VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A Computer Graphics & Visualization Mini Project Report
on

"ROUTER ARCHITECTURE"

Submitted in Partial fulfillment of the Requirements for VI Semester of the Degree of

Bachelor of Engineering
In
Computer Science & Engineering
By

DIVYA T
(1CR15CS058)

Under the Guidance of

Mr. Kiran Babu
Assoc Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,
BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Computer Graphics & visualization project work entitled **"Router Architecture"** has been carried out by **Divya T(1CR15CS058)** bonafide students of CMR Institute of Technology in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2017-2018**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. This CG project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

-----------------------	**----------------------**
Signature of Guide	**Signature of HOD**
Mr. Kiran Babu	**Dr. Jhansi Rani P**
Asso Prof.	**Professor& Head**
Dept. of CSE, CMRIT	**Dept. of CSE, CMRIT**

External Viva

Name of the examiners	Signature with date
1.	
2.	

ABSTRACT

OpenGL provides a set of commands to render a three dimensional scene. That means you provide them in an Open GL-useable form and Open GL will show this data on the screen (render it). It is developed by many companies and it is free to use. You can develop Open GL-applications without licensing.

Open GL is a hardware- and system dependent interface. An Open GL-application will work on every platform, as long as there is an installed implementation, because it is system independent, there are no functions to create windows etc., but there are helper functions for each platform. A very useful thing is GLUT.

This project includes the concepts of transformation, motion in objects. I have tried to create a simple code called "ROUTER ARCHITECTURE".

- The router consists of Input port, output port, routing processor and switch fabric. Control packets (e.g., packets carrying **routing** protocol information such as RIP, OSPF or IGMP) are forwarded from the input port to the **routing** processor and then to the output port.
- The Switch fabric Works as a bridge through which packets are delivered between line cards.

ACKNOWLEDGEMENT

Behind every success there is a master hand. A master hand will create unperturbed concentration, dedication and encouragement in everything good and bad, without whose blessing this would have never come into existence.

Firstly, I thank God for showering the blessings on me. I am grateful to my institution CMRIT for providing me a congenial atmosphere to carry out the project successfully.

I would like to express my heartfelt gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore, for extending his support.

I am highly thankful to **Dr.Jhansi Rani,** HOD of Computer Science and Engineering, CMRIT, Bangalore for her support and encouragement given to carry out the project.

I am very grateful to my guide, **Mr.Kiran Babu,** Assoc Professor, Department of Computer Science, for his able guidance and valuable advice at early stage of my project which helped me in successful completion of my project.

Finally, I would like to thank my parents and friends who helped me with the content of this report, without which the project would not have become a reality.

DIVYA T (1CR15CS058)

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Computer Graphics

- Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly.
- Computers have become a powerful medium for the rapid and economical production of pictures.
- Graphics provide a so natural means of communicating with the computer that they have become widespread.
- Interactive graphics is the most important means of producing pictures since the invention of photography and television.
- We can make pictures of not only the real world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry.
- A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

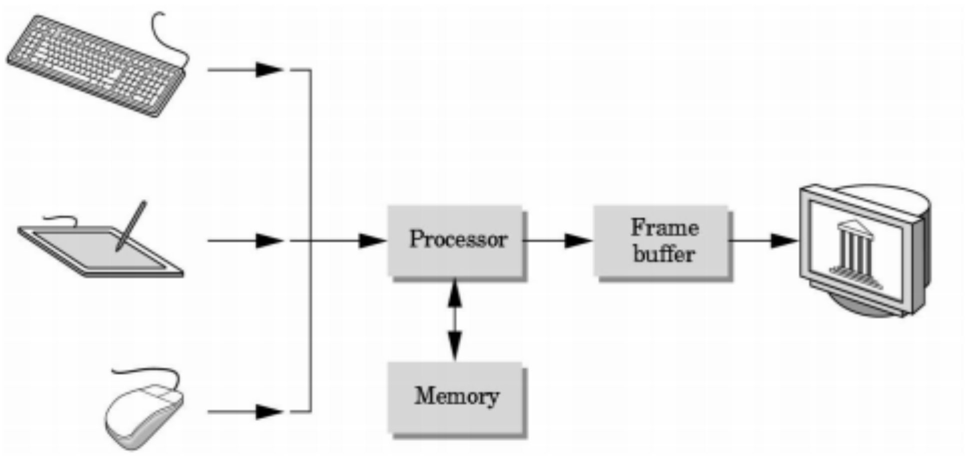


Fig 1.1: Graphic System

1.2Areas of Application of Computer Graphics

- User interfaces and Process control
- Cartography
- Office automation and Desktop publishing
- Plotting of graphs and charts
- Computer aided Drafting and designs
- Simulation and animation
- Virtual Designing
- Video games
- Web design

1.3Introduction to OpenGL

OpenGL is the premier environment for developing portable, interactive 2D and3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms.

OpenGL fosters innovation and speeds application development byincorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.

OpenGL available Everywhere: Supported on all UNIX® workstations, andshipped standard with every Windows 95/98/2000/NTandMacOS PC, no other graphics API operates on a wider range of hardware platforms and software environments.

OpenGL runs on every major operating system including Mac OS, OS/2, UNIX,Windows 95/98, Windows 2000, Windows NT, Linux, Open Step, and BeOS; it also works with every major windowing system, including Win32, MacOS, Presentation Manager, and X-Window System. OpenGL is callable from Ada, C, C++, Fortran, Python, Perl and Java and offers complete independence from network protocols and topologies.

