#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;

/\* This program demonstrates rendering Polyman in a Disco Hall. The program renders Polyman in solid form

using the function Enable(GL\_DEPTH\_TEST) to activate the z-buffer to hide hidden surfaces.

The program defines the cube as shiny plastic having reflective properties with the following statements

glEnable(GL\_COLOR\_MATERIAL);

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

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

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

Further a spotlight is set up and activated with this code

//focused spotlight with only 10 degrees one way

glLightf(GL\_LIGHT0,GL\_SPOT\_CUTOFF,10.0);

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

// point the light back to the origin

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

//enable the light

glEnable(GL\_LIGHT0);

//\*\*\*\*\*\*\*\*\*\*\* Global values\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* These values are global because the timing call back functions will only take certain parameters

hence their needs to be global variables to communicate with these functions \*/

float static theta = 0.0, theta2 = 0;;//global angular value for rotation

float scale1 = 1.0;//global scaling value for square

float dx = 0.0, dy = 0.0, dz = 0.0;//global movement value for dx and dy/

int frame = 0;

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

float ui, uj, uk;

void RenderScene(void);//this is a function to draw Polyman under lights

void loadicon(float[][3],float[][5], float[][5], float[], float[], float[], float[], float[], float[], float[], float[]); //Load the polyman icon

void drawicon(float[][3],float[][5], float[][5], float[], float[], float[], float[], float[], float[], float[], float[]); //Draw Polyman \*/

void settrans2(void);/\* sets the translation matrix for Polyman

transformation matrix for desired scale, rotation,new pos\*/

/\*performs the transformation on the icon pattern \*/

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 Polyman

//Main Program

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

{//set up window title

char header[] = "PolyDisco by Chris Stewart";

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;

}

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

void RenderScene(void)

{

float xdel = 0.25;

float nvector[7][3],px[7][5], py[7][5], pz[2], plx[4], ply[4], plz[4], pl2x[4], pl2y[4], pl3x[3], pl3y[3], pl4x[4], pl4y[4], eye[2];// These variables hold the pattern for the icon square plus line

// set up light parameters

float ambientlight[] = { 0.0,0.1,0.9,1.0 };//strong Blue ambient light

float ambientlight2[] = { 0.9,0.1,1.0,1.0 };//strong Red ambient light

float ambientlight3[] = { 1.0,1.0,1.0,1.0 };//strong Blue ambient light

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

float specular[] = { 1.0,1.0,1.0,0.0 };//specular lighting

float lightpos[] = { -2.0,4.0,4.0,1.0 };//SEE CAUTIONARY NOTE BELOW FOR COORDINATE SYSTEM

float lightpos2[] = { 2.0,4.0,4.0,1.0 };//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

//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(nvector,px, py, pz, plx, ply, pl2x, pl2y, pl3x, pl3y, eye);//Load Polyman

/\* draw Polyman \*/

glEnable(GL\_DEPTH\_TEST);

//enable lighting

glEnable(GL\_LIGHTING);

glEnable(GL\_CULL\_FACE);

glFrontFace(GL\_CCW);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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

20 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\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

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

if (theta2 < 0) {

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, 60.0);

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

// point the light back to the origin

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

//enable the light

glEnable(GL\_LIGHT0);

}

else{

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

glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambientlight2);

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, 60.0);

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

// point the light back to the origin

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

//enable the light

glEnable(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 Polyman

drawicon(nvector, px, py, pz, plx, ply, pl2x, pl2y, pl3x, pl3y, eye);

glFlush();

glEnd();

glutSwapBuffers();

return;

};//end of render scene

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

void loadicon(float nvector[][3],float px[][5], float py[][5], float pz[2], float plx[], float ply[], float pl2x[], float pl2y[], float pl3x[], float pl3y[], float eye[]) //Loads the polyman

