#include <GL/glut.h>

#include<math.h>

#include<stdio.h>

#include<iostream>

#include<stdlib.h>

using namespace std;

#define SunSize 0.2

#define EarthSize 0.1

#define MoonSize 0.016

#define Sun 0.5

static GLint axis=2;

GLfloat SpeedMultiplicator = 0.5;

GLfloat DaysPerYear = 50.0; //OK, ok... but it is soo slow with 360!

GLfloat year = 0.0; //degrees

GLfloat day = 0.0;

GLfloat moonAroundEarth = 0.0;

GLfloat moonItsSelf = 0.0;

GLfloat EarthOrbitRadius = 1.0;

GLfloat MoonOrbitRadius = 0.1;

GLfloat daySpeed = 5.0 \* SpeedMultiplicator;

GLfloat yearSpeed = DaysPerYear / 360.0 \* daySpeed \* SpeedMultiplicator;

GLfloat moonItsSelfSpeed = 5 \* SpeedMultiplicator;

GLsizei wh = 700, ww = 700; // initial window size in pixels

GLfloat size = 10.0; // used to scale size of drawn figures

int global\_x\_position, global\_y\_position; //stores where the center of the fi$

float r=1.0,g=1.0,b=1.0; // to store the global color variabe

GLfloat planes[]={-1.0,0.0,1.0,0.0};

GLfloat planet[]={0.0,-1.0,0.0,1.0};

void\* currentfont;

int sel;

int abc,efg;

GLfloat vertices[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},

{1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},

{1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};

GLfloat colors[][3] = {{0.0,0.0,0.0},{1.0,0.0,0.0},

{1.0,1.0,0.0}, {0.0,1.0,0.0}, {0.0,0.0,1.0},

{1.0,0.0,1.0}, {1.0,1.0,1.0}, {0.0,1.0,1.0}};

void setfont(void \*font)

{

currentfont=font;

}

void drawtext(float x,float y,char\* string)

{

char\* c;

glRasterPos2f(x,y);

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

{

glutBitmapCharacter(currentfont,\*c);

}

}

void title()

{

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

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

glColor3f(1.0,0.0,0.0);

drawtext(-0.9,0.9,"COMPUTER GRAPHICS AND VISUALIZATION");

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

glColor3f(1.0,1.0,1.0);

drawtext(-0.25,0.7,"PROJECT ON");

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

glColor3f(0.0,1.0,0.0);

drawtext(-0.5,0.45,"LUNAR PHASES OF MOON");

glColor3f(1.0,1.0,1.0);

drawtext(-0.3,0.25,"SUBMITTED BY");

glColor3f(1.0,1.0,0.0);

drawtext(-0.90,0.10,"NITHESH S PAVAN KUMAR S\n");

drawtext(-0.90,0.02,"(1CE17CS075) (1CE17CS080)");

glColor3f(1.0,1.0,1.0);

drawtext(-0.5,-0.1,"UNDER THE GUIDANCE OF");

glColor3f(1.0,0.0,1.0);

drawtext(-0.90,-0.25,"Mr.RAMESH B");

glColor3f(1.0,1.0,1.0);

drawtext(-0.90,-0.35,"ASST PROFESSOR ");

drawtext(-0.90,-0.45,"DEPT OF CSE ");

drawtext(-.90,-0.55,"CEC ");

glFlush();

}

void d3()

{

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

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

glColor3f(1.0,0.0,0.0);

drawtext(-0.2,0.2,"LUNAR PHASES");

glLoadIdentity();

glutSwapBuffers();

}

void d1()

{

title();

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

glLoadIdentity();

}

void d2()

{

glClear (GL\_COLOR\_BUFFER\_BIT);

title();

glutSwapBuffers();

}

void myAnyKey(int k, int x, int y)

{

glutHideWindow();

glutPostRedisplay();

}

void display();

void RenderScene(void)

{

glPushMatrix();

gluLookAt(0.0,0.0,-4.0,0.0,00.0,1.0,50.0,-30.0,40.0);

glColor3f(1.0,0.0,0.6);

glVertex2f(100.0,200.0);

glTranslatef(EarthOrbitRadius,0.0,0.0);

glutSolidSphere(SunSize,50,50);

glPushMatrix();

glRotatef(year,0.0,1.0,0.0);

glTranslatef(2,0.0,0.0);

glRotatef(-year,0.0,1.0,0.0);

glPushMatrix();

glRotatef(day,0.25,1.0,0.0);

glColor3f(1.0,1.0,0.8);

glutWireSphere(EarthSize,25,30);

glBegin(GL\_LINES);

glVertex3f(-0.0625,-0.25,0.0);

glVertex3f(0.0625,0.25,0.0);

glEnd();

glPopMatrix();

glRotatef(moonAroundEarth,0.0,1.0,0.0);

glTranslatef(MoonOrbitRadius,0.0,0.0);

glPopMatrix();

glPopMatrix();