1.3.1 The OpenGL interface

Our application will be designed to access OpenGL directly through functions in three libraries namely: gl,glu,glut.

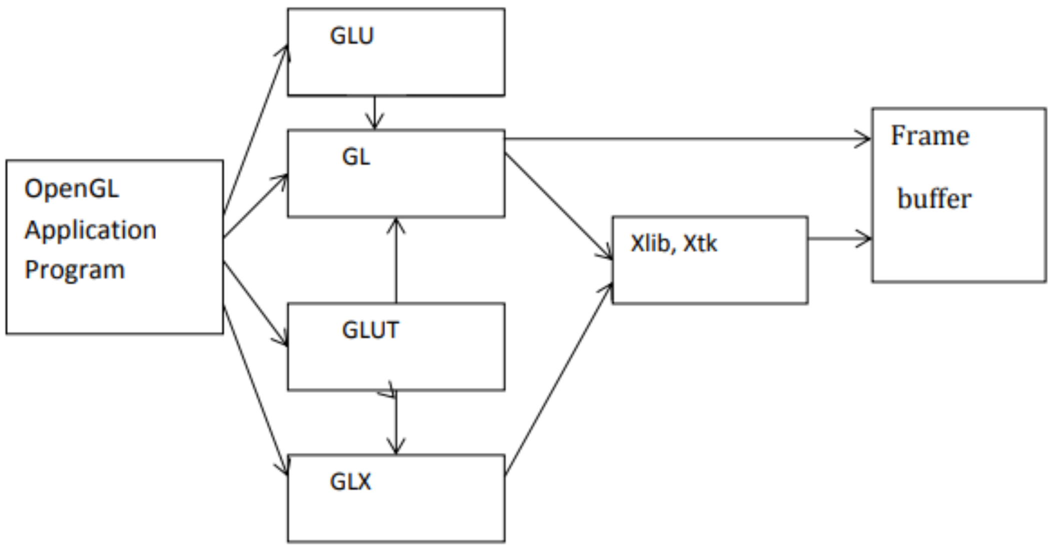


Fig 1.2: Library organization of OpenGL

1.3.2 Graphics Functions

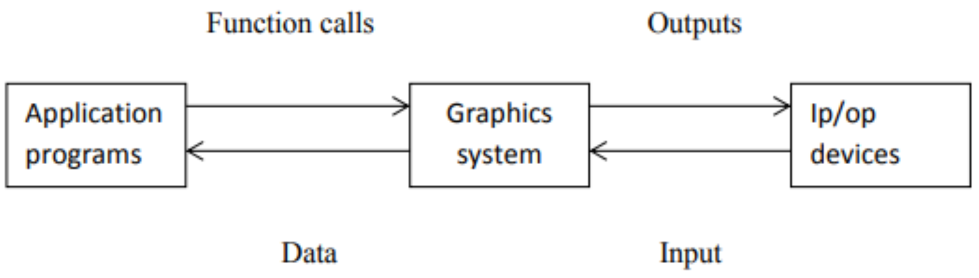


Fig 1.3: Graphics system as a black box.

Our basic model of a graphics package is a black box, a term that engineers use to denote a system whose properties are described only by its inputs and outputs. We describe an API through the functions in its library. Some of the functions are:

- The primitive functions define the low-level objects or atomic entities that our system can display.

- Attribute functions allow us to perform operations ranging from choosing the color with which we display a line segment, to picking a pattern with which to fill the

  inside of a polygon, to selecting a typeface for the titles of a graph.

- Transformation function allows carrying out transformations of objects, such as rotation, translation, and scaling.

- A set of input functions allow us to deal with the diverse forms of input that characterize modern graphics systems.

- The control functions enable us to communicate with the window systems, to initialize our programs, and to deal with any errors that take place during the execution of programs.

CHAPTER 2

REQUIREMENTS SPECIFICATION

2.1 Purpose of the requirements document

The software requirement specification is the official statement of what is required for development of particular project. It includes both user requirements and system requirements. This requirement document is utilized by variety of users starting from project manager who gives project to the engineer responsible for development of project.

It should give details of how to maintain, test, verify and what all the actions to be carried out through life cycle of project.

2.1.1 Scope of the project

The scope is to use the basic primitives defined in openGL library creating complex objects. We make use of different concepts such as glColor() , gluOrtho2D(), timer function.

2.1.2 Definition

The project **ROUTER ARCHITECTURE** is created to demonstrate OpenGL's concepts. It encompasses some of the skills learnt in our OpenGL classes such as glColor() , gluOrtho2D(), timer function.

2.1.3 Acronyms & Abbreviations

OpenGL provides a powerful but primitive set of rendering command, and all higher-level design must be done in terms of these commands.
OpenGL Utility Toolkit(GLUT): -windows-system-independent toolkit.

2.1.4 References

OpenGL tutorials

Interactive Computer Graphics (Edward Angel)

2.2 Specific requirements

2.2.1 User Requirement:

- Easy to understand and should be simple.
- The built-in functions should be utilized to maximum extent.
- OpenGL library facilities should be used.

