Nick Wright Test 2 CSC328

Code

#include<windows.h>

#include<GL/glut.h>

#include<stdlib.h>

#include<math.h>

#include<conio.h>

#include<stdio.h>

#include <iostream>

#include <iomanip>

using namespace std;

/\* Test 2

The purpose of this test is to display a solid pyramid with lighting effects \*/

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Global Values \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

float static theta = 0.0, theta2 = 0.0;

//float theta = 0.0, theta2 = 0.0;

float scale1 = 1.0;

float dx = 0.0, dy = 0.0, dz = 0.0;

int accumulator = 0;

void init(void);//this is a function to initialize the window clear color

void RenderScene(void);//this is a function to draw a square in an opened window

void loadicon(float[][3], float[][3], float[][3], float[][3], float[], float[], float[], float[][3]);

void drawicon(float[][3], float[][3], float[][3], float[][3], float[], float[], float[], float[][3]);/\*draws the icon\*/

void settrans2(void);

void SetupRC(void);//sets up the clear color

void TimerFunction(int); //this call back function is call each 30 ms and changes the location,scale and rotation of the pyramid.

//Main Program

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

{//set up window title

char header[] = "Test 2 By Nick Wright";

glutInit(&argc, argv);

// Set up the display mode with a double buffer and a depth buffer and RGB colors

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

SetupRC();

//Initialize window size and position

glutInitWindowSize(560, 440);

glutInitWindowPosition(140, 20);

// Open and Label Window

glutCreateWindow(header);

glutDisplayFunc(RenderScene);

glutTimerFunc(500, TimerFunction, 1);

glutMainLoop();

return 0;

} //end of main

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*RenderScene Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void RenderScene(void)

{

float xdel = 0.25;

float x[4][3], y[4][3], z[4][3], fcolor[5][3], bottomX[4], bottomY[4], bottomZ[4], nvector[5][3];

/\* These variables hold the pattern for the pyramid icon. x,y,z hold the four pyramid faces and bottomX, bottomY, bottomZ hold the bottom face \*/

//clear the window with the current background color

cout << "in renderscene" << endl;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//set the viewport to the window dimensions

glViewport(0, 0, 540, 440);

//Establish the clipping volume in user coordinates

glOrtho(-7.0, 7.0, -7.0, 7.0, -7.0, 7.0);

loadicon(x, y, z, fcolor, bottomX, bottomY, bottomZ, nvector);

/\* draw the cube and line \*/

glEnable(GL\_DEPTH\_TEST);

//enable lighting

glEnable(GL\_LIGHTING);

glEnable(GL\_CULL\_FACE);

glFrontFace(GL\_CCW);//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*was GL\_CCW, I think that mine is CW

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CAUTION DANGER WILL SMITH!!!! DANGER!!!\*\*\*\*\*\*\*\*\*\*\*\*

YOU MUST SWITCH TO MODELVIEW MATRIX MODE BEFORE YOU ENABLE THE LIGHT AND YOU MUST

THE REAL PROBLEM HERE SEEMS TO BE THE Angle of width of the spotlight beam described in

glLightf(GL\_LIGHT0,GL\_SPOT\_CUTOFF,20.0); values of 20 to 30 work best here. Values less than

seem to make the light too focused. Remember you must make the light wide enough to cover your object

else the polygon will not light. Finally pure colors of red, green or blue do not seem to reflect and

have a sepctular effect. Some mixture of these colors i.e.(0.5,0.4,0.3) will produce that spectular shine or flash

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*IGNORE THESE AT YOUR OWN RISK\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//light 1

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// set up light parameters

float ambientlight[] = { 0.7,0.1,0.1,1.0 };//strong red ambient light

float diffuselight[] = { 0.7,0.1,0.1,1.0 };//diffuse lighting

float specular[] = { 0.7,0.1,0.1,1.0 };//specular lighting

float lightpos[] = { -7.0,8.0,4.0,1.0 };//SEE CAUTIONARY NOTE BELOW FOR COORDINATE SYSTEM

float specref[] = { 1.0,1.0,1.0,1.0 };//set the reflectance of the material all is plastic

float spotdir[] = { 2.0,-4.0,-4.0 };//shine spot down on cube the light must shine toward the origin

// set light position, ambient, diffuse and specular strength

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

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

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

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

//focused spotlight with only 10 degrees one way

glLightf(GL\_LIGHT0, GL\_SPOT\_CUTOFF, 20.0);

glLightf(GL\_LIGHT0, GL\_SPOT\_EXPONENT, 15.0);

// point the light back to the origin

glLightfv(GL\_LIGHT0, GL\_SPOT\_DIRECTION, spotdir);

if (accumulator % 2 != 0) {

glEnable(GL\_LIGHT0);//seems to work, initially there is no light

glDisable(GL\_LIGHT1);

