VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belgaum-590018.

Computer Graphics Mini Project On

"STAR TOPOLOGY"

Submitted in partial fulfillment for the requirements of the VI Semester degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

For The Academic Year
2021-2022
By

SAIF ALI

(1DB19CS127)

Under The Guidance Of
Prof. Shivakumar.Dalali
Associate Professor
Dept. of CSE, DBIT

Department of Computer Science and Engineering

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Mysore Road, Bengaluru - 560 074.

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Bengaluru – 560 074.

DEPARTMENT OF COMPUTER SCINECE AND ENGINEERING

<u>CERTIFICATE</u>

This is to certify that the mini project report entitled **"STAR TOPOLOGY"** is a bonafide work carried out by **SAIF ALI (1DB19CS127)** in partial fulfillment of award of Degree of **Bachelor of Engineering in Computer Science and Engineering** of Visvesvaraya Technological University, Belagavi, during the academic year 2021-2022. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated. The mini project has been approved as it satisfies the academic requirements associated with the degree mentioned.

Signature of guide	**Signature of HOD**
…………………	………………….
Dr.Shivakumar Dalali	**Dr K B Shiva Kumar**
Assoc.Professor	Head of Dept.,
Dept. of CSE,	Dept. of CSE,
DBIT, Bengaluru.	DBIT, Bengaluru

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Bangalore – 560 074.

DECLARATION

I, **SAIF ALI**, student of sixth semester B.E, Department of Computer Science and Engineering, Don Bosco Institute of Technology, Kumbalagodu, Bangalore, declare that the mini project work entitled **“STAR TOPOLOGY”** has been carried out by me and submitted in partial fulfillment of the course requirements for the award of degree in **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University, Belgaum** during the academic year **2021-2022**. The matter embodied in this reporthas not been submitted to any other university or institution for the award of any other degree or diploma.

Place: Bangalore **SAIF ALI**
Date: 15/07/2022 **1DB19CS127**

ACKNOWLEDGEMENT

Here by I am submitting the CG mini project report on **"STAR TOPOLOGY"**, as per the scheme of Visvesvaraya Technological University, Belgaum.

In this connection, I would like to express my deep sense of gratitude to my beloved institution Don Bosco Institute of Engineering and also, I like to express my sincere gratitude and indebtedness to **Prof. B.S Umashankar, Principal, DBIT, Bangalore.**

I would like to express my sincere gratitude to **Dr. K.B. Shiva Kumar**, **Head of Department of Computer Science and Engineering,** for providing a congenial environment to work in and carryout my mini project.

I would like to express the deepest sense of gratitude to thank my Project Guide **Assoc.Prof. Dr Shivakumar Dalali Assoc.Professor, Department of Computer Science and Engineering,DBIT, Bangalore** for his constant help and support extended towards me during the course of the project.

Finally, I am very much thankful to all the teaching and non-teaching members of the Department of Computer Science and Engineering, my seniors, friends and my parents for their constant encouragement, support and help throughout completion of mini project.

SAIF ALI
(1DB19CS127)

ABSTRACT

The main aim and objective of this was to plan and implement the knowledge gained by studying OpenGL, to create a simple and fully functional computer graphics project. The project "**STAR TOPOLOGY**" was developed after understanding the networking concepts of star topology network. This project uses OPENGL functions to demonstrate star topology. Here, we are illustrating transfer of messages from one system to another through a HUB, which connects all the systems.

LIST OF FIGURES

LIST OF ABBREVATIONS

- OpenGl: Open Graphics Library
- GLU: OpenGL Utility Library
- GLUT : OpenGL Utility Toolkit

CONTENTS

CHAPTER 1

INTRODUCTION

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

1.1. Aim

The aim of this project is to demonstrate the working of star topology network and demonstrate he transfer of message from one system to another in a star topology network

1.2. Objective

The objective of this project is to make the user understand the concept of how the congestion is handled in a communication system. The package must also have a user-friendly interface.

1.3 Scope

It is developed in OpenGL. It has been implemented on Visual Studio platform. The 3-D graphics package designed here provides an interface for the users for handling the display and selection menu. The Mouse is the main input device used.

Chapter 2

LITERATURE SURVEY

OpenGL, a graphics software system has become a widely accepted standard for developing graphics applications. Fortunately, OpenGL is easy to learn and it possesses most of the characteristics of other popular graphics system.

- Using OpenGL library, we can create various graphics related structure.
- Different functions defined in the OpenGL can be used to give different colors and textures to the surrounding objects.
- Glut (Toolkit of OpenGL library) can be used to create window, translate and rotate the matrix of the object, to track the position of mouse, to show the render screen etc.

Example:

- To create the window Function used: glutCreateWindow ("window name")
- To define the position of window Function used: glutInitWindowposition (int x, int y) X and Y are the co-ordinates of top left part of window in pixel.
- Define the size of window Function used: glutInitWindowsize (int x, int y) x and y are the width and height of the window.

Likewise, many user-defined functions are used to manipulate the translation of the train. The works are commented in the source code. In this way I gathered most of my knowledge from the previously made similar projects and got familiar with Open GL library and its toolkit.

CHAPTER 3

PROPOSED SYSTEM

In the proposed system, OpenGL is a graphic software system designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining userinput are included in OpenGL; instead, we must work through whatever windowing system controls the hardware you’re using.

Main aim of this project is to demonstrate the working of star topology network and demonstrate he transfer of message from one system to another in a star topology network

CHAPTER 4
ANALYSIS

4.1 SOFTWARE REQUIREMENTS:

Operating system	: Windows 10
Language	: C
Compiler	: GLUT GNU Compiler
OpenGL	: Library Files and dll Files
Editor	: Visual Studio 2019

4.2 Hardware Requirements:

Processor	**:**	Intel 486/Pentium processor and above
Processor speed	**:**	500MHz or above
RAM	**:**	64 MB or above
Storage space	**:**	2 MB or above
Monitor Resolution	**:**	color monitor with minimum resolution of 640*480

CHAPTER 5

PROJECT DESIGN

This chapter documents a detailed description of the implementation of our project. We have incorporated several inbuilt OpenGL functions in this project. The following code snippet enables the easy rendering of the graphs to be visualized in the form of bar graphs.

```
{
glClear(s. . .);
glVertex2f(. . .);
glColor3f(. . .);
glMatrixMode(. . .);
glLoadIdentity(. . .);
}
```

5.1 The header files used are:

1. **#include<stdio.h>**: This is a C library function for standard input and output.
2. **#include<math.h>**: This is a C library that contains mathematic functions.
3. **#include<GL/glut.h>**: This header is included to read the glut.h, gl.h and glu.h.
4. **#include<string.h>**: This is a C library function for performing operations on strings.