2.2.2 Software Requirements:

- Platform used: UBUNTU
- Technology used: OpenGL Libraries such has OpenGL Utility library, OpenGL Utility toolkit
- Language: C

2.2.3 Hardware Requirements:

CPU Speed	1.5 GHz or higher
Processor	Intel Pentium® or higher
Memory/RAM	2 GB or higher (32-bit) 8 GB or higher (64-bit)
Display Properties	24-bit colour depth
Screen Resolution	1024 x 768 or higher at normal size (96 dpi)
Swap Space	Determined by the operating system, 500 MB or higher
Disk Space	1 GB for installing the Platform and Synthesis Applications
Video/Graphics Adapter	64 MB RAM or higher

CHAPTER 3

IMPLEMENTATION

3.1 OpenGL Function Details

- **GlutInitDisplayMode**—sets the initial display mode.
  - Declaration: void glutInitDisplayMode (unsigned int mode);
  - Remarks: The initial display mode is used when creating top-level windows, sub windows, and overlays to determine the
  - OpenGL display mode for the to-be-created window or overlay.

- **glutInitWindowposition ---** set the initial window position.
  - Declaration: void glutInitWindowPosition(int x, int y);
  - x:Window X location in pixels.
  - y:Window Y location in pixels.

- **glutInitWindowSize ---** set the initial window size.
  - Declaration: void glutInitWindowSize(intwidth,int height);
  - width: Width in pixels
  - height: Height in pixels.

- **glutCreateWindow**--- set the title to graphics window.
  - Declaration: IntglutCreateWindow(char *title);
  - Remarks: This function creates a window on the display. The string title can be used to label the window.The integer value returned can be used to set the current window when multiple windows are created.

- **glutDisplayFunc**
  - Declaration: void glutDisplayFunc(void(*func)void));
  - Remarks: This function registers the display function that is executed when the window needs to be redrawn.

- **glClear:**
  - Declaration: void glClear();
  - Remarks: This function clears the particular buffer.

- **glclearColor:**
  - Declaration: void glClearColor(GLfloat red, GLfloat green, Glfloat blue, Glfloat alpha);
  - Remarks: This function sets the color value that is used when clearing the color buffer.

- **glEnd**
  - Declaration: void glEnd();
  - Remarks: This function is used in conjunction with glBegin to delimit the vertices of an opengl primitive.

- **glMatrixMode**
  - Declaration: void glMatrixMode(GLenum mode);
  - Remarks: This function specifies which matrix will be affected by subsequent transformations mode can be GL_MODELVIEW,GL_PROJECTION or GL_TEXTURE.

- **gluOrtho2D**
  - Declaration: void glOrtho(GLdouble left, GLdouble right, GLdoublebottom,GLdouble top);
  - Remarks: It defines an orthographic viewing volume with all parameters measured from the center of the projection plane.

- **glutInit**
  - Declaration: Void glutInit(int *argc, char **argv);

- ◦ Remarks: To start thru graphics system, we must first call glutInit (),glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized.

