Nick Wright Graphics Final Project

Code

//Two Textures on a Trapezoid

#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>

#include "External Libraries\SOIL2\include\SOIL2.h"

using namespace std;

/\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Final Project \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

The purpose of this final project is to make an animated christmas card. The animeation starts with polywoman entering the stage on a biezier curve. Polyman

will then follow behind her, flying in the air in front of the moon. He will exit to the right of the screen, then re-appear, walking on the ground to the center of the stage.

He will rise and do a flip, polywoman will rock back in approval, and then they will exit the stage to the left, with polywoman retracing the biezier curve she entered on.

\*/

/\*-----------------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 = -17.0, bodyDY = 6.0, bodyDZ = -7.0; //global values for the body

float mouthTheta = 0, mouthDX = -17.0, mouthDY = 6.0, mouthDZ = -7.0; //global values for the mouth

float leg1Theta = 0, leg1DX = -17.0, leg1DY = 6.0, leg1DZ = -7.0; //global value for leg 1

float leg2Theta = 0, leg2DX = -17.0, leg2DY = 6.0, leg2DZ = -7.0; //global value for leg 2

float PMTurnTheta = 180, PMmTurn = 180, PMleg1Turn = 180, PMleg2Turn = 180;

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

float rockTheta = 0, turnTheta = 180, PWbodyDX = -7.0, PWbodyDY = -3.0, PWbodyDZ = -6.0; //global values for the body

float PWmouthTheta = 0, mTurnTheta = 180, PWmouthDX = -7.0, PWmouthDY = -3.0, PWmouthDZ = -6.0; //global values for the mouth

float PWleg1Theta = 0, l1TurnTheta = 180, PWleg1DX = -7.0, PWleg1DY = -3.0, PWleg1DZ = -6.0; //global value for leg 1

float PWleg2Theta = 0, l2TurnTheta = 180, PWleg2DX = -7.0, PWleg2DY = -3.0, PWleg2DZ = -6.0; //global value for leg 2

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

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* christmas items \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//christmas tree global variables (tree is 2d)

float treeDX = 0, treeDY = 0, treeDZ = 0.0, treeSX = 1, treeSY = 1, treeSZ = 1;

//present1 global variables (present1 is 2d)

float present1DX = 0, present1DY = 0, present1DZ = 0.0, p1SX = 1, p1SY = 1, p1SZ = 1;

//present2 global variables (present2 is 2d)

float present2DX = 0, present2DY = 0, present2DZ = 0.0, p2SX = 1, p2SY = 1, p2SZ = 1;

//moon global variables

float moonDX = 0, moonDY = 0, moonDZ = 7.0, moonSX = 0, moonSY = 0, moonSZ = 0;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* biezier curve \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

float xctrl[4], yctrl[4], 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

int nocolors = 1;

// This is a switch to allow present2 to be colored panels or white panels.

// If the panels are white, the textures show up better.

int frame = 1;

void init(void);

void RenderScene(void);

//\*\*\*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], float[][3], float[][3], float[][3]);

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

float[2], float[2], float[2], float[][3], float[][3], float[][3], 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

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

void PWloadBody(float[][4], float[][4], float[][4], float[][4], float[][4], float[][4],

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

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)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* christmas items \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//tree load, draw and matrix

void loadTree(float leafX[9], float leafY[9], float leafZ[9], float trunkX[4], float trunkY[4], float trunkZ[4]);

void drawTree(float leafX[9], float leafY[9], float leafZ[9], float trunkX[4], float trunkY[4], float trunkZ[4]);

void treeModel();

//present1 load, draw and matrix

void loadPresent1(float[4], float[4], float[4]);

void drawPresent1(float[4], float[4], float[4]);

void present1Model();

//present2 load, draw and matrix

void loadPresent2(float[][4], float[][4], float[][4], float[][3], float[][3], float[], float[], float[]);

void drawPresent2(float[][4], float[][4], float[][4], float[][3], float[][3], float[], float[], float[]);

void present2Model();

//moon load, draw and matrix

void loadMoon(float[][8], float[][8], float[][8]);

void drawMoon(float[][8], float[][8], float[][8]);

void moonModel();

// the next two draws the background

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

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

void drawlightsource(float[]); // draws the position of the light source

void settrans2(void); // sets the translation matrix for the square

// transformation matrix for desired

// scale, rotation,new pos

// performs the transformation on the icon

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

GLuint textures[4]; // This array will hold the textures for the

// OpenGL Texture objects. The call to SOIL

// creates these texture objects.

