference] by copy

Geometry: [✓ by reference] by copy

Image Orientation: [✓ y up] y down

Image Type: [✓ TYPE_4BYTE_ABGR] TYPE_3BYTE_BGR TYPE_INT_ARGB TYPE_CUSTOM RGBA TYPE_CUSTOM RGB

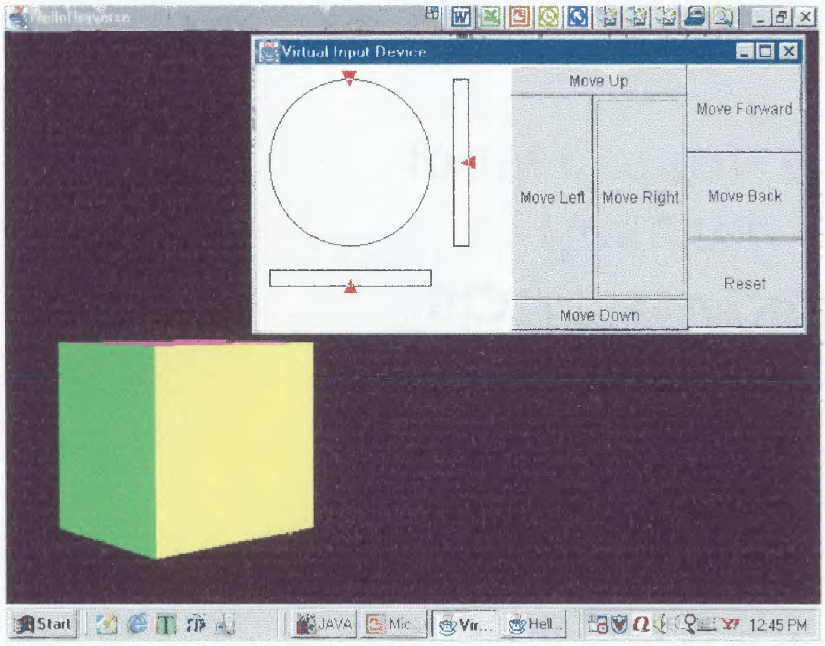
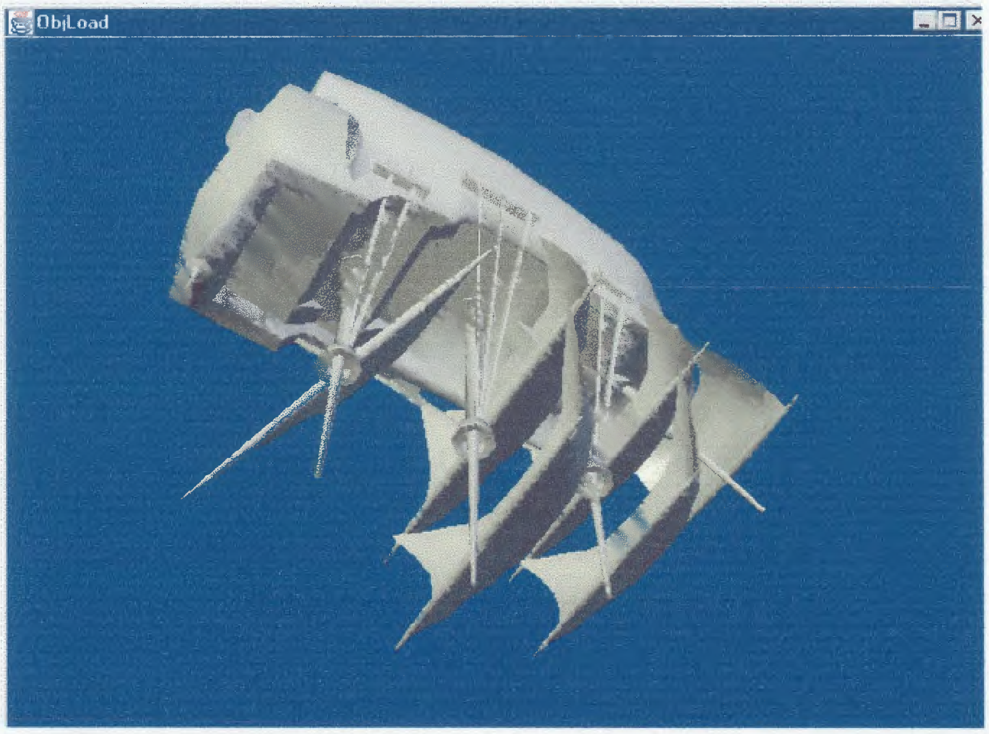


Fig 4.1.1 Example from www.java.sun.com demonstrating User Input

Examples demonstrating Java 3D's compatibility with other languages like VRML and the level of detail that can be manipulated.



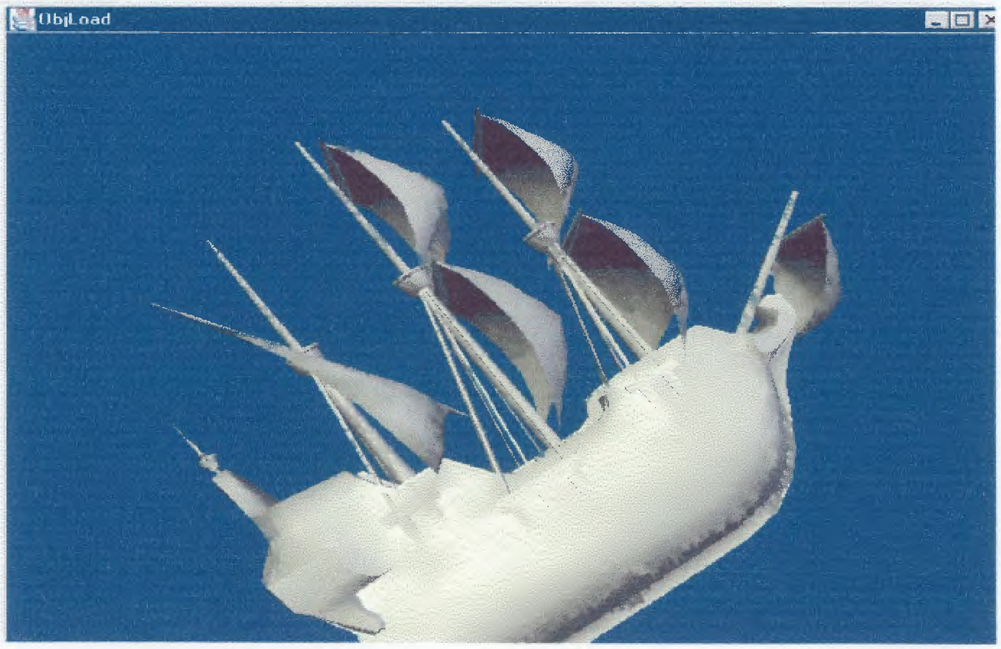


Fig 4.1.2 Example from www.java.sun.com demonstrating Compatibility with VRML and Level of Detail

