**CGV MINI PROJECT CODE**

/\*An Interactive Program to create 3d objects\*/

#include <windows.h>

#include<string.h>

#include<stdarg.h>

#include<stdio.h>

//#include<math.h>

#include <GL/glut.h>

// co ordinates of eight packets

static double x=0,y=0,z=0,i1=0;

// p[8] is used to generate 8 different packets, 4 for each user. usind the keys 1-8

// u[8] is used to Allow or deny packets using menu,

//the order is for "user 1" u[0-3]={http,telnet,ssh,smtp} & for "user 2" u[0-3]={http,telnet,ssh,smtp}

static bool p[8]={false}, u[8]={false};

//t1=false,sh1=false,sm1=false;

//static bool tt2=false,h2=false,sh2=false,sm2=false;

void \*font;

void \*currentfont;

void setFont(void \*font)

{

currentfont=font;

}

void drawstring(float x,float y,float z,char \*string)

{

char \*c;

glRasterPos3f(x,y,z);

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

{ glColor3f(0.0,1.0,1.0);

glutBitmapCharacter(currentfont,\*c);

}

}

voidstroke\_output(GLfloat x, GLfloat y, char \*format,...)voidstroke\_output(GLfloat x, GLfloat y, char \*format,...)

{

va\_list args;

char buffer[200], \*p;

va\_start(args, format);

vsprintf(buffer, format, args);

va\_end(args);

glPushMatrix();

glTranslatef(-2.5, y, 0);

glScaled(0.003, 0.005, 0.005);

for (p = buffer; \*p; p++)

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

glPopMatrix();

}

void server()

{

glPushMatrix();

glScaled(0.8,2.0,0.8);

glTranslatef(-0.5,0,-15);

glColor3f(0,1.5,1.5);

glutSolidCube(1);

glPushMatrix();

glScaled(0.5,.1,1.0);

glTranslatef(.0,3.5,0.01);

glColor3f(0.3,0.3,0.3);

glutSolidCube(1.5);

glPopMatrix();

glPushMatrix();

glScaled(0.35,.05,1.0);

glTranslatef(0.0,1,0.1);

glColor3f(0,0,0.3);

glutSolidCube(1.1);

glPopMatrix();

glPushMatrix();

glScaled(2.2,.1,1.0);

glTranslatef(0,-1,0.3);

glColor3f(0,0,1.3);

glutSolidCube(0.5);

glPopMatrix();

glPushMatrix();

glScaled(2.2,.1,1.0);

glTranslatef(0,-2,0.3);

glColor3f(0,0,1.3);

glutSolidCube(0.5);

glPopMatrix();

glPushMatrix();

glScaled(2.2,.1,1.0);

glTranslatef(0,-3,0.3);

glColor3f(0,0,1.3);

glutSolidCube(0.5);

glPopMatrix();

glPopMatrix();

}

void plane()

{

glScaled(0.2,0.1,0.3);

if(p[0] || p[4])

glColor3f(1,1,0);

if(p[1] || p[5])

glColor3f(0,1,0);

if(p[2] || p[6])

glColor3f(1,0,0);

if(p[3] || p[7])

glColor3f(0,1,1);

glutSolidSphere(1.0,40,40);

glPushMatrix();

glColor3f(0,0,0);

glTranslatef(0,0,0);

glScaled(3,0.3,0.01);

glutSolidSphere(1.0,40,40);

glPopMatrix();

glPushMatrix();

glColor3f(0,0,0);

glTranslatef(0,2,0.8);

glScaled(0.2,3,0.01);

glutSolidSphere(1.0,40,40);

glPopMatrix();

}

void wall(){

glPushMatrix();

glTranslatef(-1.5,-0.05,-6);

glColor3f(1,1,1);

glScaled(4.7,3.05,0.2);

glutSolidCube(1);

glPopMatrix();

for(float y=-5;y<=5;y+=0.65){

for(float x=-5.3; x<=1;x+=0.57){

glPushMatrix();

glColor3f(0.4,0.4,0.4);

glScaled(.7,0.3,0.5);

glTranslatef(x,y,-12);

glutSolidCube(0.5);

glPopMatrix();