//Main Program

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

{//set up window title

char header[] = "Final Project 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);

// clear the screen and set the color to transparent

SetupRC();

//Initialize window size and position

glutInitWindowSize(560, 440);

glutInitWindowPosition(140, 20);

// Open and Label Window

glutCreateWindow(header);

// start this earlier to help clean up cube

glEnable(GL\_DEPTH\_TEST);

//Now create texturez fo place on each side of the cube objects

//

textures[0] = SOIL\_load\_OGL\_texture("present.jpg", SOIL\_LOAD\_AUTO,

SOIL\_CREATE\_NEW\_ID, SOIL\_FLAG\_POWER\_OF\_TWO | SOIL\_FLAG\_INVERT\_Y);

if (!textures[0])

{

printf("soil failed to load present.jpg texture\n");

exit(0);

}

textures[1] = SOIL\_load\_OGL\_texture("mcPresent.png", SOIL\_LOAD\_AUTO,

SOIL\_CREATE\_NEW\_ID, SOIL\_FLAG\_POWER\_OF\_TWO | SOIL\_FLAG\_INVERT\_Y);

if (!textures[1])

{

printf("soil failed to load mcPresent.png texture\n");

exit(0);

}

textures[2] = SOIL\_load\_OGL\_texture("winterPhotoForFinal.jpg", SOIL\_LOAD\_AUTO,

SOIL\_CREATE\_NEW\_ID, SOIL\_FLAG\_POWER\_OF\_TWO | SOIL\_FLAG\_INVERT\_Y);

if (!textures[2])

{

printf("soil failed to load winterPhotoForFinal.jpg texture\n");

exit(0);

}

//Enable the texture state

glEnable(GL\_TEXTURE\_2D);

// clamp the image in the s direction and in the t direction

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP);

// Interpolate to the nearest pixel for color outside of image

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_NEAREST);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_NEAREST);

glutDisplayFunc(RenderScene);

glutTimerFunc(500, TimerFunction, 1);

//Now draw the scene

glutMainLoop();

return 0;

}

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

void RenderScene(void)

{

float 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] = -17.0; yctrl[0] = -15.0; //left end point

xctrl[1] = 15.0; yctrl[1] = -12.0; //point 1

xctrl[2] = -15.0; yctrl[2] = -9.0; //point 2

xctrl[3] = -3.5; yctrl[3] = -7.0; //right end point

int ncontrolpts = 4, i; // <----- change the number of control points based on how many you have (was 6)

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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], cHatX[4][3], cHatY[4][3], cHatZ[4][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];

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

//pattern for polywoman body

float PWupperX[2][4], PWupperY[2][4], PWupperZ[2][4], PWlowerX[2][4],

PWlowerY[2][4], PWlowerZ[2][4], PWeyeX[2], PWeyeY[2], PWeyeZ[2], PWnvector[8][3], PWcHatX[4][3], PWcHatY[4][3], PWcHatZ[4][3]; //polywomanman pattern;

//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];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* christmas object patterns \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//tree pattern

float leafX[9], leafY[9], leafZ[9], trunkX[4], trunkY[4], trunkZ[9];

//present1 pattern

float pointX[4], pointY[4], pointZ[4];

//present2 pattern

float x[6][4], y[6][4], z[6][4], fcolor[6][3], nvector1[6][3], lx[2], ly[2], lz[2];