}

void createplanet()

{

glPushMatrix();

glColor3f(1.0,0.0,0.0);

glTranslatef(0.8,-0.1,00);

glRotatef(0.0,1.0,0.0,0.0);

glRotatef(16.0,-1.0,7.0,0.0);

glutWireSphere(6,50,50);

glPopMatrix();

glFlush();

}

void Init(void)

{

glClearColor(0.0,0.0,0.0,0.0);

glClearDepth(10.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void Reshape(int x, int y)

{

if (y == 0) return;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(40.0,(GLdouble)x/(GLdouble)y,0.5,20.0);

glMatrixMode(GL\_MODELVIEW);

glViewport(0,0,x,y);

display();

}

void Idle(void)

{

day += daySpeed;

year += yearSpeed;

display();

}

class Moon {

int displayListId;

public:

void create() {

displayListId = glGenLists(1);

glNewList(displayListId, GL\_COMPILE);

GLfloat direction[] = {-1.0, -1.0, -1.0, 0.0};

glLightfv(GL\_LIGHT0, GL\_POSITION, direction);

glutSolidSphere(0.5, 75, 85);

glEndList();

}

void draw() {

glCallList(displayListId);

}

};

// The one and only moon.

static Moon moon;

// An orbiter is an object that flies on a circle of a certain radius on the

// xz plane. You supply the radius at construction time.

class Orbiter {

double radius;

double u;

public:

Orbiter(double radius): radius(radius), u(0.0) {}

void advance(double delta) {u += delta;}

void getPosition(double& x, double& y, double& z) {

x = radius \* cos(u);

y = 0;

z = radius \* sin(u);

}

};

// The one and only orbiter.

static Orbiter orbiter(5.0);

// Clears the window (and the depth buffer) and draws the moon as viewed from

// the current position of the orbiter.

void display()

{

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

glMatrixMode(GL\_MODELVIEW);

glPushMatrix();

glLoadIdentity();

double x, y, z;

orbiter.getPosition(x, y, z);

gluLookAt(x, y, z, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

moon.draw();

GLfloat specular0[] = {1.0, 0.0, 0.5, 1.0};

glLightfv(GL\_LIGHT0, GL\_SPECULAR, specular0);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glPopMatrix();

createplanet();

RenderScene();

glutSwapBuffers();

}

// Advances the orbiter and requests to draw the next frame.

void timer(int v)

{

orbiter.advance(0.01);

glutPostRedisplay();

glutTimerFunc(1000/60, timer, v);

}

// reshape() fixes up the projection matrix so that we always see a sphere

// instead of an ellipsoid.

void reshape(GLint w, GLint h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(40.0, GLfloat(w) / GLfloat(h), 1.0, 10.0);

}

// Enables depth testing, enables lighting for a bright yellowish diffuse

// light, and creates a moon.

void init() {

glEnable(GL\_DEPTH\_TEST);

moon.create();

}

void myReshape(GLsizei w, GLsizei h)

{

cout << "in reshape" << endl; // so we will know we are in the reshape $

glClear(GL\_COLOR\_BUFFER\_BIT); //clears the screen to black

//it is customary OpenGL style to set the viewing projection in the Reshape$

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//set up an orthographic view with the same coordinate size as the curr$

glOrtho(0.0, (GLdouble)w, 0.0, (GLdouble)h, -1.0, 1.0);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glViewport(0,0,w,h); // set viewport to use the entire window

ww = w; //reset the global width and height variables to current windo$

wh = h;

}

//callback function for left mouse button click

void accept\_left\_button\_command(int btn, int state, int x, int y)

{

cout << "in mouse click callback function"<<endl;

if(btn==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN) //this is only for left b$

{

global\_x\_position = x; //capture x mouse position in global

global\_y\_position = y; //capture y mouse position in global

glutPostRedisplay(); // post a display event

}

}

void disp(void)

{

cout << "in display" << endl; // just so you will know when display is call$

glColor3f(r,g,b); // set the color to the current color

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glFlush(); //insure that commands are executed

}

void color\_select (int color)

{

cout<<"in color select menu callback"<<endl;

}

void main\_menu\_callback(int selection)