The Java 3D core is found in the javax.media.j3d package. Java 3D relies heavily on the javax.vecmath and java.awt packages. The vecmath package provides support for point, vector and matrix data constructors and operations, while awt provides access to the java windowing toolkit. The most commonly used constructors are Point 2D, 3D like Point2d, Point 3d for double precision or Point2f, Point3f for float precision. There is also a external utility package which comes with the Java 3D core packages. The com.sun.j3d.utils is a higher-level package. It provides some 3D objects like box, sphere and operations like picking.

4.2 VISUALIZATION TECHNIQUES, 3D USER INTERFACE AND SCENE GRAPH

The *SceneGraph* is the backbone of Java 3D.In Java 3D the *VirtualUniverse* class defines the highest level of object aggregation. Everything that exists in a Java 3D scene must be attached to the *VirtualUniverse*. It could be visible objects or invisible elements like sound etc. There can be more than one *Universe* in an application but objects can exist only in one *Universe* at a time.

To help the precise placement of the objects in the universe Java3D provides a *Locale* class. A *Locale* provides a local Frame of spatial reference for the objects it contains. The *Locale* is positioned in the universe with high precision and the objects in the *Locale* are positioned with a lower precision. So a VirtualUniverse contains none, one or more *Locale* objects and a *Locale* contains one or more scene graphs. Scene Graphs hold the objects of the Universe and maintain the spatial relationship. The *VirtualUniverse* and the *Locale* provide the world coordinate space and the scene graph holds the objects that live in the virtual world. The SceneGraphObject class is an abstract class and has a child class called as *Node.* The SceneGraph is a tree like structure built from subclasses of node. I could be a G*roup* node or a L*eaf* node. A

Group node can have none, one or more children nodes, which could be *Group* or L*eaf* node. Although it can reference other objects, a L*eaf* node can have no children. The *Transform3D* class supports basic geometric operations like basic translations, rotations etc. It also provides access to raw matrix manipulations, which is utilized for creating complex geometric transformations by multiplying them.