//moon pattern

float moonX[2][8], moonY[2][8], moonZ[2][8];

float xc[4], yc[4], zc[4];

// \*\*\*\* these variables set up light parameters \*\*\*\*\*\*\*\*\*\*\*\*\*\*

float ambientlight[] = { 1.0,1.0,1.0,1.0 }; //strong white ambient light

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

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

float lightpos[] = { 0.0,4.0,4.0,1.0 }; //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[] = { 0,1.0,-5.0 }; //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(-15.0, 15.0, -15.0, 15.0, 10.0, -10.0);

//load the icons untransformed

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

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

loadMouth(mX, mY, mZ);

loadLeg1(l1x, l1y, l1z);

loadLeg2(l2x, l2y, l2z);

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

PWloadBody(PWupperX, PWupperY, PWupperZ, PWlowerX, PWlowerY, PWlowerZ, PWeyeX, PWeyeY, PWeyeZ, PWnvector, PWcHatX, PWcHatY, PWcHatZ);

PWloadMouth(PWmX, PWmY, PWmZ);

PWloadLeg1(PWl1x, PWl1y, PWl1z);

PWloadLeg2(PWl2x, PWl2y, PWl2z);

PWloadDetails(PWdetailsX, PWdetailsY, PWdetailsZ);

//christmas loads

loadTree(leafX, leafY, leafZ, trunkX, trunkY, trunkZ);

loadPresent1(pointX, pointY, pointZ);

loadPresent2(x, y, z, fcolor, nvector1, lx, ly, lz);

loadMoon(moonX, moonY, moonZ);

glEnable(GL\_DEPTH\_TEST);

/\*\*/

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 LOAD A NEW IDENTITY IDENTITY MATRIX. IF YOU DO NOT DO

//THIS AND YOU MOVE THE ICON LATER. THE LIGHT WILL FOLLOW THE ICON.

//ALSO NOTE THAT THE COORDINATE SYSTEM FOR. POSITIVE X IS TO THE Right,

//POSITIVE Y IS UP AND POSITIVE Z IS TOWARD THE VIEWER OUT OF THE SCREEN

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

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// 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, 40.0);

//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);

//enable lighting

glEnable(GL\_LIGHTING);

//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);

LoadCurtain(xc, yc, zc);

DrawCurtain(xc, yc, zc);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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, cHatX, cHatY, cHatZ);

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();

}

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

//glFlush being performed after each draw

PWbodyModel(); //body modelview matrix

drawBody(PWupperX, PWupperY, PWupperZ, PWlowerX, PWlowerY, PWlowerZ, PWeyeX, PWeyeY, PWeyeZ, nvector, PWcHatX, PWcHatY, PWcHatZ);

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();

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

if (bodyDX != 0)

{

drawMouth(PWmX, PWmY, PWmZ);

glFlush();

}

glFlush();

//christmas draws

treeModel();

drawTree(leafX, leafY, leafZ, trunkX, trunkY, trunkZ);

glFlush();

moonModel();

drawMoon(moonX, moonY, moonZ);

glFlush();

present1Model();

drawPresent1(pointX, pointY, pointZ);

glFlush();

present2Model();

drawPresent2(x, y, z, fcolor, nvector1, lx, ly, lz);

glFlush();

//now draw the light source

drawlightsource(lightpos);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

/\*

// 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, 3, xctrl), calcbiezu(Uval, 3, yctrl));

Uval += 0.05;

}

glFlush();

glEnd();

\*/

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], float cHatX[][3], float cHatY[][3], float cHatZ[][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;

//loading christmas hat

//front

cHatX[0][0] = 0; cHatY[0][0] = 2.0; cHatZ[0][0] = 0;

cHatX[0][1] = 0.5; cHatY[0][1] = 0.5; cHatZ[0][1] = 0.5;

cHatX[0][2] = -0.5; cHatY[0][2] = 0.5; cHatZ[0][2] = 0.5;