5.2 Functions used:

Init(); Here we initialize the color buffer, set the matrix mode and set window co-ordinate values.

setup(); This function is used to demonstrate the setting up of connection between client and server via TCP Handshaking.

teardown(); This function is used to demonstrate the teardown of the connection between the client and the server.

glutPostRedisplay(); It ensures that display will be drawn only once each time program goes through the event loop.

glutMainLoop(); This function whose execution will cause the program to begin an event processing loop.

CHAPTER 6

SYSTEM DESIGN

6.1 FLOWCHART/ARCHITECTURE

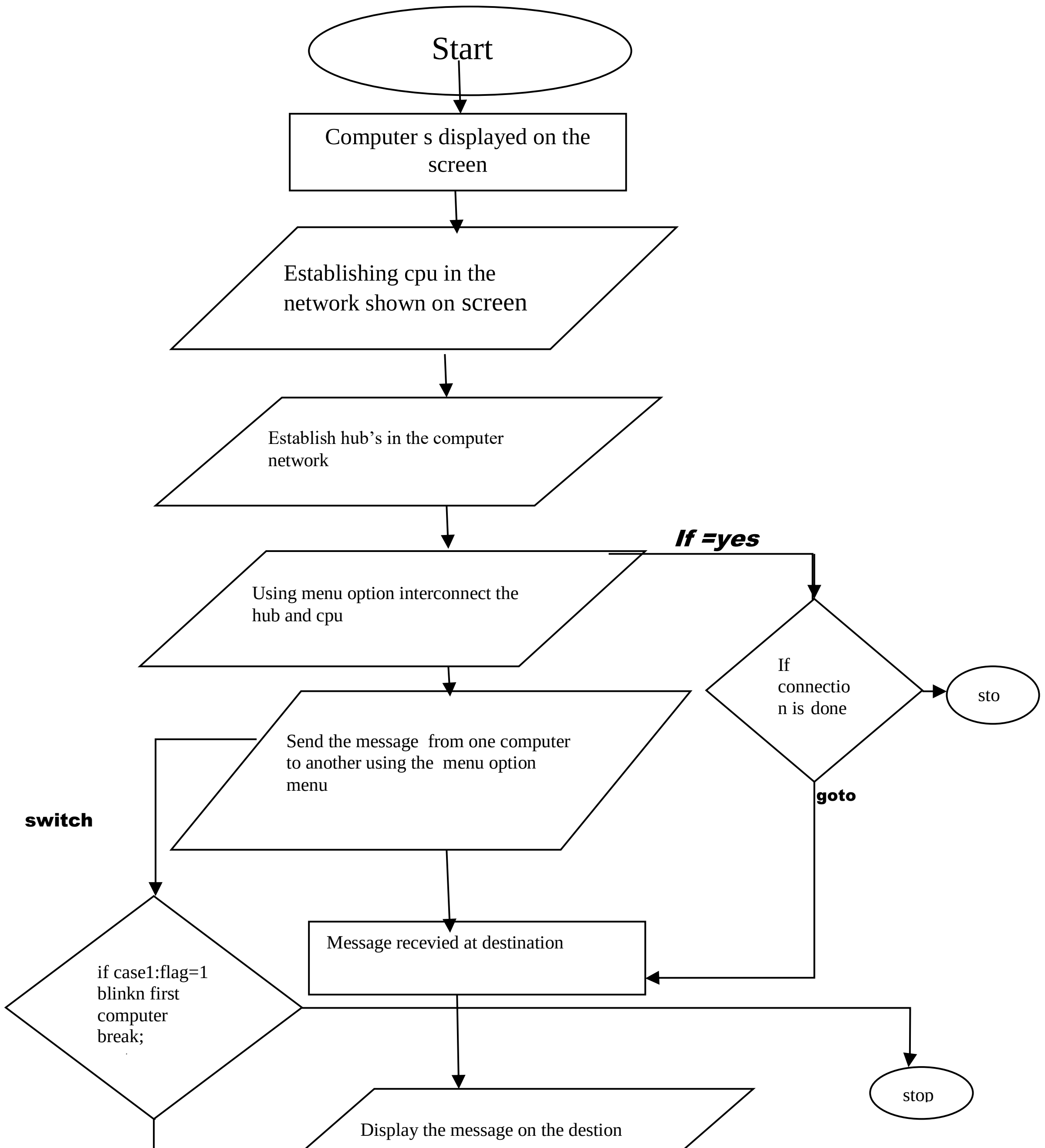
Start
Computer s displayed on the screen
Establishing cpu in the network shown on screen
Establish hub's in the computer network
If =yes
Using menu option interconnect the hub and cpu
If connectio n is done
sto
goto
Send the message from one computer to another using the menu option menu
switch
Message recevied at destination
if case1:flag=1 blinkn first computer break;
stop
Display the message on the destion screen
End

Fig6.1: Flowchart of the events in the project

6.2 CONCEPTS AND PRINCIPLES OF OPENGL

6.2.1 THE GRAPHICAL INTERFACE:

The application program has its own internal model. It draws the graphics using the facilities of the graphics system. The user views the graphics, and uses input devices, such asa mouse, to interact. Information about the users is sent back to the applications are sent back to the application, which decides what action to take. Typically, it will make changes to its internal model, which will cause the graphics to be updated, and so another loop in the interaction cycle begins.

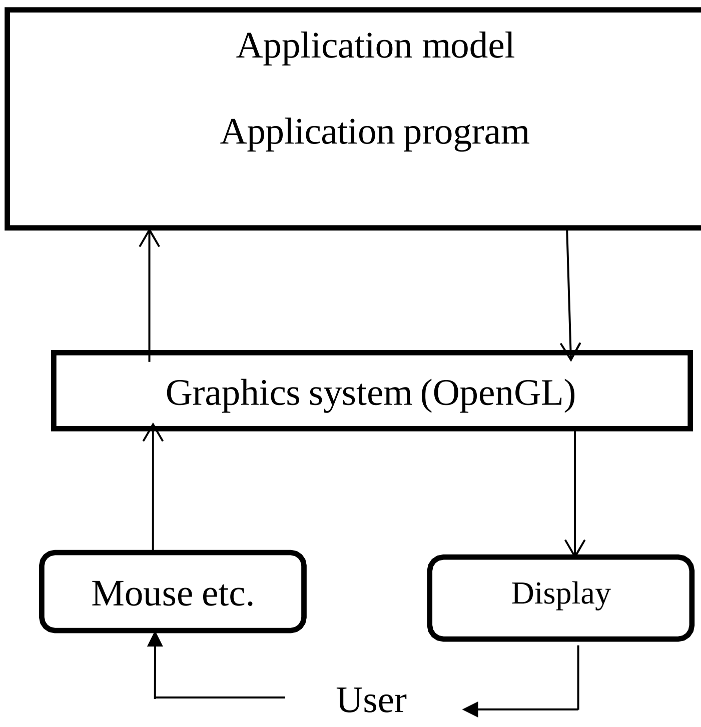

Fig 6.2: Graphical interaction

CHAPTER 7

IMPLEMENTATION

This program has been developed and implemented using OpenGL interface. Incorporate this facility includes glut.h header files.