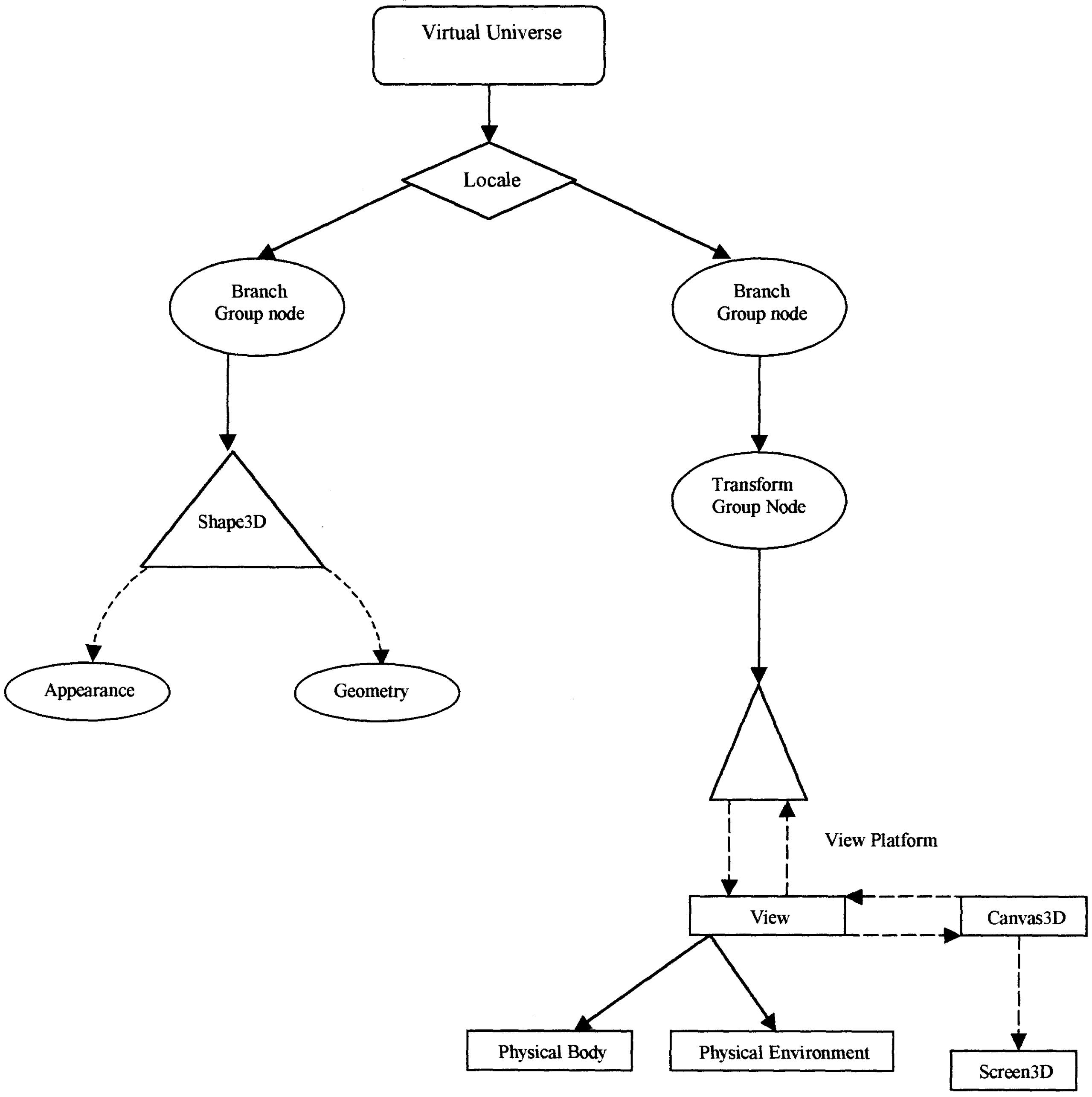


Fig 4.2.1 Example Scene Graph

4.3 Methods and Algorithms

Each *Locale* object in a *VirtualUniverse* establishes a virtual world Cartesian coordinate system. A *Locale* object serves as the reference point for the visual objects in a *VirtualUniverse*. With one *Locale* in a *SimpleUniverse* there is one coordinate system in the *VirtualUniverse*.The Coordinate system of Java 3D *VirtualUniverse* is right handed .The x-axis is positive to the right, y-axis is positive up. The z-axis is positive towards the viewer.

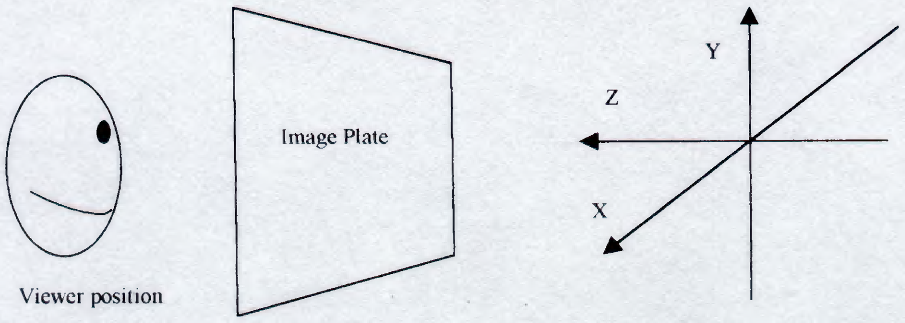


Fig 4.3.1 Orientation of axes in Virtual world

A Shape3D object defines the most common visual objects of a virtual universe. It is one of the subclasses of the Leaf class, therefore it can be a leaf object in a scene graph. However a shape object does not contain information about the shape or color of the visual object so it refers to one *Geometry* and one *Appearance* node component.

```
c = new Canvas3D (config);
// Create a simple scene and attach it to the virtual universe
BranchGroup scene = createSceneGraph ();
scene.setCapability( BranchGroup.ALLOW_BOUNDS_READ );
 SimpleUniverse u = new SimpleUniverse(c);
// This will move the ViewPlatform  so the objects in the scene can be viewed.
u.getViewingPlatform().setNominalViewingTransform();
u.addBranchGraph(scene);
```

User Defined function for adding nodes

Fig 4.3.2 Creation of a Scene Graph and attaching it to a virtual Universe

```
  public BranchGroup createSceneGraph() {
  // Create the root of the branch graph
BranchGroup objRoot = new BranchGroup();
objTransform = new TransformGroup();
objTransform.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
objTransform.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
objTransform.addChild(new GCyl());                    For creating Cylinder
objTransform.addChild(new Vert());                    For Vertical lines
gh=new Ghorizontal();                                 For horizontal lines
gh.setCapability(Shape3D.ALLOW_GEOMETRY_READ);
gh.setCapability(Shape3D.ALLOW_GEOMETRY_WRITE);
gh.setCapability(Shape3D.ALLOW_APPEARANCE_READ);
gh.setCapability(Shape3D.ALLOW_APPEARANCE_WRITE);
objTransform.addChild(gh);//new Ghorizontal());
objRoot.addChild(objTransform);                       For various transformations
      MouseRotate myMouseRotate = new MouseRotate();
      myMouseRotate.setTransformGroup(objTransform);
      myMouseRotate.setSchedulingBounds(new BoundingSphere());
      objRoot.addChild(myMouseRotate);
      MouseTranslate myMouseTranslate = new MouseTranslate();
      myMouseTranslate.setTransformGroup(objTransform);
      myMouseTranslate.setSchedulingBounds(new BoundingSphere());     Mouse Behaviors
      objRoot.addChild(myMouseTranslate);
      MouseZoom myMouseZoom = new MouseZoom();
      myMouseZoom.setTransformGroup(objTransform);
      myMouseZoom.setSchedulingBounds(new BoundingSphere());
      objRoot.addChild(myMouseZoom);

return objRoot;
} // end of CreateSceneGraph method of GCylLineapp
```

Fig 4.3.3 Creation of Branch Group to place into the Scene Graph

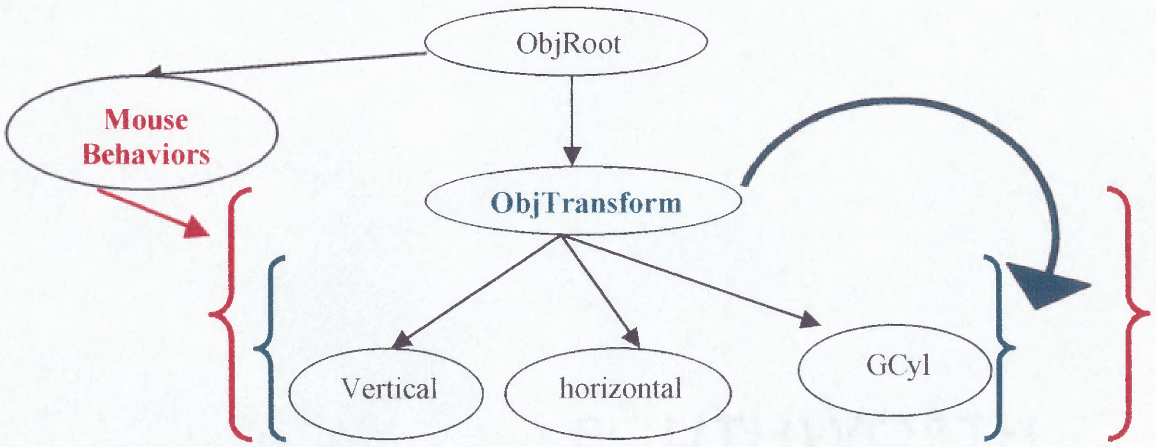


Fig 4.3.4 Router Configuration Scene graph (contained in Package CylPkg)

CHAPTER 5

CONCLUSIONS

5.1 Results and observations

During the entire research it was observed that currently the various companies have some basic 2 dimensional views of the data to be shown. The complex structure like Web sites or the router networks becomes very difficult to comprehend in such figures. It needs complex analysis and several resources to utilize such information. It was observed that once such data is Visualized in 3 Dimensions it becomes simplified even for a layman to understand the interconnections.