3.2. Code in C Language

```
#define GLUT_DISABLE_ATEXIT_HACK
#include <GL/gl.h>
#include <GL/glut.h>
#include <stdlib.h>
#include<string.h>
#include<vector>

float v[72][2]={{440,400},{560,400},{560,600},{440,600}, //Router
                {660,400},{940,400},{940,480},{660,480}, //Output Ports
                {660,520},{940,520},{940,600},{660,600}, //Output Ports
                {60,400},{340,400},{340,480},{60,480}, //Input Ports
                {60,520},{340,520},{340,600},{60,600},//Input Ports
                {680,410},{760,410},{760,470},{680,470}, //Output Boxes
                {770,410},{850,410},{850,470},{770,470}, //Output Boxes
                {680,530},{760,530},{760,590},{680,590}, //Output Boxes
                {770,530},{850,530},{850,590},{770,590}, //Output Boxes
                {860,420},{920,420},{920,460},{860,460}, //Output Boxes
                {860,540},{920,540},{920,580},{860,580}, //Output Boxes
                {150,410},{230,410},{230,470},{150,470}, //Input Boxes
                {240,410},{320,410},{320,470},{240,470}, //Input Boxes
                {150,530},{230,530},{230,590},{150,590}, //Input Boxes
                {240,530},{320,530},{320,590},{240,590}, //Input Boxes
                {80,420},{140,420},{140,460},{80,460}, //Input Boxes
                {80,540},{140,540},{140,580},{80,580}, //Input Boxes
                {440,800},{560,800},{560,870},{440,870} //Router processor
                };

float
l[26][2]={{20,750},{980,750},{280,835},{440,835},{280,835},{280,590},{280,530},{
280,470},
{400,290},{400,120},
{340,240},{660,240},
{340,180},{660,180},
{460,290},{460,120},
{520,290},{520,120},
{340,290},{340,120},
{340,120},{660,120},
{340,290},{660,290},
{660,290},{660,120}};
float offsetx=0,offsety=0;
int flag=0,color=0;
int displayscreen1,displayscreen2;

void init(int width, int height)
{
    const float ar = (float) width / (float) height;
    glViewport(0, 0, width, height);
    glLoadIdentity();
    glClearColor(1,1,1,1);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0,1000,0,1000);
    glMatrixMode(GL_MODELVIEW);

}

void arrow()
{
    glLineStipple(1, 0xffff);
    glEnable(GL_LINE_STIPPLE);
    glBegin(GL_LINE_LOOP);
    glVertex2f(500,800);
    glVertex2f(520,770);
    glVertex2f(505,780);
    glVertex2f(505,620);
    glVertex2f(520,630);
    glVertex2f(500,600);
    glVertex2f(480,630);
    glVertex2f(495,620);
    glVertex2f(495,780);
    glVertex2f(480,770);
    glEnd();
}

void print(int x, int y, char *string)
{
//set the position of the text in the window using the x and y coordinates
glRasterPos2f(x,y);
//get the length of the string to display
int len = (int) strlen(string);

//loop to display character by character
```

```
for (int i = 0; i < len; i++)
{
glutBitmapCharacter(GLUT_BITMAP_8_BY_13,string[i]);
}
glEnd();
};

void OutLine(int a,int b,int c,int d)
{
    glColor3f(0,0,0);
    glLineStipple(1, 0xffff);
    glEnable(GL_LINE_STIPPLE);
     glBegin(GL_LINE_LOOP);
        glVertex2fv(v[a]);
        glVertex2fv(v[b]);
        glVertex2fv(v[c]);
        glVertex2fv(v[d]);
        glEnd();
}

void polygon(int a,int b,int c,int d)
{
     glBegin(GL_POLYGON);
        glVertex2fv(v[a]);
        glVertex2fv(v[b]);
        glVertex2fv(v[c]);
        glVertex2fv(v[d]);
        glEnd();
}

void Input1()
{
     glBegin(GL_POLYGON);
        glVertex2i(20+offsetx,540+offsety);
        glVertex2i(40+offsetx,540+offsety);
        glColor3f(0.219, 0.835, 0.960);
        glVertex2i(40+offsetx,580+offsety);
        glVertex2i(20+offsetx,580+offsety);
        glEnd();
}
void Input2()
{
     glBegin(GL_POLYGON);
        glVertex2i(20+offsetx,420+offsety);
        glVertex2i(40+offsetx,420+offsety);
        glColor3f(0.964, 0.905, 0.054);
        glVertex2i(40+offsetx,460+offsety);
        glVertex2i(20+offsetx,460+offsety);
        glEnd();
```

```
}
void Line(int a,int b)
{
    glBegin(GL_LINES);
    glVertex2fv(l[a]);
    glVertex2fv(l[b]);
    glEnd();
}
void display1()
{
        glClearColor(0,0,0,1);
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1,1,1);
        print(340,900,"CMR INSTITUTE OF TECHNOLOGY");
        glColor3f(0.043, 0.466, 0.694);
        print(240,860,"#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,");
        print(400,840,"BANGALORE-560037");
        print(240,800,"DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING");
        print(380,720,"A  Mini Project on");
        glColor3f(1,1,1);
        print(360,700,"\"Router Architecture\"");
        glColor3f(0.043, 0.466, 0.694);
        print(30,600,"Submitted in Partial fulfillment of the Requirements for
the VI Semester of");
        print(420,580,"the Degree of");
        print(370,540,"Bachelor of Engineering");
        print(490,520,"In");
        print(320,500,"Computer Science & Engineering");
        print(490,480,"By");
        glColor3f(1,1,1);
        print(460,420,"DIVYA T");
        print(425,400,"(1CR15CS058)");
        print(420,320,"DHANUSH KUMAR S");
        print(425,300,"(1CR15CS055)");
        glColor3f(0.043, 0.466, 0.694);
        print(370,200,"Under the Guidance of");
        glColor3f(1,1,1);
        print(420,160,"Kiran Babu T S");
        glColor3f(0.043, 0.466, 0.694);
        print(320,140,"Assistant Professor,Dept of CSE");
        print(200,100,"(Click left mouse button to go to the next window)");
        print(310,50,"(Click right mouse button to exit)");
        glFlush();
}

void display()
{

    glClear(GL_COLOR_BUFFER_BIT);
```

```
glClearColor(1,1,1,1);
glColor3f(0,1,0);                    //Router
glLineWidth(2);
OutLine(0,1,2,3);
print(445,520,"Switching");
print(455,490,"Fabric");

glColor3f(0.956, 0.062, 0.105);                //Boxes next to the router
polygon(4,5,6,7);          //Right Bottom
glColor3f(0.356, 0.913, 0.266);