}

}

}

void user()

{

glPushMatrix();

glColor3f(1,1,1);

glTranslatef(0,1.2,1);

//glScaled(0.3,0.35,0.01);

glutSolidSphere(0.4,40,40);

glPopMatrix();

glPushMatrix();

glColor3f(1,1,1);

glTranslatef(0,0,1);

glScaled(0.6,1.7,0.6);

glutSolidSphere(0.6,40,40);

glPopMatrix();

glPushMatrix();

glColor3f(1,1,1);

glTranslatef(0.35,-0.1,1);

glScaled(0.2,2,0.2);

glRotatef(90,1,0,0);

glutSolidTorus(0.2,0.4,40,40);

glPopMatrix();

glPushMatrix();

glColor3f(1,1,1);

glTranslatef(-0.3,-0.1,1.2);

glScaled(0.2,2,0.2);

glRotatef(90,1,0,0);

glutSolidTorus(0.2,0.4,40,40);

glPopMatrix();

}

//PC Computers

void pc()

{

glPushMatrix();

glTranslatef(0,0,3);

glScaled(.7,.7,0);

glColor3f(0,1,0);

glutSolidCube(1);

glPushMatrix();

glColor3f(1,0,0);

glTranslatef(0,-0.65,0);

glScaled(1.1,0.3,0);

glutSolidCube(1);

glPushMatrix();

glColor3f(1,1,0);

glTranslatef(0,0,0);

glScaled(0.6,0.15,0);

glutSolidCube(1);

glPopMatrix();

glPopMatrix();

glPushMatrix();

glColor3f(1,1,1);

glTranslatef(0,0,3);

glScaled(.7,.7,0);

glutSolidCube(1);

glPopMatrix();

glPopMatrix();

}

void port(float x1,float y1){

glPushMatrix();

glTranslatef(x1,y1,-5.85);

glScaled(0.4,0.45,0.01);

glutSolidSphere(1.0,40,40);

glPushMatrix();

glColor3f(0,0,0);

glTranslatef(0,0,1);

glScaled(1,1,0.01);

glutSolidSphere(0.8,40,40);

glPopMatrix();

glPopMatrix();

}

void packet(){

glPushMatrix();

glColor3f(1,0,1);

// Move packet to spring

if(z<=10){

z+=0.1;

glTranslatef(-2,-2,5-z);

plane();

}

// http packet

if(p[0] && z>=10 || p[4] && z>=10 ){

//towards port 80

if(z>=10 && y<=3)

{

x-=0.01;

y+=0.02;

z+=0.01;

glTranslatef(-2+x,-2+y,5-z);

plane();

//glutSolidCube(0.3);

}else if

//Allow or Block

(p[0] && u[0] || p[4] && u[4] )

{if(z<=17){

z+=0.1;

}

if(z>=17)

{

z=150;

}glTranslatef(-2+x,-2+y,5-z);

plane();

//glutSolidCube(0.3);

}else

{

i1+=0.1;

for(float x4=0;x4<=3;x4++)

{

glPushMatrix();

glScalef(0.3,0.3,0.3);

if(x4==0)

glTranslatef(-7.5+i1,12+i1,-5.85);

if(x4==1)

glTranslatef(-7.5-i1,12+i1,-5.85);

if(x4==2)

glTranslatef(-7.5+i1,12-i1,-5.85);

if(x4==3)

glTranslatef(-7.5-i1,12-i1,-5.85);

//plane();

glutSolidCube(0.3);

glPopMatrix();

}

}

}

// Telnet Packets

if(p[1] && z>=10 || p[5] && z>=10 ){

//towards port 23

if(z>=10 && y<=3)

{

x-=0.002;

y+=0.01;

z+=0.005;

glTranslatef(-2+x,-2+y,5-z);

plane();

//glutSolidCube(0.3);

}else if

//Allow or Block

(p[1] && u[1] || p[5] && u[5] )

{if(z<=17){

z+=0.1;

}

if(z>=17)

{

z=150;

