National Public School   
Bengaluru

PROJECT CERTIFICATE2019-2020

This is to certify that Master Pratul V has successfully completed the project in *COMPUTER SCIENCE* prescribed by the *CBSE Board* on

3D Rendering Software

using C++ *for AISSCE* course in the Computer Science Lab of this school for the academic year 2019-2020

**Date : Signature of the Teacher Incharge**

**Name and Signature of the External Examiner**

**Name of the Candidate** Pratul V

**Reg.No** 18608861

**Examination Center**  National Public School, Rajajinagar, B’lore-10

**Date of the Practical Examination**

ACKNOWLEDGEMENT

I would like to wholeheartedly thank my Alma Mater, National Public School and our principal, Ms. Malathy Narayan, for providing us with both the opportunity and the facilities to undertake such a project. I would also express my gratitude to CBSE for making such an endeavor possible. Lastly, I am ever grateful to the faculty of the Computer Science Department for their invaluable guidance throughout.

ABOUT C++

C++ is a general purpose programming language with object oriented and high level features. It was designed by Bjarne Stroustrup in 1979 as an extension of the C programming language, providing high level features for program organization.

C++ introduces **Object Oriented Programming (OOP)** to C. The introduction of classes provides four features commonly present in OOP:

1. **Abstraction:** Allows the creation of user-defined data types that are suitable for modeling real world objects.
2. **Encapsulation:** Allows the combination of data and into associated operations into a single unit, for easier handling, and provides safety from external interference. Class members can be declared as private, public or protected to explicitly enforce encapsulation.
3. **Inheritance:** Allows the extension and reuse of existing code, for creation of a hierarchy of classes that simulate inheritance of properties in real world.
4. **Polymorphism:** In C++ this is implemented through function and operator overloading. It allows programmers to use the same name for different data, based on their signature.

INTRODUCTION

The aim of this project is to create a 3D Rendering Software that is capable of rendering a 3D model from a selected file and offers full perspective controls, that is, the user can move around in the 3D space, look around in all angles, zoom in and zoom out, etc. A new file type was also created that stores additional data defined by me that can be read by the software. As it was coded from scratch it offers complete freedom to what can be done, such as how a model is displayed, the interface for interacting with it, the inputs it can take, etc. It thus is a very useful tool for presentations and for displaying any 3D outputs.

DETAILS

Any surface can be approximately sub-divided into only triangle. Thus a 3D Object can be defined using only triangles. The data that has to be stored are the xyz coordinates of the vertices of all the triangles and for each triangle which of the 3 vertices link up to form it. Vertex index number is given by the order of adding vertex data into the array of objects of class vertex.

For rendering a 3D object the first step would be to calculate where each vertex would appear on the screen if as if the vertex was being looked at through the monitor screen. This is known as the projection of the vertex. The projection has to be defined in terms of xy coordinates. To calculate this projection following input parameters are required:

• Position of perspective in 3D space (defined with xyz coordinates)

• Direction of view of perspective (defined using 2 angles according to polar representation)

• Relative position of the vertex with respect to the position of perspective

Once projections of all vertices are calculated the 3D Object can be rendered by drawing the triangles between the set of 3 points whose coordinates are given by the projections.

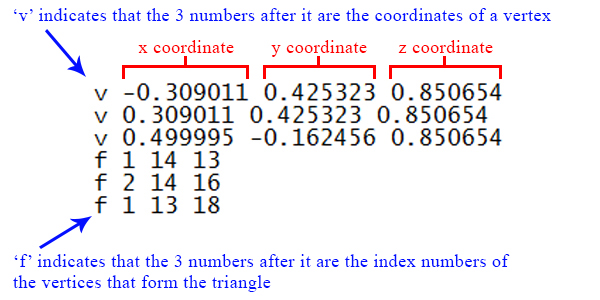
Other parameters to consider include drawing triangles furthest away first and then the triangles nearer to us, seeing whether a triangle would be to us or facing away from us(done by checking whether angle between the line joining the point of perspective to the midpoint of the triangle and the line representing the perpendicular to the flat surface of the triangle, is greater than 90 degrees)

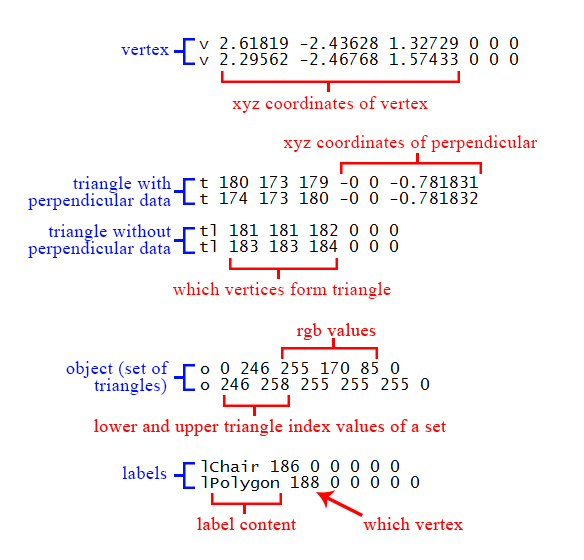
For shading angle of the perpendicular to the triangle to a point light source can be considered. For colour of an object this data can be stored additionally which tells which triangles have what colour.

Another feature that has been added to this project is the ability to place labels or text in the 3D space and whose data is also stored in the 3D model file.

File Handling:

A normal 3D model file that can be read by this program stores only the coordinates of the vertices and the index numbers of the vertices. This data is stored as lines of text in a .txt or a .obj (Wavefront obj) file. This is what and how data is being stored in the file.



For the purposes of my project I created a new file type with the extension .p3dm that stores additional data that was defined by and can be read by my program. This is how data is stored in the file.

Thus the program reads the selected file line by line and then calls appropriate functions to store values into the arrays of objects of the classes. It then calculates projections, draws triangles and waits for next input to determine which parameter has changed and accordingly calculate new projections and so the loop continues.

**Header Files:**

fstream

windows

graphics

time

ctype

math

**Structures:**

1. cord  
   To store xyz coordinates of float datatype.
2. xycord  
   To store xy coordinates of float datatype.
3. rgbcolor  
   To store red, green and blue values of a colour.

