VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A Computer Graphics & Visualization Mini Project Report

on

"Dijkstra's Algorithm"

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

Bachelor of Engineering

In

Computer Science & Engineering

By

DIANA ELIZABETH ROY
(1CR15CS056)

PRANNAY S REDDY
(1CR15CS114)

Under the Guidance of

Mr. Kartheek G C R
Asst. 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 **"Dijkstra's Algorithm"** has been carried out by **Diana Elizabeth Roy (1CR15CS056)** and **Prannay S Reddy(1CR15CS114)** 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. Kartheek G C R	**Dr. Jhansi Rani P**
Asst. Professor	**Professor &Head**
Dept. of CSE, CMRIT	**Dept. of CSE, CMRIT**

External Viva

Name of the examiners	Signature with date
1.	
2.	

ABSTRACT

This report talks about our project titled “Dijkstra’s Algorithm” implemented using OpenGL. OpenGL provides a set of commands to render a three dimensional scene. That means you provide them in an OpenGL-useable form and OpenGL will show this data on the screen (render it). It is developed by many companies and it is free to use. You can develop OpenGL applications without licensing.

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.

The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.

For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. As a result, the shortest path algorithm is widely used in network routing protocols, most notably IS-IS (Intermediate System to Intermediate System) and Open Shortest Path First (OSPF). It is also employed as a subroutine in other algorithms such as Johnson's.

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am privileged to have got all this along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank **Mr. Sanjay Jain**, Principal, CMRIT for providing me the right ecosystem and resources, in the form of this college, for doing the project.

I owe my deep gratitude to our project guide **Mr. Kartheek G C R**, Asst. Professor, CSE Dept., CMRIT, who took keen interest on our project work and guided us along, till the completion of our project by providing all the necessary information for developing a good system.

I would not forget to remember **Dr. Jhansi Rani P**, Professor and Head, Dept. of CSE, CMRIT, for her encouragement and more over for her timely support and guidance till the completion of the project work.

I am thankful to and fortunate enough to get constant encouragement, support and guidance from the college staff, student friends and family which helped us in successfully complete our project work.

TABLE OF CONTENTS

LIST OF FIGURES & TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO 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.

1.2 AREAS 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

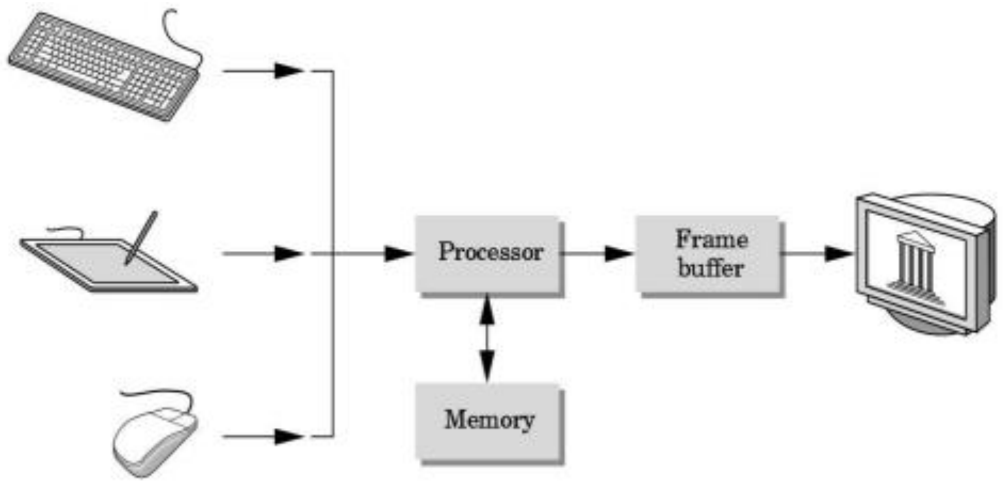


Fig 1.1: Graphic System

1.3 INTRODUCTION TO OPENGL

OpenGLis the premier environment for developing portable, interactive 2D and3D graphics applications. Since its introduction in 1992, OpenGL has become the industry'smost 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 by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all populardesktop and workstation platforms, ensuring wide application deployment.

OpenGL available Everywhere: Supported on all UNIX® workstations, and shipped 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, andX-Window System. OpenGL is callable from Ada, C, C++, Fortran, Python, Perl and Javaand offers complete independence from network protocols and topologies.