}

//light 2

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// set up light parameters

float ambientlight1[] = { 0.1,0.7,0.1,1.0 };//green ambient light

float diffuselight1[] = { 0.1,0.7,0.1,1.0 };//diffuse lighting

float specular1[] = { 1.0,1.0,1.0,1.0 };//specular lighting

float lightpos1[] = { 7.0,8.0,4.0,1.0 };//SEE CAUTIONARY NOTE BELOW FOR COORDINATE SYSTEM

float specref1[] = { 1.0,1.0,1.0,1.0 };//set the reflectance of the material all is plastic

float spotdir1[] = { -2.0,-4.0,-4.0 };//shine spot down on cube the light must shine toward the origin

// set light position, ambient, diffuse and specular strength

glLightfv(GL\_LIGHT1, GL\_POSITION, lightpos1);

glLightfv(GL\_LIGHT1, GL\_AMBIENT, ambientlight1);

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, diffuselight1);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, specular1);

//focused spotlight with only 10 degrees one way

glLightf(GL\_LIGHT1, GL\_SPOT\_CUTOFF, 20.0);

glLightf(GL\_LIGHT1, GL\_SPOT\_EXPONENT, 15.0);

// point the light back to the origin

glLightfv(GL\_LIGHT1, GL\_SPOT\_DIRECTION, spotdir1);

//enable the lights

//if statement to enable and disable lighting

if(accumulator % 2 == 0) {

glEnable(GL\_LIGHT1);

glDisable(GL\_LIGHT0);

}

//now define the material properties

glEnable(GL\_COLOR\_MATERIAL);

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

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

glMateriali(GL\_FRONT, GL\_SHININESS, 128);

glClearColor(0.5, 0.5, 0.5, 1.0);

// Clear the window and the z buffer with the background color

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

settrans2();

//now draw the square

drawicon(x, y, z, fcolor, bottomX, bottomY, bottomZ, nvector);

glFlush();

glEnd();

glutSwapBuffers();

return;

};//end of render scene

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Load Icon Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void loadicon(float x[][3], float y[][3], float z[][3], float fcolor[][3], float bottomX[], float bottomY[], float bottomZ[], float nvector[][3])

/\* This procedure loads the pyramid icon \*/

//p1 -> (0, 5, 0)

//p2 -> (2, 0, -2)

//p3 -> (2, 0, 2)

//p4 -> (-2, 0, 2)

//p5 -> (-2, 0, -2)

//p6 -> (0, 0, 0)

{/\* load front face\*/

x[0][0] = 0.0; y[0][0] = 5.0; z[0][0] = 0.0; //p1 -> (0, 5, 0)

x[0][1] = 2.0; y[0][1] = 0.0; z[0][1] = 2.0; //p3 -> (2, 0, 2)

x[0][2] = -2.0; y[0][2] = 0.0; z[0][2] = 2.0; //p4 -> (-2, 0, 2)

/\* load the color on the front face red\*/

fcolor[0][0] = 1.0; fcolor[0][1] = 0.0; fcolor[0][2] = 0.0;

//fcolor[0][0] = 1.0; fcolor[0][1] = 1.0; fcolor[0][2] = 1.0;

/\* load the normal to this face \*/

// nvector[0][0] = 0.0; nvector[0][1] = 0.0; nvector[0][2] = 1.0;

//nvector[0][0] = 0.0; nvector[0][1] = (2.0 / sqrt(13)); nvector[0][2] = (3.0 / sqrt(13));

nvector[0][0] = 0.0; nvector[0][1] = -0.3714; nvector[0][2] = -0.9285;

/\* load the right side (x) face\*/

x[1][0] = 0.0; y[1][0] = 5.0; z[1][0] = 0.0; //p1 -> (0, 5, 0)

x[1][1] = 2.0; y[1][1] = 0.0; z[1][1] = -2.0; //p2 -> (2, 0, -2)

x[1][2] = 2.0; y[1][2] = 0.0; z[1][2] = 2.0; //p3 -> (2, 0, 2)

/\* load the color on the right side face blue \*/

fcolor[1][0] = 0.0; fcolor[1][1] = 0.0; fcolor[1][2] = 1.0;

//fcolor[1][0] = 1.0; fcolor[1][1] = 1.0; fcolor[1][2] = 1.0;

// load the normal to this face pos x axis

//nvector[1][0] = 1.0; nvector[1][1] = 0.0; nvector[1][2] = 0.0;

nvector[1][0] = (3.0 / sqrt(13)); nvector[1][1] = (2.0 / sqrt(13)); nvector[1][2] = 0.0;

/\* load the back side face \*/

x[2][0] = 0.0; y[2][0] = 5.0; z[2][0] = 0.0; //p1 -> (0, 5, 0)

x[2][1] = -2.0; y[2][1] = 0.0; z[2][1] = -2.0; //p5 -> (-2, 0, -2)

x[2][2] = 2.0; y[2][2] = 0.0; z[2][2] = -2.0; //p2 -> (2, 0, -2)

/\*load the color on the back side red \*/

fcolor[2][0] = 1.0; fcolor[2][1] = 0.0; fcolor[2][2] = 0.0;