**Classes:**

1. **class vertex**
   1. void addvertex(float x, float y, float z)  
       This function is used to take 3 coordinates as input and store it in the data member of the object calling the function.
   2. void calc\_hp\_vp(cord ppos, float pha, float pva)  
      This function takes in position of perspective, direction of view of perspective(2 angle, pha expands to ‘Perspective’s Horizontal Angle’ and pva expands to ‘Perspective’s Vertical Angle’) and the calculates the projection of the vertex (object calling function) and stores it in data members hp and vp (horizontal projection and vertical projection) of the object.
2. **class triangle**
   1. Overloaded Functions:
      1. void addtriangle(int A, int B, int C, float x, float y, float z,bool AB, bool BC, bool CA)
      2. void addtriangle(int A, int B, int C, float x, float y, float z, bool perpend)  
         These functions put data into the data members of the object calling the function wherein the object represents a certain triangle. The former additionally invokes the calculate\_perpendicular() function while the latter had already received the perpendicular data.
   2. void refreshtriangles(triangle T[])  
      To update triangles once all data has been input i.e calculate perpendiculars and angles.
   3. void calculate\_midpoint()  
      To calculate the coordinates of the midpoint of the triangle.
   4. void calculate\_perpendicular()  
      To calculate the coordinates of perpendicular of the triangle relative to any of its 3 vertices.
   5. bool visibility(cord ppos)  
      To calculate angle between perpendicular and line joining midpoint to perspective so as to determine whether triangle is facing away from us or not.
   6. void calculate\_anglewithsun()  
      To calculate angle for purposes of shading.
   7. void calculate\_distance(cord ppos)  
      To calculate distance of triangle from position of perspective to determine which triangle has to be drawn first.
3. **class obj\_details**
   1. void addobject(int l, int u, int r, int g, int b)  
      Function to take in the lower index and upper index values of set of triangles with color defined by the r,g,b parameters(red, green and blue.)
   2. rgbcolor findcolor(obj\_details O[500],int sei)  
      Function to receive a index value of triangle and identify which set it belongs to and thus returns the color of the triangle.
4. **class labels**
   1. void addlabel(char textinput[],int lx)  
      Function to take in label content as input and index number of vertex where label has been placed in the 3D space.
5. **class filehandling**
   1. void createconvertedfile()  
      To convert a .txt or .obj file to the .p3dm file format that can be read by the software.
   2. void selectfile()  
      To invoke another exe “Select File.exe” that puts the address of a file selected using windows dialog into a text file “Command.txt” that is then read.
   3. int importfile()  
      To import data from selected file and once finished return 1 or 0 of if import has been successful or not.
   4. void savefile()  
      To put all data present in objects of all classes into a file with .p3dm format.
6. **class menu**
   1. void displaymenu()  
      To draw the menu.
   2. int comparemouseclick (int m\_x, int m\_y)  
      To take in xy coordinates of mouse click and identify which region it is in so as to invoke function accordingly.
   3. void displayinstructions()  
      To print text content from “Instructions.txt”

**Data Files:**

1. .p3dm (.dat) filetype
2. .txt filetype
3. Command.txt
4. Instructions.txt

PROJECT CODE

#include<windows.h>

#include<graphics.h>

#include<math.h>

#include<ctype.h>

#include<fstream>

#include<time.h>

#define pi 3.141592f

using namespace std;

//global variables

int fovp; //max screen coordinates

float fov; //max field of view

bool gtest1;//for switching between showing outward/toward faces of solid object

bool gtest2(1);//for stopping execution of rendering window

bool gtest3(0);//for showing lines of triangles

int i,j;

int v=0,t=0,o=0,l=0;

struct cord{

float x,y,z;

};

struct xycord{

float x;

float y;

};

struct rgbcolor{

unsigned int R,G,B;

};

cord campos;//camera position

void calccampos(float pha, float pva){

float dist;

dist=cos(pva)\*1;

campos.x=cos(pha)\*dist;

campos.y=sin(pha)\*dist;

campos.z=sin(pva)\*1;

}

class vertex{

public:

cord point;

int hp,vp; //coordinates of projection of point on screen

bool disp\_or\_not;

void addvertex(float x, float y, float z){

point.x=x;

point.y=y;

point.z=z;

v++;

}

void calc\_hp\_vp(cord ppos, float pha, float pva) {

float fhp,fvp;

float ha;

disp\_or\_not=1;

//calculation of relative position

cord spoint;

spoint.x=point.x-ppos.x;

spoint.y=point.y-ppos.y;

spoint.z=point.z-ppos.z;

//angles 0 and 90

bool do\_ang\_calc=1;

if(spoint.y==0&&spoint.x<0){

ha=pi;

do\_ang\_calc=0;

}

else if(spoint.y==0&&spoint.x>0){

ha=0;

do\_ang\_calc=0;

}

else if(spoint.x==0&&spoint.y>0){

ha=pi/2;

do\_ang\_calc=0;

}

else if(spoint.x==0&&spoint.y<0){

ha=-pi/2;

do\_ang\_calc=0;

}

//angle calculation

if(do\_ang\_calc){

ha=atan(spoint.y/spoint.x);

if(ha<0)

ha=-ha;

if(spoint.x<0 && spoint.y<0)

ha=-pi+ha;

else if(spoint.x>0 && spoint.y<0)

ha=-ha;

else if(spoint.x<0 && spoint.y>0)

ha=pi-ha;

}

if(ha<-pi)

ha=2\*pi+ha;

//relative horizontal angle accounting for quadrants

if(pha>0){

if(ha>0)

ha=ha-pha;

else

{

if(ha<(pha-pi))

ha=2\*pi-pha+ha;

else

ha=-pha+ha;

}

}

else {

if(ha<0)

ha=ha-pha;

else{

if(ha>(pha+pi))

ha=-2\*pi-pha+ha;

else

ha=ha-pha;

}

}

float magA,magB,angle;

magA=1;

//camera radius

magB=spoint.x\*spoint.x+spoint.y\*spoint.y+spoint.z\*spoint.z;

magB=sqrt(magB);

//cosine of angle

angle=(campos.x\*spoint.x+campos.y\*spoint.y+campos.z\*spoint.z)/(magA\*magB);

//angle is acos of above equation

angle=acos(angle);

bool skip(1);

if(!(angle>=-pi/2 && angle<=pi/2))//for checking if object is behind camera{

disp\_or\_not=hp=vp=0;

skip=0;

}

if(skip){

float t2;//refer rotationpolar

t2=spoint.x\*spoint.x+spoint.y\*spoint.y;

t2=sqrt(t2);

t2=cos(ha)\*t2;

float ae;//angle of elevation

ae=atan(spoint.z/t2);

fvp=tan(ae-pva)\*0.1;

float hypo;//hypotenuse of triangle formed by hp and vp

hypo=tan(angle)\*0.1;

if(hypo<0)

hypo=-hypo;

fhp=hypo\*hypo-fvp\*fvp;

fhp=sqrt(fhp);

if(ha<0)

fhp=-fhp;

if(!(fvp>=-0.2246 && fvp<=0.2246) || !(fhp>=-0.2246 && fhp<=0.2246))

disp\_or\_not=0;

fhp=fhp\*fovp/(tan(fov)\*0.1);

fvp=fvp\*fovp/(tan(fov)\*0.1);

hp=fhp;

vp=fvp;

}

}

}V[1200];

class triangle{

public:

int A,B,C; //index values reffering to an element of vertex array

bool AB,BC,CA; //whether line should be displayed or not

cord tri\_p,midp; //perpendicular to plane of triangle and midpoint of triangle

bool solid; //whether triangle belongs to solid object or not

float angle, anglewithsun;

float distance;

triangle(){

AB=BC=CA=1;

}

void addtriangle(int A, int B, int C, float x, float y, float z,bool AB, bool BC, bool CA){

triangle::A=A;

triangle::B=B;

triangle::C=C;

triangle::AB=AB;

triangle::BC=BC;

triangle::CA=CA;

solid=0;

tri\_p.x=x;

tri\_p.y=y;

tri\_p.z=z;

::t++;

calculate\_midpoint();

