Nick Wright Problem 4 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 4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

The purpose of this program is to have a wireframe 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, bodyDX = 7.0, bodyDY = -3.0; //global values for the body

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

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

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

//POLYWOMAN GLOBAL VARIABLES --- She is starting on the left side of the screen

float rockTheta = 0, turnTheta = 180, PWbodyDX = -7.0, PWbodyDY = -3.0; //global values for the body

float PWmouthTheta = 0, mTurnTheta = 180, PWmouthDX = -7.0, PWmouthDY = -3.0; //global values for the mouth

float PWleg1Theta = 0, l1TurnTheta = 180, PWleg1DX = -7.0, PWleg1DY = -3.0; //global value for leg 1

float PWleg2Theta = 0, l2TurnTheta = 180, PWleg2DX = -7.0, PWleg2DY = -3.0; //global value for leg 2

float rockAccumulator = 0; //accumulates to break out of rock loop for polywoman (see case 5 and 6)

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[][7], float[][7], float[][7], float[2], float[2], float[2]);

void drawBody(float[][7], float[][7], float[][7], float[2], float[2], float[2]);

//mouth functions

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

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

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

//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

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYWOMAN LOADS AND DRAWS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//void PWloadBody(float[][15], float[][15], float[][15], float[2], float[2], float[2]);

//void PWdrawBody(float[][15], float[][15], float[][15], float[2], float[2], float[2]);

void PWloadBody(float[][7], float[][7], float[][7], float[2], float[2], float[2]);

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

void PWloadLeg1(float[], float[], float[]); //loads leg 1

void PWloadLeg2(float[], float[], float[]); //loads leg 2

void PWloadDetails(float PWdetailX[][9], float PWdetailY[][9], float PWdetailZ[][9]); //loads the hair and bows

void PWdrawDetails(float PWdetailX[][9], float PWdetailY[][9], float PWdetailZ[][9]); //draws the hair and details

//\*\*\*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)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYWOMAN MODELVIEW\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

void PWleg1Model(void); //sets the MODELVIEW MATRIX for leg 1 (rotation/translation matrix)

void PWleg2Model(void); //sets the MODELVIEW MATRIX for leg 2 (rotation/translation matrix)

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[] = "Wireframes by Nick Wright (Homework 4)";

/\*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);

//initialize the window size and position

glutInitWindowSize(560, 440);

glutInitWindowPosition(140, 20);

//Initialize background color in window to red

SetupRC();

// 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;

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

//pattern for polyman body

float x[2][7], y[2][7], z[2][7], eyex[2], eyey[2], eyez[2];

//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];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYWOMAN PATTERN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//pattern for polywoman body

float PWx[2][7], PWy[2][7], PWz[2][7], PWeyex[2], PWeyey[2], PWeyez[2];

//float PWx[2][15], PWy[2][15], PWz[2][15], PWeyex[2], PWeyey[2], PWeyez[2];

//pattern for polywoman mouth

float PWmX[2][3], PWmY[2][3], PWmZ[2][3];

//pattern for polywoman legs 1 and 2

float PWl1x[4], PWl1y[4], PWl1z[4];

float PWl2x[4], PWl2y[4], PWl2z[4];

//pattern for polywoman details

float PWdetailsX[2][9], PWdetailsY[2][9], PWdetailsZ[2][9];

//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(x, y, z, eyex, eyey, eyez);

loadMouth(mX, mY, mZ);

loadLeg1(l1x, l1y, l1z);

loadLeg2(l2x, l2y, l2z);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYWOMAN LOADS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PWloadBody(PWx, PWy, PWz, PWeyex, PWeyey, PWeyez);

PWloadMouth(PWmX, PWmY, PWmZ);

PWloadLeg1(PWl1x, PWl1y, PWl1z);

PWloadLeg2(PWl2x, PWl2y, PWl2z);

PWloadDetails(PWdetailsX, PWdetailsY, PWdetailsZ);