{

nvector[0][0] = 0; nvector[0][1] = 0; nvector[0][2] = 1;

nvector[1][0] = 0; nvector[1][1] = 0; nvector[1][2] = -1;

nvector[2][0] = 0; nvector[2][1] = 1; nvector[2][2] = 0;

nvector[3][0] = 0; nvector[3][1] = -1; nvector[3][2] = 0;

nvector[4][0] = 0.832; nvector[4][1] = 0.5547; nvector[4][2] = 0;

nvector[5][0] = 0.832; nvector[5][1] = -0.5547; nvector[5][2] = 0;

if (dx == 0) {

px[0][0] = -1.125; py[0][0] = 0.0;

px[0][1] = -0.625; py[0][1] = 0.75;

px[0][2] = 0.625; py[0][2] = 0.75;

px[0][3] = 1.125; py[0][3] = -0;

px[0][4] = -1.125; py[0][4] = 0.0;

px[1][0] = -0.375; py[1][0] = 0.0;

px[1][1] = 1.125; py[1][1] = -0;

px[1][2] = 0.625; py[1][2] = -0.75;

px[1][3] = -0.625; py[1][3] = -0.75;

px[1][4] = -0.375; py[1][4] = 0.0;

}

else {

//Set the coordinates of the body

px[0][0] = -1.125; py[0][0] = 0.0;

px[0][1] = -0.625; py[0][1] = 0.75;

px[0][2] = 0.625; py[0][2] = 0.75;

px[0][3] = 1.125; py[0][3] = -0;

px[0][4] = -1.125; py[0][4] = 0.0;

px[1][0] = -1.125; py[1][0] = 0.0;

px[1][1] = 1.125; py[1][1] = -0;

px[1][2] = 0.625; py[1][2] = -0.75;

px[1][3] = -0.625; py[1][3] = -0.75;

px[1][4] = -1.125; py[1][4] = 0.0;

}

//set the right foot

plx[0] = -0.25; ply[0] = -0.5;

plx[1] = -0.25; ply[1] = -1.0;

plx[2] = -0.50; ply[2] = -1.0;

plx[3] = -0.25; ply[3] = -1.0;

//set the left foot

pl2x[0] = 0.25; pl2y[0] = -0.5;

pl2x[1] = 0.25; pl2y[1] = -1.0;

pl2x[2] = 0.0; pl2y[2] = -1.0;

pl2x[3] = 0.25; pl2y[3] = -1.0;

//set the Closed mouth

pl3x[0] = -0.375; pl3y[0] = -0.0;

pl3x[1] = -0.875; pl3y[1] = -0.5;

pl3x[2] = -0.875; pl3y[2] = -0.5;

//set the eye

eye[0] = -0.5; eye[1] = 0.5;

pz[0] = 0.5;

pz[1] = -0.5;

return;

} //end of loadicon

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

void drawicon(float nvector[][3],float px[][5], float py[][5], float pz[2], float plx[], float ply[], float pl2x[], float pl2y[], float pl3x[], float pl3y[], float eye[])

{

//draw Polyman at the transformed position

glColor3f(1.0, 1.0, 1.0);

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(-1, -1, -2);

glVertex3f(1, -1, -2);

glVertex3f(1, -1, 2);

glVertex3f(-1, -1, 2);

glVertex3f(-1, -1, -2);

glEnd();

int i;

cout << "in drawicon" << endl;

glColor3f(1.0, 0.0, 0.0);

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

for (i = 0; i < 5; i++) {

glVertex3f(px[1][i], py[1][i], pz[0]);

}

for (i = 0; i < 5; i++) {

glVertex3f(px[0][i], py[0][i], pz[0]);

}

glEnd();

i = 0;

glFrontFace(GL\_CW);

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

for (i = 0; i < 5; i++) {

glVertex3f(px[1][i], py[1][i], pz[1]);

}

for (i = 0; i < 5; i++) {

glVertex3f(px[0][i], py[0][i], pz[1]);

}

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(px[0][0], py[0][0], pz[0]);

glVertex3f(px[0][0], py[0][0], pz[1]);

glVertex3f(px[0][1], py[0][1], pz[1]);

glVertex3f(px[0][1], py[0][1], pz[0]);

glVertex3f(px[0][0], py[0][0], pz[0]);

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(px[1][0], py[1][0], pz[0]);

glVertex3f(px[1][0], py[1][0], pz[1]);

glVertex3f(px[0][0], py[0][0], pz[1]);

glVertex3f(px[0][0], py[0][0], pz[0]);

glVertex3f(px[1][0], py[1][0], pz[0]);

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(px[1][0], py[1][0], pz[0]);

glVertex3f(px[1][0], py[1][0], pz[1]);

glVertex3f(px[1][3], py[1][3], pz[1]);

glVertex3f(px[1][3], py[1][3], pz[0]);

glVertex3f(px[1][0], py[1][0], pz[0]);