}

void addtriangle(int A, int B, int C, float x, float y, float z, bool perpend){

triangle::A=A;

triangle::B=B;

triangle::C=C;

solid=1;

AB=BC=CA=1;

tri\_p.x=x;

tri\_p.y=y;

tri\_p.z=z;

if(perpend)

calculate\_perpendicular();

::t++;

}

void refreshtriangles(triangle T[]){

for(i=0;i<t;i++){

if(T[i].solid){

T[i].calculate\_midpoint();

// T[i].calculate\_perpendicular();

T[i].calculate\_anglewithsun();

}

}

}

void calculate\_midpoint(){

midp.x=(V[A].point.x+V[B].point.x+V[C].point.x)/3;

midp.y=(V[A].point.y+V[B].point.y+V[C].point.y)/3;

midp.z=(V[A].point.z+V[B].point.z+V[C].point.z)/3;

}

void calculate\_perpendicular(){

//perpendicular calculator using cross product AB AC

cord relB, relC;

//relative position of B

relB.x=V[B].point.x-V[A].point.x;

relB.y=V[B].point.y-V[A].point.y;

relB.z=V[B].point.z-V[A].point.z;

//relative position of C

relC.x=V[C].point.x-V[A].point.x;

relC.y=V[C].point.y-V[A].point.y;

relC.z=V[C].point.z-V[A].point.z;

tri\_p.x=relB.y\*relC.z-relC.y\*relB.z;

tri\_p.y=-relB.x\*relC.z+relC.x\*relB.z;

tri\_p.z=relB.x\*relC.y-relC.x\*relB.y;

}

bool visibility(cord ppos){

cord line;

//A=line joining ppos(camera) to midpoint of triangle

line.x=ppos.x-midp.x;

line.y=ppos.y-midp.y;

line.z=ppos.z-midp.z;

//tri\_p is the other line whose center is already relative to midpoint of triangle

//calculation of angle using dot product

float magA,magB;

magA=line.x\*line.x+line.y\*line.y+line.z\*line.z;

magA=pow(magA,0.5);

magB=tri\_p.x\*tri\_p.x+tri\_p.y\*tri\_p.y+tri\_p.z\*tri\_p.z;

magB=pow(magB,0.5);

//cosine of angle

angle=(line.x\*tri\_p.x+line.y\*tri\_p.y+line.z\*tri\_p.z)/(magA\*magB);

//angle is acos of above equation

angle=acos(angle);

if(gtest1)

angle=pi-angle;

if(angle<(pi/2) && angle>(-pi/2))

return 1;

else

return 0;

}

void calculate\_anglewithsun(){

cord line;

//A=line joining sun to midpoint of triangle

line.x=1;

line.y=3;

line.z=2;

//tri\_p is the other line whose center is already relative to midpoint of triangle

//calculation of angle using dot product

float magA,magB;

magA=line.x\*line.x+line.y\*line.y+line.z\*line.z;

magA=pow(magA,0.5);

magB=tri\_p.x\*tri\_p.x+tri\_p.y\*tri\_p.y+tri\_p.z\*tri\_p.z;

magB=pow(magB,0.5);

//cosine of angle

anglewithsun=(line.x\*tri\_p.x+line.y\*tri\_p.y+line.z\*tri\_p.z)/(magA\*magB);

//angle is acos of above equation

anglewithsun=acos(anglewithsun);

if(gtest1)

anglewithsun=pi-anglewithsun;

}

void calculate\_distance(cord ppos){

//calculation of relative position

cord spoint;

spoint.x=midp.x-ppos.x;

spoint.y=midp.y-ppos.y;

spoint.z=midp.z-ppos.z;

distance=spoint.x\*spoint.x+spoint.y\*spoint.y+spoint.z\*spoint.z;

distance=sqrt(distance);

}

}T[1200];

int pv(0); //pv is the no. of existing vertices before new set of trangles are added

int objc(0);

rgbcolor whit={255,255,255};

class obj\_details{

public:

int lower,upper;

rgbcolor OC;

void addobject(int l, int u, int r, int g, int b){

lower=l;

upper=u;

OC.R=r;

OC.G=g;

OC.B=b;

o++;

}

rgbcolor findcolor(obj\_details O[500],int sei){

objc=0;

while(objc<=o){

if(i>=O[objc].lower && i<O[objc].upper)

return O[objc].OC;

objc++;

}

return whit;

}

}O[500];

class labels{

public:

int vertexno;

char label\_text[30];

void addlabel(char textinput[],int lx){

vertexno=lx;

for(i=0;i<strlen(textinput);i++)

textinput[i]=textinput[i+1];

strcpy(label\_text,textinput);

l++;

}

}L[100];

void orient\_motion(float pha, cord &ppos, char ch){

int negate(-1);

if(toupper(ch)=='W'||toupper(ch)=='S') {

if(toupper(ch)=='W')

negate\*=-1;

if(pha>=0 && pha<=(pi/2)){

ppos.x+=0.1\*cos(pha)\*negate;

ppos.y+=0.1\*sin(pha)\*negate;

}

else if(pha>(pi/2) && pha<=pi){

ppos.x-=0.1\*sin(pha-pi/2)\*negate;

ppos.y+=0.1\*cos(pha-pi/2)\*negate;

}

else if(pha<0 && pha>=(-pi/2)){

ppos.x+=0.1\*cos(-pha)\*negate;

ppos.y-=0.1\*sin(-pha)\*negate;

}

else if(pha<(-pi/2) && pha>(-pi)){

ppos.x-=0.1\*sin(-pha-pi/2)\*negate;

ppos.y-=0.1\*cos(-pha-pi/2)\*negate;

}

}

else if(toupper(ch)=='A'||toupper(ch)=='D'){

if(toupper(ch)=='A')

negate\*=-1;

if(pha>=0 && pha<=(pi/2)){

ppos.x-=0.1\*sin(pha)\*negate;

ppos.y+=0.1\*cos(pha)\*negate;

}

else if(pha>(pi/2) && pha<=pi){

ppos.x-=0.1\*cos(pha-pi/2)\*negate;

ppos.y-=0.1\*sin(pha-pi/2)\*negate;

}

else if(pha<0 && pha>=(-pi/2)){

ppos.x+=0.1\*sin(-pha)\*negate;

ppos.y+=0.1\*cos(-pha)\*negate;

}

else if(pha<(-pi/2) && pha>-pi){

ppos.x+=0.1\*cos(-pha-pi/2)\*negate;

ppos.y-=0.1\*sin(-pha-pi/2)\*negate;

}

}

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

T[i].calculate\_distance(ppos);

}

int maxx=1000,maxy=1000; //screen dimensions

class filehandling{

char file\_location[180];

public:

//function to convert obj/txt file to new .p3dm file

void createconvertedfile(){

fstream imp\_,exp\_;

imp\_.open(file\_location,ios::in);

int slen;

slen=strlen(file\_location);

file\_location[slen-1]='t';

file\_location[slen-2]='a';

file\_location[slen-3]='d';

exp\_.open(file\_location,ios::out);

char type[50];

float n1,n2,n3,n4,n5,n6;

while(imp\_ >> type){

if(type[0]=='v' && type[1]==0){

imp\_>>n1>>n2>>n3;

exp\_<<'v'<<' '<<n1<<' '<<n2<<' '<<n3<<" 0 0 0"<<'\n';