//clear the window with the background color

glClear(GL\_COLOR\_BUFFER\_BIT);

//set the current drawing color to white

glColor3f(1.0, 1.0, 1.0);

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

//glFlush being performed after each draw

bodyModel(); //body modelview matrix

drawBody(x, y, z, eyex, eyey, eyez);

glFlush();

leg1Model(); //leg 1 modelview matrix

drawLeg(l1x, l1y, l1z);

glFlush();

leg2Model(); //leg 2 modelview matrix

drawLeg(l2x, l2y, l2z);

glFlush();

bodyModel();

//mouthModel(); //mouth modelview matrix

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

if (bodyDX != 0)

{

drawMouthClosed(mX, mY, mZ);

glFlush();

}

else

{

drawMouthOpen(mX, mY, mZ);

glFlush();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYWOMAN TRANFORMATIONS AND DRAWS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//glFlush being performed after each draw

PWbodyModel(); //body modelview matrix

drawBody(PWx, PWy, PWz, PWeyex, PWeyey, PWeyez);

glFlush();

PWleg1Model(); //leg 1 modelview matrix

drawLeg(PWl1x, PWl1y, PWl1z);

glFlush();

PWleg2Model(); //leg 2 modelview matrix

drawLeg(PWl2x, PWl2y, PWl2z);

glFlush();

PWbodyModel();

PWdrawDetails(PWdetailsX, PWdetailsY, PWdetailsZ);

glFlush();

PWbodyModel();

//PWmouthModel(); //mouth modelview matrix

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

if (bodyDX != 0)

{

drawMouthClosed(PWmX, PWmY, PWmZ);

glFlush();

}

else

{

drawMouthOpen(PWmX, PWmY, PWmZ);

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;

}

//polywoman walks in from the left

PWbodyDX += 0.075;

PWmouthDX += 0.075;

PWleg1DX += 0.075;

PWleg2DX += 0.075;

if (PWleg1DY > -3) {

PWleg1DY -= 0.1; PWleg2DY += 0.1;

}

else {

PWleg1DY += 0.1; PWleg2DY -= 0.1;

}

//use body position to change frame

if (bodyDX <= 0)

{

//polyman polywoman

bodyDX = 0; PWbodyDX = -3.5;

mouthDX = 0; PWmouthDX = -3.5;

leg1DX = 0; PWleg1DX = -3.5;

leg2DX = 0; PWleg2DX = -3.5;

leg1DY = -3.0; PWleg1DY = -3.0;

leg2DY = -3.0; PWleg2DY = -3.0;

frame = 2;

}

break;

case 2:

//frame 2 polyman opens his mouth and jumps into the air (y = 5)

//body parameters

bodyDY += 0.2;

//leg1 parameters

leg1DY += 0.2;

//leg2 parameters

leg2DY += 0.2;

//mouth parameters

mouthDY += 0.2;

if (bodyDY > 5.0)

{

bodyDY = 5.0;

leg1DY = 5.0;

leg2DY = 5.0;

mouthDY = 5.0;

frame = 3;

}

break;

case 3:

//frame 3 polyman rotates 360 degrees

//using negtaive theta value to make him do a backflip

//body parameters

bodyTheta -= 5.0;

//leg1 parameters

leg1Theta -= 5.0;

//leg2 parameters

leg2Theta -= 5.0;

//mouth parameters

mouthTheta -= 5.0;

if (bodyTheta <= -360.0)

{

frame = 4;

bodyTheta = 0.0;

leg1Theta = 0.0;

leg2Theta = 0.0;

mouthTheta = 0.0;

}

break;

case 4:

//frame 4 polyman lands back down on the ground (y = -3.0) polyman also closes his mouth

//body parameters

bodyDY -= 0.2;

//leg1 parameters

leg1DY -= 0.2;

//leg2 parameters

leg2DY -= 0.2;