polygon(8,9,10,11);         //Right Top
glColor3f(0.964, 0.905, 0.054);
polygon(12,13,14,15);        //Left Bottom
glColor3f(0.219, 0.835, 0.960);
polygon(16,17,18,19);        //Left Top
glColor3f(0,0,0);
print(80,620,"Input port");
print(20,560,"A");
print(700,620,"Output port");
print(960,560,"C");

glLineWidth(1);
glColor3f(1,1,1);                    //Output ports
polygon(20,21,22,23);
OutLine(20,21,22,23);
glColor3f(1,1,1);
polygon(24,25,26,27);
OutLine(24,25,26,27);
glColor3f(1,1,1);
polygon(28,29,30,31);
OutLine(28,29,30,31);
glColor3f(1,1,1);
polygon(32,33,34,35);
OutLine(32,33,34,35);
glColor3f(1,1,1);
polygon(36,37,38,39);
OutLine(36,37,38,39);
glColor3f(1,1,1);
polygon(40,41,42,43);
OutLine(40,41,42,43);
print(80,500,"Input port");
print(20,430,"B");
print(700,500,"Output port");
print(960,430,"D");

glColor3f(1,1,1);
polygon(44,45,46,47);                    //Input ports
OutLine(44,45,46,47);
glColor3f(1,1,1);
```

```
polygon(48,49,50,51);
OutLine(48,49,50,51);
glColor3f(1,1,1);
polygon(52,53,54,55);
OutLine(52,53,54,55);
glColor3f(1,1,1);
polygon(56,57,58,59);
OutLine(56,57,58,59);
glColor3f(1,1,1);
polygon(60,61,62,63);
OutLine(60,61,62,63);
glColor3f(1,1,1);
polygon(64,65,66,67);
OutLine(64,65,66,67);

glColor3f(0.964, 0.054, 0.549);
polygon(68,69,70,71);
OutLine(68,69,70,71);           //Routing Processor
print(400,900,"Routing Processor");

glPushAttrib(GL_ENABLE_BIT);
glLineStipple(1, 0x0f0f);
glEnable(GL_LINE_STIPPLE);
Line(0,1);
Line(2,3);
Line(4,5);
Line(6,7);
glPushAttrib(GL_ENABLE_BIT);
glLineStipple(1, 0xffff);
glEnable(GL_LINE_STIPPLE);
Line(8,9);
Line(10,11);
Line(12,13);
Line(14,15);
Line(16,17);
Line(18,19);
Line(20,21);
Line(22,23);
Line(24,25);
print(20,800,"Routing, management");
print(20,780,"control plane");
print(19,760,"(software)");
print(20,720,"Forwarding" );
print(20,700,"data plane(hardware)");
print(440,300,"Output Ports");
print(360,200,"A");
print(270,180,"Input");
print(270,160,"Ports");
print(360,140,"B");
```

```
    print(420,260,"C");
    print(480,260,"D");
    print(540,270,"Routing");
    print(540,250,"Processor");
    print(420,200,"m");
    print(480,200,"n");
    print(420,140,"b");
    print(480,140,"v");
    print(580,200,"c");
    print(580,140,"x");

    glColor3f(0.086, 0.662, 0.831);
    arrow();
    if(flag==1&& color==0){
        glColor3f(0.356, 0.913, 0.266);
        Input1();
    }
    if(flag==1&& color==1){
        glColor3f(0.956, 0.062, 0.105);
        Input1();
    }
    if(flag==1&& color==2){
        glColor3f(0.964, 0.054, 0.549);
        Input1();
    }
    if(flag==2&& color==0){
        glColor3f(0.356, 0.913, 0.266);
         Input2();
    }
     if(flag==2&& color==1){
        glColor3f(0.956, 0.062, 0.105);
        Input2();
    }
    if(flag==2&& color==2){
        glColor3f(0.964, 0.054, 0.549);
        Input2();
    }
     glColor3f(0,0,0);
     print(200,50,"(Click Right mouse button to exit and go to Home page)");
     glFlush();
}

void key(unsigned char key,int x0,int y0)
{
    if(key=='m')
    {
        flag=1;
        color=0;
        for(offsetx=20;offsetx<940;offsetx++)
        display();
        offsetx=0;
    }
    if(key=='n')
    {
        flag=1;
        color=1;
        for(offsetx=20;offsetx<460;offsetx++)
            display();
        for(offsety=0;offsety>-120;offsety--)
            display();
        for(offsetx=460;offsetx<940;offsetx++)
            display();
        offsetx=0;
        offsety=0;
    }
    if(key=='b'){
            flag=2;
        color=0;
        for(offsetx=20;offsetx<460;offsetx++)
            display();
        for(offsety=0;offsety<120;offsety++)
            display();
        for(offsetx=460;offsetx<940;offsetx++)
            display();
        offsetx=0;
        offsety=0;
    }
    if(key=='v')
    {
         flag=2;
        color=1;
        for(offsetx=20;offsetx<940;offsetx++)
           display();
        offsetx=0;
    }

    if(key=='c')
    {
        flag=1;
        color=2;
        for(offsetx=20;offsetx<250;offsetx++)
            display();
        for(offsety=0;offsety<275;offsety++)
            display();
        for(offsetx=250;offsetx<470;offsetx++)
            display();
        offsetx=0;
        offsety=0;
    }
    if(key=='x')
    {
        flag=2;
        color=2;
        for(offsetx=20;offsetx<250;offsetx++)
            display();
        for(offsety=0;offsety<395;offsety++)
            display();
        for(offsetx=250;offsetx<470;offsetx++)
            display();
        offsetx=0;
        offsety=0;
    }
}

void mouse1(int button,int state,int x,int y){
    if(button==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
        glutDestroyWindow(displayscreen2);
}

void mouse(int button,int state,int x,int y){
    if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
    {
        glutInitWindowSize(640,640);
        glutInitWindowPosition(0,0);
        glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
        displayscreen2=glutCreateWindow("Router Architecture");
        glEnable(GL_LINE_SMOOTH);
        glutReshapeFunc(init);
        glutMouseFunc(mouse1);
        glutKeyboardFunc(key);
        glutDisplayFunc(display);
        glClearColor(0,0,0,0);
        glutMainLoop();
    }
    if(button==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
        glutDestroyWindow(displayscreen1);
}

/* Program entry point */
int main(int argc, char *argv[])
{
    glutInit(&argc, argv);
    glutInitWindowSize(640,640);
    glutInitWindowPosition(0,0);
    glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
    displayscreen1=glutCreateWindow("Home Page");
    glEnable(GL_LINE_SMOOTH);
    glutReshapeFunc(init);
    glutMouseFunc(mouse);
    glutKeyboardFunc(key);
    glutDisplayFunc(display1);
    glClearColor(0,0,0,0);
    glutMainLoop();
}
```