//right side

cHatX[1][0] = 0; cHatY[1][0] = 2.0; cHatZ[1][0] = 0;

cHatX[1][1] = 0.5; cHatY[1][1] = 0.5; cHatZ[1][1] = -0.5;

cHatX[1][2] = 0.5; cHatY[1][2] = 0.5; cHatZ[1][2] = 0.5;

//backside

cHatX[2][0] = 0; cHatY[2][0] = 2.0; cHatZ[2][0] = 0;

cHatX[2][1] = -0.5; cHatY[2][1] = 0.5; cHatZ[2][1] = -0.5;

cHatX[2][2] = 0.5; cHatY[2][2] = 0.5; cHatZ[2][2] = -0.5;

//left side

cHatX[3][0] = 0; cHatY[3][0] = 2.0; cHatZ[3][0] = 0;

cHatX[3][1] = -0.5; cHatY[3][1] = 0.5; cHatZ[3][1] = 0.5;

cHatX[3][2] = -0.5; cHatY[3][2] = 0.5; cHatZ[3][2] = -0.5;

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], float cHatX[][3], float cHatY[][3], float cHatZ[][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();

//drawing hat

int hatSide;

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

{

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_POLYGON);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* uncomment this line once you have the nvectors!

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

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

glVertex3f(cHatX[hatSide][i], cHatY[hatSide][i], cHatZ[hatSide][i]); //not sure that this loop is correct??

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);

glRotatef(PMTurnTheta, 0.0, 1.0, 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, leg1DZ);

glRotatef(PMleg1Turn, 0.0, 1.0, 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, leg2DZ);

glRotatef(PMleg2Turn, 0.0, 1.0, 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 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 PWnvector[][3], float PWcHatX[][3], float PWcHatY[][3], float PWcHatZ[][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;

//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;

//loading coords for the other sides

//top

PWnvector[2][0] = 0; PWnvector[2][1] = 1; PWnvector[2][2] = 0;

//bottom

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

//upper back

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

//lower back

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

//upper front

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

//lower front

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

//loading christmas hat

//front

PWcHatX[0][0] = 0; PWcHatY[0][0] = 2; PWcHatZ[0][0] = 0;

PWcHatX[0][1] = 0.5; PWcHatY[0][1] = 0.5; PWcHatZ[0][1] = 0.5;

PWcHatX[0][2] = -0.5; PWcHatY[0][2] = 0.5; PWcHatZ[0][2] = 0.5;

//right side

PWcHatX[1][0] = 0; PWcHatY[1][0] = 2; PWcHatZ[1][0] = 0;

PWcHatX[1][1] = 0.5; PWcHatY[1][1] = 0.5; PWcHatZ[1][1] = -0.5;

PWcHatX[1][2] = 0.5; PWcHatY[1][2] = 0.5; PWcHatZ[1][2] = 0.5;

//backside

PWcHatX[2][0] = 0; PWcHatY[2][0] = 2; PWcHatZ[2][0] = 0;

PWcHatX[2][1] = -0.5; PWcHatY[2][1] = 0.5; PWcHatZ[2][1] = -0.5;

PWcHatX[2][2] = 0.5; PWcHatY[2][2] = 0.5; PWcHatZ[2][2] = -0.5;

//left side

PWcHatX[3][0] = 0; PWcHatY[3][0] = 2; PWcHatZ[3][0] = 0;

PWcHatX[3][1] = -0.5; PWcHatY[3][1] = 0.5; PWcHatZ[3][1] = 0.5;

PWcHatX[3][2] = -0.5; PWcHatY[3][2] = 0.5; PWcHatZ[3][2] = -0.5;

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();

PWbodyDX = calcbiezu(uval, 3, xctrl);

PWbodyDY = calcbiezu(uval, 3, yctrl);

glTranslatef(PWbodyDX, PWbodyDY, PWbodyDZ);

//glTranslatef(PWbodyDX, PWbodyDY, PWbodyDZ);

// 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();

PWleg1DX = calcbiezu(uval, 3, xctrl);