}glTranslatef(-2+x,-2+y,5-z);

plane();

}else

{

i1+=0.1;

for(float x4=0;x4<=3;x4++)

{

glPushMatrix();

glScalef(0.3,0.3,0.3);

if(x4==0)

glTranslatef(-5.6+i1,12+i1,-5.85);

if(x4==1)

glTranslatef(-5.6+i1,12-i1,-5.85);

if(x4==2)

glTranslatef(-5.6-i1,12+i1,-5.85);

if(x4==3)

glTranslatef(-5.6-i1,12-i1,-5.85);

glutSolidCube(0.3);

glPopMatrix();

}

}

}

// SSH Packets

if(p[2] && z>=10 || p[6] && z>=10 ){

//towards port 23

if(z>=10 && y<=3)

{

x+=0.002;

y+=0.01;

z+=0.005;

glTranslatef(-1.8+x,-2+y,5-z);

plane();

}else if

//Allow or Block

(p[2] && u[2] || p[6] && u[6] )

{if(z<=17){

z+=0.1;

}

if(z>=17)

{

z=150;

}glTranslatef(-1.8+x,-2+y,5-z);

plane();

}else

{

i1+=0.1;

for(float x4=0;x4<=3;x4++)

{

glPushMatrix();

glScalef(0.3,0.3,0.3);

if(x4==0)

glTranslatef(-3.6+i1,12+i1,-5.85);

if(x4==1)

glTranslatef(-3.6+i1,12-i1,-5.85);

if(x4==2)

glTranslatef(-3.6-i1,12+i1,-5.85);

if(x4==3)

glTranslatef(-3.6-i1,12-i1,-5.85);

glutSolidCube(0.3);

glPopMatrix();

}

}

}

// SMTP Packets

if(p[3] && z>=10 || p[7] && z>=10 ){

//towards port 23

if(z>=10 && y<=2)

{

x+=0.0081;

y+=0.01;

z+=0.005;

glTranslatef(-2+x,-2+y,5-z);

plane();

}else if

//Allow or Block

(p[3] && u[3] || p[7] && u[7] )

{ if(z<=17){

z+=0.1;

}

if(z>=17)

{

z=150;

}

glTranslatef(-2+x,-2+y,5-z);

plane();

}

else

{

i1+=0.1;

for(float x4=0;x4<=3;x4++)

{

glPushMatrix();

glScalef(0.3,0.3,0.3);

if(x4==0)

glTranslatef(-1.5+i1,11+i1,-5.85);

if(x4==1)

glTranslatef(-1.5+i1,11-i1,-5.85);

if(x4==2)

glTranslatef(-1.5-i1,11+i1,-5.85);

if(x4==3)

glTranslatef(-1.5-i1,11-i1,-5.85);

glutSolidCube(0.3);

glPopMatrix();

}

}

}

glPopMatrix();

}

void spring(){

glPushMatrix();

glColor3f(0,1,1);

glTranslatef(-2.3,-2.5,-5.3);

if(z>=9.6 && z<=10.2){

glScaled(0.3,-1.5+z/10,0.3);

}

glScaled(0.4,1.5,0.4);

glRotatef(90,1,0,0);

glutWireTorus(0.25,0.4,30,4);

glPopMatrix();

}

void switch1()

{

glPushMatrix();

glColor3f(0,1,1.2);

glTranslatef(-2,-2,5);

glScaled(1.5,0.5,2);

glutSolidCube(0.5);

glPopMatrix();

}

void info()

{

glPushMatrix();

glTranslatef(0,0,-15);

//Port 80

glColor3f(1,1,0);

port(6.8,4.1);

//Port 22 lower right

glColor3f(0.1,1,0.1);

port(6.8,3.1);

//Port 25 left

glColor3f(1,0,0);

port(6.8,2.1);