CHAPTER 4

DESCRIPTION AND SNAPSHOTS

4.1 DESCRIPTION

- The Objective of this project demonstrate the working of the router and explain its working.
- A **router** is a device that forwards data packets along networks. A **router** is connected to at least two networks, commonly two LANs or WANs or a LAN and its ISP's network. **Routers** are located at gateways, the places where two or more networks connect.
- A router consists of :-
    - **Input ports:**An input port performs several key functions. It performs the physical layer function of terminating an incoming physical link at a router. An input port also performs link-layer functions needed to interoperate with the link layer at the other side of the

incoming link. Perhaps most crucially, the lookup function is also performed at the input port. It is here that the forwarding table is consulted to determine the router output port to which an arriving packet will be forwarded via the switching fabric. Control packets (for example, packets carrying routing protocol information) are forwarded from an input port to the routing processor.

- **Switching fabric.** The switching fabric connects the router's input ports to its output ports. This switching fabric is completely contained with the router - a network inside of a network router!
- **Output ports.** An output port stores the datagrams that have been forwarded to it through the switching fabric, and then transmits the datagrams on the outgoing link. The output port thus performs the reverse data link and physical layer functionality as the input port.
- **Routing processor.** The routing processor executes the routing protocols, maintains the routing tables, and performs network management functions, within the router. Since we cover these topics elsewhere in this book, we defer discussion of these topics to elsewhere.

● A router's input ports, output ports, and switching fabric together implement the forwarding function and are almost always implemented in hardware. These forwarding functions are sometimes collectively referred to as the router forwarding plane.

● A router's control functions i.e. executing the routing protocols, responding to attached links that go up or down, and performing management functions,operate at the millisecond or second timescale. These router control plane functions are usually implemented in software and execute on the routing processor (typically a traditional CPU).

● The switching fabric is at the very heart of a router, as it is through this fabric that the packets are actually switched (that is, forwarded) from an input port to an output port. Switching can be accomplished in a number of ways:

➢ *Switching via memory*: The simplest, earliest routers were traditional computers, with switching between input and output ports being done under direct control of the CPU (routing processor). Input and output ports functioned as traditional I/O devices in a traditional operating system.

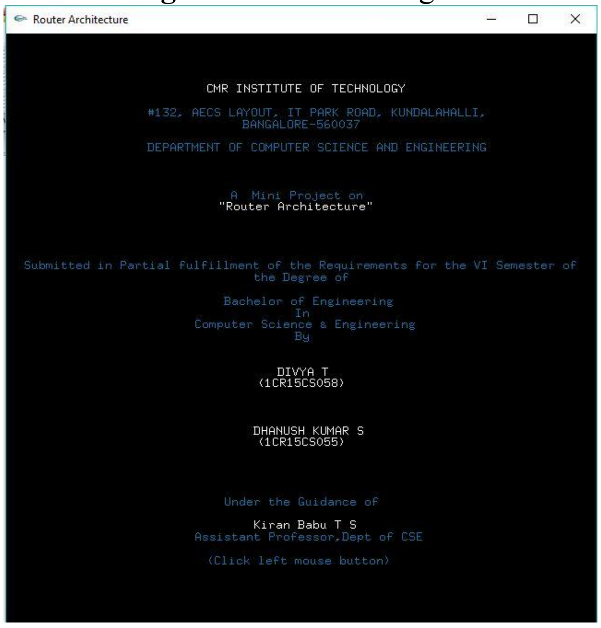
➢ *Switching via a bus*: In this approach, an input port transfers a packet directly to the output port over a shared bus, without intervention by the routing processor. This is typically done by having the input port pre-pend a switch-internal label (header) to the packet indicating the local output port to which this packet is being transferred and transmitting the packet onto the bus. The packet is received by all output ports, but only the port that matches the label will keep the packet. The label is then removed at the output port, as this label is only used within the switch to cross the bus.

➢ *Switching via an interconnection network*: One way to overcome the bandwidth limitation of a single, shared bus is to use a more sophisticated interconnection network, such as those that have been used in the past to interconnect processors in a multiprocessor computer architecture. Acrossbar switch is an interconnection network consisting of 2N buses that connect N input ports to N output ports. Each vertical bus intersects each horizontal bus at a crosspoint, which can be opened or closed at any time by the switch fabric controller (whose logic is part of the switching fabric itself). When a packet arrives from port Aand needs to be forwarded to port Y, the switch controller closes the crosspoint at the intersection of busses Aand Y, and port Athen sends the packet onto its bus, which is picked up (only) by bus Y. Note that a packet from port B can be forwarded to port X at the same time, since the A-to-Y and B-to-X packets use different input and output busses. Thus, unlike the previous two switching approaches, crossbar networks are capable of forwarding multiple packets in parallel. However, if two packets from two different input ports are destined to the same output port, then one will have to wait at the input, since only one packet can be sent over any given bus at a time. More sophisticated interconnection networks use multiple stages of switching elements to allow packets from different input ports to proceed towards the same output port at the same time through the switching fabric.

- Routers may provide connectivity within enterprises, between enterprises and the Internet, or between internet service providers' (ISPs') networks. The largest routers (such as the Cisco CRS-1 or Juniper PTX) interconnect the various ISPs, or may be used in large enterprise networks.Smaller routers usually provide connectivity for typical home and office networks.