PWleg1DY = calcbiezu(uval, 3, yctrl);

glTranslatef(PWleg1DX, PWleg1DY, PWleg1DZ);

// 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();

PWleg2DX = calcbiezu(uval, 3, xctrl);

PWleg2DY = calcbiezu(uval, 3, yctrl);

glTranslatef(PWleg2DX, PWleg2DY, PWleg2DZ);

// 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();

//drawing some bows

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

glColor3f(2.0, 0.5, 1.0);

glPointSize(4);

glBegin(GL\_POINTS);

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

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

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

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

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

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

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

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

glEnd();

glFlush();

}

//Christmas Functions

//tree functions

void loadTree(float leafX[9], float leafY[9], float leafZ[9], float trunkX[4], float trunkY[4], float trunkZ[4]) {

//load leafs

//top triangle

leafX[0] = 1; leafY[0] = 1; leafZ[0] = 0.0;

leafX[1] = 0; leafY[1] = 2; leafZ[1] = 0.0;

leafX[2] = -1; leafY[2] = 1; leafZ[2] = 0.0;

//middle triangle

leafX[3] = 1; leafY[3] = 0; leafZ[3] = 0.0;

leafX[4] = 0; leafY[4] = 1; leafZ[4] = 0.0;

leafX[5] = -1; leafY[5] = 0; leafZ[5] = 0.0;

//bottom triangle

leafX[6] = 1; leafY[6] = -1; leafZ[6] = 0.0;

leafX[7] = 0; leafY[7] = 0; leafZ[7] = 0.0;

leafX[8] = -1; leafY[8] = -1; leafZ[8] = 0.0;

//tree trunk

trunkX[0] = 0.5; trunkY[0] = -1; trunkZ[0] = 0.0;

trunkX[1] = -0.5; trunkY[1] = -1; trunkZ[1] = 0.0;

trunkX[2] = -0.5; trunkY[2] = -2; trunkZ[2] = 0.0;

trunkX[3] = 0.5; trunkY[3] = -2; trunkZ[3] = 0.0;

}

void drawTree(float leafX[9], float leafY[9], float leafZ[9], float trunkX[4], float trunkY[4], float trunkZ[4]) {

//start with the leaves

glColor3f(0.0, 1.0, 0); //setting color to green

//top leaves

glFrontFace(GL\_CCW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(leafX[0], leafY[0], leafZ[0]);

glVertex3f(leafX[1], leafY[1], leafZ[1]);

glVertex3f(leafX[2], leafY[2], leafZ[2]);

glEnd();

glFlush();

//middle leaves

glBegin(GL\_POLYGON);

glVertex3f(leafX[3], leafY[3], leafZ[3]);

glVertex3f(leafX[4], leafY[4], leafZ[4]);

glVertex3f(leafX[5], leafY[5], leafZ[5]);

glEnd();

glFlush();

//bottom leaves

glBegin(GL\_POLYGON);

glVertex3f(leafX[6], leafY[6], leafZ[6]);

glVertex3f(leafX[7], leafY[7], leafZ[7]);

glVertex3f(leafX[8], leafY[8], leafZ[8]);

glEnd();

glFlush();

//tree trunk

glColor3f(0.64, 0.16, 0.16); //setting color to brown

glBegin(GL\_POLYGON);

glVertex3f(trunkX[0], trunkY[0], trunkZ[0]);

glVertex3f(trunkX[1], trunkY[1], trunkZ[1]);

glVertex3f(trunkX[2], trunkY[2], trunkZ[2]);

glVertex3f(trunkX[3], trunkY[3], trunkZ[3]);

glEnd();

glFlush();

}

void treeModel() {

//sets the modelviel matrix for the body

cout << "in tree modelview matrix" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(treeDX, treeDY, treeDZ);

glScalef(treeSX, treeSY, treeSZ);

return;