//mouth parameters

mouthDY -= 0.2;

if (bodyDY <= -3.0)

{

bodyDY = -3.0;

mouthDY = -3.0;

leg1DY = -3.0;

leg2DY = -3.0;

frame = 5;

}

break;

case 5:

//polywoman rocks backwards

rockTheta += 5.0;

PWmouthTheta += 5.0;

PWleg1Theta += 5.0;

PWleg2Theta += 5.0;

rockAccumulator += 5.0; //using this as an accumulator so that polywoman only rocks a few times

if (rockAccumulator == 90)

{

rockTheta = 0.0;

PWmouthTheta = 0.0;

PWleg1Theta = 0.0;

PWleg2Theta = 0.0;

frame = 7;

break;

}

else if (rockTheta == 30)

{

frame = 6;

}

break;

case 6:

//polywoman rocks forwards

rockTheta -= 5.0;

PWmouthTheta -= 5.0;

PWleg1Theta -= 5.0;

PWleg2Theta -= 5.0;

if (rockTheta == -30)

{

frame = 5;

}

break;

case 7:

//polywoman rotates so that they may both walk off stage together

turnTheta -= 5.0;

mTurnTheta -= 5.0;

l1TurnTheta -= 5.0;

l2TurnTheta -= 5.0;

if (turnTheta <= 0)

{

turnTheta = 0;

mTurnTheta = 0;

l1TurnTheta = 0;

l2TurnTheta = 0;

frame = 8;

}

break;

case 8:

//frame 5 polyman walks off of the stage to the left

//polyman polywoman

bodyDX -= 0.15; PWbodyDX -= 0.15; //body parameters

mouthDX -= 0.15; PWmouthDX -= 0.15; //mouth parameters

leg1DX -= 0.15; PWleg1DX -= 0.15; //leg1 parameters

leg2DX -= 0.15; PWleg2DX -= 0.15; //leg2 parameters

//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;

}

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

if (PWleg1DY > -3) {

PWleg1DY -= 0.1; PWleg2DY += 0.1;

}

else {

PWleg1DY += 0.1; PWleg2DY -= 0.1;

}

if (bodyDX <= -6.0)

{

//polyman polywoman

bodyDX = -6.5; PWbodyDX = -6.5;

mouthDX = -6.5; PWmouthDX = -6.5;

leg1DX = -6.5; PWleg1DX = -6.5;

leg2DX = -6.5; PWleg2DX = -6.5;

leg1DY = -3.0; PWleg1DY = -3.0;

leg2DY = -3.0; PWleg2DY = -3.0;

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

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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 drawMouthClosed(float mX[][3], float mY[][3], float mZ[][3])

{

//this function will draw the mouth closed

//draw front closed mouth

//setting color to cyan

glColor3f(0.0, 1.0, 1.0);

glBegin(GL\_LINE\_STRIP);

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

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

glEnd();

glFlush();

//draw back closed mouth

//setting color to blue

glColor3f(0.0, 0.0, 1.0);

glBegin(GL\_LINE\_STRIP);

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

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

glEnd();

glFlush();

//connect front and back

//setting color to white

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_LINE\_STRIP);

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]);

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

glEnd();

glFlush();

return;

}//end of drawMouthClosed

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

{

//this function will draw the mouth open

//draw front open mouth

//setting color to cyan

glColor3f(0.0, 1.0, 1.0);

glBegin(GL\_LINE\_STRIP);

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();

//draw back open mouth

//setting color to blue

glColor3f(0.0, 1.0, 1.0);

glBegin(GL\_LINE\_STRIP);

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();

//connect front and back

//setting color to white

glColor3f(1.0, 1.0, 1.0);

//top mouth

glBegin(GL\_LINE\_STRIP);

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

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

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

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

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

glEnd();

glFlush();

//bottom mouth

glBegin(GL\_LINE\_STRIP);

