CSC328 Graphics Problem 7 Poly Skies – Nick Wright

Code

#include<windows.h>

#include<GL/glut.h>

#include<stdlib.h>

#include<math.h>

#include<conio.h>

#include<stdio.h>

#include <iostream>

#include <iomanip>

#include <gl/glut.h>

using namespace std;

/\*-----------------Global Variables------------------\*/

//theta = global angular value for rotation

//dx and dy = global movement values for x and y, respectively

//POLYMAN GLOBAL VARIABLES --- He is starting on the right side of the screen

float bodyTheta = 0, bodyTheta2 = 180, bodyTheta3 = 20, bodyDX = 7.0, bodyDY = -3.0, bodyDZ = 1.0; //global values for the body

//float mouthTheta = 0, mouthTheta2 = 180, mouthDX = 7.0, mouthDY = -3.0, mouthDZ = 1.0; //global values for the mouth

float leg1Theta = 0, leg1Theta2 = 180, leg1Theta3 = 20, leg1DX = 7.0, leg1DY = -3.0, leg1DZ = 1.0; //global value for leg 1

float leg2Theta = 0, leg2Theta2 = 180, leg2Theta3 = 20, leg2DX = 7.0, leg2DY = -3.0, leg2DZ = 1.0; //global value for leg 2

float slopeTheta = 0, slopeTheta2 = 0, slopeDX = 0.0, slopeDY = -2.5, slopeDZ = -2.0; //global values for the floor

//float rockAccumulator = 0; //accumulates to break out of rock loop

//int frame = 1; //frame value

float xctrl[10], yctrl[10], uval = 0.0; // these are for Biezier Control points for the path for the of the cube. <- chnage the values of this

float calcbiezu(float, int, float[]); //calclated biez at a point u

int fact(int); //calclates factorial

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

//\*\*\*CREATING THE LOADS AND DRAWS\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYMAN LOADS AND DRAWS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//body functions

void loadBody(float[][4], float[][4], float[][4], float[][4], float[][4], float[][4],

float[2], float[2], float[2], float[][3]);

void drawBody(float[][4], float[][4], float[][4], float[][4], float[][4], float[][4],

float[2], float[2], float[2], float[][3]);

//mouth functions

void loadMouth(float[][3], float[][3], float[][3]); //loads the mouth

void drawMouth(float[][3], float[][3], float[][3]); //draws the mouth

//leg functions

void drawLeg(float[], float[], float[]); //draws legs

void loadLeg1(float[], float[], float[]); //loads leg 1

void loadLeg2(float[], float[], float[]); //loads leg 2

void loadSlope(float[], float[]);

void drawSlope(float[], float[]);

//\*\*\*CREATING THE MODELVIEW MATRICIES\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYMAN MODELVIEW\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void bodyModel(void); //sets the MODELVIEW MATRIX for the body and mouth (rotation/translation matrix)

void leg1Model(void); //sets the MODELVIEW MATRIX for leg 1 (rotation/translation matrix)

void leg2Model(void); //sets the MODELVIEW MATRIX for leg 2 (rotation/translation matrix)

void slopeModel(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, scaleand rotation

// of the square.

//Main Program

int main(int argc, char\*\* argv) { //set up window title

char header[] = "Assignment 7 Biezier Curve 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(540, 440);

glutInitWindowPosition(0, 0);

// Open and Label Window

glutCreateWindow(header);

glutDisplayFunc(RenderScene);

glutTimerFunc(500, TimerFunction, 1);

//Now draw the scene

glutMainLoop();

return 0;

}

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

void RenderScene(void) {

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYMAN PATTERN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//pattern for polyman body

float upperX[2][4], upperY[2][4], upperZ[2][4], lowerX[2][4],

lowerY[2][4], lowerZ[2][4], eyeX[2], eyeY[2], eyeZ[2], nvector[8][3]; //polyman pattern

//pattern for mouth

float mX[2][3], mY[2][3], mZ[2][3];

//pattern for legs 1 and 2

float l1x[4], l1y[4], l1z[4];

float l2x[4], l2y[4], l2z[4];