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

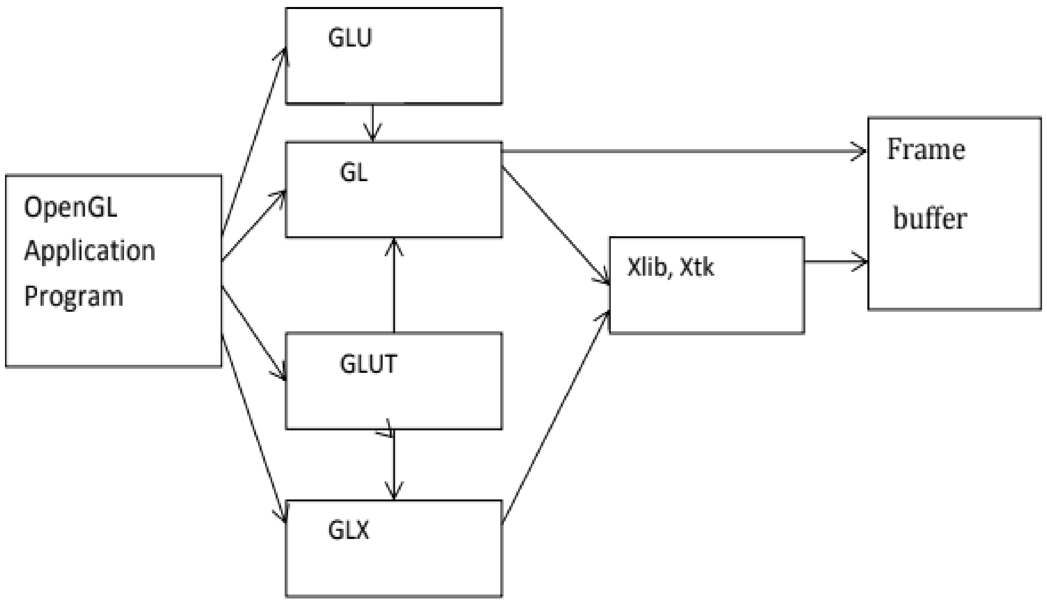


Fig 1.2: Library organization of OpenGL

1.3.1 GRAPHICS FUNCTIONS

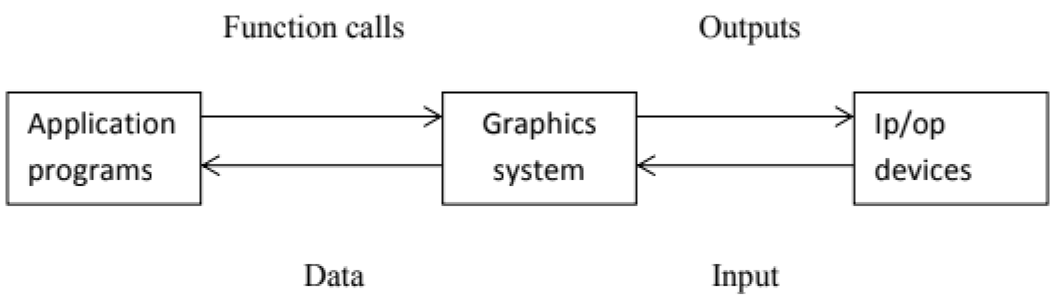


Fig 1.3: Graphics system as a black box

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

SYSTEM REQUIREMENTS

2.1 USER REQUIREMENTS

- 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 HARDWARE REQUIREMENTS

- Processor-Intel or AMD (Advanced Micro Devices)
- RAM-512MB (MINIMUM)
- Hard Disk-1MB (MINIMUM)
- Mouse
- Keyboard
- Monitor

2.3 SOFTWARE REQUIREMENTS

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

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 theOpenGL display mode for the to-be-created window oroverlay.

- **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 stringtitle can be used to label the window.The integer value returned canbe used to set the current window when multiple windows arecreated.

- **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: voidglClearColor(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 todelimit the vertices of an opengl primitive.

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

- **gluOrtho2D**
  - Declaration:

    voidglOrtho(GLdouble left, GLdouble right, GLdoublebottom,GLdouble top);
  - Remarks: It defines an orthographic viewing volume with allparameters measured from the center of the projection plane.

- **glutPassiveMotion**
  - Declaration:

    void passiveMotionFunc(int x,int y);
  - Remarks: It is used to converting screen resolution to ortho 2d spec and do calculations to find the angle of the lazer.

- **glutInit**
  - Declaration:

    voidglutInit(int *argc, char **argv);
  - Remarks: To start thru graphics system, we must first callglutInit (),glutInit will initialize the GLUT library andnegotiate a session with the window system. During thisprocess, glutInit may cause the termination of the GLUTprogram with an error message to the user if GLUT cannotbe properly initialized.

