**CHAPTER 1**

**INTRODUCTION**

* 1. **Computer Graphics**

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

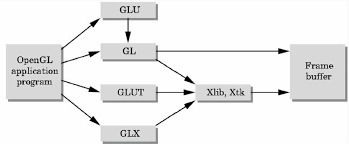
have no inherent geometry, such as survey results.

**1.2 INTRODUCTION TO OPENGL**

Most of our application will be designed to access OpenGL directly through functions in three libraries. Functions in the main GL (or OpenGL in windows) library have names that begin with the letters gl and are stored in a library usually referred to as GL (or OpenGL in windows). The second is the **OpenGL Utility Library** (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with letters glu.

To interface with the window system and to get input from external devices into our programs, we need at least one more system-specific library that provides the “glue” between the window system and OpenGL. For the X window system, this library is functionality that should be expected in any modern windowing system.

**Fig 2.1** shows the organization of the libraries for an X Window System environment. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus can be recompiled with the GLUT library for other window systems.

****

**Fig 1.2 Library organization**

**1.1.1: OpenGL Command Syntax**

OpenGL commands use the prefix **gl** and initial capital letters for each word making up the command name. Similarly, OpenGL defined constants begin with GL\_, use all capital letters and use underscores to separate words (like GL\_COLOR\_BUFFER\_BIT).

**1.3 Project Goal**

The aim of this project is to provide a virtual image for the combination of both structured and unstructured

information of “”. The package a user friendly interface.

**1.4 Scope of project**

It is developed in DEV C++. It has been implemented on WINDOWS platform. The 3-D graphics package designed here provides an interface for the users for handling the movement of spaceship and shoot the asteroids. The Keyboard and mouse is the main input device used.

**CHAPTER 2**

**Introduction**

**CHAPTER 3**

**REQUIREMENTS AND SPECIFICATION**

**Hardware Constraints**

* Processor : Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz 2.40 GHz
* Ram: 8.0 GB
* Display : VGA Color Monitor

**Software Constraints**

* Operating System : Windows 10/XP/Vista
* Language : Open Gl
* Compiler : Dev c++

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 Complete Source Code:**

#include<GL/glut.h>

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<string.h>

#include<unistd.h>

#define LIGHT\_ON 0

#define LIGHT\_OFF 1

int pos[16] = {10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85};

int peg[3] = {50,150,250};

int moves[10000][3];

int max\_moves;

int POLES[3][10];

int top[3]={-1,-1,-1};

int NUM\_DISKS=3;

int cnt,counter,speed=-500;

int line1=90,line2=85;

float ycoordinate;

int lightflag=1,animationFlag=1,randomColorFlag=0;

void push(int p,int disk)

{

POLES[p][++top[p]] = disk;

}

void pop(int p)

{

top[p]--;

}

void tower(int n,int src,int temp,int dst)

{

if(n>0)

{

tower(n-1,src,dst,temp);

moves[cnt][0] = n;

moves[cnt][1] = src;

moves[cnt][2] = dst;

cnt++;

tower(n-1,temp,src,dst);

}

}

void drawPegs()

{

int i;

glColor3f(0.5,0.0,0.1);

for(i=0;i<3;i++)

{

glPushMatrix();

glTranslatef(peg[i],5,0);

glRotatef(-90,1,0,0);

glutSolidCone(2,70,20,20);

glutSolidTorus(2,45, 20, 20);

glPopMatrix();

}

}

void printString(char \*text)

{

int len=strlen(text),i;

for(i=0;i<len;i++)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,text[i]);

}

void drawText()

{

glColor3f(1,1,1);

glRasterPos3f(-70,line1,0);

printString("Move :");

char str[5];

sprintf(str, "%d", counter);

glRasterPos3f(-40,line1,0);

printString(str);

glRasterPos3f(-70,line2,0);

printString("Disk");

char str1[10];

sprintf(str1, "%d", moves[counter][0]);

glRasterPos3f(-50,line2,0);

printString(str1);

glRasterPos3f(-40,line2,0);

printString("from");

char src[2];

if(moves[counter][1]==0)strcpy(src,"A");

else if(moves[counter][1]==1)strcpy(src,"B");

else strcpy(src,"C");

glRasterPos3f(-20,line2,0);

printString(src);

glRasterPos3f(-10,line2,0);