//pattern for the slope

float slopeX[3], slopeY[3];

float x1, y1, xdel = 0.25;

float Uval;

// Biezier u value going from 0 to 1 to drive the cube. The cube values are x(u), y(u)

// Set Up AThe Control Points

xctrl[0] = 0.0; yctrl[0] = 10.0; //left end point

xctrl[1] = 7.0; yctrl[1] = 1.0; //point 1

xctrl[2] = 9.0; yctrl[2] = 5.0; //point 2

xctrl[3] = 5.0; yctrl[3] = 9.0; //point 3

xctrl[4] = 4.0; yctrl[4] = 6.0; //point 4

xctrl[5] = 5.0; yctrl[5] = 5.0; //point 5

xctrl[6] = 6.0; yctrl[6] = 4.0; //point 6

xctrl[7] = 7.0; yctrl[7] = 3.0; //point 7

xctrl[8] = 8.0; yctrl[8] = 2.0; //point 8

xctrl[9] = 7.5; yctrl[9] = 0.0; //right end point

int ncontrolpts = 10, i; // <----- change the number of control points based on how many you have (was 6)

//clear the window with the current background color

cout << "in renderscene" << endl;

glClearColor(0.0, 0.0, 0.0, 1.0); //set clear color to black

glClear(GL\_COLOR\_BUFFER\_BIT); //note clear color was set inn SetupRC

glEnable(GL\_DEPTH\_TEST);

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

glLoadIdentity();

//set the current drawing color to white

glColor3f(1.0, 1.0, 1.0);

//set the viewport to the window dimensions

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//set the viewport to the window dimensions

glViewport(0, 0, 540, 440);

//Establish the clipping volume in user coordinates

glOrtho(-2.0, 11.0, -2.0, 11.0, 5.0, -5.0);

//\*\*\*\*\*\*\*\*\*\*\*\*THE FOLLOWING DRAWS THE BIEZIER CURVE CREATED BY THE CONTROL POINTS AND THE AXIS FOR THE CONTROL POINTS\*\*\*\*\*\*\*\*\*\*\*\*

//set the drawing color to white

glColor3f(1.0, 1.0, 1.0);

//drawing triangle (slope)

glBegin(GL\_POLYGON);

glColor3f(1.0, 1.0, 1.0);

glVertex2f(0, 10);

glVertex2f(0, 0);

glVertex2f(7.5, 0);

glFlush();

glEnd();

//drawing the triangle to represent the slope

// now draw the CONTROL POINTS

glPointSize(5.0);

//loop through all the points

glBegin(GL\_POINTS);

glColor3f(0.0, 1.0, 0.0);

for (i = 0; i < ncontrolpts; i++) glVertex2f(xctrl[i], yctrl[i]);

glEnd();

// DRAW THE BIEZIER CURVE FOR THE CUBE TO FOLLOW

// change the draw to red

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_LINE\_STRIP);

Uval = 0.0;

for (i = 0; i <= 20; i++) {

//calculate the x,y coordinates for this uval <-------------- i <= 20 was initial value

glVertex2f(calcbiezu(Uval, 9, xctrl), calcbiezu(Uval, 9, yctrl));

Uval += 0.05;

}

glFlush();

glEnd();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*THIS IS THE END OF THE BIEZIER CURVE DRAW\*\*\*\*\*\*\*

/\*

//drawing triangle (slope)

glBegin(GL\_POLYGON);

glColor3f(1.0, 1.0, 1.0);

glVertex2f(0, 10);

glVertex2f(0, 0);

glVertex2f(7.5, 0);

glFlush();

glEnd();

\*/

//load the icons untransformed

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYMAN LOADS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

loadBody(upperX, upperY, upperZ, lowerX, lowerY, lowerZ, eyeX, eyeY, eyeZ, nvector);

loadMouth(mX, mY, mZ);

loadLeg1(l1x, l1y, l1z);

loadLeg2(l2x, l2y, l2z);