}

//tree functions

void loadPresent1(float pointX[], float pointY[], float pointZ[]) {

pointX[0] = 0.5; pointY[0] = 0.5; pointZ[0] = 0;

pointX[1] = -0.5; pointY[1] = 0.5; pointZ[1] = 0;

pointX[2] = -0.5; pointY[2] = -0.5; pointZ[2] = 0;

pointX[3] = 0.5; pointY[3] = -0.5; pointZ[3] = 0;

}

void drawPresent1(float pointX[], float pointY[], float pointZ[]) {

// first bind the texture to the polygon

glBindTexture(GL\_TEXTURE\_2D, textures[0]);

// first load the s and t arrays to clamp the texture to the polygon

float s[4], t[4];

/\*

s[0] = 0.0; t[0] = 1.0;// bind the upper left of texture to upper left corner of poly

s[1] = 0.0; t[1] = 0.0;//bind the lower left of texture to lower left corner of poly

s[2] = 1.0; t[2] = 0.0;//bind the lower right of texture to lower right corner of poly

s[3] = 1.0; t[3] = 1.0;//bind the upper right of texture to upper right of corner of poly

\*/

s[0] = 1.0; t[0] = 0.0;// bind the upper left of texture to upper left corner of poly

s[1] = 0.0; t[1] = 0.0;//bind the lower left of texture to lower left corner of poly

s[2] = 0.0; t[2] = 1.0;//bind the lower right of texture to lower right corner of poly

s[3] = 1.0; t[3] = 1.0;//bind the upper right of texture to upper right of corner of poly

// now set up the polygon for curtains

glBegin(GL\_POLYGON);

//set normal to curtains for lights

glNormal3f(0.0, 0.0, 1.0);

// now draw curtain polygon with curtain texture bound to it.

for (int i = 0; i <= 3; i++)

{//Place the texture coordinate on the surface of polygon for curtain clamp

// it to this vertex corner.

glTexCoord2f(s[i], t[i]);

glVertex3f(pointX[i], pointY[i], pointZ[i]);

}//

glFlush();

glEnd();

return;

/\*

glColor3f(0.0, 1.0, 0); //setting color to green

glFrontFace(GL\_CCW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(pointX[0], pointY[0], pointZ[0]);

glVertex3f(pointX[1], pointY[1], pointZ[1]);

glVertex3f(pointX[2], pointY[2], pointZ[2]);

glVertex3f(pointX[3], pointY[3], pointZ[3]);

glEnd();

glFlush();

\*/

}

void present1Model() {

//sets the modelviel matrix for the body

cout << "in tree modelview matrix" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(present1DX, present1DY, present1DZ);

glScalef(p1SX, p1SY, p1SZ);

glRotatef(180, 0, 0, 1);

return;

}

//present2 functions

void loadPresent2(float x[][4], float y[][4], float z[][4], float fcolor[][3],

float nvector1[][3], float xl[], float yl[], float zl[])

{ /\* this procedure loads a trapzoidal icon \*/

/\* load front face\*/

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

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

x[0][2] = 1.0; y[0][2] = -1.0; z[0][2] = 1.0;

x[0][3] = 1.0; y[0][3] = 1.0; z[0][3] = 1.0;

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

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

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

nvector1[0][0] = 0.0; nvector1[0][1] = 0.0; nvector1[0][2] = 1.0;

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

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

x[1][1] = 1.0; y[1][1] = -1.0; z[1][1] = 1.0;

x[1][2] = 1.0; y[1][2] = -1.0; z[1][2] = -1.0;

x[1][3] = 1.0; y[1][3] = 1.0; z[1][3] = -1.0;

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

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

// load the normal to this face pos x axis

nvector1[1][0] = 1.0; nvector1[1][1] = 0.0; nvector1[1][2] = 0.0;

/\* load the back side face \*/

x[2][0] = 1.0; y[2][0] = 1.0; z[2][0] = -1.0;

x[2][1] = 1.0; y[2][1] = -1.0; z[2][1] = -1.0;

x[2][2] = -1.0; y[2][2] = -1.0; z[2][2] = -1.0;

x[2][3] = -1.0; y[2][3] = 1.0; z[2][3] = -1.0;