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

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

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

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

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

glEnd();

glFlush();

return;

}

void loadBody(float x[][7], float y[][7], float z[][7], float eyex[2], float eyey[2], float eyez[2]) {

//load the front face

x[0][0] = (-9.0 / 8); y[0][0] = (0); z[0][0] = (1.0 / 2);

x[0][1] = (-5.0 / 8); y[0][1] = (3.0 / 4); z[0][1] = (1.0 / 2);

x[0][2] = (5.0 / 8); y[0][2] = (3.0 / 4); z[0][2] = (1.0 / 2);

x[0][3] = (9.0 / 8); y[0][3] = (0); z[0][3] = (1.0 / 2);

x[0][4] = (5.0 / 8); y[0][4] = (-3.0 / 4); z[0][4] = (1.0 / 2);

x[0][5] = (-5.0 / 8); y[0][5] = (-3.0 / 4); z[0][5] = (1.0 / 2);

//set color for font face (cyan)

x[0][6] = 0; y[0][6] = 1.0; z[0][6] = 1.0;

//load the front eye

eyex[0] = -1.0 / 2; eyey[0] = 1.0 / 2; eyez[0] = 1.0 / 2;

//load the back face

x[1][0] = (-9.0 / 8); y[1][0] = (0); z[1][0] = (-1.0 / 2);

x[1][1] = (-5.0 / 8); y[1][1] = (3.0 / 4); z[1][1] = (-1.0 / 2);

x[1][2] = (5.0 / 8); y[1][2] = (3.0 / 4); z[1][2] = (-1.0 / 2);

x[1][3] = (9.0 / 8); y[1][3] = (0); z[1][3] = (-1.0 / 2);

x[1][4] = (5.0 / 8); y[1][4] = (-3.0 / 4); z[1][4] = (-1.0 / 2);

x[1][5] = (-5.0 / 8); y[1][5] = (-3.0 / 4); z[1][5] = (-1.0 / 2);

//set color for font face (blue)

x[1][6] = 0; y[1][6] = 0; z[1][6] = 1.0;

//load the back eye

eyex[1] = -1.0 / 2; eyey[1] = 1.0 / 2; eyez[1] = -1.0 / 2;

return;

}//end of loadBody

void drawBody(float x[][7], float y[][7], float z[][7], float eyex[2], float eyey[2], float eyez[2])

{

int i;

glColor3f(1.0, 1.0, 1.0);

//draw upper head

glBegin(GL\_LINE\_STRIP);

//2,2',1',1,2

glVertex3f(x[0][1], y[0][1], z[0][1]);

glVertex3f(x[1][1], y[1][1], z[1][1]);

glVertex3f(x[1][0], y[1][0], z[1][0]);

glVertex3f(x[0][0], y[0][0], z[0][0]);

glVertex3f(x[0][1], y[0][1], z[0][1]);

glEnd();

glFlush();

//draw underside

glBegin(GL\_LINE\_STRIP);

//6,6',5',5

glVertex3f(x[0][5], y[0][5], z[0][5]);

glVertex3f(x[1][5], y[1][5], z[1][5]);

glVertex3f(x[1][4], y[1][4], z[1][4]);

glVertex3f(x[0][4], y[0][4], z[0][4]);

glVertex3f(x[0][5], y[0][5], z[0][5]);

glEnd();

glFlush();

//draw black lower

glBegin(GL\_LINE\_STRIP);

//4,4',5',5

glVertex3f(x[0][3], y[0][3], z[0][3]);

glVertex3f(x[1][3], y[1][3], z[1][3]);

glVertex3f(x[1][4], y[1][4], z[1][4]);

glVertex3f(x[0][4], y[0][4], z[0][4]);

glVertex3f(x[0][3], y[0][3], z[0][3]);

glEnd();

glFlush();

//draw back upper

glBegin(GL\_LINE\_STRIP);