// Port 23

glColor3f(0,1,1);

port(6.8,1.1);

glPushMatrix();

setFont(GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,0);

drawstring(4,2.9,0.0,"(80)");

setFont(GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,0);

drawstring(3.3,2.9,0.0,"HTTP");

setFont(GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,0);

drawstring(4,2.1,0.0,"(23)");

setFont(GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,0);

drawstring(3,2.1,0.0,"TELNET");

setFont(GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,0);

drawstring(4,1.4,0.0,"(22)");

setFont(GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,0);

drawstring(3.5,1.4,0.0,"SSH");

setFont(GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,0);

drawstring(4,0.7,0.0,"(25)");

setFont(GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,0);

drawstring(3.3,0.7,0.0,"SMTP");

setFont(GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,0);

drawstring(-15.1,-3.8,-2.0,"SWITCH");

setFont(GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,0);

drawstring(-15.1,-0.8,-2.0,"USER 1");

setFont(GLUT\_BITMAP\_TIMES\_ROMAN\_24);

glColor3f(0,0,0);

drawstring(-15.1,-7,-6.5,"USER 2");

glPopMatrix();

glPopMatrix();

}

// Draw WALL

void firewall()

{

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

glLoadIdentity();

info();

if(z<=15){

glTranslatef(0.0f,0.0-z/4,-10.0+z/8);

glRotatef(z\*3,0.1f,0.0f,0.0f);

}else

{

glTranslatef(0.0f,-3.75,-8.125);

glRotatef(45,0.1f,0.0f,0.0f);

}

glPushMatrix();

glScaled(0.5,0.5,0.5);

//Wall

wall();

switch1();

//Port 80

glColor3f(0.1,1,0.1);

port(-2.2,0.1);

//Port 22 lower right

glColor3f(1,0,0);

port(-1.4,-0.1);

//Port 25 left

glColor3f(1,1,0);

port(-3,0.2);

// Port 23

glColor3f(0,1,1);

port(-0.5,-0.0);

for(int i=0;i<8;i++){

if(p[i])

packet();

}

spring();

server();

glPushMatrix();

glTranslatef(-1,0,0);

server();

glPopMatrix();

glPushMatrix();

glTranslatef(-2,0,0);

server();

glPopMatrix();

//PC 1

glPushMatrix();

glTranslatef(-2.35,-3,1);

pc();

glPopMatrix();

//PC 2

glPushMatrix();

glTranslatef(-0.8,-3,1);

pc();

glPopMatrix();

//PC 3

glPushMatrix();

glTranslatef(-4,-3,1);

pc();

glPopMatrix();

//USER 1

glPushMatrix();

glTranslatef(-4.2,-3,5);

glScaled(0.4,0.4,0.4);

user();

glPopMatrix();

//USER 2

glPushMatrix();

glTranslatef(-2.7,-3,5);

glScaled(0.4,0.4,0.4);

user();

glPopMatrix();

//USER 3

glPushMatrix();

glTranslatef(-1.3,-3,5);

glScaled(0.4,0.4,0.4);

user();

glPopMatrix();

glPopMatrix();

glFlush();

glutSwapBuffers();

}

void doInit()

{

/\* Background and foreground color \*/

glClearColor(0.6,0.7,0.8,0.0);

glColor3f(.0,1.0,1.0);

glViewport(0,0,640,480);

/\* Select the projection matrix and reset it then

setup our view perspective \*/

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(30.0f,(GLfloat)640/(GLfloat)480,0.1f,200.0f);

/\* Select the modelview matrix, which we alter with rotatef() \*/

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glClearDepth(2.0f);

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LEQUAL);

}

void display()

{

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

glLoadIdentity();

glTranslatef(0.0f,0.0f,-13.0f);

stroke\_output(-2.0, 1.7, "s -> Start");

stroke\_output(-2.0, 1, "q--> quit");

GLfloat mat\_ambient[]={0.0f,1.0f,2.0f,1.0f};

GLfloat mat\_diffuse[]={0.0f,1.5f,.5f,1.0f};

GLfloat mat\_specular[]={5.0f,1.0f,1.0f,1.0f};

GLfloat mat\_shininess[]={50.0f};

glMaterialfv(GL\_FRONT,GL\_AMBIENT,mat\_ambient);

glMaterialfv(GL\_FRONT,GL\_DIFFUSE,mat\_diffuse);

glMaterialfv(GL\_FRONT,GL\_SPECULAR,mat\_specular);

glMaterialfv(GL\_FRONT,GL\_SHININESS,mat\_shininess);