4.2 SOURCE CODE

```
#include <GL/glut.h>
#include <GL/gl.h>
#include<math.h>
#include<stdio.h>
#include <iostream>
using namespace std;

#define GL_PI 3.14
#define MAX 25
int n,i=1,a[25],b[25],cost[25][25],tree[25][25],src,l[2],dist[10];
char s[20],*s1;
void *currentfont;

void reverse(char str[], int length)
{
    int start = 0;
    int end = length -1;
    while (start < end)
    {
        swap(*(str+start), *(str+end));
        start++;
        end--;
    }
}

char* itoa(int num, char* str, int base)
{
    int i = 0;
    bool isNegative = false;

    /* Handle 0 explicitely, otherwise empty string is printed for 0
*/
    if (num == 0)
    {
        str[i++] = '0';
        str[i] = '\0';
        return str;
    }

    // In standard itoa(), negative numbers are handled only with
    // base 10. Otherwise numbers are considered unsigned.
    if (num < 0 && base == 10)
    {
        isNegative = true;
        num = -num;
    }

    // Process individual digits
    while (num != 0)
    {
        int rem = num % base;
        str[i++] = (rem > 9)? (rem-10) + 'a' : rem + '0';
        num = num/base;
    }

    // If number is negative, append '-'
    if (isNegative)
        str[i++] = '-';

    str[i] = '\0'; // Append string terminator

    // Reverse the string
    reverse(str, i);

    return str;
}

//FUNCTION TO SELECT BITMAP FONT
void setFont(void *font)
{
     currentfont=font;
}
//FUNCTION TO DRAW BITMAP string at (x,y)
void drawstring(GLfloat x,GLfloat y,char *string)
{
   char *c;
   glRasterPos2f(x,y);

   for(c=string;*c!='\0';*c++)
   {
        glutBitmapCharacter(currentfont,*c);
   }
}

//FUNCTION TO DELAY
void delay()
{
     for(int i=0;i<12000;i++)
        for(int j=0;j<12000;j++);
}
//DISPLAY FUNCTION FOR TITLE PAGE
void title()
{
     glLineWidth(3.0);
     glColor3f(1.0,1.0,0.0);
     glBegin(GL_LINE_LOOP);
```

```
                                glVertex2f(10,10);
                                glVertex2f(10,490);
                                glVertex2f(490,490);
                                glVertex2f(490,10);
      glEnd();

      setFont(GLUT_BITMAP_HELVETICA_18);
      glColor3f(1.0,1.0,1.0);
      drawstring(100,440,"Topic: Dijktra's Algorithm");

      glColor3f(1.0,1.0,1.0);
      drawstring(100,400,"Submitted by");

      glColor3f(0.0,1.0,0.0);
      drawstring(100,360,"DIANA ELIZABETH ROY");
      drawstring(100,340,"PRANNAY S REDDY");


      glColor3f(0.0,1.0,0.0);
      drawstring(100,320,"VI CSE");

      glColor3f(0.0,1.0,0.0);
      drawstring(100,280,"1CR15CS056");
      drawstring(100,240,"1CR15CS114");

      //delay();
      glColor3f(1.0,1.0,1.0);
      drawstring(100,100,"Right click in My Window for options");
glFlush();
}
//DISPLAY FUNCTION FOR INITIALIZING (DRAWING) THE  INPUT AND OUTPUT
AREAS
void initial()
{
      glClear(GL_COLOR_BUFFER_BIT);

      setFont(GLUT_BITMAP_HELVETICA_18);

      glColor3f(0.0,0.0,0.0);
      drawstring(20,230,"Input Area");
      drawstring(20,470,"Output Area");

      glColor3f(0.0,0.0,0.0);
      glLineWidth(3.0);
      glBegin(GL_LINES);
                                glVertex2f(10,10);
                                glVertex2f(10,490);

                                glVertex2f(10,490);
                                glVertex2f(490,490);
```

```
                           glVertex2f(490,490);
                           glVertex2f(490,10);

                           glVertex2f(490,10);
                           glVertex2f(10,10);

                           glVertex2f(10,250);
                           glVertex2f(490,250);
     glEnd();
//glBegin(GL_LINES);
//glVertex2f(10,250);
//glVertex2f(490,250);
//glEnd();
     glFlush();
}

//BLANK DISPLAY FUNCTION
void display (void)
{
     glFlush();

}

//DRAW A BITMAP NUMBER i at (x,y)
void raster(int x,int y,int i)
{
     char z=i+'0';
     glRasterPos2f(x-5,y-5);
     glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,z);
}


//DRAW THE NODES (SQUARES)
void drawSquare(int x, int y)
{

     if(i<=n)
     {
             y = 500-y;                             //Convert from
screen coordinates
             glPointSize(40);

             if(i==src)
                       glColor3f(0.7f, 0.4f, 0.0f);
             else
                       glColor3f(0.5f, 0.5f, 0.8f);

             glBegin(GL_POINTS);
                                glVertex2f(x , y);
             glEnd();
```

```
                a[i]=x;
                b[i]=y;

                glColor3f(0.0f, 1.0f, 0.0f);
                s1=itoa(i,s,10);
                drawstring(x-5,y-5,s1);

                glFlush();
     }
     i=i+1;
}

//READ DATA: |V|,COST MATRIX, SOURCE VERTEX
void read()

{


      printf("Enter the number of vertices\n");
      scanf("%d",&n);
      printf("Enter the cost matrix\n");
      for(int j=1;j<=n;j++)
                for(int k=1;k<=n;k++)
                {
                        scanf("%d",&cost[j][k]);
                        if(cost[j][k]==0)
                                         cost[j][k]=999;
                }

       printf("Enter the source\n");
       scanf("%d",&src);

       printf("\nGO TO MY WINDOW AND CLICK RIGHT BUTTON FOR NEXT
OPTION\n");
       initial(); //Draw the initial screen
}

//DRAW THE EDGES
void drawline()
{
   int j,k,x1,x2,y1,y2;
   for(j=1;j<=n;j++)
   {
    for(k=1;k<=n;k++)
    {
     if(cost[j][k]!=999 && j<k)
     {
      x1=a[j];
      y1=b[j];
      x2=a[k];
      y2=b[k];
```

```
        glColor3f(0.0,0.5,0.0);

        glLineWidth(3);
        glBegin(GL_LINES);
                          glVertex2i(x1,y1);
                          glVertex2i(x2,y2);
        glEnd();

        s1=itoa(cost[j][k],s,10);
        drawstring((x1+x2-16)/2,(y1+y2+22)/2,s1);
        glFlush();
       }

       if(cost[j][k]!=cost[k][j] && cost[j][k]!=999 && j>k)
       {
        x1=a[j];
        y1=b[j];
        x2=a[k];