//fcolor[2][0] = 1.0; fcolor[2][1] = 1.0; fcolor[2][2] = 1.0;

// load the normal to this face neg z axis

//nvector[2][0] = 0.0; nvector[2][1] = 0.0; nvector[2][2] = -1.0;

nvector[2][0] = 0.0; nvector[2][1] = 0.3714; nvector[2][2] = 0.9285;

/\* load the left side x face \*/

x[3][0] = 0.0; y[3][0] = 5.0; z[3][0] = 0.0; //p1 -> (0, 5, 0)

x[3][1] = -2.0; y[3][1] = 0.0; z[3][1] = 2.0; //p4 -> (-2, 0, 2)

x[3][2] = -2.0; y[3][2] = 0.0; z[3][2] = -2.0; //p5 -> (-2, 0, -2)

/\* load the color on the back side blue \*/

fcolor[3][0] = 0.0; fcolor[3][1] = 0.0; fcolor[3][2] = 1.0;

//fcolor[3][0] = 1.0; fcolor[3][1] = 1.0; fcolor[3][2] = 1.0;

// load the normal to this face neg x axis

//nvector[3][0] = -1.0; nvector[3][1] = 0.0; nvector[3][2] = 0.0;

nvector[3][0] = (-3.0 / sqrt(13)); nvector[3][1] = (2.0 / sqrt(13)); nvector[3][2] = 0.0;

/\* load the bottom side \*/

bottomX[0] = -2.0; bottomY[0] = 0.0; bottomZ[0] = 2.0; //p4 -> (-2, 0, 2)

bottomX[1] = 2.0; bottomY[1] = 0.0; bottomZ[1] = 2.0; //p3 -> (2, 0, 2)

bottomX[2] = 2.0; bottomY[2] = 0.0; bottomZ[2] = -2.0; //p2 -> (2, 0, -2)

bottomX[3] = -2.0; bottomY[3] = 0.0; bottomZ[3] = -2.0; //p5 -> (-2, 0, -2)

/\* load the color on the top green \*/

fcolor[4][0] = 0.0; fcolor[4][1] = 1.0; fcolor[4][2] = 0.0;

//fcolor[4][0] = 1.0; fcolor[4][1] = 1.0; fcolor[4][2] = 1.0;

// load the normal to this face pos y axis

nvector[4][0] = 0.0; nvector[4][1] = 1.0; nvector[4][2] = 0.0;

return;

}/\* end of load icon \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function drawicon \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void drawicon(float x[][3], float y[][3], float z[][3], float fcolor[][3], float bottomX[], float bottomY[], float bottomZ[], float nvector[][3])

{

/\* this function draws the cube at the transformed position \*/

int i, face;

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

{// render each face of the cube

// note we are doing color tracking on the material color

glColor3f(fcolor[face][0], fcolor[face][1], fcolor[face][2]);

glBegin(GL\_POLYGON);

glNormal3f(nvector[face][0], nvector[face][1], nvector[face][2]); //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* uncomment this line once you have the nvectors!

for (i = 0; i <= 2; i++)

glVertex3f(x[face][i], y[face][i], z[face][i]); //not sure that this loop is correct??

glEnd();

}

//color for bottom of pyramid

glColor3f(fcolor[4][0], fcolor[4][1], fcolor[4][2]);

//draw the bottom of the pyramid

glBegin(GL\_POLYGON);

glNormal3f(nvector[4][0], nvector[4][1], nvector[4][2]);

glVertex3f(bottomX[0], bottomY[0], bottomZ[0]);

glVertex3f(bottomX[1], bottomY[1], bottomZ[1]);

glVertex3f(bottomX[2], bottomY[2], bottomZ[2]);

glVertex3f(bottomX[3], bottomY[3], bottomZ[3]);

glEnd();

glFlush();

return;

}//end of draw icon

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function settrans2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void settrans2(void)

//Sets the translation matrix for the cube

{

cout << "in settrans2" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(dx, dy, dz);

glRotatef(theta, 0.0, 1.0, 0.0);// note that the angle theta is in degrees, not radians

glRotatef(theta2, 1.0, 1.0, 1.0);

return;

} //end of settrans2

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Function SetupRC\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Setup the rendering state

void SetupRC(void)

{// this function sets the clear color of an open window and clears the open window

// Set clear color to green

glClearColor(0.0, 0.0, 1.0, 1.0);

return;

}//end of SetupRC

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Functioner Timer\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TimerFunction(int value)

//this call back function is call each 30 ms and changes the location,scale and rotation

// of the square.

{

theta += 2.0;

theta2 += 2.0;

if (theta >= 720.0)theta = 0.0;

//trying out functionality for switching lights off and on

if (int(theta) % 90 == 0.0) {

accumulator += 1;

}

// Redraw the scene with new coordinates

glutPostRedisplay();

glutTimerFunc(33, TimerFunction, 1);

}

output

A picture containing shape

Description automatically generatedA picture containing chart

Description automatically generatedA picture containing graphical user interface

Description automatically generatedA picture containing shape

Description automatically generated