NETWORKING

#include "stdafx.h"

#include <windows.h>

#include<string.h>

#include<stdarg.h>

#include<stdio.h>

#include <GL/glut.h>

#include <math.h>

static double x=0.0;

static double move=-60;

static float rx[100]={0}, ry[100]={0};

//control waves

static double w1=0,w2=0,w3=0;

static bool transmit=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);

}

}

void stroke\_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 satellite()

{

glRotatef(60,1,0,0);

//body

glPushMatrix();

glColor3f(0.2,0.2,0.2);

glScaled(1,0.6,1);

glTranslatef(3.0,0,0.0);

glutSolidCube(0.4);

glPopMatrix();

//Solar Panels

glPushMatrix();

glColor3f(0.3,0.3,0.3);

glTranslatef(3,0,0.0);

//glRotatef(45,1,0,0);

glScaled(3.7,0.0,1);

glutSolidCube(0.4);

glPopMatrix();

glPushMatrix();

glColor3f(0.2,0.1,0.1);

glTranslatef(3.0,0,-0.4);

glScaled(0.5,0.5,0.5);

glutSolidSphere(0.3,50,50);

glPopMatrix();

glPushMatrix();

glColor3f(0.2,0.2,0.1);

glTranslatef(3.0,0,0.4);

glScaled(0.4,0.4,0.3);

glutSolidTorus(0.3,0.2,20,20);

glPopMatrix();

}

// Second Screen

void sat2(double ang)

{

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

glLoadIdentity();

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

glRotatef(ang,0.0f,1.0f,0.0f);

//earth

glPushMatrix();

glColor3f(0.0,1.0,0.0);

//glScaled(0.8,0.04,0.8);

//glTranslatef(0.0,0,0.0);

glutSolidSphere(2.0,50,50);

glPopMatrix();

satellite();

glFlush();

glutSwapBuffers();

}

void building(float x1,float y1,float z1)

{

//Main Structure

glPushMatrix();

glColor3f(0.5,0.5,0.5);

glTranslatef(x1,y1,z1);

glScaled(0.5,1.5,0.5);

glutSolidCube(2);

glPopMatrix();

//Dish on top

glPushMatrix();

glColor3f(1,1,0);

glTranslatef(x1,y1+1.8,z1);

glRotatef(60,1,0,0);

glScaled(0.5,1.5,0.5);

glutSolidCone(0.5,1,20,20);

glPopMatrix();

//windows

glPushMatrix();

glColor3f(0.1,0,0);

glTranslatef(x1-0.2,y1+0.7,z1);

glScaled(0.5,0.5,0.5);

//glutSolidCube(.3);

for(float j=-3;j<1.5;j+=.8)

{

for(float i=0;i<1;i+=0.8)

{

glPushMatrix();

glTranslatef(i,j,1);

glutSolidCube(0.4);

glPopMatrix();

}

}

glPopMatrix();

}

void waves()

{

glPushMatrix();

glTranslatef(0,1,0);

glScaled(0.05,0.5,0.1);

glutSolidCube(0.5);

glPopMatrix();

glPushMatrix();

glRotatef(-8,0,0,1);

glTranslatef(0.01,1,0);

glScaled(0.05,0.5,0.1);

glutSolidCube(0.5);

glPopMatrix();

glPushMatrix();

glRotatef(8,0,0,1);

glTranslatef(-0.01,1,0);

glScaled(0.05,0.6,0.1);

glutSolidCube(0.5);

glPopMatrix();

}

void sat1()

{

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

glLoadIdentity();

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

//glRotatef(x,0.0f,1.0f,0.0f);

//Moon

glPushMatrix();

glColor3f(1,1,1);

glTranslatef(-3.8,2.8,0);

glScaled(0.5,0.5,0.1);

glutSolidSphere(0.6,50,50);

glPopMatrix();

//Earth

glPushMatrix();

glColor3f(0.0,1.0,0.0);

glTranslatef(0,-12,0);

//glScaled(0.8,0.04,0.8);

glutSolidSphere(10.0,50,50);

glPopMatrix();

//Building Center

glPushMatrix();

glColor3f(0,1,1);

glRotatef(10,1,0,0);

building(1.2,-1.2,3.2);

glPopMatrix();

//Building left

glPushMatrix();

glColor3f(0,1,1);

glRotatef(5,0,0,1);

building(-3.8,-1.2,0);

glPopMatrix();

//signal

glPushMatrix();

glColor3f(0,0,1);

if(transmit)

{

glRotatef(-25,0,0,1);

glTranslatef(-1.25,-1.6+w1,0);

}

else

glTranslatef(1,20,3.3);

waves();

glPopMatrix();

//Main Dish

//Tower

glPushMatrix();

glColor3f(1,1,1);

glTranslatef(-1,-2,4);

glRotatef(270,1,0,0);

glScaled(1.0,1,2.0);

glutWireCone(0.5,1,4,10);

glPopMatrix();