        y2=b[k];

        glColor3f(1.0,0.5,0.0);
        glBegin(GL_LINES);
                           glVertex2i(x1+10,y1+18);
                           glVertex2i(x2+10,y2+18);
        glEnd();

        s1=itoa(cost[j][k],s,10);
        drawstring((x1+x2+20)/2,(y1+y2+36)/2,s1);
        glFlush();
       }
      }
     }
}


void shortestpath()
{

       //START OF DIJIKSTRA's
       int w,j,u,v,k,p,q,x1,y1,x2,y2,x,y;
       int dist[MAX],visit[MAX],parent[MAX],mincost,min;
       for(w=1;w<=n;w++)
       {
                         visit[w]=0;
                         dist[w]=cost[src][w];
                         parent[w]=src;
       }

       visit[src]=1;
     mincost=0;
     k=1;

     for(w=1;w<n;w++)
     {
                    min=999;
                    u=-1;

                    for(j=1;j<=n;j++)
                    {
                                   if(!visit[j]&&dist[j]<min)
                                   {

min=dist[j];
                                                                 u=j;
                                   }
                    }

                    visit[u]=1;
                    mincost=mincost+dist[u];

                    tree[k][1]=parent[u];
                    tree[k++][2]=u;

                    for(v=1;v<=n;v++)
                    {

if(!visit[v]&&(cost[u][v]+dist[u]<dist[v]))
                                   {

dist[v]=dist[u]+cost[u][v];

parent[v]=u;
                                   }
                    }
     }

     printf("\nThe cost from source to other vertices are\n\n");
     for(w=1;w<=n;w++)
     {
                     if(w!=src)
                              printf("%d-->%d =
%d\n",src,w,dist[w]);
     }

      //END OF DIJIKSTRAS


      for(int r=1;r<=n;r++)
      {
              x=a[r];
```

```
                    y=b[r];

                    glPointSize(25);
                    if(r==src)
                                glColor3f(0.7f, 0.4f, 0.0f);
                    else
                                glColor3f(0.5f, 0.5f, 0.8f);

                    glBegin(GL_POINTS);
                                         glVertex2f(x,y+250);
                    glEnd();

                    glColor3f(0.0,1.0,0.0);

                    s1=itoa(r,s,10);
                    drawstring(x,y+250,s1);

                    glFlush();

//}

                    for(int x=1;x<n;x++)
                    {
                            p=tree[x][1];
                            q=tree[x][2];

                            x1=a[p];
                            y1=b[p];
                            x2=a[q];
                            y2=b[q];

                            if(p<q)
                            {
                                   glColor3f(0.0,0.5,0.0);
                                   glBegin(GL_LINES);
                                                     glVertex2i(x1,y1+250);
                                                     glVertex2i(x2,y2+250);
                                   glEnd();


                                   s1=itoa(cost[p][q],s,10);
                                   drawstring((x1+x2)/2,(y1+y2+500)/2,s1);
                            }

                            else
                            {
                                   glColor3f(1.0,0.5,0.0);
                                   glBegin(GL_LINES);
                                                     glVertex2i(x1,y1+250);
                                                     glVertex2i(x2,y2+250);
                                   glEnd();
                                s1=itoa(cost[p][q],s,10);
                                drawstring((x1+x2)/2,(y1+y2+500)/2,s1);
                         }
                 }
                 glFlush();
         }
}


void mouse(int bin, int state , int x , int y)
{
     if(bin==GLUT_LEFT_BUTTON&&state==GLUT_DOWN)
        drawSquare(x,y);

}

void top_menu(int option)
{

     switch(option)
     {
                   case 1:
                        read();
                        glutPostRedisplay();
                        break;
                   case 2:
                        drawline();
                        glutPostRedisplay();
                        break;
                   case 3:
                        shortestpath();
                        glutPostRedisplay();
                        break;
                   case 4:
                        exit(0);
     }
}


void init (void)
{
     glClearColor (1.0, 1.0, 1.0, 1.0);
     glClear(GL_COLOR_BUFFER_BIT);
     glViewport( 0,0, 500, 500 );
     glMatrixMode( GL_PROJECTION );
     glOrtho( 0.0, 500.0, 0.0, 500.0, 1.0, -1.0 );
     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity();
     glFlush();
```

```
}
void myInit1()
{
   glClearColor(0.0,0.0,0.0,0.0);
   glColor3f(0.0f,0.0f,0.0f);
   glPointSize(5.0);
   gluOrtho2D(0.0,500.0,0.0,500.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   setFont(GLUT_BITMAP_HELVETICA_18);
}


void display1(void)
{
   glClear(GL_COLOR_BUFFER_BIT);
   title();

}
int main (int argc,char** argv)
{
    glutInit(&argc,argv);

    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
    glutInitWindowPosition(820,100);
    glutInitWindowSize(450,450);
    glutCreateWindow("Front Sheet");
    glutDisplayFunc(display1);
    myInit1();

    glutInitDisplayMode( GLUT_SINGLE|GLUT_RGB );
    glutInitWindowSize(500,500);
    glutInitWindowPosition(0,0);
    glutCreateWindow("My Window");
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
    glutCreateMenu(top_menu);
    glutAddMenuEntry("Read Cost Matrix",1);
    glutAddMenuEntry("Display Weighted Graph",2);
    glutAddMenuEntry("Display Shortest Path",3);
    glutAddMenuEntry("Exit",4);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    printf("\nGO TO MY WINDOW AND CLICK RIGHT BUTTON FOR NEXT
OPTION\n");
    init();

    glutMainLoop();
}
```