4.2 SCREENSHOTS

Fig 4.2.1 Welcome Page



CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A Mini Project on "Router Architecture"

Submitted in Partial fulfillment of the Requirements for the VI Semester of the Degree of

Bachelor of Engineering In Computer Science & Engineering By

DIVYA T (1CR15CS058)

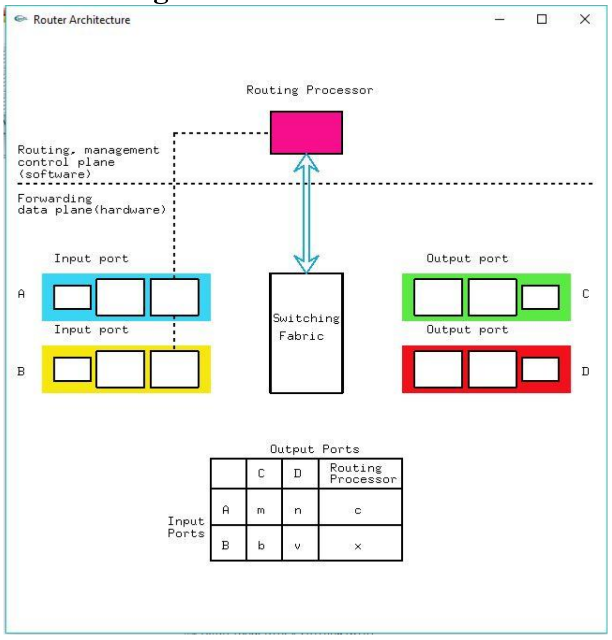
DHANUSH KUMAR S (1CR15CS055)

Under the Guidance of

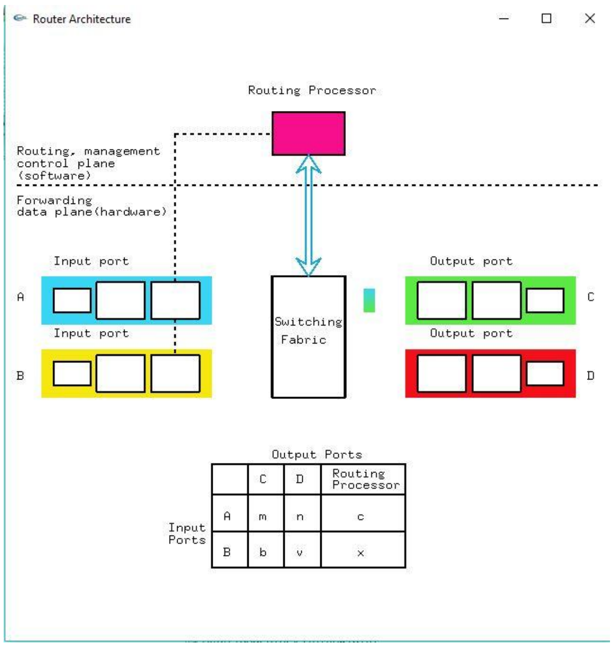
Kiran Babu T S
Assistant Professor,Dept of CSE

(Click left mouse button)