//Dish

glPushMatrix();

glColor3f(1,1,1);

glTranslatef(-1.08,0.2,3);

glRotatef(60,1,0,0);

glScaled(0.7,1.3,0.7);

glutSolidCone(0.4,0.5,20,20);

glPopMatrix();

//Building right

glPushMatrix();

glColor3f(0,1,1);

glRotatef(-5,0,0,1);

building(3.8,-1.2,0);

glPopMatrix();

//Saltellite

glPushMatrix();

glTranslatef(-3,3.0,0);

satellite();

glPopMatrix();

//Ack to right building

glPushMatrix();

if(transmit)

{

glRotatef(50,0,0,1);

glTranslatef(2.8,3.2-w2,0);

}

else

glTranslatef(1,20,3.3);

waves();

glPopMatrix();

//Ack to Left building

glPushMatrix();

if(transmit)

{

glRotatef(-50,0,0,1);

glTranslatef(-2.8,3.2-w2,0);

}

else

glTranslatef(1,20,3.3);

waves();

glPopMatrix();

//Ack to Center building

glPushMatrix();

if(transmit)

{

glRotatef(23,0,0,1);

glTranslatef(1,3.2-w3,3.3);

}

else

glTranslatef(1,20,3.3);

waves();

glPopMatrix();

//stars

glPointSize(5);

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

{

for(int i=0;i<100;i++)

{

rx[j]=rand()/500;

ry[i]=rand()/500;

glBegin(GL\_POINTS);

glColor3f(0,2,2);

glVertex3f(-6+rx[j],ry[i],-5);

glEnd();

}

}

glPushMatrix();

//glScaled(1.1,2.0,0.1);

glTranslatef(0.0,0.0,-2.0);

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

glColor3f(1,1,1);

drawstring(1,3.7,-1.0,"Satellite");

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

glColor3f(1,1,1);

drawstring(-4.4,.5,-1.0,"Reciever");

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

glColor3f(1,1,0);

drawstring(0,-2,7,"Reciever");

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

glColor3f(1,1,1);

drawstring(-1.5,-1,-1.0,"Transmitter");

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

glColor3f(1,1,1);

drawstring(3.2,1,3,"Reciever");

glPopMatrix();

glFlush();

glutSwapBuffers();

}

// Third Screen

void sat3(double ang)

{

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

glLoadIdentity();

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

glRotatef(ang,0.0f,1.0f,0.0f);

//earth

glPushMatrix();

glColor3f(0.3,0.6,1);

//glScaled(0.8,0.04,0.8);

//glTranslatef(0.0,0,0.0);

glutSolidSphere(2.0,50,50);

glPopMatrix();

satellite();

glFlush();

glutSwapBuffers();

}

void e()

{

x-=0.07;

sat2(x);

}

void s()

{

x-=0.07;

sat2(x);

}

void S()

{

x += .07;

if(transmit)

{

if(w1<=4.2)

w1+=0.01;

if(w1>=2.5 && w2<=6.9)

w2+=0.01;

if(w1>=2.5 && w3<=5)

w3+=0.01;

}

sat1();

}

void doInit()

{

/\* Background and foreground color \*/

glClearColor(0.0,0.0,0.0,0);

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);

glEnable( GL\_COLOR\_MATERIAL );

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/S--> Start");

stroke\_output(-2.0, 0.9, "t--> Transmit");

stroke\_output(-2.0, 0.0, "q/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[]={1.7f,1.7f,1.7f,1.0f};

GLfloat light\_position3[]={0.0f,8.0f,10.0f,0.0f};

glLightfv(GL\_LIGHT0,GL\_POSITION,light\_position3);

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

GLfloat lightIntensity1[]={1.7f,1.7f,1.7f,1.0f};

GLfloat light\_position31[]={-2.0f,8.0f,10.0f,0.0f};

glLightfv(GL\_LIGHT1,GL\_POSITION,light\_position31);

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

glEnable(GL\_COLOR\_MATERIAL);

glFlush();

glutSwapBuffers();

}

void menu(int id)

{

switch(id)

{

case 1:glutIdleFunc(S);

break;

case 2:glutIdleFunc(s);

break;

case 5:exit(0);

break;

}

glFlush();

glutSwapBuffers();

glutPostRedisplay();

}

void mykey(unsigned char key,int x,int y)

{

if(key=='s')

{

glutIdleFunc(s);

}

if(key=='S')

{

glutIdleFunc(S);

}

if(key=='e')

{

glutIdleFunc(e);

}

if(key=='t')

{

transmit=!transmit;

if(!transmit)

{

w1=0;

w2=0;

w3=0;

}

glutIdleFunc(S);

}

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("Networking");

glutDisplayFunc(display);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glEnable(GL\_LIGHT1);

glShadeModel(GL\_SMOOTH);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_NORMALIZE);

glutKeyboardFunc(mykey);

glutCreateMenu(menu);

glutAddMenuEntry("Pyramid 's'",1);

glutAddMenuEntry("Reverse Pyramid 'S'",2);

glutAddMenuEntry("Quit 'q'",5);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

doInit();

glutMainLoop();

return 0;

}