CHAPTER 4

DISCUSSION

4.1 OVERVIEW

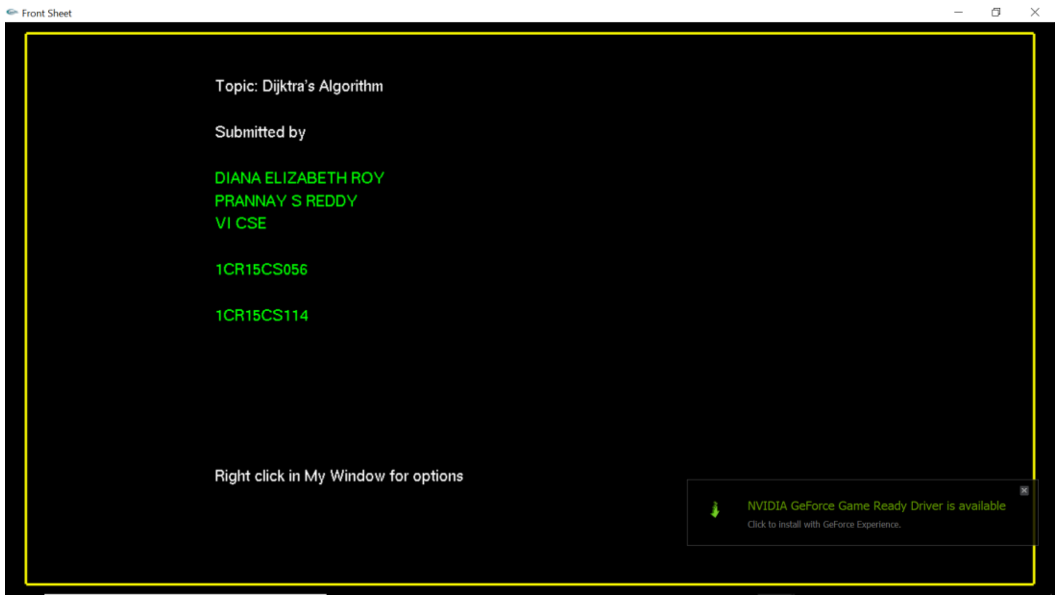
In this section, we will discuss the execution of the code and explain the overall appearance of the project.

On the in-built Ubuntu Terminal, we first need to locate the directory containing the file to run the program using the command:

g++ projFINAL.cpp –lGL –GLU -lglut

This command creates a "./a.out " file which must be run by typing it in the Ubuntu Terminal.

4.1 SCREEN SNAPSHOTS


Front Sheet

Topic: Dijktra's Algorithm

Submitted by

DIANA ELIZABETH ROY
PRANNAY S REDDY
VI CSE

1CR15CS056

1CR15CS114

Right click in My Window for options

NVIDIA GeForce Game Ready Driver is available
Click to install with GeForce Experience.