The most interesting aspect is when the frequency matrix is mapped onto the Cylinder, which gives a real world application to this thesis, by creating as tool for analyzing the network load.

A module was created to constantly update the matrix dynamically so that a simulation of network load could be performed and the cylinder constantly updated the data thus creating a very strong visualization factor to it. This could be very useful in detecting of security breaches where the number of horizontal lines going into a point increase dramatically could be an alarm for a network.

5.2 Manipulations after Visualization

The cylinder could be redrawn, after giving it a numeric field, which represents the row and column number. This could be used to simulate manipulation of a network for load balancing purposes. The research also involved applying this Visualization technique on web sites. The result in both cases, Router configurations and Web sites, were positive , i.e. in both cases manipulation was done much more efficiently due to the easier understanding of 3D figures and better representation.

APPENDIX A

CODE LISTING

```
/*

1.

This program draws two line strip arrays for the top and bottom of the cylinder in a class
GCyl which has a GCylGeometry function that returns the geometry of top and bottom of
the cyl.

2.

It has a class Vert which draws a line array between the top and bottom of the cylineder
It has s function called rVert which returns the geometry object with the line strip

3.

I has a class which Draws horizontal links from various vertical lines .

4.

The Scenegraph adds two children nodes with objects of these geometries.

5.

Reads matrix from a file and incorporates into the program to create the GMatrix

6.

Color Transformations incorporated, interaction for color etc. can be done though can be
modified for much higher levels of changes like check boxes, list, textboxes etc.

*/

import java.awt.*;
import java.awt.event.*;
import java.awt.event.WindowAdapter;
import java.awt.Color;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.BorderLayout;
import java.awt.Frame;
import javax.swing.*;
import javax.swing.event.*;
import java.applet.Applet;
import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.behaviors.mouse.*;
import javax.media.j3d.*;
import javax.vecmath.*;
import java.util.StringTokenizer;
import java.io.*;
import javax.vecmath.GMatrix;

/*
*
*Class for creating Standard Colors
*
*/
class Colors
{

public static final Color3f red     = new Color3f(1.0f,0.0f,0.0f);
public static final Color3f green   = new Color3f(0.0f,1.0f,0.0f);
public static final Color3f blue    = new Color3f(0.0f,0.0f,1.0f);
public static final Color3f yellow  = new Color3f(1.0f,1.0f,0.0f);
public static final Color3f cyan    = new Color3f(0.0f,1.0f,1.0f);
public static final Color3f magenta = new Color3f(1.0f,0.0f,1.0f);
public static final Color3f white   = new Color3f(1.0f,1.0f,1.0f);
public static final Color3f black   = new Color3f(0.0f,0.0f,0.0f);

}


public class CylPkg extends Applet implements ActionListener{
/*
*
*Variable Declaration
*
*/
int N;                          //VARIABLE TO HOLD DIMENSION OF MATRIX
n+1=N
Point3f topc[];
Point3f botc[];
TransformGroup objTransform;
static int objectCount;//for Standards button manipulation
JLabel text, clicked;
JButton button, clickButton,colButton,standards;
JPanel panel,fpanel1,f1;
```

```
JTextField t1,t2;
JList l1,l2;
char flag;
Canvas3D c;
GMatrix A;
Color3f col,col1,col2;
Ghorizontal gh;
private boolean _clickMeMode = true;

//frame for text boxes and matrix switching
public Color3f CArr[]=new
Color3f[]{Colors.red,Colors.green,Colors.blue,Colors.yellow,Colors.cyan,Colors.magent
a,Colors.white,Colors.black};

/*
*
*Class Constructor declaration
*
*/

public CylPkg() {

//Local Variables
double nbr;float percent=0.2f;

//Swing elements for user Input for Matrix switching and color manipulation
                setLayout(new BorderLayout());
                GraphicsConfiguration config =
                SimpleUniverse.getPreferredConfiguration();
                JPanel p = new JPanel();
                p.setLayout(new FlowLayout());
                text = new JLabel("Enter row and column to switch");
                button = new JButton("Click to switch");
                colButton=new JButton("Color Change");
                standards=new JButton("Standard Matrices");
                t1=new JTextField(5);
                t2=new JTextField(5);

//Variables for Color manipulation
                col1=new Color3f(1.0f,1.0f,1.0f);
                col2=new Color3f(1.0f,1.0f,1.0f);
                //declaring color constants
                l1=new JList(new String[]{"red","green","blue", "yellow","cyan",
                "magenta", "white","black"});
                l2=new JList(new String[]{"red","green","blue", "yellow", "cyan",
                "magenta","white","black"});
                l1.setSelectedIndex(0);
                l1.setVisibleRowCount(1);
                ListSelectionModel lsm = l1.getSelectionModel();
                lsm.setSelectionMode
                (ListSelectionModel.SINGLE_SELECTION);
                JScrollPane pane = new JScrollPane (l1);
                l2.setSelectedIndex(0);
                l2.setVisibleRowCount(1);
                ListSelectionModel lsm2 = l2.getSelectionModel();
                lsm.setSelectionMode(ListSelectionModel.SINGLE_SELECTION
                );
                JScrollPane pane2 = new JScrollPane (l2);

                //Adding Action Listeners
                button.addActionListener(this);
                colButton.addActionListener(this);
                standards.addActionListener(this);
                 p.add(t1);
                 p.add(t2);
                 p.add(button);
                 p.add(pane);
                 p.add(pane2);
                 p.add(colButton);
                 p.add(standards);
                 add("North", p);

                //adding Canvas
                 c = new Canvas3D(config);
                 add("Center",c);
                 add("South",text);
/*
*
*Creating GMatrix
*
*/

                FileRead f=new FileRead();
                A=new GMatrix(f.getGM());
                N=f.getN()+1;
                topc=new Point3f[N];
                botc=new Point3f[N];
                System.out.println("VALUE OF N IS "+N);
                System.out.println("lengths "+botc.length);
```

```
// Create a simple scene and attach it to the virtual universe

                        BranchGroup scene = createSceneGraph();
                        scene.setCapability( BranchGroup.ALLOW_BOUNDS_READ );
                        SimpleUniverse u = new SimpleUniverse(c);

// This will move the ViewPlatform back a bit so the objects in the scene can be viewed.

                        u.getViewingPlatform().setNominalViewingTransform();
                        u.addBranchGraph(scene);
                        System.out.println("obj count is"+objectCount);

}//end constr


 //Another constructor with string filename
 public CylPkg(String fileName) {

                       double nbr;float percent=0.2f;
                        setLayout(new BorderLayout());
                        GraphicsConfiguration config =
                        SimpleUniverse.getPreferredConfiguration();
                        JPanel p = new JPanel();
                        p.setLayout(new FlowLayout());
                        text = new JLabel("Enter row and column to switch");
                        button = new JButton("Click to switch");
                        colButton=new JButton("Color Change");
                        standards=new JButton("Standard Matrices");
                         t1=new JTextField(5);
                         t2=new JTextField(5);
                         col1=new Color3f(1.0f,1.0f,1.0f);
                          col2=new Color3f(1.0f,1.0f,1.0f);