printString("to");

char dst[2];

if(moves[counter][2]==0)strcpy(dst,"A");

else if(moves[counter][2]==1)strcpy(dst,"B");

else strcpy(dst,"C");

glRasterPos3f(0,line2,0);

printString(dst);

glColor3f(0.6,0.3,0.5);

glBegin(GL\_POLYGON);

glVertex3f(-75,93,-5);

glVertex3f(-75,83,-5);

glVertex3f(10,83,-5);

glVertex3f(10,93,-5);

glEnd();

glColor3f(1,0,0);

glRasterPos3f(peg[0],70,0);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'A');

glRasterPos3f(peg[1],70,0);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'B');

glRasterPos3f(peg[2],70,0);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'C');

}

void drawSolved()

{

glColor3f(1,1,0);

glRasterPos3f(-60,87,0);

printString("Solved !!");

glColor3f(0.6,0.3,0.5);

glBegin(GL\_POLYGON);

glVertex3f(-75,93,-5);

glVertex3f(-75,83,-5);

glVertex3f(10,83,-5);

glVertex3f(10,93,-5);

glEnd();

glColor3f(1,0,0);

glRasterPos3f(peg[0],70,0);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'A');

glRasterPos3f(peg[1],70,0);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'B');

glRasterPos3f(peg[2],70,0);

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24,'C');

}

void display()

{

int i,j,k;

if(randomColorFlag)

glClearColor((rand()%100)/100.0,(rand()%100)/100.0,(rand()%100)/100.0,0);

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

if(lightflag)glEnable(GL\_LIGHTING);

glPushMatrix();

gluLookAt(0,ycoordinate,0,0,0,-1,0,1,0);

drawPegs();

for(i=0;i<3;i++)

{

k=0;

for(j=0;j<=top[i];j++)

{

glPushMatrix();

glTranslatef(peg[i],pos[k++],0);

glRotatef(90,1,0,0);

glColor3f(0.1\*POLES[i][j],0.2\*POLES[i][j],0);

glutSolidTorus(2.0, 4\*POLES[i][j], 20, 20);

glPopMatrix();

}

}

glPopMatrix();

glDisable(GL\_LIGHTING);

if(counter==max\_moves)

drawSolved();

else

drawText();

if(lightflag)glEnable(GL\_LIGHTING);

glutSwapBuffers();

}

void lighting()

{

GLfloat shininess[] = {50};

GLfloat white[] = {0.6,0.6,0.6,1};

glEnable(GL\_COLOR\_MATERIAL);

glColorMaterial(GL\_FRONT\_AND\_BACK,GL\_AMBIENT\_AND\_DIFFUSE);

GLfloat light\_ambient[] = { 0.0, 0.0, 0.0, 1.0 };

GLfloat light\_position[] = {100,60, 10, 0.0 };

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, white);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, white);

glMaterialfv(GL\_FRONT, GL\_SHININESS, shininess);

glEnable(GL\_LIGHT0);

}

void init()

{

glClearColor(0.0,0.0,0.0,0);

glColor3f(1,0,0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(-80,350,-10,100,-100,100);

glMatrixMode(GL\_MODELVIEW);

glEnable(GL\_DEPTH\_TEST);

lighting();

}

void animate(int n,int src,int dest)

{

int i;

if(speed<=0)speed=1;

for(i=pos[top[src]+1];i<90;i+=speed)

{

glPushMatrix();

glTranslatef(peg[src],i,0);

glRotatef(85,1,0,0);

glColor3f(0.1\*n,0.2\*n,0);

glutSolidTorus(2.0, 4\*n, 20, 20);

glPopMatrix();

glutSwapBuffers();

display();

}

if(peg[src]<peg[dest])

for(i=peg[src];i<=peg[dest];i+=speed)

{

glPushMatrix();

glTranslatef(i,90,0);

glRotatef(85,1,0,0);

glColor3f(0.1\*n,0.2\*n,0);

glutSolidTorus(2.0, 4\*n, 20, 20);

glPopMatrix();

glutSwapBuffers();

display();

}

else

for(i=peg[src];i>=peg[dest];i-=speed)

{

glPushMatrix();

glTranslatef(i,90,0);

glRotatef(85,1,0,0);

glColor3f(0.1\*n,0.2\*n,0);

glutSolidTorus(2.0, 4\*n, 20, 20);

glPopMatrix();

glutSwapBuffers();

display();