}

else if(type[0]=='f'&& type[1]==0){

imp\_>>n1>>n2>>n3;

exp\_<<'t'<<' '<<n1<<' '<<n2<<' '<<n3<<" 0 0 0"<<'\n';

}

// T[t].addtriangle(int(n1)-1,int(n2)-1,int(n3)-1,1);

}

imp\_.close();

exp\_.close();

}

//function to put address of selected file into text file "Command.txt"

void selectfile(){

fstream cmd;

cmd.open("Command.txt",ios::out);

cmd<<"%";

cmd.close();

bool flag1(0);

ShellExecute( NULL, NULL, "Select File.exe", NULL, NULL, SW\_SHOW );

ifstream location\_;

while(!flag1){

location\_.open("Command.txt");

file\_location[0]=0;

location\_.getline(file\_location,180);

//junk eliminator

int j;

for(i=0;i<strlen(file\_location);i++)

if(file\_location[i]==':'){

for(j=i-1;j<=strlen(file\_location);j++)

file\_location[j-i+1]=file\_location[j];

break;

}

flag1=1;

if(file\_location[0]=='%'){

Sleep(200);

flag1=0;

}

location\_.close();

}

//cout<<"Press Y to convert .obj file format to .3dp file format if file is .obj format else press N to continue"<<endl;

//convertfile();

}

int importfile(){

bool flag(0);

while(!flag){

flag=1;

if(file\_location[0]=='%'){

settextstyle(3,0,4);

char str[50]="Error | Select file again";

outtextxy(maxx-300,maxy,str);

flag=0;

selectfile();

}

}

//identification of .p3dm file

char fextension[5];

for(i=0;i<strlen(file\_location);i++){

if(file\_location[i]=='.'){

for(int j=i+1;j<=strlen(file\_location);j++){

fextension[j-i-1]=file\_location[j];

}

}

}

if(strcmpi("obj",fextension)==0||strcmpi("txt",fextension)==0){

{

cleardevice();

settextstyle(3,0,4);

char str[100]="PRESS Y TO CONVERT AND CREATE A NEW FILE OF .p3dm FILE FORMAT";

outtextxy(maxx/2-75,maxy-50,str);

}

char yornochoice;

yornochoice=getch();

if(toupper(yornochoice)=='Y')

createconvertedfile();

else{

cleardevice();

settextstyle(3,0,4);

char str[50]="ERROR IN FILE TYPE. SELECT AGAIN";

outtextxy(maxx-300,maxy,str);

return 1;

}

}

else if(strcmpi("p3dm",fextension)==0){

char datfileloc[180];

strcpy(datfileloc,file\_location);

int slend;

slend=strlen(datfileloc);

datfileloc[slend-1]=0;

datfileloc[slend-2]='t';

datfileloc[slend-3]='a';

datfileloc[slend-4]='d';

rename(file\_location,datfileloc);

file\_location[slend-1]=0;

file\_location[slend-2]='t';

file\_location[slend-3]='a';

file\_location[slend-4]='d';

}

else{

{

cleardevice();

settextstyle(3,0,4);

char str[50]="ERROR IN FILE TYPE. SELECT AGAIN";

outtextxy(maxx-300,maxy,str);

}

return 1;

}

char type[50];

float n1,n2,n3,n4,n5,n6;

ifstream file\_(file\_location);

while(file\_ >> type >> n1 >> n2 >> n3 >> n4 >> n5 >> n6){

if(type[0]=='v')

V[v].addvertex(n1,n2,n3);

else if(type[0]=='t' && n4==n5 && n5==n6 && n6==0){

if(type[1]=='l')

T[t].addtriangle(int(n1)-1+pv,int(n3)-1+pv,int(n2)-1+pv,n4,n5,n6,0,1,0);

else

T[t].addtriangle(int(n1)-1+pv,int(n3)-1+pv,int(n2)-1+pv,n4,n5,n6,1);

}

else if(type[0]=='t'){

T[t].addtriangle(int(n1)-1+pv,int(n3)-1+pv,int(n2)-1+pv,n4,n5,n6,0);

}

else if(type[0]=='o' )

O[o].addobject(n1,n2,n3,n4,n5);

else if(type[0]=='l')

L[l].addlabel(type,n1);

// T[t].addtriangle(int(n1)-1,int(n2)-1,int(n3)-1,1);

}

T[0].refreshtriangles(T);

file\_.close();

char p3dmfileloc[180];

strcpy(p3dmfileloc,file\_location);

int slen;

slen=strlen(p3dmfileloc);

p3dmfileloc[slen+1]=0;

p3dmfileloc[slen]='m';

p3dmfileloc[slen-1]='d';

p3dmfileloc[slen-2]='3';

p3dmfileloc[slen-3]='p';

rename(file\_location,p3dmfileloc);

return 0;

}

void savefile()

{

fstream cmd;

cmd.open("Command.txt",ios::out);

cmd<<"%";

cmd.close();

bool flag1(0);

ShellExecute( NULL, NULL, "Save File.exe", NULL, NULL, SW\_SHOW );

ifstream location\_;

char p3dmfileloc[180],datfileloc[180];

while(!flag1){

location\_.open("Command.txt");

p3dmfileloc[0]=0;

location\_.getline(p3dmfileloc,180);

//junk eliminator

int j;

for(i=0;i<strlen(p3dmfileloc);i++)

if(p3dmfileloc[i]==':'){

for(j=i-1;j<=strlen(p3dmfileloc);j++)

p3dmfileloc[j-i+1]=p3dmfileloc[j];

break;

}

flag1=1;

if(p3dmfileloc[0]=='%'){

Sleep(200);

flag1=0;

}

location\_.close();

}

strcpy(datfileloc,p3dmfileloc);

datfileloc[strlen(datfileloc)-4]=0;

strcat(datfileloc,"dat");

ofstream exp\_(datfileloc);

char labeltemp[30];

for(i=0;i<l;i++){

labeltemp[0]='l';

labeltemp[1]=0;

strcat(labeltemp,L[i].label\_text);

int vno=L[i].vertexno+1;

exp\_<<labeltemp<<' '<<L[i].vertexno+1<<" 0 0 0 0 0"<<'\n';

}

exp\_.close();

remove(p3dmfileloc);

rename(datfileloc,p3dmfileloc);

}

}F;

class menu{

public:

void displaymenu(){

char str[40];

settextstyle(3,0,7);

strcpy(str,"3D RENDERING SOFTWARE");

outtextxy(maxx/6,maxy-250,str);

settextstyle(3,0,4);

strcpy(str,"BY PRATUL VENKATESH");

outtextxy(maxx/6,maxy-170,str);

strcpy(str,"MENU");

outtextxy(maxx/6,maxy-90,str);

strcpy(str,"OPEN FILE");

outtextxy(maxx/6+30,maxy,str);

strcpy(str,"INSTRUCTIONS");

outtextxy(maxx/6+30,maxy+90,str);

strcpy(str,"EXIT PROGRAM");

outtextxy(maxx/6+30,maxy+180,str);

setfillstyle(0,BLACK);

rectangle(maxx/6,maxy-15,maxx/6+325,maxy+50);

rectangle(maxx/6,maxy+75,maxx/6+325,maxy+140);

rectangle(maxx/6,maxy+165,maxx/6+325,maxy+230);