//3,3',4',4

glVertex3f(x[0][2], y[0][2], z[0][2]);

glVertex3f(x[1][2], y[1][2], z[1][2]);

glVertex3f(x[1][3], y[1][3], z[1][3]);

glVertex3f(x[0][3], y[0][3], z[0][3]);

glVertex3f(x[0][2], y[0][2], z[0][2]);

glEnd();

glFlush();

//draw top

glBegin(GL\_LINE\_STRIP);

//3,3',2',2

glVertex3f(x[0][2], y[0][2], z[0][2]);

glVertex3f(x[1][2], y[1][2], z[1][2]);

glVertex3f(x[1][1], y[1][1], z[1][1]);

glVertex3f(x[0][1], y[0][1], z[0][1]);

glVertex3f(x[0][2], y[0][2], z[0][2]);

glEnd();

glFlush();

//draw the back face

glColor3f(x[1][6], y[1][6], z[1][6]);

glBegin(GL\_LINE\_STRIP);

for (i = 5; i >= 0; i--) //for (i = 0; i <= 5; i++)

{

glVertex3f(x[1][i], y[1][i], z[1][i]); //glVertex3f(x[1][0], y[1][0], z[1][0]);

}

glEnd();

glFlush();

//draw the back eye

//setting the point size to 3

glPointSize(3);

glBegin(GL\_POINTS);

glVertex3f(eyex[1], eyey[1], eyez[1]);

glEnd();

glFlush();

//draw the front face

glColor3f(x[0][6], y[0][6], z[0][6]);

glBegin(GL\_LINE\_STRIP);

for (i = 5; i >= 0; i--) //for (i = 0; i <= 5; i++)

{

glVertex3f(x[0][i], y[0][i], z[0][i]); //glVertex3f(x[0][0], y[0][0], z[0][0]);

}

glEnd();

glFlush();

//draw the front eye

//setting the point size to 3

glPointSize(3);

glBegin(GL\_POINTS);

glVertex3f(eyex[0], eyey[0], eyez[0]);

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, 0.0);

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

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, 0.0);

glRotatef(leg1Theta, 0.0, 0.0, 1.0);// note that the angle theta is in degrees, not radians

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, 0.0);

glRotatef(leg2Theta, 0.0, 0.0, 1.0);// note that the angle theta is in degrees, not radians

glRotatef(30.0, 1.0, 0.0, 0.0);

return;

}//end of leg2Model

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*POLYWOMAN FUNCTIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void PWloadBody(float PWx[][7], float PWy[][7], float PWz[][7], float PWeyex[2], float PWeyey[2], float PWeyez[2])

{

//load the front face

PWx[0][0] = (-9.0 / 8); PWy[0][0] = (0); PWz[0][0] = (1.0 / 2);

PWx[0][1] = (-5.0 / 8); PWy[0][1] = (3.0 / 4); PWz[0][1] = (1.0 / 2);

PWx[0][2] = (5.0 / 8); PWy[0][2] = (3.0 / 4); PWz[0][2] = (1.0 / 2);

PWx[0][3] = (9.0 / 8); PWy[0][3] = (0); PWz[0][3] = (1.0 / 2);

PWx[0][4] = (5.0 / 8); PWy[0][4] = (-3.0 / 4); PWz[0][4] = (1.0 / 2);

PWx[0][5] = (-5.0 / 8); PWy[0][5] = (-3.0 / 4); PWz[0][5] = (1.0 / 2);

//set color for front face (lilac)

PWx[0][6] = 2.0; PWy[0][6] = 0.5; PWz[0][6] = 1.0;

//load the front eye

PWeyex[0] = -1.0 / 2; PWeyey[0] = 1.0 / 2; PWeyez[0] = 1.0 / 2;

//load the back face

PWx[1][0] = (-9.0 / 8); PWy[1][0] = (0); PWz[1][0] = (-1.0 / 2);