//loadSlope(slopeX, slopeY);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYMAN TRANFORMATIONS AND DRAWS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//glFlush being performed after each draw

bodyModel(); //body modelview matrix

drawBody(upperX, upperY, upperZ, lowerX, lowerY, lowerZ, eyeX, eyeY, eyeZ, nvector);

glFlush();

leg1Model(); //leg 1 modelview matrix

drawLeg(l1x, l1y, l1z);

glFlush();

leg2Model(); //leg 2 modelview matrix

drawLeg(l2x, l2y, l2z);

glFlush();

bodyModel();

//slopeModel();

//drawSlope(slopeX, slopeY);

glEnd();

glutSwapBuffers();

return;

}; //end of render scene

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYMAN FUNCTIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void loadMouth(float mX[][3], float mY[][3], float mZ[][3])

{

//load the front mouth

mX[0][0] = (-9.0 / 8); mY[0][0] = (0); mZ[0][0] = (1.0 / 2);

mX[0][1] = (-3.0 / 8); mY[0][1] = (0); mZ[0][1] = (1.0 / 2);

mX[0][2] = (-5.0 / 8); mY[0][2] = (-3.0 / 4); mZ[0][2] = (1.0 / 2);

//load the back mouth

mX[1][0] = (-9.0 / 8); mY[1][0] = (0); mZ[1][0] = (-1.0 / 2);

mX[1][1] = (-3.0 / 8); mY[1][1] = (0); mZ[1][1] = (-1.0 / 2);

mX[1][2] = (-5.0 / 8); mY[1][2] = (-3.0 / 4); mZ[1][2] = (-1.0 / 2);

}//end of loadMouth

void drawMouth(float mX[][3], float mY[][3], float mZ[][3])

{

//front mouth

glColor3f(1.0, 1.0, 0); //setting color to yellow

glFrontFace(GL\_CCW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(mX[0][0], mY[0][0], mZ[0][0]);

glVertex3f(mX[0][1], mY[0][1], mZ[0][1]);

glVertex3f(mX[0][2], mY[0][2], mZ[0][2]);

glEnd();

glFlush();

//back mouth

glColor3f(1.0, 1.0, 1.0); //setting color to white

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(mX[1][0], mY[1][0], mZ[1][0]);

glVertex3f(mX[1][1], mY[1][1], mZ[1][1]);

glVertex3f(mX[1][2], mY[1][2], mZ[1][2]);

glEnd();

glFlush();

//side mouth

glColor3f(2.0, 0.5, 1.0);//lilac

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(mX[0][0], mY[0][0], mZ[0][0]);

glVertex3f(mX[1][0], mY[1][0], mZ[1][0]);

glVertex3f(mX[1][2], mY[1][2], mZ[1][2]);

glVertex3f(mX[0][2], mY[0][2], mZ[0][2]);

glEnd();

glFlush();

return;

}

void loadBody(float upperX[][4], float upperY[][4], float upperZ[][4], float lowerX[][4], float lowerY[][4],

float lowerZ[][4], float eyeX[2], float eyeY[2], float eyeZ[2], float nvector[][3]) {

//upper coords front face

upperX[0][0] = -9.0 / 8; upperY[0][0] = 0; upperZ[0][0] = 1.0 / 2;

upperX[0][1] = -5.0 / 8; upperY[0][1] = 3.0 / 4; upperZ[0][1] = 1.0 / 2;

upperX[0][2] = 5.0 / 8; upperY[0][2] = 3.0 / 4; upperZ[0][2] = 1.0 / 2;

upperX[0][3] = 9.0 / 8; upperY[0][3] = 0; upperZ[0][3] = 1.0 / 2;

//lower coords front face

lowerX[0][0] = -5.0 / 8; lowerY[0][0] = -3.0 / 4; lowerZ[0][0] = 1.0 / 2;

lowerX[0][1] = -3.0 / 8; lowerY[0][1] = 0; lowerZ[0][1] = 1.0 / 2;

lowerX[0][2] = 9.0 / 8; lowerY[0][2] = 0; lowerZ[0][2] = 1.0 / 2;

lowerX[0][3] = 5.0 / 8; lowerY[0][3] = -3.0 / 4; lowerZ[0][3] = 1.0 / 2;