}

int comparemouseclick (int m\_x, int m\_y){

if(m\_x>maxx/6 && m\_x<maxx/6+325 && m\_y>maxy-15 && m\_y<maxy+50)

return 1;

else if(m\_x>maxx/6 && m\_x<maxx/6+325 && m\_y>maxy+75 && m\_y<maxy+140)

return 2;

else if(m\_x>maxx/6 && m\_x<maxx/6+325 && m\_y>maxy+165 && m\_y<maxy+230)

return 3;

else

return 0;

}

void displayinstructions(){

ifstream f("Instructions.txt");

int lspacing=maxy/10+50;

char str[80];

settextstyle(3,0,5);

f.getline(str,80,'\n');

outtextxy(maxx/6,lspacing,str);

lspacing+=40;

settextstyle(3,0,3);

while(f.getline(str,80,'\n')){

outtextxy(maxx/6,lspacing,str);

lspacing+=30;

}

}

}M;

int main(){

bool menu(1);

char ch;

cord ppos={-5,5,1.5};

float pha=-pi/4; //angle of view in xy plane

float pva=0; //vertical factor

calccampos(pha,pva);

rgbcolor rgb; //for shading

int fn(40); //fov

int poly[8]; //for solid fill

char\* label\_text;

int shading\_factor;

int order[1200];

int pos;

j=0;

int no\_of\_triangles\_to\_display(0);

float max,pmax(100);

int m\_x, m\_y;//mouse click x and y

gtest1=1;

bool mouse\_continuity(0);

initwindow(maxx,maxy,"3D Viewer");

//to maximize window

HWND hWnd = ::GetActiveWindow();

ShowWindow( hWnd, SW\_MAXIMIZE );

do{

fov=fn\*pi/180;

maxx=getmaxx();

maxx/=2;

maxy=getmaxy();

maxy/=2;

if(menu){

int menuchoice(0);

cleardevice();

M.displaymenu();

int m\_x(0),m\_y(0);

while(!(menuchoice>=1 && menuchoice<=3)){

if (ismouseclick(WM\_LBUTTONDOWN))

getmouseclick(WM\_LBUTTONDOWN, m\_x, m\_y);

if (ismouseclick(WM\_LBUTTONUP))

clearmouseclick(WM\_LBUTTONUP);

menuchoice=M.comparemouseclick(m\_x,m\_y);

delay(50);

}

if(menuchoice==1){

/\*1. Open File\*/

F.selectfile();

while(F.importfile())

{

F.selectfile();

}

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

T[i].calculate\_distance(ppos);

menu=0;

}

else if(menuchoice==2){

cleardevice();

M.displayinstructions();

getch();

menuchoice=0;

}

else if(menuchoice==3){

return 0;

}

}

else{

if(maxx>maxy)

fovp=maxx;

else

fovp=maxy;

for(i=0;i<v;i++){

V[i].calc\_hp\_vp(ppos,pha,pva);

}

//sorting

j=0;

pmax=100;

no\_of\_triangles\_to\_display=0;

while(j<t){

max=0;

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

if(T[i].distance>max && T[i].distance<pmax){

max=T[i].distance;

pos=i;

}

pmax=max;

if(T[pos].visibility(ppos)==1 || !T[pos].solid){

order[no\_of\_triangles\_to\_display]=pos;

no\_of\_triangles\_to\_display++;

}

j++;

}

//drawing triangles

cleardevice();

for(j=0;j<no\_of\_triangles\_to\_display;j++){

i=order[j];

rgb=O[0].findcolor(O,i);

if(T[i].solid && (V[T[i].A].disp\_or\_not && V[T[i].B].disp\_or\_not && V[T[i].C].disp\_or\_not)){

//shading

if(T[i].angle<0)

shading\_factor=255-255\*((-T[i].anglewithsun/(pi/2))\*0.5f);

else

shading\_factor=255-255\*((T[i].anglewithsun/(pi/2))\*0.5f);

rgb.R=rgb.R\*shading\_factor/255;

rgb.G=rgb.G\*shading\_factor/255;

rgb.B=rgb.B\*shading\_factor/255;

setcolor((0x04000000 | RGB(rgb.R,rgb.G,rgb.B)));

setfillstyle(1,(0x04000000 | RGB(rgb.R,rgb.G,rgb.B)));

poly[0]=maxx-V[T[i].A].hp;

poly[1]=maxy-V[T[i].A].vp;

poly[2]=maxx-V[T[i].B].hp;

poly[3]=maxy-V[T[i].B].vp;

poly[4]=maxx-V[T[i].C].hp;

poly[5]=maxy-V[T[i].C].vp;

fillpoly(3,poly);

}

else{

setcolor((0x04000000 | RGB(rgb.R,rgb.G,rgb.B)));

setfillstyle(1,(0x04000000 | RGB(rgb.R,rgb.G,rgb.B)));

if(T[i].AB && V[T[i].A].disp\_or\_not && V[T[i].B].disp\_or\_not)

line(maxx-V[T[i].A].hp,maxy-V[T[i].A].vp,maxx-V[T[i].B].hp,maxy-V[T[i].B].vp);

if(T[i].BC && V[T[i].B].disp\_or\_not && V[T[i].C].disp\_or\_not)

line(maxx-V[T[i].B].hp,maxy-V[T[i].B].vp,maxx-V[T[i].C].hp,maxy-V[T[i].C].vp);

if(T[i].CA && V[T[i].C].disp\_or\_not && V[T[i].A].disp\_or\_not)

line(maxx-V[T[i].C].hp,maxy-V[T[i].C].vp,maxx-V[T[i].A].hp,maxy-V[T[i].A].vp);

}

if(gtest3 && T[i].solid){

//lines

setcolor(WHITE);

if(T[i].AB && V[T[i].A].disp\_or\_not && V[T[i].B].disp\_or\_not)

line(maxx-V[T[i].A].hp,maxy-V[T[i].A].vp,maxx-V[T[i].B].hp,maxy-V[T[i].B].vp);

if(T[i].BC && V[T[i].B].disp\_or\_not && V[T[i].C].disp\_or\_not)

line(maxx-V[T[i].B].hp,maxy-V[T[i].B].vp,maxx-V[T[i].C].hp,maxy-V[T[i].C].vp);

if(T[i].CA && V[T[i].C].disp\_or\_not && V[T[i].A].disp\_or\_not)

line(maxx-V[T[i].C].hp,maxy-V[T[i].C].vp,maxx-V[T[i].A].hp,maxy-V[T[i].A].vp);