Fig 4.1: Front Page

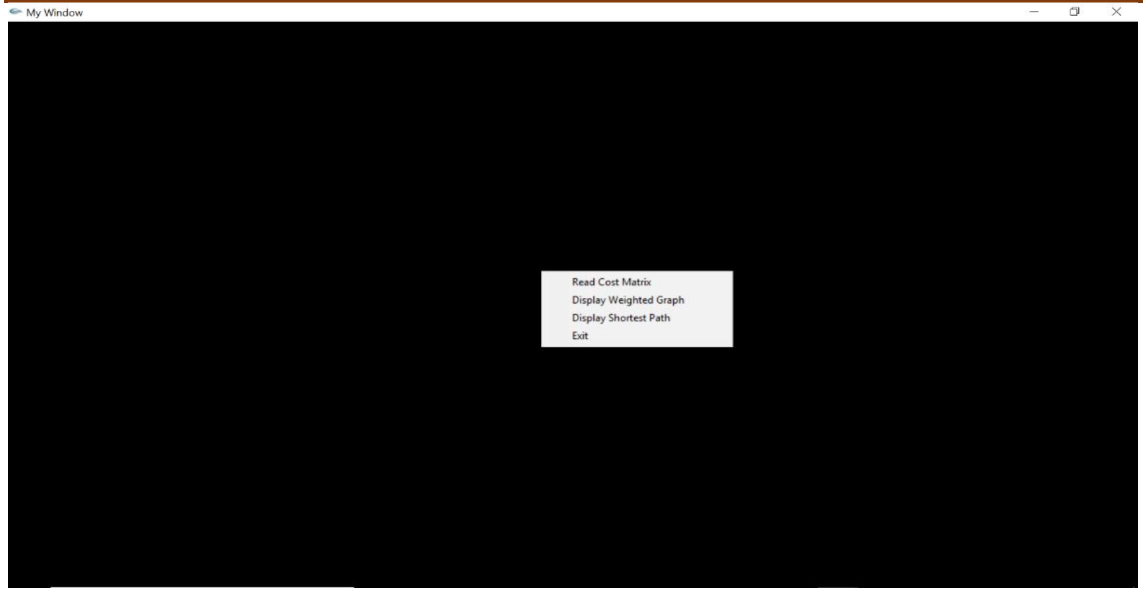
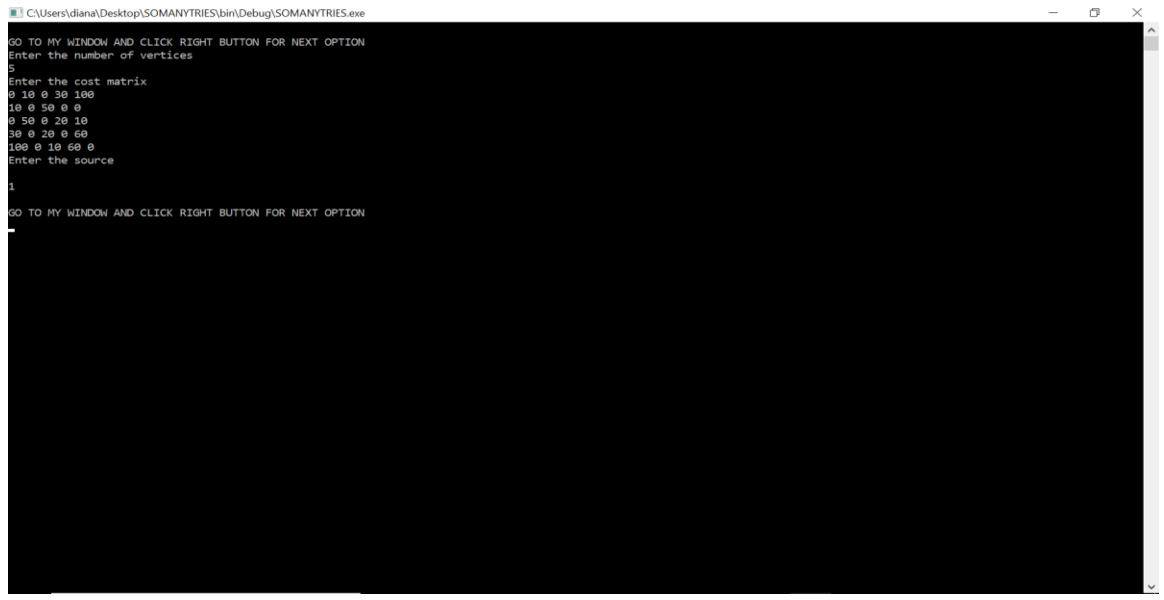


Fig 4.2: On right clicking in my window


C:\Users\diana\Desktop\SOMANYTRIES\bin\Debug\SOMANYTRIES.exe

```
GO TO MY WINDOW AND CLICK RIGHT BUTTON FOR NEXT OPTION
Enter the number of vertices
5
Enter the cost matrix
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0
Enter the source

1

GO TO MY WINDOW AND CLICK RIGHT BUTTON FOR NEXT OPTION
_
```