//eye coordinates front face

eyeX[0] = -1.0 / 2; eyeY[0] = 1.0 / 2; eyeZ[0] = 1.0 / 2;

//loading nvector for front face

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

//upper coords back face

upperX[1][0] = -9.0 / 8; upperY[1][0] = 0; upperZ[1][0] = -1.0 / 2;

upperX[1][1] = -5.0 / 8; upperY[1][1] = 3.0 / 4; upperZ[1][1] = -1.0 / 2;

upperX[1][2] = 5.0 / 8; upperY[1][2] = 3.0 / 4; upperZ[1][2] = -1.0 / 2;

upperX[1][3] = 9.0 / 8; upperY[1][3] = 0; upperZ[1][3] = -1.0 / 2;

//lower coords back face

lowerX[1][0] = -5.0 / 8; lowerY[1][0] = -3.0 / 4; lowerZ[1][0] = -1.0 / 2;

lowerX[1][1] = -3.0 / 8; lowerY[1][1] = 0; lowerZ[1][1] = -1.0 / 2;

lowerX[1][2] = 9.0 / 8; lowerY[1][2] = 0; lowerZ[1][2] = -1.0 / 2;

lowerX[1][3] = 5.0 / 8; lowerY[1][3] = -3.0 / 4; lowerZ[1][3] = -1.0 / 2;

//eye coords back face

eyeX[1] = -1.0 / 2; eyeY[1] = 1.0 / 2; eyeZ[1] = -1.0 / 2;

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

//loading coords for the other sides

//top

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

//bottom

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

//upper back

nvector[4][0] = (3 / sqrt(13)); nvector[4][1] = (2 / sqrt(13)); nvector[4][2] = 0;

//lower back

nvector[5][0] = (3 / sqrt(13)); nvector[5][1] = (-2 / sqrt(13)); nvector[5][2] = 0;

//upper front

nvector[6][0] = (-3 / sqrt(13)); nvector[6][1] = (2 / sqrt(13)); nvector[6][2] = 0;

//lower front

nvector[4][0] = (-3 / sqrt(13)); nvector[4][1] = (-2 / sqrt(13)); nvector[4][2] = 0;

return;

}//end of loadBody

void drawBody(float upperX[][4], float upperY[][4], float upperZ[][4], float lowerX[][4], float lowerY[][4],

float lowerZ[][4], float eyeX[2], float eyeY[2], float eyeZ[2], float nvector[][3])

{

int i;

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

//back face

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

//glColor3f(1.0, 1.0, 0.0);

glColor3f(1.0, 1.0, 0.0);

for (i = 3; i >= 0; i--)

{

glVertex3f(upperX[0][i], upperY[0][i], upperZ[0][i]);

}

glEnd();

glFlush();

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

//glColor3f(1.0, 1.0, 0.0);

glColor3f(1.0, 1.0, 0.0);

for (i = 3; i >= 0; i--)

{

glVertex3f(lowerX[0][i], lowerY[0][i], lowerZ[0][i]);

}

glEnd();

glFlush();

glFrontFace(GL\_CCW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

//eye 1

glColor3f(0.0, 0.0, 0.0);

glPointSize(4);

glBegin(GL\_POINTS);

glVertex3f(eyeX[0], eyeY[0], eyeZ[0]);

glEnd();

glFlush();

//front face

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

glColor3f(1.0, 1.0, 1.0); //-------------------------------------------------changed front face to black

//glColor3f(1.0, 1.0, 1.0);

for (i = 3; i >= 0; i--)

{

glVertex3f(upperX[1][i], upperY[1][i], upperZ[1][i]);

}

glEnd();

glFlush();

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

glColor3f(1.0, 1.0, 1.0); //--------------------------------------------------changed front face to black

//glColor3f(1.0, 1.0, 1.0);

for (i = 3; i >= 0; i--)