//declaring color constants
l1=new JList(new String[]{"red","green", "blue","yellow","cyan", "magenta", "white",
"black" });
l2=new JList(new String[]{"red","green", "blue","yellow","cyan","magenta", "white",
"black" });
l1.setSelectedIndex(0);
l1.setVisibleRowCount(1);
ListSelectionModel lsm = l1.getSelectionModel();
lsm.setSelectionMode (ListSelectionModel.SINGLE_SELECTION);
JScrollPane pane = new JScrollPane (l1);
l2.setSelectedIndex(0);
l2.setVisibleRowCount(1);
ListSelectionModel lsm2 = l2.getSelectionModel();
```

```
lsm.setSelectionMode (ListSelectionModel.SINGLE_SELECTION);
JScrollPane pane2 = new JScrollPane (l2);
button.addActionListener(this);
colButton.addActionListener(this);
standards.addActionListener(this);
p.add(t1);
p.add(t2);
p.add(button);
p.add(pane);
p.add(pane2);
p.add(colButton);
add("North", p);
c = new Canvas3D(config);
add("Center",c);
add("South",text);

 //creating GMatrix
                    FileRead f=new FileRead(fileName);
                    A=new GMatrix(f.getGM());
                    N=f.getN()+1;
                    topc=new Point3f[N];
                    botc=new Point3f[N];

                    System.out.println("VALUE OF N IS "+N);
                    System.out.println("lengths "+botc.length);

                    // Create a simple scene and attach it to the virtual universe
                    BranchGroup scene = createSceneGraph();
                    scene.setCapability( BranchGroup.ALLOW_BOUNDS_READ );
                    SimpleUniverse u = new SimpleUniverse(c);

                   // This will move the ViewPlatform back a bit so the
                   // objects in the scene can be viewed.
                     u.getViewingPlatform().setNominalViewingTransform();

                      u.addBranchGraph(scene);

}//end constr

public BranchGroup createSceneGraph() {

               // Create the root of the branch graph
               BranchGroup objRoot = new BranchGroup();
               objTransform = new TransformGroup();
               objTransform.setCapability(TransformGroup.ALLOW_TRANSFORM_
               WRITE);
```

```
                objTransform.setCapability(TransformGroup.ALLOW_TRANSFORM_R
                EAD);
                objTransform.addChild(new GCyl());
                objTransform.addChild(new Vert());
                gh=new Ghorizontal();
                gh.setCapability(Shape3D.ALLOW_GEOMETRY_READ);
                gh.setCapability(Shape3D.ALLOW_GEOMETRY_WRITE);
                gh.setCapability(Shape3D.ALLOW_APPEARANCE_READ);
                gh.setCapability(Shape3D.ALLOW_APPEARANCE_WRITE);
                objTransform.addChild(gh);//new Ghorizontal());
                objRoot.addChild(objTransform);

                //Adding Mouse Behaviors
                MouseRotate myMouseRotate = new MouseRotate();
                myMouseRotate.setTransformGroup(objTransform);
                myMouseRotate.setSchedulingBounds(new BoundingSphere());
                objRoot.addChild(myMouseRotate);

                MouseTranslate myMouseTranslate = new MouseTranslate();
                myMouseTranslate.setTransformGroup(objTransform);
                myMouseTranslate.setSchedulingBounds(new BoundingSphere());
                objRoot.addChild(myMouseTranslate);

                //Zoom In / Zoom out
                MouseZoom myMouseZoom = new MouseZoom();
                myMouseZoom.setTransformGroup(objTransform);
                myMouseZoom.setSchedulingBounds(new BoundingSphere());
                objRoot.addChild(myMouseZoom);

                return objRoot;
   } // end of CreateSceneGraph method of GCylLineapp


public void actionPerformed(ActionEvent event)
{
              Object source = event.getSource();
              if (source.equals(button))
              {
                ///matrix transformations
                //show original matrix
               text.setText("Switching Matrix");
               //algorithm to switch matrix.
               double tmp;
               //extract variables from text boxes
               int n1,n2;
```

```
            n1=Integer.parseInt(t1.getText());
            n2=Integer.parseInt(t2.getText());
            text.setText("values are "+t1.getText()+t2.getText());
            try{
            for(int i=0;i<N-1;i++)
             {
                                        //switching row
                                        tmp=A.getElement(n1,i);
                                        A.setElement(n1,i,A.getElement(n2,i));
                                        A.setElement(n2,i,tmp);
                                        String str=A.getElement(i,n1)+"
                                        "+A.getElement(i,n2);
                                        System.out.println("row
                                         switch\t"+i+"\t"+A.getElement(i,n1)+"  " +
                                        A.getElement(i,n2));
                                        //col
                                        tmp=A.getElement(i,n1);
                                        A.setElement(i,n1,A.getElement(i,n2));
                                        A.setElement(i,n2,tmp);


                  }//for end

             //redrawing horizontal lines
             try{
                                        requestFocus();
                                        c.postSwap();
                                        gh.setGeometry(gh.rhorz());}
                                        catch(BadTransformException be)
                                       {
                                        System.out.println("BadTransformException in
                                        here");}
                  }//try end
             catch(ArrayIndexOutOfBoundsException e)
             {
             System.out.println("Array Out of Bounds somewhere");
             }//end catch
      }//if true condition end
      else
      {
        //////Color transformations
        if(source.equals(colButton))
        {
                                               float n1,n2,n3;
                                               int c11=l1.getSelectedIndex();;
                                               int c22=l2.getSelectedIndex();
                                        col1=CArr[c11];
                                        col2=CArr[c22];
                                        gh.setGeometry(gh.rhorz());
                                        }//end if
                                        else
                                        if(source.equals(standards))
                                        {
                                        final Frame frCyl = new Frame();
                                        frCyl.add(new Standards());
                                        frCyl.pack();
                                        frCyl.setVisible( true );
                                        frCyl.setSize( 700, 700 );
                                        }

                    }//end if

        }//end action performed
        /*
        *
        *Main Method
        *
        */
        // The following allows this to be run as an application  as well as an applet

          public static void main(String[] args) {
            Frame frame = new MainFrame(new CylPkg(), 556, 556);
          } // end of main method of CylPkg

        /*
        *
        *Class Ghorizontal
        *
        */
        public class Ghorizontal extends Shape3D{

        public Ghorizontal() {
        this.setGeometry(rhorz());
              } // end of GCyl constructor

        private Geometry rhorz()
        {

                        int NoLines=N;
                        LineArray tfa;
                        Point3f pinit[];
                        Point3f pend[];
                    int k=0;
                    double a,b;
                    float x,y;
                    float w=-0.4f;