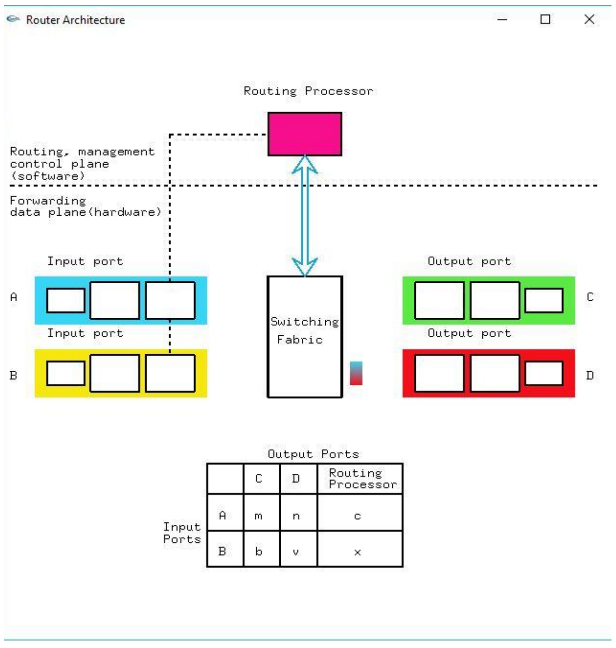
Fig 4.2.2 Router Architecture



	C	D	Routing Processor
A	m	n	c
B	b	v	x



	C	D	Routing Processor
A	m	n	c
B	b	v	x



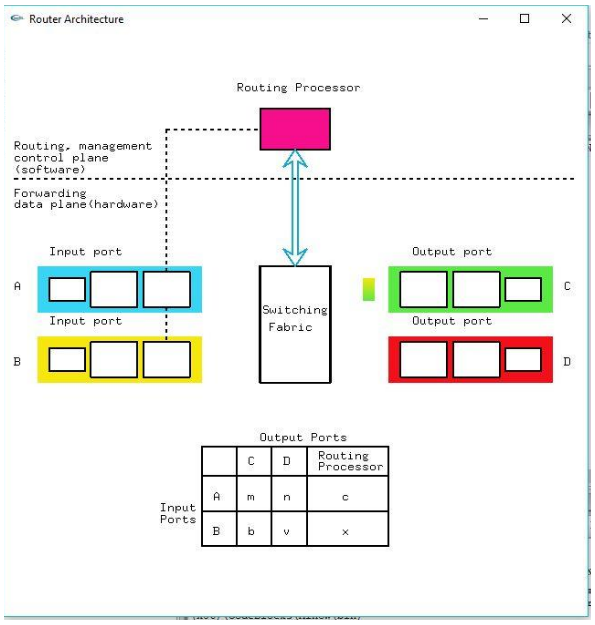
	C	D	Routing Processor
A	m	n	c
B	b	v	x

Fig 4.2.3 Packet moving from Input port A to output port C

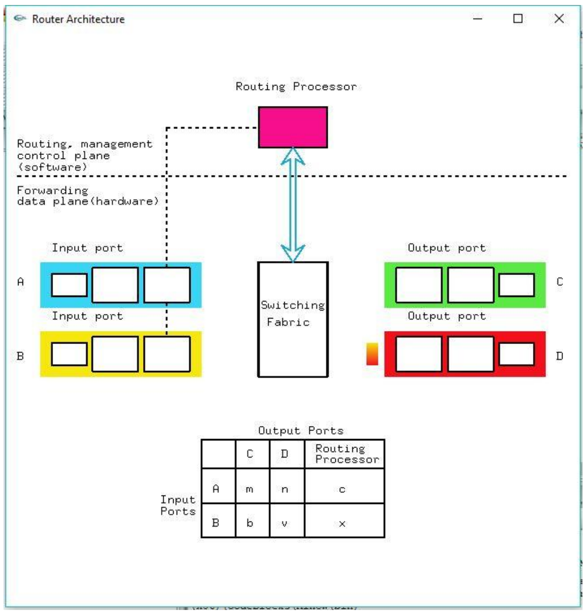
Fig 4.2.4 Packet moving from Input port A to output port D

Fig 4.2.5 Packet moving from Input port B to Output port C

Fig 4.2.6 Packet moving from Input port B to Output Port D

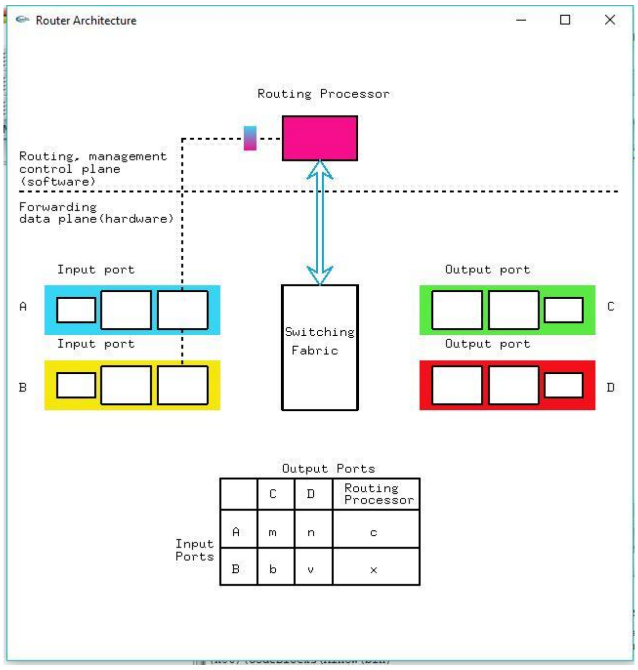


	C	D	Routing Processor
A	m	n	c
B	b	v	x



	C	D	Routing Processor
A	m	n	c
B	b	v	x

Fig 4.2.7 Packet moving from Input port A to Routing Processor



	C	D	Routing Processor
A	m	n	c
B	b	v	x

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

By implementing this project I got to know how to use some of the built in functions effectively and how to interact efficiently and easily. I got a good exposure of how games, animation and simulations are developed, by working on this project.

The OpenGL Utility Toolkit (GLUT) is a programming interface with ANSI C bindings for writing window system independent OpenGL programs.

One of the major accomplishments in the specification of OpenGL was the isolation of window system dependencies from OpenGL's rendering model. The result is that OpenGL is window system independent. Window system operations such as the creation of a rendering window and the handling of window system events are left to the native window system to define. Necessary interactions between OpenGL and the window system such as creating and binding an OpenGL context to a window are described separately from the OpenGL specification in a window system dependent specification.

The GLUT application-programming interface (API) requires very few routines to display a graphics scene rendered using OpenGL. The GLUT API (like the OpenGL API) is stateful. Most initial GLUT state is defined and the initial state is reasonable for simple programs. The GLUT routines also take relatively few parameters. No pointers are returned. The only pointers passed into GLUT are pointers to character strings (all strings passed to GLUT are copied, not referenced) and opaque font handles.

BIBLIOGRAPHY

- Google for problem solving
- Donald Hearn & Pauline Baker: Computer Graphics-OpenGL Version,3rd Edition, Pearson Education,2011
- Edward Angel: Interactive computer graphics- A Top Down approach with OpenGL, 5th edition. Pearson Education, 2011
- M MRaikar, Computer Graphics using OpenGL, Fillip Learning / Elsevier, Bangalore / New Delhi (2013)
- https://stackoverflow.com/
- "Computer Graphics", Addison-Wesley 1997 James D Foley, Andries Van Dam, StevenK Feiner, John F Hughes.
- F.S.Hill and Stephen M.Kelly, "Computer Graphics using OpenGl ", 3rd edition
- http://www.opengl.org
- http:// www.openglprojects.in
- http://www.openglprogramming.com/
- http://www.google.com/