4.1 Attributes

1. Void glColor3[b I f d ub us ui](TYPE r,TYPEg,TYPE b)

Sets the present RGB colors. Valid types are byte(b),int (i), float (f), double (d), unsigned byte(ub), unsigned short(us) and unsigned int (ui).

2. Void glClearColor(GLclampfg,Glclampfg,Glclampfb, Glclampfa)

Sets the present RGBA clear color used when clearing the color buffer. Variable of Glclampfare floating point numbers between 0.0 and1.0.

4.2 Working with the window system

1. Void glFlush()

Forces any buffered openGL commands to execute.

2. Void glutInit(int argc,char **argv)

Initializes GLUT. The arguments from main are passed in and can be used by the application.

3. Void glutCreateWindow(char*title)

Creates a window on the display. The string title can be used by the window. The return value provides reference to the windows that can be used when there are multiple windows.

4. Void glutInitDisplayMode(unsigned int mode)

Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model(GLUT_RGB, GLUT_INDEX) and buffering (GLUT_SINGLE,GLUT_DOUBLE).

5. void glutInitWindowSize(intwidth,int height)

Specifies the initial height and width of the window in pixels.

6. void glutInitWindowPosition(intx,int y)

Specifies the initial position of top-left corner of the window in pixels.

7. void glViewport(intx,inty,Glsizei height)

Specifies a width x height viewport in pixels whose lower left corner is at (x,y) measured from origin of the window.

8. void glutMainLoop()

Causes the program to enter an event processing loop. It should be the last statement in main.

9. Void glutDisplayFunc(void(*func)(void))

Requests that the display callback be executed after the current callback returns.

10. void glutSwapBuffers()

Swaps the front and back buffers.

4.3 Interaction

1. void glutReshapeFunc(void *f(intwidth,int height)

Registers the reshape callback function f. The callback function returns the height and width of the new window. The reshape callback invokes display callback.

4.4 Transformation

1. void glMatrixmode(Glenum mode)

Specifies which matrix will be affected by subsequent transactions. Mode can be GL_MODEL_VIEW,GL_PROJECTION or GL_TEXTURES.

2. Void glLoadIdentity()

Sets the current transformation matrix to an identity matrix.

3. void glPushMatrix() & void glPopMatrix()

Pushes to and pops from the matrix stack corresponding to the current matrix mode.

4. void glRotate[fd](TYPE x, TYPE y,TYPE z)

Alters the current matrix by a rotation of the angle degree about the axis(dx,dy,dz).TYPE is either Glfloat or Gldouble.

5. void glTranslate[fd](TYPE x, TYPE y,TYPE z)

Alters the current matrix by a displacement of(x,y,z). TYPE is of either Glfloat or Gldouble.

4.5 Viewing

1. void glOrtho2D(Gldoubleleft,Gldoubleright,Gldoublebottom,Gldouble top)

Defines a two dimensional viewing rectangle in the plane z=0.