}

}

//label rednering

for(j=0;j<l;j++){

if(V[L[j].vertexno].disp\_or\_not){

label\_text=L[j].label\_text;

settextstyle(3,HORIZ\_DIR,2);

setcolor(WHITE);

outtextxy(maxx-V[L[j].vertexno].hp,maxy-V[L[j].vertexno].vp,label\_text);

setcolor(YELLOW);

setfillstyle(1,YELLOW);

poly[0]=maxx-V[L[j].vertexno].hp-6;

poly[1]=maxy-V[L[j].vertexno].vp+8;

poly[2]=maxx-V[L[j].vertexno].hp-6;

poly[3]=maxy-V[L[j].vertexno].vp+14;

poly[4]=maxx-V[L[j].vertexno].hp-12;

poly[5]=maxy-V[L[j].vertexno].vp+14;

poly[6]=maxx-V[L[j].vertexno].hp-12;

poly[7]=maxy-V[L[j].vertexno].vp+8;

fillpoly(4,poly);

}

}

setcolor(YELLOW);

line(maxx+10,maxy,maxx-10,maxy);

line(maxx,maxy+10,maxx,maxy-10);

{

settextstyle(3,0,1);

char str[30]="Press H for HELP";

outtextxy(20,maxy\*2-30,str);

}

delay(10);

ch=getch();

switch (toupper(ch))

{

case 'W':

orient\_motion(pha,ppos,ch);

break;

case 'S':

orient\_motion(pha,ppos,ch);

break;

case 'A':

orient\_motion(pha,ppos,ch);

break;

case 'D':

orient\_motion(pha,ppos,ch);

break;

case 'E':

ppos.z=ppos.z+0.1;

break;

case 'F':

ppos.z=ppos.z-0.1;

break;

case 'J':

if(pha<pi)

pha+=pi/180;

calccampos(pha,pva);

break;

case 'L':

if(pha>-pi)

pha-=pi/180;

calccampos(pha,pva);

break;

case 'K':

if(pva>-pi/2)

pva-=pi/180;

calccampos(pha,pva);

break;

case 'I':

if(pva<pi/2)

pva+=pi/180;

calccampos(pha,pva);

break;

case 'G':

if(fn<55)

fn+=1;

break;

case 'R':

if(fn>29)

fn-=1;

break;

case 'H':

setcolor(BLACK);

setfillstyle(1,BLACK);

poly[0]=maxx/6-40;

poly[1]=maxy/10+30;

poly[2]=maxx/6-40;

poly[3]=maxy+200;

poly[4]=maxx/6+510;

poly[5]=maxy+200;

poly[6]=maxx/6+510;

poly[7]=maxy/10+30;

fillpoly(4,poly);

setcolor(YELLOW);

M.displayinstructions();

getch();

break;

case 'Q':

if(gtest1)

gtest1=0;

else

gtest1=1;

break;

case 'M':

return 0;

break;

case 'B':

if(gtest3)

gtest3=0;

else

gtest3=1;

break;

default:

break;

}

orient\_motion(pha,ppos,ch);

//excess horizontal orienation correction

if(pha>pi)

pha=-2\*pi+pha;

else if(pha==-pi || pha<-pi)

pha=2\*pi+pha;

//break;

}

}while(gtest2);

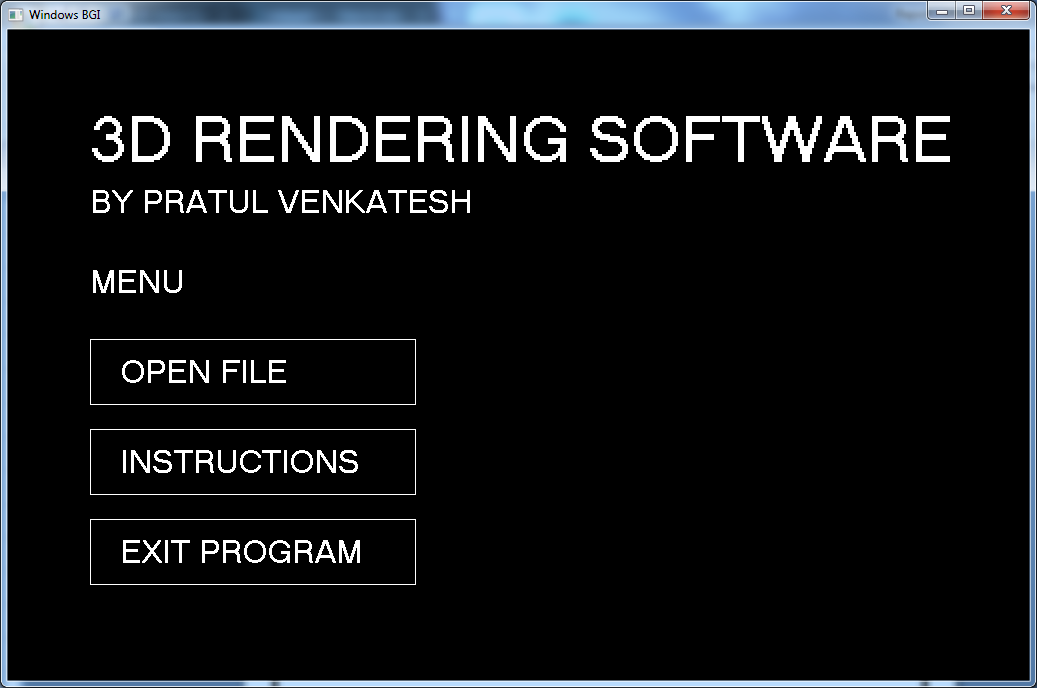
ShowWindow( hWnd, SW\_SHOW );

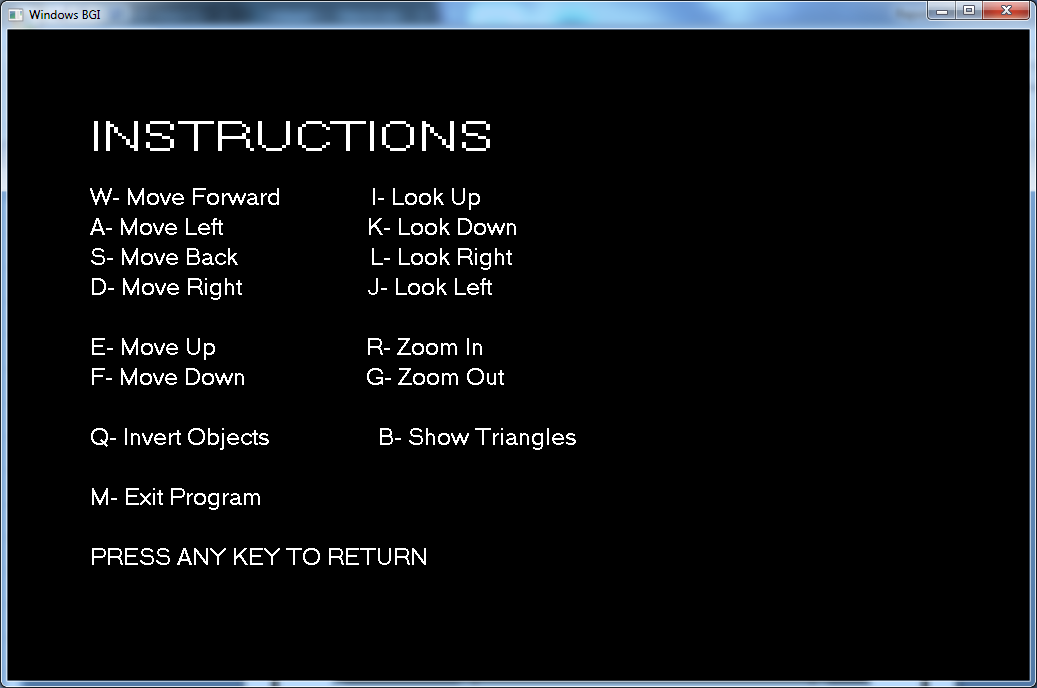
getch();