PWx[1][1] = (-5.0 / 8); PWy[1][1] = (3.0 / 4); PWz[1][1] = (-1.0 / 2);

PWx[1][2] = (5.0 / 8); PWy[1][2] = (3.0 / 4); PWz[1][2] = (-1.0 / 2);

PWx[1][3] = (9.0 / 8); PWy[1][3] = (0); PWz[1][3] = (-1.0 / 2);

PWx[1][4] = (5.0 / 8); PWy[1][4] = (-3.0 / 4); PWz[1][4] = (-1.0 / 2);

PWx[1][5] = (-5.0 / 8); PWy[1][5] = (-3.0 / 4); PWz[1][5] = (-1.0 / 2);

//set color for back face (purple)

PWx[1][6] = 0.7; PWy[1][6] = 0; PWz[1][6] = 0.7;

//load the back eye

PWeyex[1] = -1.0 / 2; PWeyey[1] = 1.0 / 2; PWeyez[1] = -1.0 / 2;

//trying with no loads for the additional connecting sides

return;

} //end of polywoman load body

void PWloadMouth(float PWmX[][3], float PWmY[][3], float PWmZ[][3])

{

//load the front mouth

PWmX[0][0] = (-9.0 / 8); PWmY[0][0] = (0); PWmZ[0][0] = (1.0 / 2);

PWmX[0][1] = (-3.0 / 8); PWmY[0][1] = (0); PWmZ[0][1] = (1.0 / 2);

PWmX[0][2] = (-5.0 / 8); PWmY[0][2] = (-3.0 / 4); PWmZ[0][2] = (1.0 / 2);

//load the back mouth

PWmX[1][0] = (-9.0 / 8); PWmY[1][0] = (0); PWmZ[1][0] = (-1.0 / 2);

PWmX[1][1] = (-3.0 / 8); PWmY[1][1] = (0); PWmZ[1][1] = (-1.0 / 2);

PWmX[1][2] = (-5.0 / 8); PWmY[1][2] = (-3.0 / 4); PWmZ[1][2] = (-1.0 / 2);

}//end of loadMouth

void PWloadLeg1(float PWl1x[], float PWl1y[], float PWl1z[])

{

//this function will load leg 1

PWl1x[0] = -1.0 / 4; PWl1y[0] = -1.0 / 2; PWl1z[0] = 1.0 / 2;

PWl1x[1] = -1.0 / 4; PWl1y[1] = -1.0; PWl1z[1] = 1.0 / 2;

PWl1x[2] = -1.0 / 2; PWl1y[2] = -1.0; PWl1z[2] = 1.0 / 2;

//leg1 color (lilac)

PWl1x[3] = 2.0; PWl1y[3] = 0.5; PWl1z[3] = 1.0;

return;

}//end of loadLeg1

void PWloadLeg2(float PWl2x[], float PWl2y[], float PWl2z[])

{

//this function will load leg 2

PWl2x[0] = 1.0 / 4; PWl2y[0] = -1.0 / 2; PWl2z[0] = -1.0 / 2;

PWl2x[1] = 1.0 / 4; PWl2y[1] = -1.0; PWl2z[1] = -1.0 / 2;

PWl2x[2] = 0; PWl2y[2] = -1.0; PWl2z[2] = -1.0 / 2;

//leg2 color (purple)

PWl2x[3] = 0.7; PWl2y[3] = 0; PWl2z[3] = 0.7;

return;

}//end of loadLeg1

//function PWbodyModel

void PWbodyModel()

{

//sets the modelviel matrix for the body

cout << "in Poly Woman bodyModel" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(PWbodyDX, PWbodyDY, 0.0);

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

glRotatef(rockTheta, 0.0, 0.0, 1.0); //rotation for rocking movement

glRotatef(turnTheta, 0.0, 1.0, 0.0); //rotation for turning

glRotatef(30.0, 1.0, 0.0, 0.0);

return;