CHAPTER 8

SOURCE CODE

```
#include<glut.h>
#include <string.h>
float dis=0.0,tr=0.0,tr1=0.0,tr2=0.0,tr3=0.0,tr4=0.0,tr5=0.0,tr6=0.0, tr7=0.0;
float tr8=0.0,tr9=0.0,tr10=0.0,tr11=0.0,tr12=0.0;
int cy_flag=0, cy_flag1=0,cy_flag2=0;
int flag=0,flag1=0,flag2=0,flag3=0,flag4=0,flag5=0;
int blink_flag=0,blink_flag2=0;
float dis1=0;
float dis2=0;
float cy_height_1=-0.1;
float cy_height_2=-0.1;
float cy_height_3=-0.1;
float cy_height_3_1=-0.1;
float cy_height_4=-0.1;
float con_blend=0.0;
int connec=0,hub1=0;
float Hub_R=0, Hub_G=4;

float c1r=-0.9,c1g=-0.9,c1b=0.9;
float c2r=-0.9,c2g=-0.9,c2b=0.9;
float c3r=-0.9,c3g=-0.9,c3b=0.9;
float c4r=-0.9,c4g=-0.9,c4b=0.9;

void anim();

void Cylinder(float cy_height)
{
        static GLUquadricObj *q;
        q = gluNewQuadric();
        gluQuadricNormals (q,GLU_TRUE);
        gluQuadricTexture (q,GLU_TRUE);
        gluCylinder (q, 0.4, 0.4, cy_height, 20, 20);
}

void computer(float scr, float scg, float scb)
{
/////////////////Screen/////////////////////
glScalef(1,1,0.3);

//glColor4f(-3.0,-3.0,-3,1);
glLineWidth(2);
//glColor4f(-0.9,-0.9,0.9,0.8);
glColor4f(scr-2,scg-2,scb-2,0.8);
glutSolidCube(9.7);
glScalef(1.4,1.4,1);
```

```
///////////////Border/////////////////
glColor4f(0.5,0.5,0.5,1.8);
glPushMatrix();//Bottom
        glTranslatef(0,-3.9,0);
        glScalef(2.8,0.3,1);
        glutSolidCube(3);
glPopMatrix();

glPushMatrix(); //Left
        glTranslatef(-3.9,-0.18,0);
        glScalef(0.3,2.8,1);
        glutSolidCube(3);
glPopMatrix();

glPushMatrix();//Top
        glTranslatef(-0.16,3.9,0);
        glScalef(2.8,0.3,1);
        glutSolidCube(3);
glPopMatrix();

glPushMatrix(); //Right
        glTranslatef(3.9,-0.0,0);
        glScalef(0.3,2.9,1);
        glutSolidCube(3);
glPopMatrix();


/////////////////Stand///////////////////////
glColor4f(0.1,0.1,0.1,1.8);
glPushMatrix();
        glTranslatef(0,-5.3,0);
        glScalef(1,0.5,1);
        glutSolidCube(3);
glPopMatrix();


//////////////////Keyboard//////////////////
glColor4f(-0.1,-0.1,-0.1,0.8);
glPushMatrix();
        glTranslatef(0.5,-7.3,0);
        glScalef(9,0.6,20.5);
        glutSolidCube(1);
        glColor4f(0.6,0.6,0.6,0.8);
        glutWireCube(1);
glPopMatrix();


//////////////////Keys//////////////////
glColor4f(4.9,4.9,4.9,1.8);
```

```
glBegin(GL_LINE_STRIP);
        glVertex3f(4.5,-7.5,0);
        glVertex3f(-3.0,-7.5,0);
        glVertex3f(-3.0,-7.5,-3);
        glVertex3f(4.5,-7.5,-3);
        glVertex3f(4.5,-7.5,-6);
        glVertex3f(-3.0,-7.5,-6);
        glVertex3f(-3.0,-7.5,-9);
        glVertex3f(4.5,-7.5,-9);
        glVertex3f(4.5,-7.5,-12);
        glVertex3f(-3.0,-7.5,-12);
        glVertex3f(-3.0,-7.5,0);
        glVertex3f(-2.0,-7.5,0);
        glVertex3f(-2.0,-7.5,-12);
        glVertex3f(-2.0,-7.5,-12);
        glVertex3f(-2.0,-7.5,0);
        glVertex3f(-1.0,-7.5,0);
        glVertex3f(-1.0,-7.5,-12);
        glVertex3f(-0.0,-7.5,-12);
        glVertex3f(-0.0,-7.5,0);
        glVertex3f(1.0,-7.5,0);
        glVertex3f(1.0,-7.5,-12);
        glVertex3f(2.0,-7.5,-12);
        glVertex3f(2.0,-7.5,0);
        glVertex3f(3.0,-7.5,0);
        glVertex3f(3.0,-7.5,-12);
        glVertex3f(4.0,-7.5,-12);
        glVertex3f(4.0,-7.5,0);
        glVertex3f(4.5,-7.5,0);
        glVertex3f(4.5,-7.5,-12);
glEnd();
}
void cpu()
{
        /////////////////////////CPU/////////////////////
        glColor4f(0.0,0.0,0.0,1.9);
        //cabinet
        glPushMatrix();
                glScalef(0.3,1,1);
                glTranslatef(-33,-3.3,0);
                glutSolidCube(15);
                glColor4f(1.5,0.0,0.0,1.9);
                glutWireCube(15);
        glPopMatrix();
        //power bottom outer
        glPushMatrix();
                glTranslatef(-9.7,-3.0,0);
                glColor4f(1.0,1.0,1.0,0.5);
                glutSolidSphere(0.7,20,35);
        glPopMatrix();
```

```
//power button inner
glPushMatrix();
        glTranslatef(-9.7,-3.0,0);
        glColor4f(0.5,0.5,0.5,1.8);
        glutSolidSphere(0.6,20,35);
glPopMatrix();
//cpu button (reset) outer
glPushMatrix();
                glTranslatef(-9.7,-5.0,0);
                glColor4f(1.0,1.5,1.5,0.5);
                glutSolidSphere(0.38,20,10);
        glPopMatrix();
// Button inner(Reset)
glPushMatrix();
        glTranslatef(-9.7,-5.0,0);
        glColor4f(0.5,0.5,0.5,1.2);
        glutSolidSphere(0.3,20,10);

glPopMatrix();
//CD rom
glPushMatrix();
        glScalef(0.7,0.2,1);
        glTranslatef(-14.0,14.3,0);
        glutSolidCube(5);
        glColor4f(1.0,1.0,1.0,0.2);
        glutWireCube(5);
glPopMatrix();
//CD open key
glPushMatrix();
        glScalef(0.3,0.1,1);
        glTranslatef(-28.0,17.3,0);
        glutSolidCube(2);
        glColor4f(3.5,3.5,3.5,1.0);
glPopMatrix();

//Usb ports
glPushMatrix();
        glScalef(0.3,0.15,1);
        glTranslatef(-28.0,-65.3,0);
        glColor4f(1.5,1.5,1.5,0.8);
        glutSolidCube(2);
glPopMatrix();

glPushMatrix();
        glScalef(0.3,0.15,1);
        glTranslatef(-33.0,-65.3,0);
        glutSolidCube(2);
        glColor4f(1.5,1.5,1.5,0.8);
glPopMatrix();
```

```
        glPushMatrix();