GLfloat lightIntensity[]={3.7f,0.7f,0.7f,1.0f};

GLfloat light\_position[]={0.0f,5.5f,4.0f,0.0f};

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

glLightfv(GL\_LIGHT0,GL\_DIFFUSE,lightIntensity);

GLfloat lightIntensity1[]={3.7f,0.7f,0.7f,1.0f};

GLfloat light\_position1[]={0.0f,-5.5f,4.0f,0.0f};

glLightfv(GL\_LIGHT1,GL\_POSITION,light\_position1);

glLightfv(GL\_LIGHT1,GL\_DIFFUSE,lightIntensity1);

glEnable(GL\_COLOR\_MATERIAL);

glFlush();

glutSwapBuffers();

}

void select\_menu(int id)

{

switch(id)

{

case 1:

u[0]=true;

glutIdleFunc(firewall);

break;

case 2:

u[1]=true;

glutIdleFunc(firewall);

break;

case 3:

u[2]=true;glutIdleFunc(firewall);

break;

case 4:

u[3]=true;

glutIdleFunc(firewall);

break;

case 5:

u[0]=false;glutIdleFunc(firewall);

break;

case 6:

u[1]=false;

glutIdleFunc(firewall);

break;

case 7:

u[2]=false;

glutIdleFunc(firewall);

break;

case 8:

u[3]=false;glutIdleFunc(firewall);

break;

case 9:

u[4]=true;glutIdleFunc(firewall);

break;

case 10:

u[5]=true;

glutIdleFunc(firewall);

break;

case 11:

u[6]=true;glutIdleFunc(firewall);

break;

case 12:

u[7]=true;glutIdleFunc(firewall);

break;

case 13:

u[4]=false;glutIdleFunc(firewall);

break;

case 14:

u[5]=false;glutIdleFunc(firewall);

break;

case 15:

u[6]=false;glutIdleFunc(firewall);

break;

case 16:

u[7]=false;

break;

case 18: p[0]=true;

p[1]=p[2]=p[3]=p[4]=p[5]=p[6]=p[7]=false;

x=y=z=i1=0;

break;

case 19:

p[1]=true;

p[0]=p[2]=p[3]=p[4]=p[5]=p[6]=p[7]=false;

x=y=z=i1=0;

break;

case 20:

p[2]=true;

p[1]=p[0]=p[3]=p[4]=p[5]=p[6]=p[7]=false;

x=y=z=i1=0;

break;

case 21:

p[3]=true;

p[1]=p[2]=p[0]=p[4]=p[5]=p[6]=p[7]=false;

x=y=z=i1=0;

break;

case 22: p[4]=true;

p[1]=p[2]=p[3]=p[0]=p[5]=p[6]=p[7]=false;

x=y=z=i1=0;

break;

case 23:

p[5]=true;

p[1]=p[2]=p[3]=p[4]=p[0]=p[6]=p[7]=false;

x=y=z=i1=0;

break;

case 24:

p[6]=true;

p[1]=p[2]=p[3]=p[4]=p[5]=p[0]=p[7]=false;

x=y=z=i1=0;

break;

case 25:

p[7]=true;

p[1]=p[2]=p[3]=p[4]=p[5]=p[6]=p[0]=false;

x=y=z=i1=0;

break;

case 26:

u[0]=u[1]=u[2]=u[3]=true;

break;

case 27:

u[4]=u[5]=u[6]=u[7]=true;

break;

case 17:exit(0);

break;

}

glFlush();

glutSwapBuffers();

glutPostRedisplay();

}

void myMenu(){

int menu[10];

// Allow access to user 1

menu[0] = glutCreateMenu(select\_menu);

glutAddMenuEntry("Allow All ",26);

glutAddMenuEntry("HTTP ",1);

glutAddMenuEntry("Telnet ",2);

glutAddMenuEntry("SSH ",3);

glutAddMenuEntry("SMTP ",4);