}

for(i=70;i>pos[top[dest]+1];i-=speed)

{

glPushMatrix();

glTranslatef(peg[dest],i,0);

glRotatef(85,1,0,0);

glColor3f(0.1\*n,0.2\*n,0);

glutSolidTorus(2.0, 4\*n, 20, 20);

glPopMatrix();

glutSwapBuffers();

display();

}

}

void mouse(int btn,int mode,int x,int y)

{

if(btn == 4 && mode == GLUT\_DOWN)

{

if(counter<max\_moves)

{

pop(moves[counter][1]);

if(animationFlag)

animate(moves[counter][0],moves[counter][1],moves[counter][2]);

push(moves[counter][2],moves[counter][0]);

counter++;

}

}

if(btn == 3 && mode == GLUT\_DOWN)

{

if(counter>0)

{

counter--;

pop(moves[counter][2]);

if(animationFlag)

animate(moves[counter][0],moves[counter][2],moves[counter][1]);

push(moves[counter][1],moves[counter][0]);

}

}

glutPostRedisplay();

}

void restart()

{

int i;

memset(POLES,0,sizeof(POLES));

memset(moves,0,sizeof(POLES));

memset(top,-1,sizeof(top));

cnt=0,counter=0;

ycoordinate=0.1;

max\_moves = pow(2,NUM\_DISKS)-1;

for(i=NUM\_DISKS;i>0;i--)

{

push(0,i);

}

tower(NUM\_DISKS,0,1,2);

}

void processMenuLighting(int option)

{

switch(option)

{

case LIGHT\_OFF:

glDisable(GL\_LIGHTING);

lightflag=0;

break;

case LIGHT\_ON:

glEnable(GL\_LIGHTING);

lightflag=1;

break;

}

glutPostRedisplay();

}

void processMenuMain2(int option)

{

}

void processMenuCamera(int option)

{

switch(option)

{

case 0:ycoordinate+=0.1;break;

case 1:ycoordinate-=0.1;break;

}

glutPostRedisplay();

}

void processMenuRestart(int option)

{

if(option==0)

{

restart();

glutPostRedisplay();

}

}

void processMenuExit(int option)

{

if(option==0)exit(0);

}

void processMenuAnimate(int option)

{

switch(option)

{

case 0:

animationFlag=1;

break;

case 1:

animationFlag=0;

}

}

void processMenuSolveCompletely(int option)

{

int temp=animationFlag;

animationFlag=0;

int i,j;

while(counter<max\_moves)

{

mouse(4,GLUT\_DOWN,0,0);

display();

for(i=0;i<100000;i++)

for(j=0;j<100;j++);

}

animationFlag=temp;

}

void processMenuBgColor(int option)

{

switch(option)

{

case 0:glClearColor(0,0,0,0);randomColorFlag=0;break;

case 1:glClearColor(1,1,1,0);randomColorFlag=0;break;

case 2:glClearColor(1,0,0,0);randomColorFlag=0;break;

case 3:glClearColor(0,1,0,0);randomColorFlag=0;break;

case 4:glClearColor(0,0,1,0);randomColorFlag=0;break;

case 5:randomColorFlag=1;break;

}

glutPostRedisplay();

}

void createGLUTMenus2()

{

int menu = glutCreateMenu(processMenuLighting);

glutAddMenuEntry("On",LIGHT\_ON);

glutAddMenuEntry("Off",LIGHT\_OFF);

int menuExit = glutCreateMenu(processMenuExit);

glutAddMenuEntry("Yes",0);

glutAddMenuEntry("No",1);

int menuCamera = glutCreateMenu(processMenuCamera);

glutAddMenuEntry("+0.1",0);

glutAddMenuEntry("-0.1",1);

int menuRestart = glutCreateMenu(processMenuRestart);

glutAddMenuEntry("Yes",0);

glutAddMenuEntry("No",1);

int menuAnimate = glutCreateMenu(processMenuAnimate);

glutAddMenuEntry("On",0);

glutAddMenuEntry("Off",1);

int menuBgColor = glutCreateMenu(processMenuBgColor);

glutAddMenuEntry("Black",0);

glutAddMenuEntry("White",1);

glutAddMenuEntry("Red",2);