                glScalef(0.3,0.15,1);
                glTranslatef(-38.0,-65.3,0);
                glutSolidCube(2);
                glColor4f(1.5,1.5,1.5,0.8);
        glPopMatrix();


}

void connector()
{
        glPushMatrix();
        glRotatef(45,1,1,0);
        glColor4f(0.3,0.3,0.3,con_blend);
        glutSolidCube(2);
        glColor4f(10,10,10,con_blend);
        glutWireCube(2.0);
        glPopMatrix();
}

void line()
{
        //////////////First pipe////////////////////////
        glPushMatrix();
                glColor4f(0,0,0,0.5);
                glTranslatef(-18.5,10.5,0);
                glRotatef(40,1,0,0);
                Cylinder(cy_height_1);
        glPopMatrix();

        glPushMatrix();
                glColor4f(0,0,0,0.5);
                glTranslatef(-18.5,0.0,0);
                glRotatef(180,1,0,1);
                Cylinder(cy_height_1);
        glPopMatrix();

        ////////////////2nd  Pipe////////////////////
        glPushMatrix();
                glColor4f(0,0,0,0.5);
                glTranslatef(18.7,11.3,0);
                glRotatef(45,1,0,0);
                Cylinder(cy_height_4);
        glPopMatrix();

        glPushMatrix();
                glColor4f(0,0,0,0.5);
                glTranslatef(18.9,0.0,0);
                glRotatef(-90,0,1,0);
                Cylinder(cy_height_4);
        glPopMatrix();

///////////////Third Pipe////////////////////
        glPushMatrix();
                glColor4f(0,0,0,0.5);
                glTranslatef(-18.7,-14.5,0);
                glRotatef(15,1,0,0);
                Cylinder(cy_height_2);
        glPopMatrix();

        glPushMatrix();
                glColor4f(0,0,0,0.5);
                glTranslatef(-18.5,-20.5,0);
                glRotatef(180,1,0,1);
                Cylinder(cy_height_1);
        glPopMatrix();

        glPushMatrix();
                glColor4f(0,0,0,0.5);
                glTranslatef(-0.8,-20.5,0);
                glRotatef(-90,1,0,0);
                Cylinder(cy_height_3_1);
        glPopMatrix();



////////////Fourth Pipe//////////////////
        glPushMatrix();
        glColor4f(0,0,0,0.5);
        glTranslatef(18.6,-15.0,0);
        glRotatef(15,1,0,0);
        Cylinder(cy_height_3);
        glPopMatrix();

        glPushMatrix();
        glColor4f(0,0,0,0.5);
        glTranslatef(18.9,-20.7,0);
        glRotatef(-90,0,1,0);
        Cylinder(cy_height_4);
        glPopMatrix();

        glPushMatrix();
        glColor4f(0,0,0,0.5);
        glTranslatef(0.8,-20.5,0);
        glRotatef(-90,1,0,0);
        glPopMatrix();
        Cylinder(cy_height_3_1);


}
```

```
void hub(float scr, float scg, float scb)
{

        glColor4f(1,0.0,0,0.5);
        glPushMatrix();
                glScalef(1,0.5,0.5);
                glutSolidCube(6);
        glPopMatrix();

        glColor4f(0,0.0,0,1);
        glPushMatrix();
                glTranslatef(-2.3,0,0);
                glutSolidCube(0.5);
                glTranslatef(1.5,0,0);
                glutSolidCube(0.5);
                glTranslatef(1.5,0,0);
                glutSolidCube(0.5);
                glTranslatef(1.5,0,0);
                glutSolidCube(0.5);
        glPopMatrix();

        glColor4f(0.2,0.2,0.2,0.1);
        glPushMatrix();
                glScalef(0.9,0.47,0.5);
                glutSolidCube(7);
                glutWireCube(7);
        glPopMatrix();

        glColor4f(0.0,0.0,0.0,0.25);
        glPushMatrix();
                glScalef(4.5,0.5,0);
                glutSolidTetrahedron();
        glPopMatrix();

        glColor4f(0.0,0.0,0.0,0.2);
        glPushMatrix();
                glScalef(1.8,0.23,0);
                glTranslatef(0,8.4,0);
                glutSolidCube(3);
                glColor4f(0.7,0.7,0.7,0.2);
                glutWireCube(3);
        glPopMatrix();


        glColor4f(scr,scg,scb,1);
        glPushMatrix();
                glTranslatef(2.0,1.95,0);
                glColor4f(0.3,1.2,0.2,0.6);
                glutSolidSphere(0.3,20,5);
        glPopMatrix();
```

```
	glFlush();
}
void anim()
{
	//Blinking effect of source computer
		//if(tr2<=12)
		if(blink_flag%2!=0)
		{
			if(c1b<6)
				c1b+=0.5;
			if(c1b>=6)
				c1b=0;
		}
		if(tr2>12 || blink_flag%2==0){
			blink_flag=2;
			c1r=-0.9;c1g=-0.9;c1b=0.9; //Change color of source computer back to
original color
		}
	if(cy_flag==1)
	{
		if(tr<=12)
			tr+=0.07;
		if(tr>12 && tr1<=37.3)
			tr1+=0.07;
		if(tr1>37.2 && tr2<=12)
		{
			flag = 1;
			tr2+=0.07;
		}


		if(tr2>12)//Blinking effect of 2nd computer
		{
			if(c2b<5)
				c2b+=0.5;
			else if(c2b>=5)
				c2b=0;
			blink_flag=2;
		}
	}

	//data flow b/w 1st computer to 3rd computer
	if(cy_flag1==1)
	{
		if(tr3<=12)
			tr3+=0.07;
		if(tr3>12 && tr4<=17.75)
			tr4+=0.07;
		if(tr4>17.75 && tr5<=20.3)
		{
			flag = 1;
			tr5+=0.07;
		}
		if(tr5>20.3 && tr6<=17.9)
		{
			flag1=1;
			tr6+=0.07;
		}
		if(tr6>17.8 && tr7<=5.5)
		{
			flag2=1;
			tr7+=0.07;
		}

	//Blinking effect of source computer
		//if(tr6<=17.8)
		//{
		//	if(c1b<6)
		//		c1b+=0.5;
		//	if(c1b>=6)
		//		c1b=0;
		//}
		//else {