{

glVertex3f(lowerX[1][i], lowerY[1][i], lowerZ[1][i]);

}

glEnd();

glFlush();

//front eye

glColor3f(0.0, 0.0, 0.0);

glPointSize(4);

glBegin(GL\_POINTS);

glVertex3f(eyeX[1], eyeY[1], eyeZ[1]);

glEnd();

glFlush();

//connecting pieces

//upper head

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glNormal3f(nvector[6][0], nvector[6][1], nvector[6][2]);

glColor3f(0.5, 0.5, 1.0);

glVertex3f(upperX[0][1], upperY[0][1], upperZ[0][1]);

glVertex3f(upperX[1][1], upperY[1][1], upperZ[1][1]);

glVertex3f(upperX[1][0], upperY[1][0], upperZ[1][0]);

glVertex3f(upperX[0][0], upperY[0][0], upperZ[0][0]);

glEnd();

glFlush();

//top

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

glColor3f(1, 0, 0); //red

glVertex3f(upperX[0][2], upperY[0][2], upperZ[0][2]);

glVertex3f(upperX[1][2], upperY[1][2], upperZ[1][2]);

glVertex3f(upperX[1][1], upperY[1][1], upperZ[1][1]);

glVertex3f(upperX[0][1], upperY[0][1], upperZ[0][1]);

glEnd();

glFlush();

//upper back

glFrontFace(GL\_CCW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

glColor3f(0.0, 1.0, 0.0); //bronze

glVertex3f(upperX[0][2], upperY[0][2], upperZ[0][2]);

glVertex3f(upperX[1][2], upperY[1][2], upperZ[1][2]);

glVertex3f(upperX[1][3], upperY[1][3], upperZ[1][3]);

glVertex3f(upperX[0][3], upperY[0][3], upperZ[0][3]);

glEnd();

glFlush();

//upper mouth

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glColor3f(0.0, 0.5, 0.5); //blue green

glVertex3f(upperX[0][0], upperY[0][0], upperZ[0][0]);

glVertex3f(upperX[1][0], upperY[1][0], upperZ[1][0]);

glVertex3f(upperX[1][3], upperY[1][3], upperZ[1][3]);

glVertex3f(upperX[0][3], upperY[0][3], upperZ[0][3]);

glEnd();

glFlush();

//lower mouth

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glColor3f(1.0, 0.5, 0.0); //orange

glVertex3f(lowerX[0][1], lowerY[0][1], lowerZ[0][1]);

glVertex3f(lowerX[1][1], lowerY[1][1], lowerZ[1][1]);

glVertex3f(lowerX[1][0], lowerY[1][0], lowerZ[1][0]);

glVertex3f(lowerX[0][0], lowerY[0][0], lowerZ[0][0]);

glEnd();

glFlush();

//lower back

glFrontFace(GL\_CCW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

glColor3f(0.1, 0.0, 0.0); //brown

glVertex3f(lowerX[0][2], lowerY[0][2], lowerZ[0][2]);

glVertex3f(lowerX[1][2], lowerY[1][2], lowerZ[1][2]);

glVertex3f(lowerX[1][3], lowerY[1][3], lowerZ[1][3]);

glVertex3f(lowerX[0][3], lowerY[0][3], lowerZ[0][3]);

glEnd();

glFlush();

//bottom

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

glColor3f(0.5, 1.0, 1.0); //cyan

glVertex3f(lowerX[0][0], lowerY[0][0], lowerZ[0][0]);

glVertex3f(lowerX[1][0], lowerY[1][0], lowerZ[1][0]);

glVertex3f(lowerX[1][3], lowerY[1][3], lowerZ[1][3]);

glVertex3f(lowerX[0][3], lowerY[0][3], lowerZ[0][3]);

glEnd();

glFlush();

return;

}//end of drawbody

void loadLeg1(float l1x[], float l1y[], float l1z[])

{

//this function will load leg 1

l1x[0] = -1.0 / 4; l1y[0] = -1.0 / 2; l1z[0] = 1.0 / 2;

l1x[1] = -1.0 / 4; l1y[1] = -1.0; l1z[1] = 1.0 / 2;

l1x[2] = -1.0 / 2; l1y[2] = -1.0; l1z[2] = 1.0 / 2;