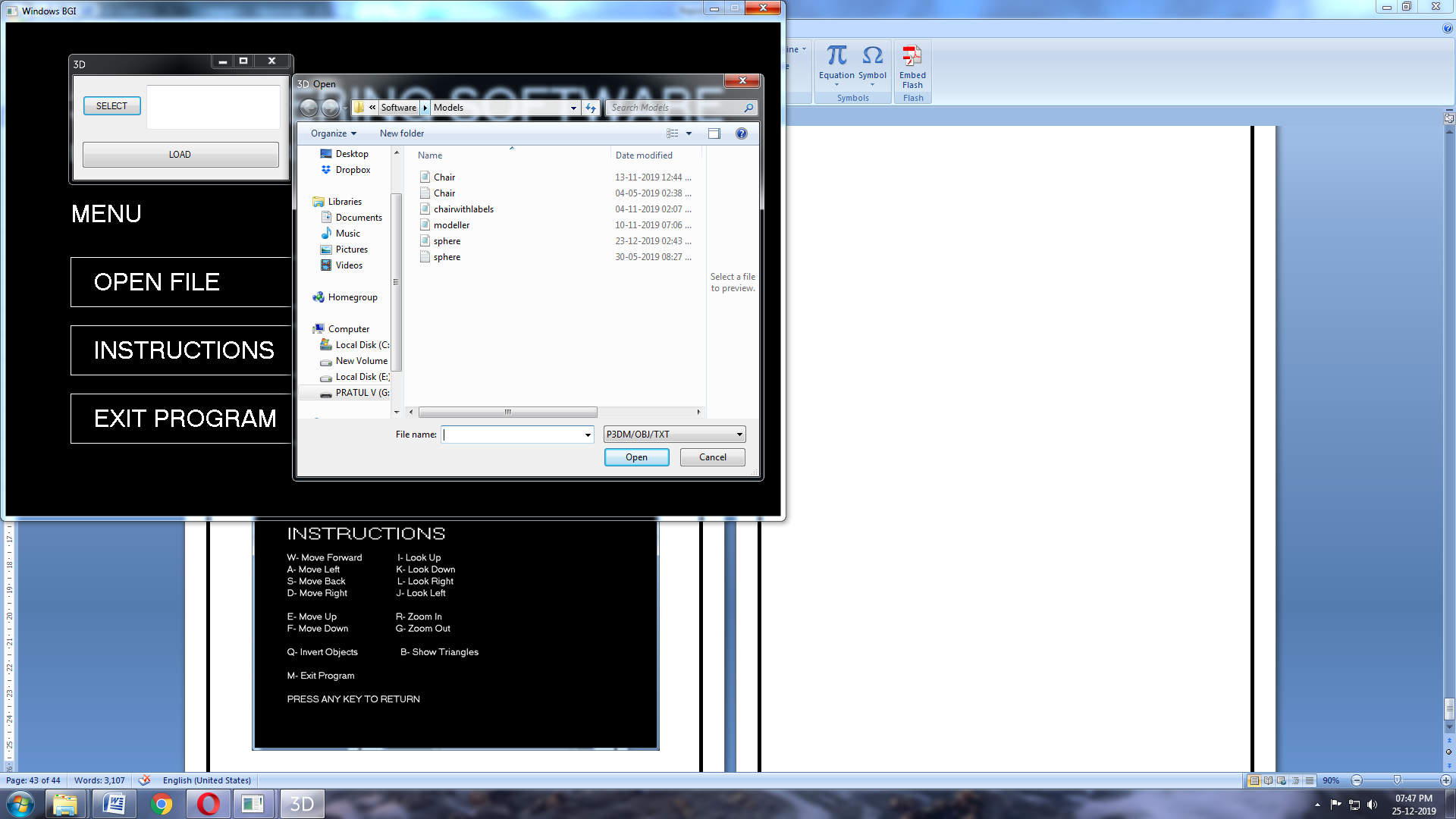
return 0;