		//	c1r=-0.9;c1g=-0.9;c1b=0.9; //Change color of source computer back to
original color
		//}
		if(tr6>17.8)//Blinking effect of 3rd computer
		{
			if(c3b<6)
				c3b+=0.5;
			if(c3b>=6)
				c3b=0;
			if(tr6<17.9)
				blink_flag=2;
		}

	}

	//data movement 4m 1st to 4th
	if(cy_flag2==1)
	{
		if(tr8<=12)
			tr8+=0.07;
		if(tr8>12 && tr9<=19.5)
			tr9+=0.07;
		if(tr9>19.5 && tr10<=20.3)
		{
			flag3 = 1;
			tr10+=0.07;
		}
		if(tr10>20.3 && tr11<=17.9)
		{
```

```
                        flag4=1;
                        tr11+=0.07;
                }
                if(tr11>17.8 && tr12<=5.5)
                {
                        flag5=1;
                        tr12+=0.07;
                        if(tr11<17.9)
                                blink_flag=2;
                }

        ////Blinking effect of source computer
        //      if(tr6<=17.8)
        //      {
        //              if(c1b<6)
        //                      c1b+=0.5;
        //              if(c1b>=6)
        //                      c1b=0;
        //      }
        //      else {
        //              c1r=-0.9;c1g=-0.9;c1b=0.9; //Change color of source computer back to
original color
        //      }
                if(tr6>17.8)//Blinking effect of destination computer
                {
                        if(c3b<6)
                                c3b+=0.5;
                        if(c3b>=6)
                                c2b=0;
                }
        }

        if(cy_height_1<=16.5)
                cy_height_1+=0.1;
        if(cy_height_2<=20.0)
                cy_height_2+=0.1;
        if(cy_height_3<=20)
                cy_height_3+=0.1;
        if(cy_height_3_1<=21)
                cy_height_3_1+=0.1;
        //if(cy_height_3_1>21)
        //      //cy_flag=1;
        if(cy_height_4<=17)
                cy_height_4+=0.1;


        glutPostRedisplay();
}
void animate(int value)
{
        if(hub1==1)
```

```
	{
		if(dis<=27.0)
			dis+=0.2;

		else if(con_blend<=1.0)
			con_blend+=0.01;
	}
	glutPostRedisplay();
	glutTimerFunc(30,animate,0);
}
void text1(float a, float b, char *ptr,float r1, float g1, float b1)
{
	int len1=strlen(ptr);
	glColor3f(r1,g1,b1);
	glRasterPos2d(a,b);
	for(int i=0;i<len1;i++)
		glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10 ,ptr[i]);
}
void text3(float a, float b, char *ptr)
{
	int len1=strlen(ptr);
	glColor3f(8.9,2.0,1.5);
	glRasterPos2d(a,b);
	for(int i=0;i<len1;i++)
		glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12,ptr[i]);
}
void text4(float a, float b, char *ptr)
{
	int len1=strlen(ptr);
	glColor3f(4.9,2.0,5.5);
	glRasterPos2d(a,b);
	for(int i=0;i<len1;i++)
		glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24  ,ptr[i]);
}

void text2(float a, float b, char *ptr, float red, float green, float blue)
{
	int len1=strlen(ptr);
	glColor3f(red, green, blue);
	glRasterPos2d(a,b);
	for(int i=0;i<len1;i++)
		glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,ptr[i]);
}

void demo_menu(int id)
{
	switch(id)
	{
	case 1:
		break;
	case 4: connec=1;
```

```
                break;
        case 5: hub1=1;
                break;
        }
        glutPostRedisplay();
}

void option_menu(int id)
{
        switch(id)
        {
        case 1:
                        break;
        }
        glutPostRedisplay();
}

void computer1(int id)
{
        switch(id)
        {
        case 1:cy_flag=1;
                 blink_flag=1;
                break;
        case 2:cy_flag1=1;
                 blink_flag=3;
                        break;
        case 3:cy_flag2=1;
                 blink_flag=5;
                        break;
        }
        glutPostRedisplay();
}



void mymouse(int btn,int state,int x,int y)
{
        if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
        {
                        //glutIdleFunc(anim);
        }
        if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
        {

        }
        if(btn==GLUT_MIDDLE_BUTTON && state==GLUT_DOWN)
        {
                        exit(0);
        }
}
```

```
void packet1()
{

        glPushMatrix();
                glTranslatef(-15.6,19,0);
                glColor4f(0.9,1.5,0,1);
                glutSolidSphere(0.4,10,10);
        glPopMatrix();
}
void packet2()
{

        glPushMatrix();
                glTranslatef(-15.6,19,0);
                glColor4f(0.9,1.5,0,1);
                glutSolidSphere(0.4,10,10);
        glPopMatrix();
}
void packet3()
{

        glPushMatrix();
                glTranslatef(-15.6,19,0);
                glColor4f(0.9,1.5,0,1);
                glutSolidSphere(0.4,10,10);
        glPopMatrix();
}
void display(void)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        GLfloat amb[] = {0.4,0.3,0.3,0.5};
        GLfloat diff[] = {1.0,1.0,0.5,0.5};
        GLfloat spec[] = {1.0,1.0,1.0,0.5};
        GLfloat shi[] = {50.0};
        glMaterialfv(GL_FRONT, GL_AMBIENT, amb);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, diff);
        glMaterialfv(GL_FRONT, GL_SPECULAR, spec);
        glMaterialfv(GL_FRONT, GL_SHININESS, shi);

        GLfloat lightintensity[] = {1.0, 1.0, 1.0, 1.0};