glutAddMenuEntry("Green",3);

glutAddMenuEntry("Blue",4);

glutAddMenuEntry("Random",5);

int menuSolveCompletely = glutCreateMenu(processMenuSolveCompletely);

glutAddMenuEntry("Start",0);

glutCreateMenu(processMenuMain2);

glutAddSubMenu("Lighting",menu);

glutAddSubMenu("Move Camera",menuCamera);

glutAddSubMenu("Animation",menuAnimate);

glutAddSubMenu("Background Color",menuBgColor);

glutAddSubMenu("Solve Completely",menuSolveCompletely);

glutAddSubMenu("Restart",menuRestart);

glutAddSubMenu("Exit",menuExit);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

void processMenuMain1(int option)

{

}

void processMenuNumDisks(int option)

{

NUM\_DISKS=option;

restart();

glutPostRedisplay();

}

void createGLUTMenus1()

{

int menu = glutCreateMenu(processMenuNumDisks);

glutAddMenuEntry("3",3);

glutAddMenuEntry("4",4);

glutAddMenuEntry("5",5);

glutAddMenuEntry("6",6);

glutAddMenuEntry("7",7);

glutAddMenuEntry("8",8);

glutAddMenuEntry("9",9);

glutAddMenuEntry("10",10);

int menuExit = glutCreateMenu(processMenuExit);

glutAddMenuEntry("Yes",0);

glutAddMenuEntry("No",1);

glutCreateMenu(processMenuMain1);

glutAddSubMenu("Number of Disks",menu);

glutAddSubMenu("Exit",menuExit);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

void strokeString(float x,float y,float sx,float sy,char \*string,int width)

{

char \*c;

glLineWidth(width);

glPushMatrix();

glTranslatef(x,y,0);

glScalef(sx,sy,0);

for(c=string; \*c != '\0'; c++)

{

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, \*c);

}

glPopMatrix();

}

void initfirst()

{

glClearColor(0,0,0,0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0,1000,0,1000,-1,1);

glMatrixMode(GL\_MODELVIEW);

}

void first()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,0,0);

// strokeString(50,850,0.15,0.15,"BANGALORE INSTITUTE OF TECHNOLOGY",2);

// strokeString(100,750,0.3,0.3,"DEPARTMENT OF COMPUTER SCIENCE",4);

// strokeString(300,670,0.3,0.3,"AND ENGINEERING",4);

strokeString(200,500,0.3,0.3,"AN OPENGL MINI PROJECT ON",2);

glColor3f(0,1,1);

strokeString(250,420,0.4,0.4,"TOWER OF HANOI",6);

strokeString(150,220,0.2,0.2,"NUMBER OF DISKS:",3);

glColor3f(1,1,0);

char str[5];

sprintf(str, "%d", NUM\_DISKS);

strokeString(450,220,0.2,0.2,str,3);

glColor3f(1,0,0);

strokeString(50,100,0.17,0.17,"1 . Set the number of disks using the menu",2);

strokeString(50,50,0.17,0.17,"2 . Press (Enter) to start the simulation",2);

// strokeString(650,200,0.15,0.15,"By:",2);

// glColor3f(1,1,0);

// strokeString(650,160,0.18,0.18,"Dwarakanandan.B.M",2);

// strokeString(650,30,0.2,0.2,"Girija.J(Associate Professor)",2);

// glColor3f(1,0,0);

// strokeString(650,120,0.18,0.18,"1BI13CS052",2);

// strokeString(650,60,0.15,0.15,"Under the guidance of.",2);

glutSwapBuffers();

}

void keyboard2(unsigned char c, int x, int y){}

void keyboard(unsigned char c, int x, int y)

{

switch(c)

{

case 13:

restart();

init();

glutDisplayFunc(display);

createGLUTMenus2();

glutKeyboardFunc(keyboard2);

glutMouseFunc(mouse);

break;

}

glutPostRedisplay();

}

int main(int argc,char\*\* argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB);

glutInitWindowSize(1024,720);

glutInitWindowPosition(100,100);

glutCreateWindow("tower of hannoi");

initfirst();

glutDisplayFunc(first);

createGLUTMenus1();

glutKeyboardFunc(keyboard);

glutMainLoop();

return 0;

}

**4.2 SCREENSHOTS**

**CHAPTER 5**

**CONCLUSION**