{

cout<<"in main menu callback function"<<endl;

if (selection == 0) exit(0); // exit the program if exit is selected

else if (selection == 1) //clear screen

{

GLfloat diffuse0[] = {1.0, 0.0, 0.5, 1.0};

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, diffuse0);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

}

else if (selection == 2)

{

GLfloat ambient0[] = {0.0, 1.0, 0.0, 1.0};

glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambient0);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

}

else if (selection == 3)

{

GLfloat diffuse0[] = {0.0, 0.0, 1.0, 1.0};

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, diffuse0);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

}

}

void set\_up\_menu(void)

{

int c\_menu\_id, shape\_menu\_id,c\_menus\_id,facts; // menu identifiers for submenus

c\_menu\_id = glutCreateMenu(color\_select);

glutAddMenuEntry("New moon",1);

glutAddMenuEntry("Waxing Crescent",2);

glutAddMenuEntry("First Quarter",3);

glutAddMenuEntry("Waxing Gibbous",4);

glutAddMenuEntry("Full Moon",5);

glutAddMenuEntry("Waning Gibbous",6);

glutAddMenuEntry("Last Quarter",7);

glutAddMenuEntry("waning Crescent",8);

c\_menus\_id = glutCreateMenu(color\_select);

glutAddMenuEntry("\*The rise and fall of the tides on earth is caused by the moons.",1);

glutAddMenuEntry("\*The is moon is drifting away 3.8cm away from the earth every year.",2);

glutAddMenuEntry("\*A person would weigh much less on the moon. ",3);

glutAddMenuEntry("\*The moon has no atmosphere.",4);

glutAddMenuEntry("\*The moon has quakes. ",5);

glutAddMenuEntry("\*The moon is the 5th largest naturnal satellite \n in the solar system.",6);

// create the main menu

//we don't need the main menu id so don't save it

glutCreateMenu(main\_menu\_callback);

glutAddSubMenu("PHASES OF MOON",c\_menu\_id );

glutAddSubMenu("Facts",c\_menus\_id);

glutAddMenuEntry("Pink Color",1);

glutAddMenuEntry("Blue Color", 3);

glutAddMenuEntry("Green Color",2);

glutAddMenuEntry("Exit",0);

glutAttachMenu(GLUT\_RIGHT\_BUTTON); //attach menu to right button

}

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

{

switch(key)

{

case '1':

sel=1;

break;

case '2':

sel=2;

break;

case '3':

sel=3;

break;

case '4':

sel=4;

break;

case 'q':

exit(0);

break;

}

}

void mySpecialKey(int k, int x, int y)

{

switch(k)

{

case GLUT\_KEY\_LEFT:

axis=0;

break;

case GLUT\_KEY\_RIGHT:

axis=1;

break;

case GLUT\_KEY\_UP:

axis=2;

break;

case GLUT\_KEY\_DOWN:

exit(0);

}

}

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

{

GLubyte image[64][64][3];

int i,j,c;

for(i=0;i<64;i++)

{

for(j=0;j<64;j++)

{

c=(((i&0x8)==0)^((j&0x8))==0)\*255;

image[i][j][0]=(GLubyte) c;

image[i][j][1]=(GLubyte) c;

image[i][j][2]=(GLubyte) c;

}

}

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_DOUBLE | GLUT\_RGB|GLUT\_DEPTH);

glutInitWindowPosition(0,0);

glutInitWindowSize(600,600);

abc=glutCreateWindow("FIRST WINDOW");

glutDisplayFunc(d1);

glutDisplayFunc(d2);

glClearColor(1.0,0.0,1.0,0.0);

glutKeyboardFunc(keys);

glutSpecialFunc(myAnyKey);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB|GLUT\_DEPTH);

glutInitWindowPosition(670,0);

glutInitWindowSize(600,600);

glutCreateWindow(argv[0]);

glutReshapeFunc (myReshape);

glutMouseFunc (accept\_left\_button\_command);

glutDisplayFunc(disp);

set\_up\_menu(); // build pop up menu

glutReshapeFunc(Reshape);

glutIdleFunc(Idle);

glutDisplayFunc(display);

glutTimerFunc(100,timer,0);

glutReshapeFunc(reshape);

init();

glutKeyboardFunc(keys);

glutSpecialFunc(myAnyKey);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

glutInitWindowPosition(0,0);

glutInitWindowSize(1200,680);

glutCreateWindow("WINDOW");

glutDisplayFunc(d3);

glClearColor(1.0,0.0,0.5,0.0);

glutKeyboardFunc(keys);

glutSpecialFunc(myAnyKey);

glutMainLoop();

return 0;

}