        GLfloat lightposition[] = {50.0, 25.0, 15.7, 0.0};
        glLightfv(GL_LIGHT0, GL_POSITION, lightposition);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, lightintensity);

        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glEnable(GL_COLOR_MATERIAL);
        glShadeModel(GL_SMOOTH);
        //glEnable(GL_DEPTH_TEST);
        glEnable(GL_NORMALIZE);
```

```
	// 1st computer
	glPushMatrix();
		glPushMatrix();
			glScalef(0.6,0.6,0.6);
			glTranslatef(-21,28,0);
			//glRotatef(-20, 0.0, 1.0, 0.0);
			cpu();
		glPopMatrix();
	glScalef(0.6,0.6,0.6);
	glRotatef(30, 1, 1, 0.0);
	glTranslatef(-23,30,0);
	computer(c1r,c1g,c1b);			//these parameters show blinking effect and back
to normal position.

	glPopMatrix();



		//2nd Computer
	glPushMatrix();
		glPushMatrix();
			glScalef(0.6,0.6,0.6);
			glTranslatef(41,28,0);
			//glRotatef(20, 0, 0.1, 0.0);
			cpu();
		glPopMatrix();
	glScalef(0.6,0.6,0.6);
	glRotatef(30, 1.0, 1.0, 0.0);
	glTranslatef(20,28,0);
	computer(c2r,c2g,c2b);

	glPopMatrix();

	//3rd computer
	glPushMatrix();
		glPushMatrix();
			glScalef(0.6,0.6,0.6);
			glTranslatef(-21,-17,0);
			//glRotatef(20, 0, 0.1, 0.0);
			cpu();
		glPopMatrix();
	glScalef(0.6,0.6,0.6);
	glRotatef(30, 1.0, 1.0, 0.0);
	glTranslatef(-19,-17,0);
	computer(c3r,c3g,c3b);

	glPopMatrix();

	//4th Computer
	glPushMatrix();
		glPushMatrix();
```

```
                glScalef(0.6,0.6,0.6);
                glTranslatef(41,-16,0);
                //glRotatef(30, 1, 0, 1);
                cpu();
        glPopMatrix();
glScalef(0.6,0.6,0.6);
glRotatef(30, 1.0, 1.0, 0.0);
glTranslatef(22,-19,0);
computer(c4r,c4g,c4b);

glPopMatrix();
if(dis<27)
{
        text1(-13.9,17.5,"I WANT TO SHARE ",5,5,5);
        text1(-13.9,16.5,"SOME DATA WITH MY",5,5,5);
        text1(-13.9,15.5," NEIGHBOUR.",5,5,5);
        text1(-13.9,14.5," BUT HOW??",5,5,5);
}
else if(tr<11)
        text1(4,1,"HEY DONT PANIC, IM HERE TO ASSIST YOU.",-5,-5,-5);
else if(tr2<12)
        text2(-12.8,15.5,"Hello", 0.8,1,0);
else if(tr2>12)
        text2(11.8,15.9,"Hello",0.8,1,0);
//Hub Movement to the centre
glPushMatrix();
        glTranslatef(-27+dis,0,0);
                hub(Hub_R,Hub_G,0);
glPopMatrix();
if(dis>20)
{
        if(connec==1)
        {
                line();
                glutIdleFunc(anim);
        }
        glPushMatrix();
                glTranslatef(-18.5,0,0);
                connector();
        glPopMatrix();

        glPushMatrix();
                glTranslatef(18.5,0,0);
                connector();
        glPopMatrix();

        glPushMatrix();
                glTranslatef(-18.7,-20.5,0);
                connector();
        glPopMatrix();
```

```
		glPushMatrix();
			glTranslatef(18.7,-20.5,0);
			connector();
		glPopMatrix();

		glPushMatrix();
			glTranslatef(0,-20.5,0);
			glScalef(2,1,1);
			connector();
		glPopMatrix();

		glPushMatrix();
			if(tr<=12)
				glTranslatef(-3,-7-tr,1);
			else if(tr>12)
				glTranslatef(-3+tr1,-7-tr,1);
			if(flag==1)
				glTranslatef(0,tr2,0);
			packet1();
		glPopMatrix();

		glPushMatrix();
			if(tr3<=12)
				glTranslatef(-3,-7-tr3,1);
			else if(tr3>12)
				glTranslatef(-3+tr4,-7-tr3,1);
			if(flag==1)
				glTranslatef(0,0-tr5,0);
			if(flag1==1)
				glTranslatef(0-tr6,0,0);
			if(flag2==1)
				glTranslatef(0,0+tr7,0);
			packet2();
		glPopMatrix();

		//data movement from 1st to 4th
		glPushMatrix();
			if(tr8<=12)
				glTranslatef(-3,-7-tr8,1);
			else if(tr8>12)
				glTranslatef(-3+tr9,-7-tr8,1);
			if(flag3==1)
				glTranslatef(0,0-tr10,0);
			if(flag4==1)
				glTranslatef(0+tr11,0,0);
			if(flag5==1)
				glTranslatef(0,0+tr12,0);
			packet3();
		glPopMatrix();

	}
```

```
        text3(-19.4,16,"APPLE");
        text3(17.9,16.2,"APPLE");
        text3(-19.4,-10.5,"APPLE");
        text3(17.9,-10,"APPLE");
        text4(-4, 20, "STAR TOPOLOGY");


        glutSwapBuffers();
        glFlush();
}

void mypoint()
{
        glClearColor(0.5 ,0.9,0.7,0.5);
        glOrtho(-23.0,23.0,-23.0,23.0,-43.0,43.0);
        glEnable(GL_BLEND);
        glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
}
int main(int argc,char *argv)
{
        glutInit(&argc,&argv);
        glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
        glutCreateWindow("STAR TOPOLOGY");
        mypoint();
        glutDisplayFunc(display);

        int     sub_menu_1 = glutCreateMenu(computer1);
        glutAddMenuEntry("To computer 2",1);
        glutAddMenuEntry("To computer 3",2);
        glutAddMenuEntry("To computer 4",3);

        int     sub_menu = glutCreateMenu(option_menu);
        glutAddSubMenu("From computer 1",sub_menu_1);

        /*glutAddMenuEntry("From computer 2",2);
        glutAddMenuEntry("From computer 3",3);
        glutAddMenuEntry("From computer 4",4);*/

        glutCreateMenu(demo_menu);
        glutAddMenuEntry("HUB",5);
        glutAddMenuEntry("CONNECTION",4);
        glutAddSubMenu("SEND MESSAGE",sub_menu);

        glutAttachMenu(GLUT_RIGHT_BUTTON);
        glutMouseFunc(mymouse);
        anim();
        animate(0);
        glutFullScreen();
        glutMainLoop();
        return(0);}
```