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

glNormal3f(nvector[3][0], nvector[3][1], nvector[3][2]);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(px[1][2], py[1][2], pz[0]);

glVertex3f(px[1][2], py[1][2], pz[1]);

glVertex3f(px[1][3], py[1][3], pz[1]);

glVertex3f(px[1][3], py[1][3], pz[0]);

glVertex3f(px[1][2], py[1][2], pz[0]);

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

glNormal3f(nvector[5][0], nvector[5][1], nvector[5][2]);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(px[1][2], py[1][2], pz[0]);

glVertex3f(px[1][2], py[1][2], pz[1]);

glVertex3f(px[0][3], py[0][3], pz[1]);

glVertex3f(px[0][3], py[0][3], pz[0]);

glVertex3f(px[1][2], py[1][2], pz[0]);

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

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

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(px[0][3], py[0][3], pz[0]);

glVertex3f(px[0][3], py[0][3], pz[1]);

glVertex3f(px[0][2], py[0][2], pz[1]);

glVertex3f(px[0][2], py[0][2], pz[0]);

glVertex3f(px[0][3], py[0][3], pz[0]);

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

//Top

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

glColor3f(0.0, 0.5, 0.5);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(px[0][2], py[0][2], pz[1]);

glVertex3f(px[0][1], py[0][1], pz[1]);

glVertex3f(px[0][1], py[0][1], pz[0]);

glVertex3f(px[0][2], py[0][2], pz[0]);

glVertex3f(px[0][2], py[0][2], pz[1]);

glEnd();

glColor3f(1.0, 1.0, 1.0);

//now draw the line

glBegin(GL\_LINES);

glVertex3f(plx[0], ply[0], pz[1]);

for (i = 1; i <= 3; i++)

{

glVertex3f(plx[i], ply[i], pz[1]);

}//glVertex2f(plxp[2],plyp[2]);

glEnd();

glBegin(GL\_LINES);

glVertex3f(pl2x[0], pl2y[0], pz[0]);

for (i = 1; i <= 3; i++)

{

glVertex3f(pl2x[i], pl2y[i], pz[0]);

}

glEnd();

//Draw Eye

glBegin(GL\_POINTS);

glColor3f(0.0, 0.0, 0.0);

glVertex3f(eye[0], eye[1], pz[0]);

glEnd();

glBegin(GL\_POINTS);

glColor3f(0.0, 0.0, 0.0);

glVertex3f(eye[0], eye[1], pz[1]);

//Draw mouth

glColor3f(0.0, 0.0, 0.0);

glEnd();

if (dx == 0) {//Open mouth

}

else {//ClosedMouth

glBegin(GL\_LINES);

glColor3f(0.0, 0.0, 0.0);

glVertex3f(pl3x[1], pl3y[1], pz[0]);

glVertex3f(pl3x[0], pl3y[0], pz[0]);

glVertex3f(pl3x[2], pl3y[2], pz[0]);

glEnd();

glColor3f(0.0, 0.0, 0.0);

glVertex3f(pl3x[1], pl3y[1], pz[1]);

glVertex3f(pl3x[0], pl3y[0], pz[1]);

glVertex3f(pl3x[2], pl3y[2], pz[1]);

}

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(30, 1.0, 0.0, 0.0);

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

glRotatef(theta2, 0.0, 1.0, 0.0);

return;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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.

{

int count = 0;

switch (frame)

{

case 0: //frame 1 polyman starts at right (7, -3) and walks to the middle right (3.5,-3)

//Polyman parameters

theta2 = theta2 - 10;

if (theta2 == -170) {

frame =1;

}

else {

frame =0;

}

break;

case 1: //frame 1 polyman starts at right (7, -3) and walks to the middle right (3.5,-3)

//Polyman parameters

theta2 = theta2 + 10;

if (theta2 == 0) {

frame = 2;

}

else {

frame = 1;

}

break;

case 2: //frame 1 polyman starts at right (7, -3) and walks to the middle right (3.5,-3)

//Polyman parameters

//we key on Polyman's position to change the frame

theta = theta + 5;

count++;

if (count == 72) {

frame = 0;

}frame = 3;

break;

case 3:// frame 2 polyman jumps to y=5

theta = theta + 5;

count++;

if (count == 64) {

frame = 0;

}frame = 2;

break;

}

// Redraw the scene with new coordinates

glutPostRedisplay();

glutTimerFunc(100, TimerFunction, 1);

}