// Block user 1

menu[1] = glutCreateMenu(select\_menu);

glutAddMenuEntry("HTTP ",5);

glutAddMenuEntry("Telnet ",6);

glutAddMenuEntry("SSH ",7);

glutAddMenuEntry("SMTP ",8);

//Allow user 2

menu[2] = glutCreateMenu(select\_menu);

glutAddMenuEntry("Allow All ",27);

glutAddMenuEntry("HTTP ",9);

glutAddMenuEntry("Telnet ",10);

glutAddMenuEntry("SSH ",11);

glutAddMenuEntry("SMTP ",12);

//Block user 2

menu[3]=glutCreateMenu(select\_menu);

glutAddMenuEntry("HTTP ",13);

glutAddMenuEntry("Telnet ",14);

glutAddMenuEntry("SSH ",15);

glutAddMenuEntry("SMTP ",16);

// Send User 1

menu[4] = glutCreateMenu(select\_menu);

glutAddMenuEntry("HTTP ",18);

glutAddMenuEntry("Telnet ",19);

glutAddMenuEntry("SSH ",20);

glutAddMenuEntry("SMTP ",21);

// Send User 2

menu[5] = glutCreateMenu(select\_menu);

glutAddMenuEntry("HTTP ",22);

glutAddMenuEntry("Telnet ",23);

glutAddMenuEntry("SSH ",24);

glutAddMenuEntry("SMTP ",25);

menu[6]=glutCreateMenu(NULL);

glutAddSubMenu("Allow",menu[0]);

glutAddSubMenu("Block",menu[1]);

glutAddSubMenu("Send",menu[4]);

menu[7]=glutCreateMenu(NULL);

glutAddSubMenu("Allow",menu[2]);

glutAddSubMenu("Block",menu[3]);

glutAddSubMenu("Send",menu[5]);

glutCreateMenu(select\_menu);

glutAddSubMenu("User 1",menu[6]);

glutAddSubMenu("User 2",menu[7]);

glutAddMenuEntry("Quit 'q'",17);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

void mykey(unsigned char key,int x1,int y1)

{

if(key=='s')

{ glutIdleFunc(firewall);

}

if(key=='1')

{

p[0]=true;

p[1]=p[2]=p[3]=p[4]=p[5]=p[6]=p[7]=false;

x=y=z=i1=0;

}

if(key=='2')

{

p[1]=true;

p[0]=p[2]=p[3]=p[4]=p[5]=p[6]=p[7]=false;

x=y=z=i1=0;

}

if(key=='3')

{

p[2]=true;

p[1]=p[0]=p[3]=p[4]=p[5]=p[6]=p[7]=false;

x=y=z=i1=0;

}

if(key=='4')

{

p[3]=true;

p[1]=p[2]=p[0]=p[4]=p[5]=p[6]=p[7]=false;

x=y=z=i1=0;

}

if(key=='5')

{

p[4]=true;

p[1]=p[2]=p[3]=p[0]=p[5]=p[6]=p[7]=false;

x=y=z=i1=0;

}

if(key=='6')

{

p[5]=true;

p[1]=p[2]=p[3]=p[4]=p[0]=p[6]=p[7]=false;

x=y=z=i1=0;

}

if(key=='7')

{

p[6]=true;

p[1]=p[2]=p[3]=p[4]=p[5]=p[0]=p[7]=false;

x=y=z=i1=0;

}

if(key=='8')

{

p[7]=true;

p[1]=p[2]=p[3]=p[4]=p[5]=p[6]=p[0]=false;

x=y=z=i1=0;

}

if(key=='S')

{

glutIdleFunc(firewall);

}

if(key=='q'||key=='Q')

{

exit(0);

}

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGB);

glutInitWindowSize(1000,480);

glutInitWindowPosition(0,0);

glutCreateWindow("Firewall");

glutDisplayFunc(display);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHT1);

glShadeModel(GL\_SMOOTH);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_NORMALIZE);

glutKeyboardFunc(mykey);

myMenu();

doInit();

glutMainLoop();

return 0;

}