CHAPTER 9

SNAPSHOTS

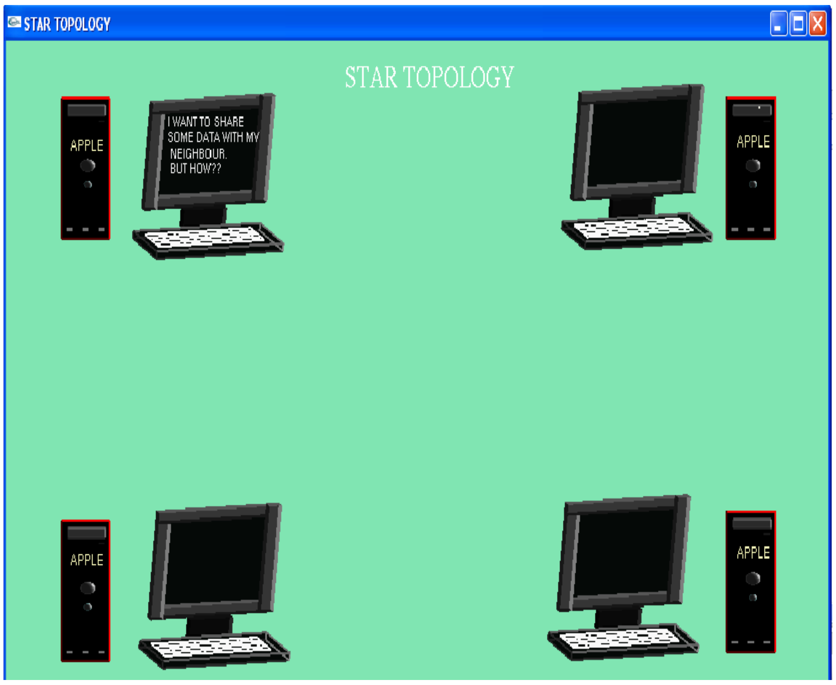

Fig 9.1 Display screen with Computer and cpu

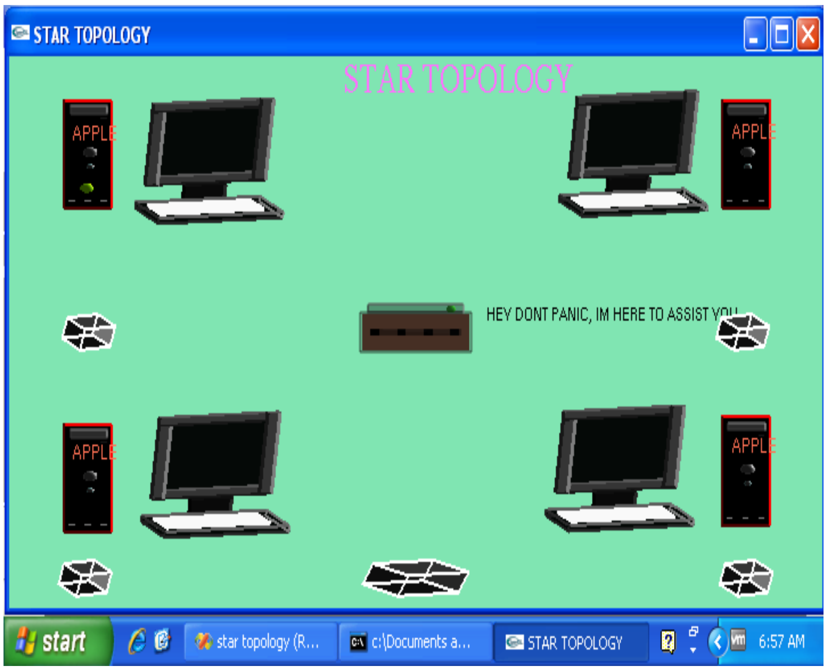

Fig 9.2: Display of Hub

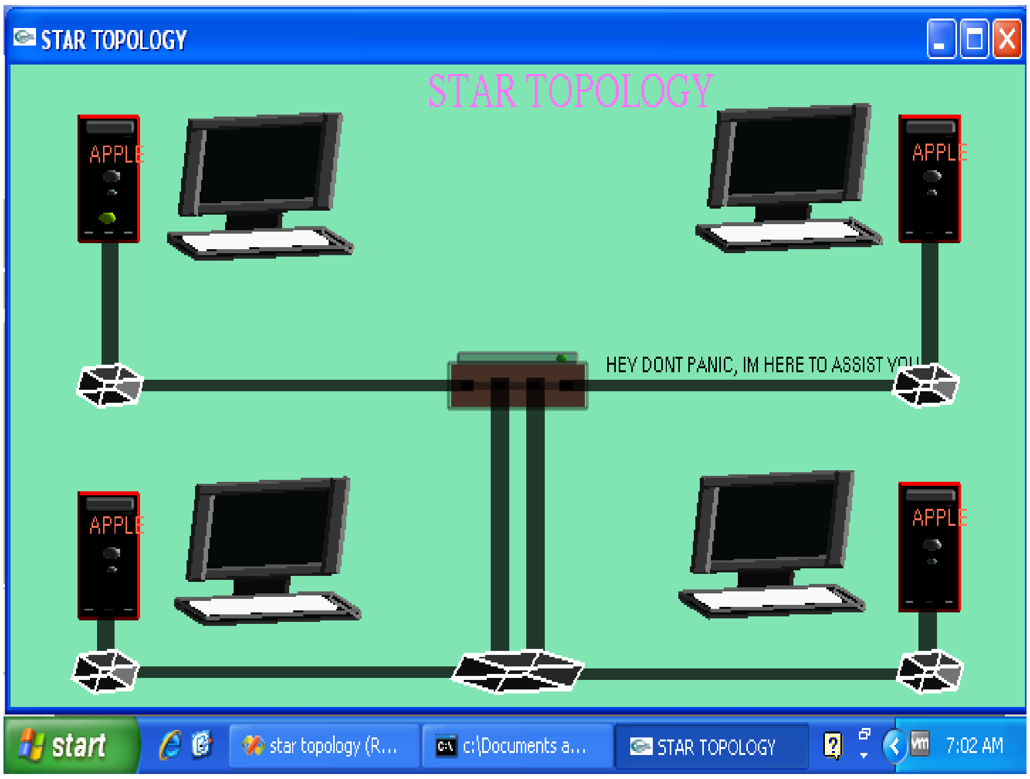

Fig 9.3 LINK ESTABLISHMENT B/W CPU AND HUB

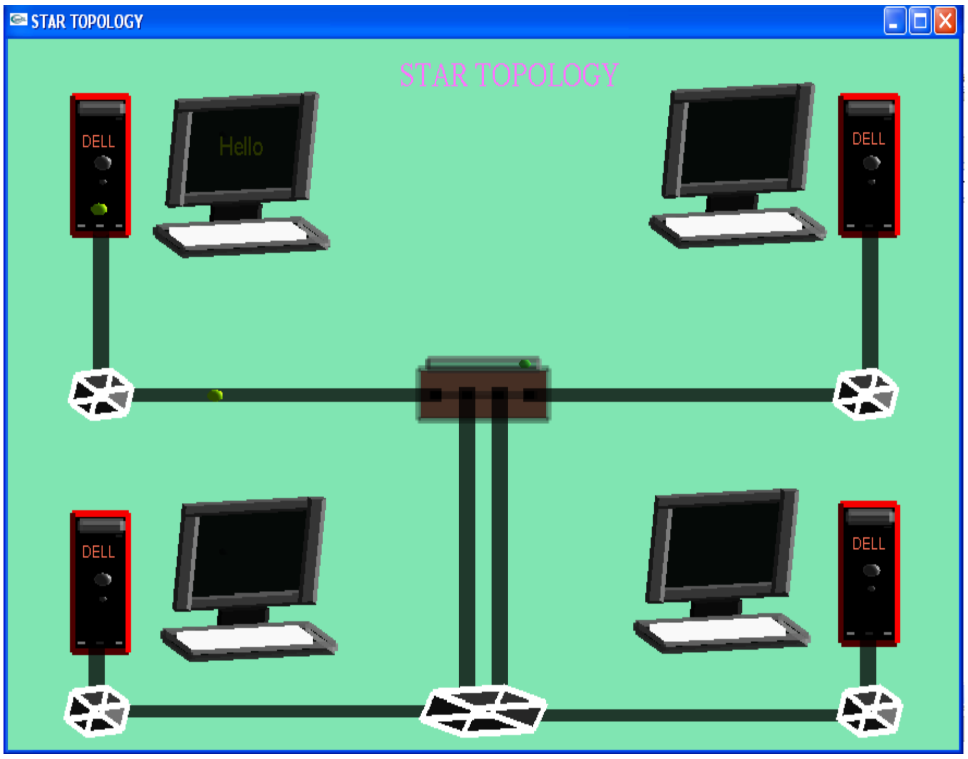

Fig 9.3 SENDING AND RECEIVING MESSAGE

CHAPTER 10
CONCLUSION

An attempt has been made to develop an openGL which meets necessary requirements of the user successfully. Since it is user friendly it enables user to interact efficiently and easily.

The development of mini project has given us a good exposure to openGL by which we have learnt some of the techniques which helps in the development of animation .

Hence it is helpful for us to even take this field as our career too and develop some other features in openGL and provide as token of contribution to the graphics world.

FUTURE ENHANCEMENT:

This project has been designed using C++, which works on the windows platform.The project can be designed using other languages and better graphical interfaces. The following features could have been incorporated.

- ✓ Circular and priority queues can be implemented.
- ✓ The project could have been entirely in 3D instead of 2D

REFERENCES

[1] Edward Angel’s Interactive Computer Graphics Pearson Education 5th Edition

[2] Interactive computer Graphics - A top down approach using open GL by Edward Angel

[3] Jackie L.Neider,Mark Warhol, Tom R.Davis,"OpenGL Red Book", Second Revised Edition,2005.

[4] Donald D Hearn and M.PaulineBaker,"Computer Graphics with OpenGL",3rd Edition.

WEBSITES

[5] www.OpenGLRedbook.com

[6] www.OpenGLsimpleexamples.com

[7] www.OpenGLprogramminguide.com

[8] www.basic4glwikispaces.com

[9] www.opengl.org