Fig 4.3: Enter input in terminal

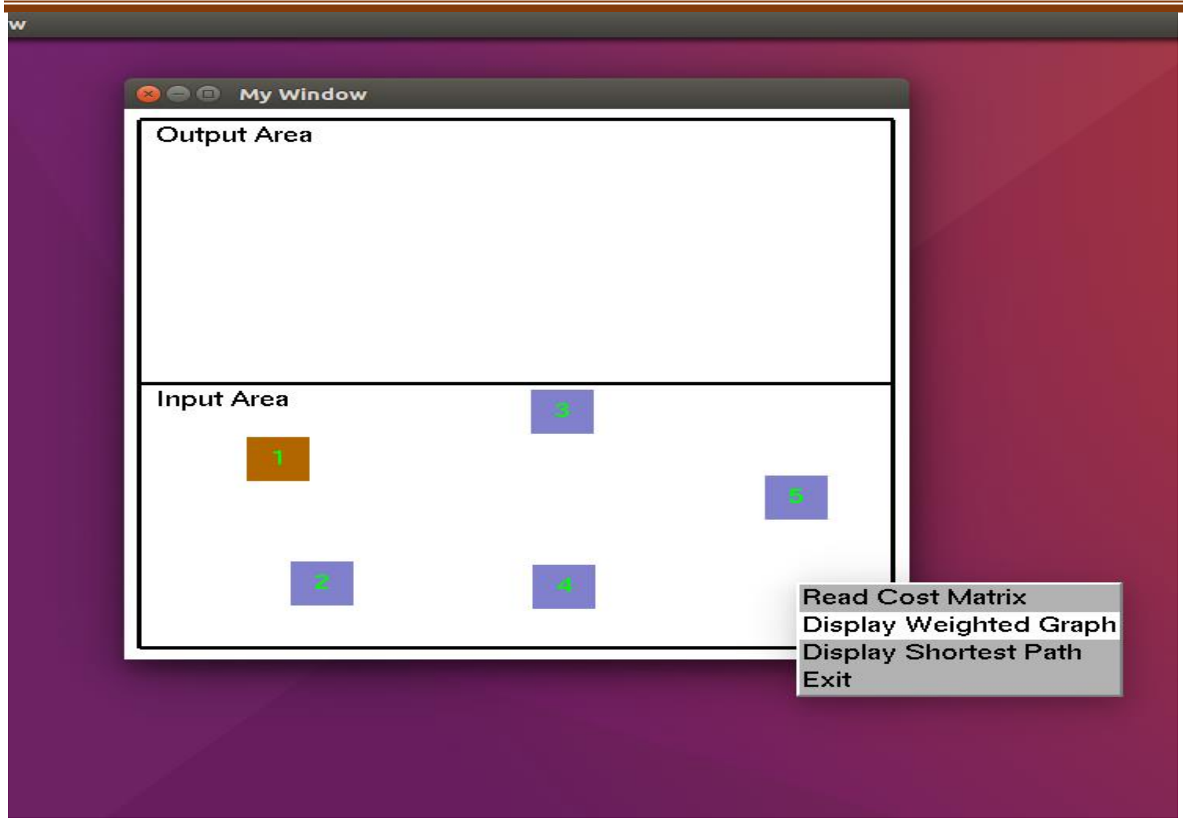


Fig 4.4: Choose the nodes by clicking the mouse

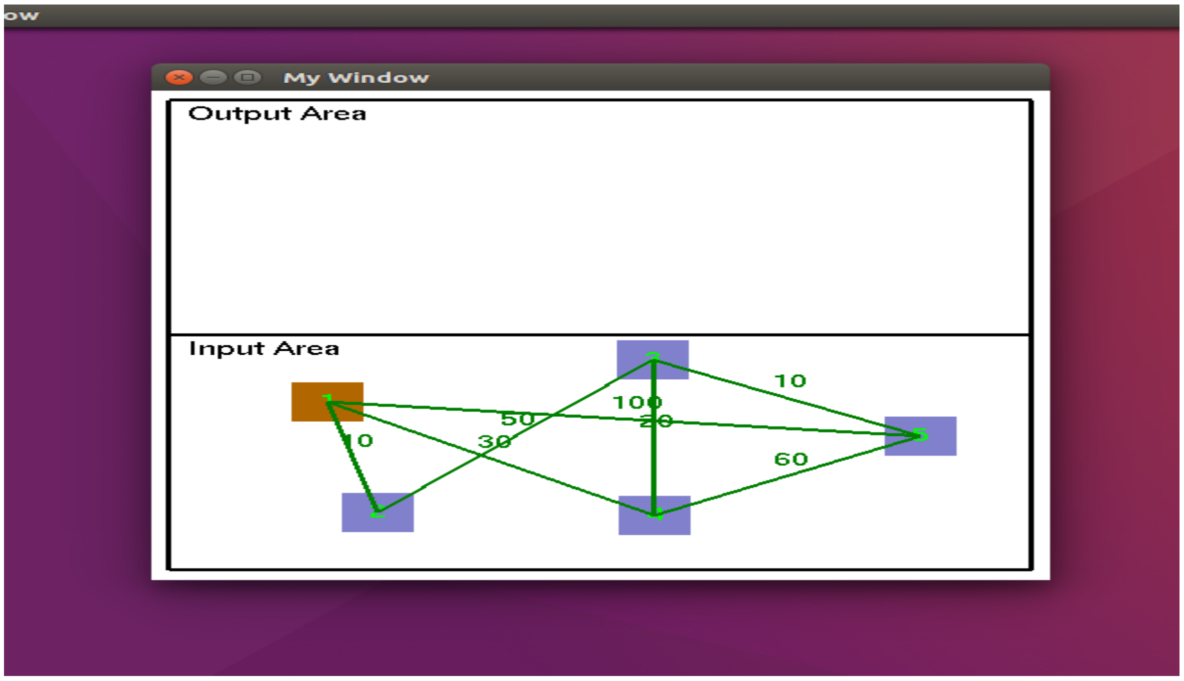


Fig 4.5: Display Weighted graph

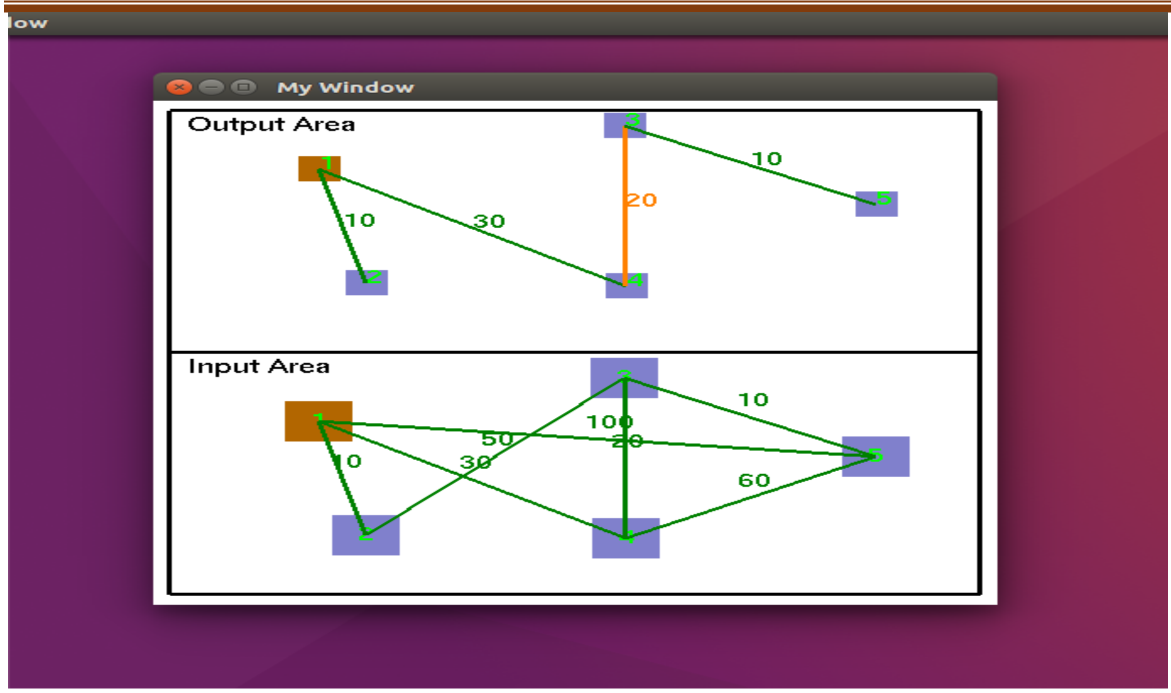
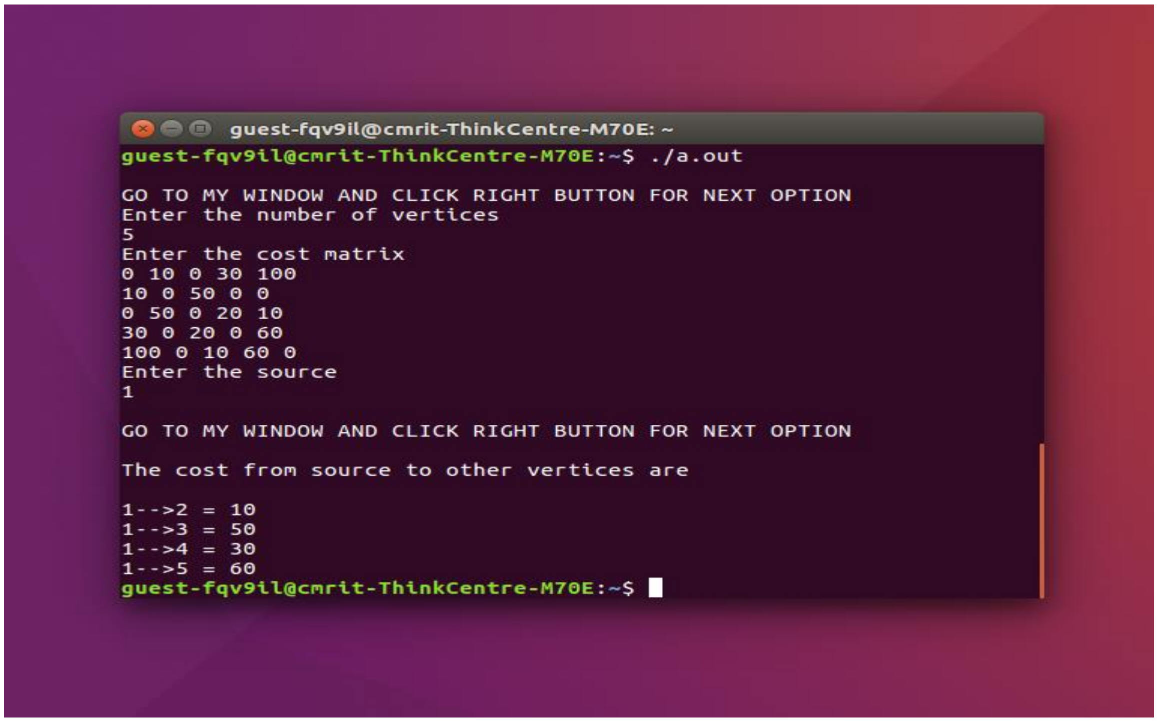


Fig 4.6: Display Shortest path



```
guest-fqv9il@cmrit-ThinkCentre-M70E:~$ ./a.out

GO TO MY WINDOW AND CLICK RIGHT BUTTON FOR NEXT OPTION
Enter the number of vertices
5
Enter the cost matrix
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0
Enter the source
1

GO TO MY WINDOW AND CLICK RIGHT BUTTON FOR NEXT OPTION

The cost from source to other vertices are

1-->2 = 10
1-->3 = 50
1-->4 = 30
1-->5 = 60
guest-fqv9il@cmrit-ThinkCentre-M70E:~$
```

Fig 4.7: Final Output in terminal

CHAPTER 5

CONCLUSION

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 algorithms 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.

REFERENCES

[1] “Computer Graphics with OpenGL” 4th edition, Donald D. Hearn, M. Pauline Baker, Warren Carithers

[2] http://www.opengl.org

[3] http://www.wikipedia.org

[4]http://www.openglprogramming.com

[5] Q&A’s on Stack Overflow - https://stackoverflow.com/questions/tagged/opengl