}//end of bodyModel

//function leg1Model

void PWleg1Model()

{

//sets the modelviel matrix for leg1

cout << "in leg1Model" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(PWleg1DX, PWleg1DY, 0.0);

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

glRotatef(PWleg1Theta, 0.0, 0.0, 1.0);

glRotatef(l1TurnTheta, 0.0, 1.0, 0.0);

glRotatef(30.0, 1.0, 0.0, 0.0);

return;

}//end of leg1Model

//function leg2Model

void PWleg2Model()

{

//sets the modelviel matrix for leg2

cout << "in leg2Model" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(PWleg2DX, PWleg2DY, 0.0);

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

glRotatef(PWleg2Theta, 0.0, 0.0, 1.0);

glRotatef(l2TurnTheta, 0.0, 1.0, 0.0);

glRotatef(30.0, 1.0, 0.0, 0.0);

return;

}//end of leg2Model

void PWloadDetails(float PWdetailX[][9], float PWdetailY[][9], float PWdetailZ[][9])

{

//polywoman front details

//color (lilac)

PWdetailX[0][0] = 2.0; PWdetailY[0][0] = 0.5; PWdetailZ[0][0] = 1.0;

//hair

PWdetailX[0][1] = -1.0 / 4; PWdetailY[0][1] = 3.0 / 4; PWdetailZ[0][1] = 1.0 / 2;

PWdetailX[0][2] = -1.0 / 4; PWdetailY[0][2] = 1.0 / 4; PWdetailZ[0][2] = 1.0 / 2;

PWdetailX[0][3] = 0.0; PWdetailY[0][3] = 3.0 / 4; PWdetailZ[0][3] = 1.0 / 2;

PWdetailX[0][4] = 0.0; PWdetailY[0][4] = 1.0 / 4; PWdetailZ[0][4] = 1.0 / 2;

PWdetailX[0][5] = 1.0 / 4; PWdetailY[0][5] = 3.0 / 4; PWdetailZ[0][5] = 1.0 / 2;

PWdetailX[0][6] = 1.0 / 4; PWdetailY[0][6] = 1.0 / 4; PWdetailZ[0][6] = 1.0 / 2;

PWdetailX[0][7] = 1.0 / 2; PWdetailY[0][7] = 3.0 / 4; PWdetailZ[0][7] = 1.0 / 2;

PWdetailX[0][8] = 1.0 / 2; PWdetailY[0][8] = 1.0 / 4; PWdetailZ[0][8] = 1.0 / 2;

//polywoman rear details

//color (purple)

PWdetailX[1][0] = 0.7; PWdetailY[1][0] = 0.0; PWdetailZ[1][0] = 0.7;

//hair

PWdetailX[1][1] = -1.0 / 4; PWdetailY[1][1] = 3.0 / 4; PWdetailZ[1][1] = -1.0 / 2;

PWdetailX[1][2] = -1.0 / 4; PWdetailY[1][2] = 1.0 / 4; PWdetailZ[1][2] = -1.0 / 2;

PWdetailX[1][3] = 0.0; PWdetailY[1][3] = 3.0 / 4; PWdetailZ[1][3] = -1.0 / 2;

PWdetailX[1][4] = 0.0; PWdetailY[1][4] = 1.0 / 4; PWdetailZ[1][4] = -1.0 / 2;

PWdetailX[1][5] = 1.0 / 4; PWdetailY[1][5] = 3.0 / 4; PWdetailZ[1][5] = -1.0 / 2;

PWdetailX[1][6] = 1.0 / 4; PWdetailY[1][6] = 1.0 / 4; PWdetailZ[1][6] = -1.0 / 2;

PWdetailX[1][7] = 1.0 / 2; PWdetailY[1][7] = 3.0 / 4; PWdetailZ[1][7] = -1.0 / 2;

PWdetailX[1][8] = 1.0 / 2; PWdetailY[1][8] = 1.0 / 4; PWdetailZ[1][8] = -1.0 / 2;