                    float r=0.4f;
                    int ctlines=0;


ctlines=N*N;//increase ctlines irrespectiev of even / odd number
tfa=new LineArray (500*500,LineArray.COORDINATES|LineArray.COLOR_3);
pinit=new Point3f[2*ctlines];
pend=new Point3f[2*ctlines];
System.out.println("pend length = "+pend.length);
                         for (int i = 0; i < pinit.length; i++)
                        {
                        pinit[i] = new Point3f();
                        pend[i]=new Point3f();
                        }

                        float zp=botc[1].distance(topc[1]);
                        zp=zp/(N-2);
                        Point3f l22,l;int t=0;
                        for(int i=0;i<N-1;i++)
                        {
                        x=botc[i].x;
                        y=botc[i].y;
                        w=botc[1].z+zp*i;
                        for(int j=0;j<N-2;j++)
                        {
                        double q=A.getElement(i,j);

                        if((q==1.0)&&(i!=j))
                         {      if(i==0)
                        {
                                tfa.setCoordinate(k,botc[1]);
                                k++;
                                t=j+1;
                                tfa.setCoordinate(k,botc[t]);
                                k++;
                        }
                        else
                        {
                                if(j<12)
                                col=col1;//new Color3f(1.0f,0.0f,0.0f);
                                else
                                col=col2;//new Color3f(0.0f,1.0f,0.0f);
```

```
                                        l=new Point3f(x,y,w);
                                        tfa.setCoordinate(k,l);
                                        tfa.setColor(k,col);
                                        k++;
                                        t=j+1;
                                        x=botc[t].x;
                                        y=botc[t].y;
                                        l22=new Point3f(x,y,w);
                                        tfa.setCoordinate(k,l22);
                                        tfa.setColor(k,col);
                                        k++;
                                        x=botc[i+1].x;
                                        y=botc[i+1].y;
                                        }

                                }
                                }//end of for
                        }
return tfa;
} // end of method ghorz
}//end class

/*
*
*class Vert
*
*/

public class Vert extends Shape3D
{

        public Vert()
        {
                this.setGeometry(rVert());
                this.setAppearance(VertAppearance());
        } // end

private Appearance VertAppearance () {

          Appearance appearance = new Appearance();
          PolygonAttributes polyAttrib = new PolygonAttributes();

          //polyAttrib.setPolygonMode(PolygonAttributes.POLYGON_LINE);
          polyAttrib.setCullFace(PolygonAttributes.CULL_NONE);
```

```
            appearance.setPolygonAttributes(polyAttrib);
            PointAttributes pointAttrs = new PointAttributes();
            pointAttrs.setPointSize(3.0f);
           ColoringAttributes colorAttrs = new ColoringAttributes();
           colorAttrs.setColor(1.0f, 1.0f, 0.0f);
           appearance.setPointAttributes(pointAttrs);
           appearance.setColoringAttributes(colorAttrs);
return appearance;

} // end of method gCylAppearance of class GCyl


private Geometry rVert()
{

         LineArray vl=new LineArray (2*N,LineArray.COORDINATES);
         Point3f pbot[]=new Point3f[N];
         Point3f ptop[]=new Point3f[N];
        Point3f pbotc[]=new Point3f[N];
        LineStripArray tfa;

          int     totalN = 2*(N+1);
          Point3f coords[] = new Point3f[totalN];
          int     stripCounts[] ={ N+1, N+1};//, N+1};
          float   r = 0.4f;
          float   w = 0.4f;
          int     n;
          double  a;
          float   x, y;


          //initialise coords
         for (int i = 0; i < coords.length; i++)
          coords[i] = new Point3f();


//horizontal lines for top and bottom

          for(a = 0,n = 0; n < N; a = 2.0*Math.PI/(N-1)*++n ){
                x = (float) (r * Math.cos(a));
                y = (float) (r * Math.sin(a));
                coords[0*(N+1)+n+1] = new Point3f(x, y, w);
                coords[1*(N+1)+n] = new Point3f(x, y,-w);
        }
        coords[0]=coords[1];
```

```
tfa = new LineStripArray (totalN,LineStripArray.COORDINATES,stripCounts);
tfa.setCoordinates(0, coords);

//top and bottom coords
tfa.setCapability(Shape3D.ALLOW_GEOMETRY_READ);
int i,j;

for ( i = 0; i < topc.length; i++)
topc[i] = new Point3f();
tfa.getCoordinates(0,topc);

for (j = 0; j < botc.length; j++)
{
botc[j] = new Point3f();
}
tfa.getCoordinates(i,botc);

for (j = 0; j < pbotc.length; j++)
{
pbotc[j] = new Point3f();
}

for(i=0,j=0;i<topc.length;i++,j++)
{
vl.setCoordinate(j,topc[i]);
j++;
vl.setCoordinate(j,botc[i]);
}
System.out.println(botc.length+" lengths   "+ topc.length);
return vl;
}//end vert lines

}//end vert class

/*
*class GCyl
*/
 public class GCyl extends Shape3D{

        public GCyl() {
        this.setGeometry(gCylGeometry());
        this.setAppearance(gCylAppearance());
        } // end of GCyl constructor
```

```
private Geometry gCylGeometry() {

            LineStripArray tfa;
            int    totalN = 2*(N+1);
            Point3f coords[] = new Point3f[totalN];
            int    stripCounts[] ={ N+1, N+1};//, N+1};
            float  r = 0.4f;
            float  w = 0.4f;
            int    n;
            double  a;
            float  x, y;
            for (int i = 0; i < coords.length; i++)
                coords[i] = new Point3f();

//horizontal lines for top and bottom
tfa = new LineStripArray (totalN,LineStripArray.COORDINATES,stripCounts);

for(a = 0,n = 0; n <N+1; a = 2.0*Math.PI/(N-1)*++n )
        {
                                    x = (float) (r * Math.cos(a));
                                    y = (float) (r * Math.sin(a));
                                    coords[0*(N+1)+n] = new Point3f(x, y, w);
                                    coords[1*(N+1)+n] = new Point3f(x, y,-w);
        }

tfa.setCoordinates(0, coords);
return tfa;
} // end of method gCylGeometry in class GCyl

private Appearance gCylAppearance () {

                 Appearance appearance = new Appearance();
                 PolygonAttributes polyAttrib = new PolygonAttributes();
                 polyAttrib.setCullFace(PolygonAttributes.CULL_NONE);
                 appearance.setPolygonAttributes(polyAttrib);
                PointAttributes pointAttrs = new PointAttributes();
                pointAttrs.setPointSize(3.0f);
               ColoringAttributes colorAttrs = new ColoringAttributes();
               colorAttrs.setColor(1.0f, 0.0f, 1.0f);
               appearance.setPointAttributes(pointAttrs);
               appearance.setColoringAttributes(colorAttrs);
                return appearance;

        } // end of method gCylAppearance of class GCyl

  } // end of class GCyl
```

```
/*
*File raeding class
*/


public class FileRead {

        private  int x=0;
        private String s;
        private GMatrix gm;
        private int N;
        FileRead()
        {

                 //Read from file
                 try {
                  BufferedReader br=new BufferedReader(new
                  FileReader("Test.doc"));//mat.txt"));


                s=br.readLine();//for value of N
                System.out.println(s);
                N=Integer.parseInt(s);
                // creating GMatrix
                gm=new GMatrix(N,N);
                System.out.println("GMatrix created");

                try{
                        Thread.sleep(1000);
                }
                catch(InterruptedException ie)