//cyan color

l1x[3] = 0; l1y[3] = 1.0; l1z[3] = 1.0;

return;

}//end of loadLeg1

void drawLeg(float l1x[], float l1y[], float l1z[])

{

//this function will draw leg 1

//setting color

glColor3f(l1x[3], l1y[3], l1z[3]);

glBegin(GL\_LINE\_STRIP);

glVertex3f(l1x[0], l1y[0], l1z[0]);

glVertex3f(l1x[1], l1y[1], l1z[1]);

glVertex3f(l1x[2], l1y[2], l1z[2]);

glEnd();

glFlush();

return;

}//end of drawLeg1

void loadLeg2(float l2x[], float l2y[], float l2z[])

{

//this function will load leg 2

l2x[0] = 1.0 / 4; l2y[0] = -1.0 / 2; l2z[0] = -1.0 / 2;

l2x[1] = 1.0 / 4; l2y[1] = -1.0; l2z[1] = -1.0 / 2;

l2x[2] = 0; l2y[2] = -1.0; l2z[2] = -1.0 / 2;

//blue color

l2x[3] = 0; l2y[3] = 0; l2z[3] = 1.0;

return;

}//end of loadLeg1

void bodyModel()

{

//float bodyTheta = 0, bodyDX = -6.0, bodyDY = -3.0;

//sets the modelviel matrix for the body

cout << "in bodyModel" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

/\*

glTranslatef(bodyDX, bodyDY, bodyDZ);

// note that the angle theta is in degrees, not radians

glRotatef(bodyTheta, 0.0, 0.0, 1.0);

glRotatef(bodyTheta2, 0.0, 1.0, 0.0);

glRotatef(30.0, 1.0, 0.0, 0.0);

\*/

// Set the Biezier location for the x,y, draw dx(uval), dy(uval), Note that the annimation for movement is in the TimerFunction

bodyDX = calcbiezu(uval, 9, xctrl);

bodyDY = calcbiezu(uval, 9, yctrl);

glTranslatef(bodyDX, bodyDY, bodyDZ);

glRotatef(bodyTheta2, 0.0, 1.0, 0.0); // note that the angle theta is in degrees, not radians

//glRotatef(bodyTheta, 1.0, 1.0, 1.0);

glRotatef(bodyTheta, 0.0, 0.0, 1.0);

glRotatef(bodyTheta3, 1.0, 0.0, 0.0); //rotate along x axis

return;

}//end of bodyModel

//function leg1Model

void leg1Model()

{

//float leg1Theta = 0, leg1DX = -6.0, leg1DY = -3.0;

//sets the modelviel matrix for leg1

cout << "in leg1Model" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

/\*

glTranslatef(leg1DX, leg1DY, leg1DZ);

// note that the angle theta is in degrees, not radians

glRotatef(leg1Theta, 0.0, 0.0, 1.0);

glRotatef(leg1Theta2, 0.0, 1.0, 0.0);

glRotatef(30.0, 1.0, 0.0, 0.0);

\*/

// Set the Biezier location for the x,y, draw dx(uval), dy(uval), Note that the annimation for movement is in the TimerFunction

leg1DX = calcbiezu(uval, 9, xctrl);

leg1DY = calcbiezu(uval, 9, yctrl);

glTranslatef(leg1DX, leg1DY, leg1DZ);

glRotatef(leg1Theta2, 0.0, 1.0, 0.0); // note that the angle theta is in degrees, not radians

//glRotatef(leg1Theta, 1.0, 1.0, 1.0);

glRotatef(leg1Theta, 0.0, 0.0, 1.0);

glRotatef(leg1Theta3, 1.0, 0.0, 0.0); //rotate along x axis

return;

}//end of leg1Model

//function leg2Model

void leg2Model()

{

//float leg2Theta = 0, leg2DX = -6.0, leg2DY = -3.0;