}

void PWdrawDetails(float PWdetailX[][9], float PWdetailY[][9], float PWdetailZ[][9])

{

//draw connecting pieces across top of head

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

glBegin(GL\_LINES);

glVertex3f(PWdetailX[0][1], PWdetailY[0][1], PWdetailZ[0][1]);

glVertex3f(PWdetailX[1][1], PWdetailY[1][1], PWdetailZ[1][1]);

glEnd(); glFlush();

glBegin(GL\_LINES);

glVertex3f(PWdetailX[0][3], PWdetailY[0][3], PWdetailZ[0][3]);

glVertex3f(PWdetailX[1][3], PWdetailY[1][3], PWdetailZ[1][3]);

glEnd(); glFlush();

glBegin(GL\_LINES);

glVertex3f(PWdetailX[0][5], PWdetailY[0][1], PWdetailZ[0][1]);

glVertex3f(PWdetailX[1][5], PWdetailY[1][1], PWdetailZ[1][1]);

glEnd(); glFlush();

glBegin(GL\_LINES);

glVertex3f(PWdetailX[0][7], PWdetailY[0][7], PWdetailZ[0][7]);

glVertex3f(PWdetailX[1][7], PWdetailY[1][7], PWdetailZ[1][7]);

glEnd(); glFlush();

//draw back

glColor3f(PWdetailX[1][0], PWdetailY[1][0], PWdetailZ[1][0]);

glBegin(GL\_LINES);

glVertex3f(PWdetailX[1][1], PWdetailY[1][1], PWdetailZ[1][1]);

glVertex3f(PWdetailX[1][2], PWdetailY[1][2], PWdetailZ[1][2]);

glEnd(); glFlush();

glBegin(GL\_LINES);

glVertex3f(PWdetailX[1][3], PWdetailY[1][3], PWdetailZ[1][3]);

glVertex3f(PWdetailX[1][4], PWdetailY[1][4], PWdetailZ[1][4]);

glEnd(); glFlush();

glBegin(GL\_LINES);

glVertex3f(PWdetailX[1][5], PWdetailY[1][5], PWdetailZ[1][5]);

glVertex3f(PWdetailX[1][6], PWdetailY[1][6], PWdetailZ[1][6]);

glEnd(); glFlush();

glBegin(GL\_LINES);

glVertex3f(PWdetailX[1][7], PWdetailY[1][7], PWdetailZ[1][7]);

glVertex3f(PWdetailX[1][8], PWdetailY[1][8], PWdetailZ[1][8]);

glEnd(); glFlush();

//draw front

glColor3f(PWdetailX[0][0], PWdetailY[0][0], PWdetailZ[0][0]);

glBegin(GL\_LINES);

glVertex3f(PWdetailX[0][1], PWdetailY[0][1], PWdetailZ[0][1]);

glVertex3f(PWdetailX[0][2], PWdetailY[0][2], PWdetailZ[0][2]);

glEnd(); glFlush();

glBegin(GL\_LINES);

glVertex3f(PWdetailX[0][3], PWdetailY[0][3], PWdetailZ[0][3]);

glVertex3f(PWdetailX[0][4], PWdetailY[0][4], PWdetailZ[0][4]);

glEnd(); glFlush();

glBegin(GL\_LINES);

glVertex3f(PWdetailX[0][5], PWdetailY[0][5], PWdetailZ[0][5]);

glVertex3f(PWdetailX[0][6], PWdetailY[0][6], PWdetailZ[0][6]);

glEnd(); glFlush();

glBegin(GL\_LINES);

glVertex3f(PWdetailX[0][7], PWdetailY[0][7], PWdetailZ[0][7]);

glVertex3f(PWdetailX[0][8], PWdetailY[0][8], PWdetailZ[0][8]);

glEnd(); glFlush();

}

Output