                {
                        System.out.println("th intr");
                }


//READING DATA
for(int i=0;i<N;i++)
{

s=br.readLine();
StringTokenizer t=new StringTokenizer(s," ");
System.out.println("\n");
for(int j=0;j<N;j++)
{
x=Integer.parseInt(t.nextToken());
System.out.println(x);
gm.setElement(i,j,x);
x=0;
}


}

//displaying from GMatrix

System.out.println("DISPLAYING FROM GMATRIX");

try{
        Thread.sleep(1000);
}
catch(InterruptedException ie)
{
        System.out.println("th intr");
}

for(int i=0;i<N;i++)
for(int j=0;j<N;j++)
System.out.println(gm.getElement(i,j));
} catch(java.io.IOException e)
{
                System.out.println("Cannot read from text.txt");
}

}//end of constr

        GMatrix getGM()
        {
        return gm;
        }

int getN()
{
        return N;
}
```

```
//other constructor for string
FileRead(String fileName)
        {
         //Read from file
         try {
                BufferedReader br=new BufferedReader(new FileReader(fileName));
                s=br.readLine();//for value of N
                System.out.println(s);
                N=Integer.parseInt(s);
                // creating GMatrix
                gm=new GMatrix(N,N);
                System.out.println("GMatrix created");

                try{
                        Thread.sleep(1000);
                }
                catch(InterruptedException ie)
                {
                        System.out.println("th intr");
                }


                //READING DATA
                for(int i=0;i<N;i++)
                {

                s=br.readLine();
                StringTokenizer t=new StringTokenizer(s," ");
                System.out.println("\n");
                for(int j=0;j<N;j++)
                {
                x=Integer.parseInt(t.nextToken());
                System.out.println(x);
                gm.setElement(i,j,x);
                x=0;
                }


                }

                System.out.println("DISPLAYING FROM GMATRIX");

                try{
                        Thread.sleep(1000);
                }
                catch(InterruptedException ie)
                {
                        System.out.println("th intr");
                }

                for(int i=0;i<N;i++)
                for(int j=0;j<N;j++)
                System.out.println(gm.getElement(i,j));
                } catch(java.io.IOException e)
                {
                System.out.println("Cannot read from text.txt");
                }
        }//end of constr
}//end of class

} // end of class CylPkg

class Standards extends Applet implements ActionListener{
        // ,ListSelectionListener {


  JLabel text;
  JButton standards;
  JPanel panel;
  JList l1;//,l2;

public Standards()
{
        JPanel p = new JPanel();
        p.setLayout(new FlowLayout());
        text = new JLabel("Click on the Matrix Standard to see");
        standards=new JButton("Generate Matrices");
        //declaring color constants
        String[] str={"Band Diagonal","Block Trianular","Block
        TriDiagonal","SinglyBordered Block Diagonal","DoublyBordered Block
        Diagonal","SinglyBordered Block Trianular","Bordered Band Triangular","Singly
        Bordered band Diagonal","Doubly Bordered band
        Diagonal","Example1","Example2","Example3","Example4","Example5","ICIM
        S","Enikia"};
        l1=new JList(str);
        l1.setSelectedIndex(0);
        l1.setVisibleRowCount(1);
        ListSelectionModel lsm = l1.getSelectionModel();
        lsm.setSelectionMode
        (ListSelectionModel.SINGLE_SELECTION);
        JScrollPane pane = new JScrollPane (l1);
```

```
standards.addActionListener(this);
p.add(text);
p.add(l1);
p.add(pane);
p.add(standards);
add(p);
}//end constr

public void actionPerformed(ActionEvent event)
{
    Object source = event.getSource();
    if (source.equals(standards))
    {
            String fileName=new String();
            System.out.println("nothing");
            String matrixName=(String)l1.getSelectedValue();
            //code for matrix
            System.out.println("Matrix type selected is"+matrixName);
            int x=l1.getSelectedIndex();
            System.out.println("Matrix type selected is"+matrixName+"index
            is"+x);
            //Generating Figure according to the Standard matrix form
            System.out.println("Matrix type selected is"+ x);
                switch(x)
                {
                            case 0:System.out.println("Band Diagonal");
                            fileName="BandDiag.doc";
                            break;
            case 1:System.out.println("Block Triangular");
                            fileName="BlockTriangular.doc";
                            break;
            case 2:System.out.println("Block TriDiagonal");
                            fileName="BlocktriDiag.doc";
                            break;
            case 3:System.out.println("SinglyBordered Block Diagonal");
                            fileName="SinglyborBlkD.doc";
                            break;
            case 4:System.out.println("DoublyBordered Block Diagonal");
                            fileName="DoublyBordBD.doc";
                            break;
            case 5:System.out.println("SinglyBordered Block Trianular");
                            fileName="SBBlkTri.doc";
                            break;
            case 6:System.out.println("Bordered Band Triangular");
                            fileName="BorderedBandTr.doc";
                            break;
```

```
                    case 7:System.out.println("Singly Bordered band Diagonal");
                                    fileName="SinglyBorBDiag.doc";
                                    break;
                    case 8:System.out.println("Doubly Bordered band Diagonal");
                                    fileName="DoublyborBlkD.doc";
                                    break;
                    case 9:System.out.println("Example1");
                                    fileName="Example1.doc";
                                    break;
                    case 10:System.out.println("Example2");
                                    fileName="Example2.doc";
                                    break;
                    case 12:System.out.println("Example3");
                                    fileName="Example3.doc";
                                    break;
                    case 13:System.out.println("Example4");
                                    fileName="Example4.doc";
                                    break;
                    case 14:System.out.println("Example5");
                                    fileName="Example5.doc";
                    case 15:System.out.println("ICIMS");
                                    fileName="icims.txt";
                    case 16:System.out.println("Enikia");
                                    fileName="enikia.txt";
                                    break;
                    default:System.out.println("Select another choice");
                                    }//end switch

                                    final Frame frCyl = new Frame();
                                    frCyl.add(new CylPkg(fileName));
                                    frCyl.pack();
                                    frCyl.setVisible( true );
                                    frCyl.setSize( 500, 500 );
                                    frCyl.addWindowListener( new
                                    WindowAdapter()
                                    {
                                    public void windowClosed( WindowEvent e
)                                    {
                                     frCyl.dispose();
                                     }
                                    });
                    }//end action performed
}
} // end of class Standards
```

