Nick Wright Problem 6 Polydisco 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;

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* HOMEWORK 6 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* NICK WRIGHT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The purpose of this program is to have a solid of polyman and polywoman walk onto the stage from opposite

sides. When polyman reaches the center of the screen he will open his mouth, jump and do a flip. Polywoman

will open her mouth and rock back and forth in approval. Both icons will then turn and walk off the left side of

the screen.

\*/

/\*-----------------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 = 0, bodyDX = 7.0, bodyDY = -3.0, bodyDZ = 1.0; //global values for the body

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

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

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

float floorTheta = 0, floorTheta2 = 0, floorDX = 0.0, floorDY = -2.5, floorDZ = -2.0; //global values for the floor

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

int frame = 1;

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

void RenderScene(void);//this is a function to draw the scene 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 loadFloor(float[], float[], float[], float nvectorX, float nvectorY, float nvectorZ);

void drawFloor(float[], float[], float[], float nvectorX, float nvectorY, float nvectorZ);

//\*\*\*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 floorModel(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 square

//Main Program

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

{

//set up the window title

char header[] = "Polydisco by Nick Wright (Homework 6)";

/\*glutInit() initializes GLUT. Takes the command line arguments which are used to

initialize the native window system.

This function must be called before any other GLUT functions.\*/

glutInit(&argc, argv);

//set up the display mode with a single buffer and rgb colors

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

SetupRC();

//initialize the window size and position

glutInitWindowSize(560, 440);

glutInitWindowPosition(140, 20);

//Initialize background color in window to red

// Open and Label Window

glutCreateWindow(header);

glutDisplayFunc(RenderScene);

glutTimerFunc(30, TimerFunction, 1);

//now draw the scene

glutMainLoop();

return 0;

}

//Render Scene Function

void RenderScene(void)

{

float xdel = 0.25;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FLOOR PATTERN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

float floorX[4], floorY[4], floorZ[4];

float nvectorX = 0, nvectorY = 0, nvectorZ = 0;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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];

//clear the window with the current background color

cout << "in renderscene" << endl;

//set the current drawing color to white

glColor3f(1.0, 1.0, 1.0);

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, 5.0, -5.0);

//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);

loadFloor(floorX, floorY, floorZ, nvectorX, nvectorY, nvectorZ);

glEnable(GL\_DEPTH\_TEST);

//more lighting

//enable lighting

glEnable(GL\_LIGHTING);

glEnable(GL\_CULL\_FACE); //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* try taking this line out later

glFrontFace(GL\_CCW); //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* try taking this line out later

//do not ignore following 1 lines of code

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

//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[] = { -4.0,4.0,4.0,1.0 };//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[] = { 4.0,-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 (frame >= 2) {

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[] = { 4.0,4.0,4.0,1.0 };//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[] = { -4.0,-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 (frame <=2) {

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 with the background color

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

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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();

//if the body is not at x = 0, the mouth will be drawn closed, else it will be drawn open

if (bodyDX != 0)

{

drawMouth(mX, mY, mZ);

glFlush();

}

floorModel();

drawFloor(floorX, floorY, floorZ, nvectorX, nvectorY, nvectorZ);

glFlush();

glFlush();

glEnd();

glutSwapBuffers();

return;

}//end of renderscene

//function timer

void TimerFunction(int value)

{

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

switch (frame)

{

case 1:

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

//body parameters

bodyDX -= 0.15;

//mouth parameters

mouthDX -= 0.15;

//leg parameters

leg1DX -= 0.15;

leg2DX -= 0.15;

//if else statement to make the legs move up and down

if (leg1DY > -3) {

leg1DY -= 0.1; leg2DY += 0.1;

}

else {

leg1DY += 0.1; leg2DY -= 0.1;

}

//use body position to change frame

if (bodyDX <= 0)

{

//polyman

bodyDX = 0;

mouthDX = 0;

leg1DX = 0;

leg2DX = 0;

leg1DY = -3.0;

leg2DY = -3.0;

frame = 2;

}

break;

case 2:

//frame 2 polyman opens his mouth and rotates

bodyTheta += 2.0;

bodyTheta2 += 2.0;

leg1Theta += 2.0;

leg1Theta2 += 2.0;

leg2Theta += 2.0;

leg2Theta2 += 2.0;

if (bodyTheta == 360)

{

bodyTheta = 0.0;

bodyTheta2 = 0.0;

leg1Theta = 0.0;

leg1Theta2 = 0.0;

leg2Theta = 0.0;

leg2Theta2 = 0.0;

frame = 3;

}

break;

case 3:

//frame 3 polyman shuffles forwards

//polyman

bodyDX -= 0.15;

mouthDX -= 0.15;

leg1DX -= 0.15;

leg2DX -= 0.15;

if (bodyDX <= -1.20)

{

bodyDX = -1.20;

leg1DX = -1.20;

leg2DX = -1.20;

mouthDX = -1.20;

frame = 4;

}

break;

case 4:

//frame 4 polyman shuffles backwards

bodyDX += 0.15;

mouthDX += 0.15;

leg1DX += 0.15;

leg2DX += 0.15;

if (bodyDX >= 1.20)

{

bodyDX = 1.20;

leg1DX = 1.20;

leg2DX = 1.20;

mouthDX = 1.20;

frame = 5;

}

break;

case 5:

//frame 5 polyman shuffles forwards once more

//polyman

bodyDX -= 0.15;

mouthDX -= 0.15;

leg1DX -= 0.15;

leg2DX -= 0.15;

if (bodyDX <= 0.0)

{

bodyDX = 0.0;

leg1DX = 0.0;

leg2DX = 0.0;

mouthDX = 0.0;

frame = 2;

}

break;

}

// Redraw the scene with new coordinates

glutPostRedisplay();

glutTimerFunc(30, TimerFunction, 1);

}

//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 blue

glClearColor(0.0, 0.0, 1.0, 1.0);

return;

}//end of setuprc

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* floor functions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void loadFloor(float floorX[], float floorY[], float floorZ[], float nvectorX, float nvectorY, float nvectorZ) {

//p1

floorX[0] = -2; floorY[0] = -3; floorZ[0] = -2;

//p2

floorX[1] = 2; floorY[1] = -3; floorZ[1] = -2;

//p3

floorX[2] = 2; floorY[2] = -3; floorZ[2] = 2;

//p4

floorX[3] = -2; floorY[3] = -3; floorZ[3] = 2;

//vector

nvectorX = 0.0;

nvectorY = 1.0;

nvectorZ = 0.0;

return;

}//end of loadFloor

void drawFloor(float floorX[], float floorY[], float floorZ[], float nvectorX, float nvectorY, float nvectorZ) {

glColor3f(1.0, 1.0, 1.0);

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glBegin(GL\_POLYGON);

glNormal3f(nvectorX, nvectorY, nvectorZ);

/\* clockwise

glVertex3f(floorX[0], floorY[0], floorZ[0]);

glVertex3f(floorX[1], floorY[1], floorZ[1]);

glVertex3f(floorX[2], floorY[2], floorZ[2]);

glVertex3f(floorX[3], floorY[3], floorZ[3]);

\*/

glVertex3f(floorX[0], floorY[0], floorZ[0]);

glVertex3f(floorX[3], floorY[3], floorZ[3]);

glVertex3f(floorX[2], floorY[2], floorZ[2]);

glVertex3f(floorX[1], floorY[1], floorZ[1]);

glEnd();

glFlush();

}//end of drawFloor

void floorModel()

{

//sets the modelviel matrix for the floor

cout << "in floorModel" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(floorDX, floorDY, floorDZ);

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

glRotatef(floorTheta, 0.0, 0.0, 1.0);

glRotatef(floorTheta2, 0.0, 1.0, 0.0);

glRotatef(-30.0, 1.0, 0.0, 0.0);

return;

}//end of bodyModel

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

//function bodyModel

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);

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);

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);

return;

}//end of leg2Model

output

A picture containing graphical user interface

Description automatically generatedGraphical user interface

Description automatically generatedA picture containing text

Description automatically generatedGraphical user interface, application

Description automatically generated