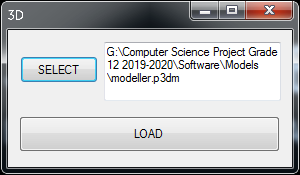
}

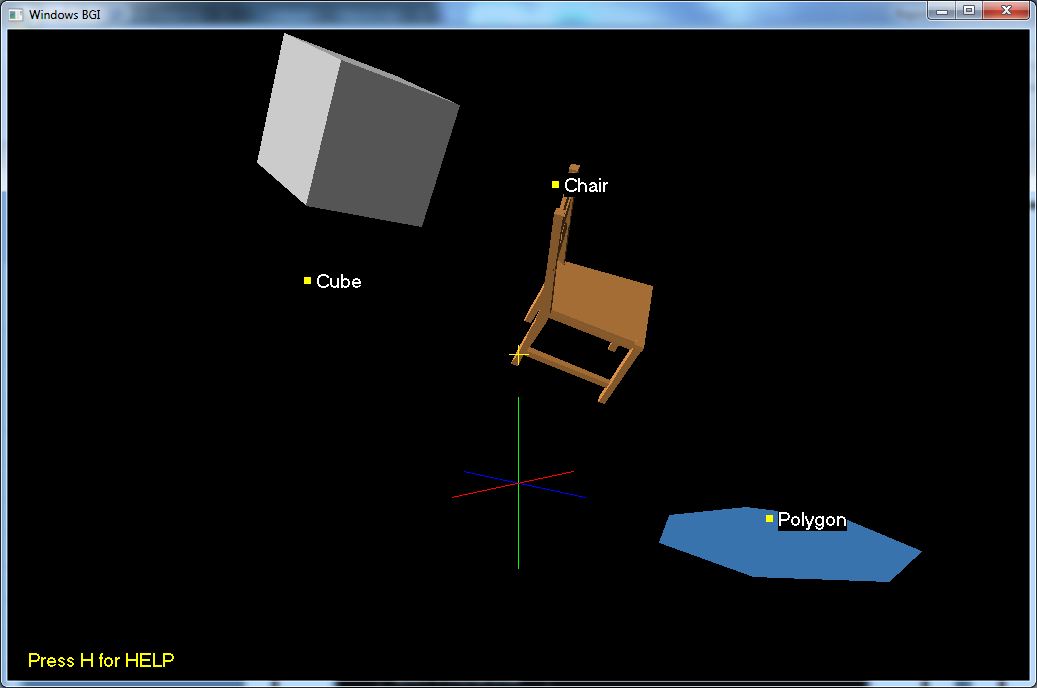
SAMPLE OUTPUT

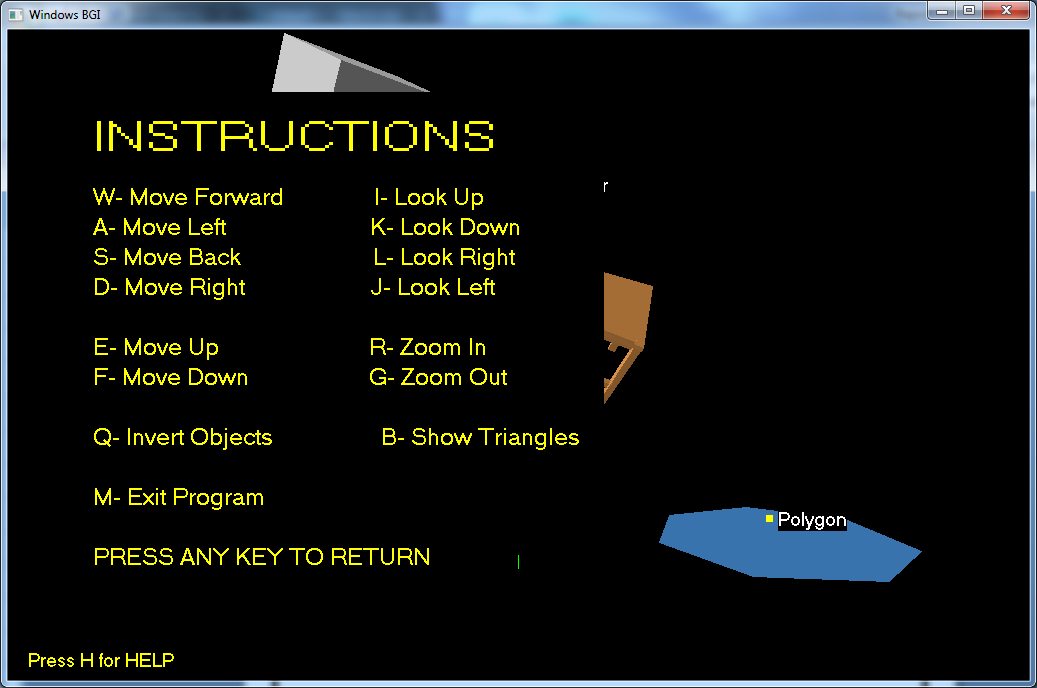


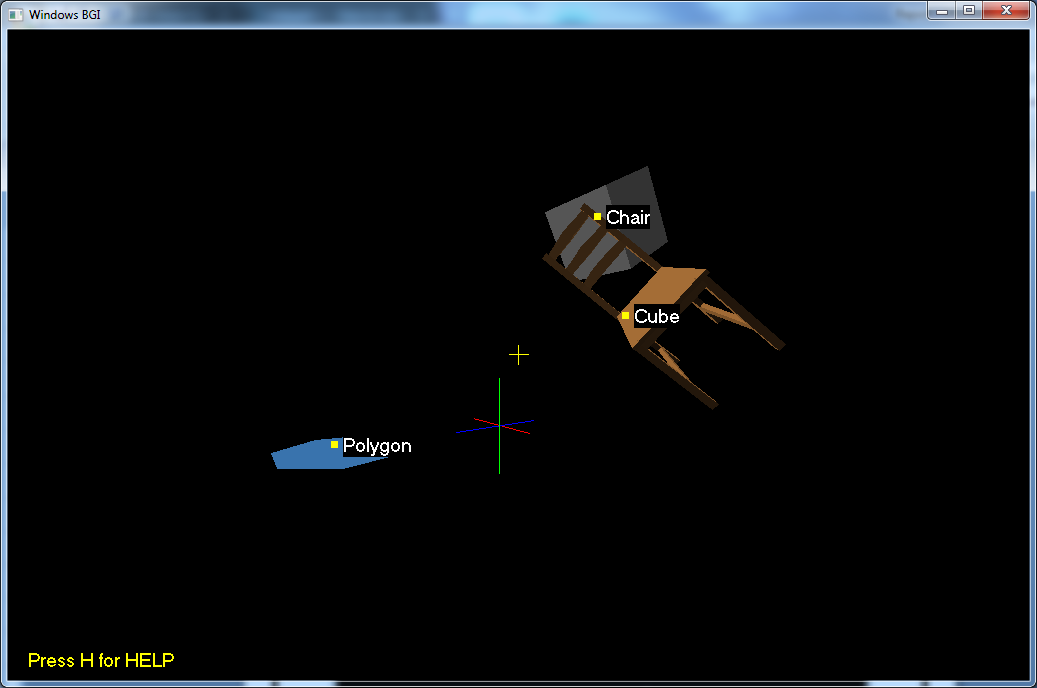


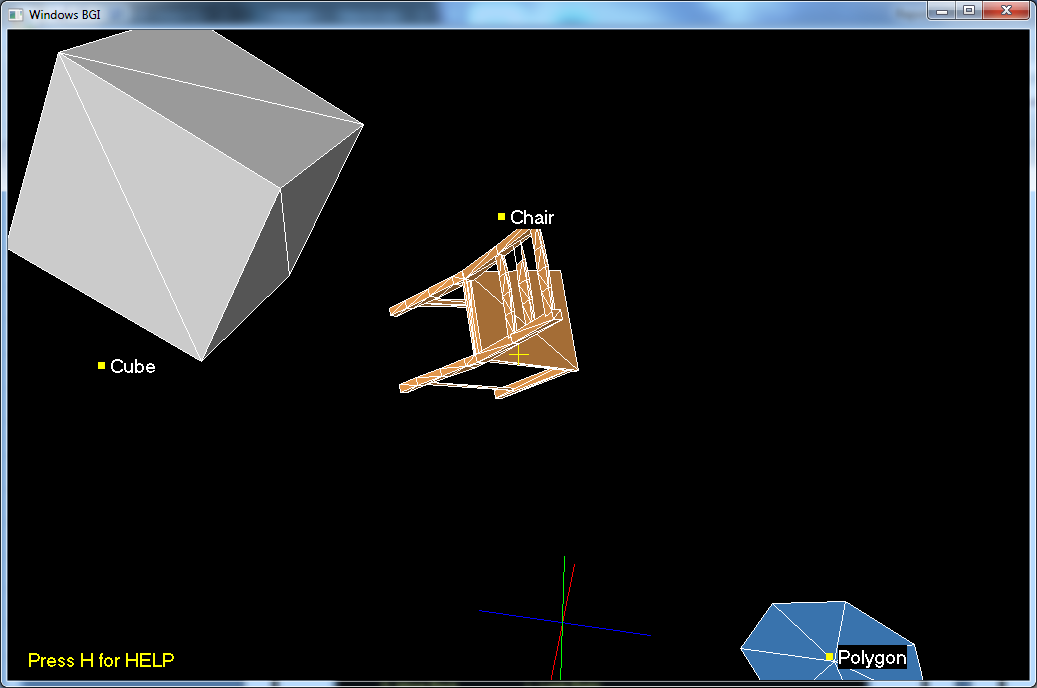












BIBLIOGRAPHY

* ***“Borland Graphics Interface (BGI) Documentation”*** https://www.cs.colorado.edu/~main/cs1300/doc/bgi/index.html
* https://www.geeksforgeeks.org
* ***“Computer Science with C++”*** by Sumita Arora