REFERENCES

[1] Andrew S. Tanenbaum, *Computer Networks*, Prentice Hall, India, 1999.

[2] Comer, D.E, *Internetworking with TCP/IP*, Vol.1, 3rd ed., Engelwood Cliffs , NJ:Prentice Hall 1995.

[3] Day, J.D, and Zimmerman, H.: "The OSI Reference Model", *Proc. Of the IEEE*, vol 71, pp.1334-1340, Dec.1983.

[4] McBryan, O.:"GENVL and WWWW: Tools of Taming the Web," *Proc. Cambridge Security WorkShop*, Springer-Verlag, pp. 1-17, 1994.

[5] Ford, L.R., Jr., and Fulkerson, *D.R.: Flows in Networks*, Princeton, NJ: Princeton University Press, 1962

[6] Ford, P.S., Rekhter, y., and Braun, H.-W.: "Improving the Routing and addressing of IP," *IEEE Network Magazine*, vol.7, pp. 10-15, May/June 1993.

[7] Barilleaux, Jon , *3D User Interfaces*, 1st ed,Greenwich,CT Manning 2000.

[8] Ammeraal, Leen, Computer Graphics for Java Programmers,West Sussex, England,Wiley & Sons 2000.

[9] RFC's 1131, 1245, 1253, 1583.

[10] http://archives.math.utk.edu/ICTCM/EP-10/C31/html/paper.html On 11/21/00 12:40 P.M.

[11] http://hissa.nist.gov/dads/HTML/adjcncymtrxr.html On 11/21/00 12:45 PM.

[12] http://www.cs.oberlin.edu/classes/dragn/labs/graphs/graphs33.html 11/21/00 12:49 P.M.

[13] http://www.ietf.org/rfc/rfc2328.txt 11/21/00 12:57 PM.

[14] http://www.cse.bris.ac.uk/comms/cccjp/mrtg-shark/cse-a-cpu.html

11/20/00 11:17 P.M.

[15] http://www.java.sun.com 12/4/2000 12.22 P.M.

[16] Bela, Bollobas, *Modern Graph Theory*, Springer-Verlag volume 184, 1998.