//sets the modelviel matrix for leg2

cout << "in leg2Model" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

/\*

glTranslatef(leg2DX, leg2DY, leg2DZ);

// note that the angle theta is in degrees, not radians

glRotatef(leg2Theta, 0.0, 0.0, 1.0);

glRotatef(leg2Theta2, 0.0, 1.0, 0.0);

glRotatef(30.0, 1.0, 0.0, 0.0);

\*/

leg2DX = calcbiezu(uval, 9, xctrl);

leg2DY = calcbiezu(uval, 9, yctrl);

glTranslatef(leg2DX, leg2DY, leg2DZ);

// note that the angle theta is in degrees, not radians

glRotatef(leg2Theta2, 0.0, 1.0, 0.0); //rotate along y axis

//glRotatef(leg2Theta, 1.0, 1.0, 1.0);

glRotatef(leg2Theta, 0.0, 0.0, 1.0); //rotate along z axis

glRotatef(leg2Theta3, 1.0, 0.0, 0.0); //rotate along x axis

return;

}//end of leg2Model

void loadSlope(float slopeX[], float slopeY[]) {

//p1

slopeX[0] = 0; slopeY[0] = 10;

//p2

slopeX[1] = 0; slopeY[1] = 0;

//p3

slopeX[2] = 7.5; slopeY[2] = 0;

/\*

//drawing triangle (slope)

glBegin(GL\_POLYGON);

glColor3f(1.0, 1.0, 1.0);

glVertex2f(0, 10);

glVertex2f(0, 0);

glVertex2f(7.5, 0);

glFlush();

glEnd();

\*/

return;

}//end of loadFloor

void drawSlope(float slopeX[], float slopeY[]) {

glBegin(GL\_POLYGON);

glColor3f(1.0, 1.0, 1.0);

glVertex2f(slopeX[0], slopeY[0]);

glVertex2f(slopeX[1], slopeY[1]);

glVertex2f(slopeX[2], slopeY[2]);

glFlush();

glEnd();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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, 0.0, 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

// Not4e that the uval is used to update the Biezier positon of the Cube x (uval), y(uaval).the

// calls to the Biezier function are in settrans2()

// of the square.

{

//bodytheta

if (bodyDX >= 3.5 && bodyDX <= 7) {// && bodyDX <= 5

bodyTheta -= 5.0;

leg1Theta -= 5.0;

leg2Theta -= 5.0;

}

if (bodyDX >= 7.01) {

bodyTheta = 0.0;

leg1Theta = 0.0;

leg2Theta = 0.0;

}

/\*

if (bodyDX >= 5) {

bodyTheta = 0.0;

leg1Theta = 0.0;

leg2Theta = 0.0;

}

\*/

uval += 0.01;

if (uval >= 1.0) uval = 1.0;

// Redraw the scene with new coordinates

glutPostRedisplay();

glutTimerFunc(33, TimerFunction, 1);

}

float calcbiezu(float u, int n, float cp[]) { //This function calculates the biezier value at u for the control points cp..

float val = 0.0;

int i;

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

val += cp[i] \* float(fact(n)) / float((fact(i) \* fact(n - i))) \* pow(u,

float(i)) \* pow(float(1.0 - u), float(n - i));

}

return val;

}

int fact(int n) {

// Variable Declaration

//This function calculates n!

int counter, fct = 1;

if (n == 0) return 1;

//for Loop Block

for (int counter = 1; counter <= n; counter++) {

fct = fct \* counter;

}

return fct;

}

void slopeModel()

{

//sets the modelviel matrix for the floor

cout << "in floorModel" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(slopeDX, slopeDY, slopeDZ);

// note that the angle theta is in degrees, not radians

glRotatef(slopeTheta, 0.0, 0.0, 1.0);

glRotatef(slopeTheta2, 0.0, 1.0, 0.0);

glRotatef(0, 1.0, 0.0, 0.0);

return;

}//end of bodyModel

Output

Graphical user interface

Description automatically generated with medium confidenceGraphical user interface

Description automatically generatedGraphical user interface

Description automatically generated