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

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

// load the normal to this face neg z axis

nvector1[2][0] = 0.0; nvector1[2][1] = 0.0; nvector1[2][2] = -1.0;

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

x[3][0] = -1.0; y[3][0] = 1.0; z[3][0] = 1.0;

x[3][1] = -1.0; y[3][1] = 1.0; z[3][1] = -1.0;

x[3][2] = -1.0; y[3][2] = -1.0; z[3][2] = -1.0;

x[3][3] = -1.0; y[3][3] = -1.0; z[3][3] = 1.0;

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

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

// load the normal to this face neg x axis

nvector1[3][0] = -1.0; nvector1[3][1] = 0.0; nvector1[3][2] = 0.0;

/\*loat the top side\*/

x[4][0] = 1.0; y[4][0] = 1.0; z[4][0] = 1.0;

x[4][1] = 1.0; y[4][1] = 1.0; z[4][1] = -1.0;

x[4][2] = -1.0; y[4][2] = 1.0; z[4][2] = -1.0;

x[4][3] = -1.0; y[4][3] = 1.0; z[4][3] = 1.0;

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

fcolor[4][0] = 0.5; fcolor[4][1] = 0.5; fcolor[4][2] = 0.0;

// load the normal to this face pos y axis

nvector1[4][0] = 0.0; nvector1[4][1] = 1.0; nvector1[4][2] = 0.0;

/\*load the bottom side \*/

x[5][0] = 1.0; y[5][0] = -1.0; z[5][0] = 1.0;

x[5][1] = -1.0; y[5][1] = -1.0; z[5][1] = 1.0;

x[5][2] = -1.0; y[5][2] = -1.0; z[5][2] = -1.0;

x[5][3] = 1.0; y[5][3] = -1.0; z[5][3] = -1.0;

/\* load the color on bottom yellow \*/

fcolor[5][0] = 0.0; fcolor[5][1] = 0.5; fcolor[5][2] = 0.5;

// load the normal to this face neg y axis

nvector1[5][0] = 0.0; nvector1[5][1] = -1.0; nvector1[5][2] = 0.0;

/\*load the line \*/

xl[0] = 0.0; yl[0] = 3.0; zl[0] = 0.0;

xl[1] = 0.0; yl[1] = -3.0; zl[1] = 0.0;

return;

} /\* end of load icon \*/

void drawPresent2(float x[][4], float y[][4], float z[][4], float fcolor[][3],

float nvector1[][3], float xl[], float yl[], float zl[])

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

//float s1[4]={0.0,0.0,1.0,1.0},t1[4]={0.0,1.0,1.0,0.0};

float s[6][4], t[6][4];

int i, face;

// load face 0 the red face by hand.

s[0][0] = 0.0; t[0][0] = 1.0;

s[0][1] = 0.0; t[0][1] = 0.0;

s[0][2] = 1.0; t[0][2] = 0.0;

s[0][3] = 1.0; t[0][3] = 1.0;

// now map every other texture on the 4 conrners of the figure.

//loading face 1

s[1][0] = 0.0; t[1][0] = 0.0;

s[1][1] = 0.0; t[1][1] = 1.0;

s[1][2] = 1.0; t[1][2] = 1.0;

s[1][3] = 1.0; t[1][3] = 0.0;

// loading face 2

s[2][0] = 0.0; t[2][0] = 0.0;

s[2][1] = 0.0; t[2][1] = 1.0;

s[2][2] = 1.0; t[2][2] = 1.0;

s[2][3] = 1.0; t[2][3] = 0.0;

//loading face 3

s[3][0] = 0.0; t[3][0] = 0.0;

s[3][1] = 0.0; t[3][1] = 1.0;

s[3][2] = 1.0; t[3][2] = 1.0;

s[3][3] = 1.0; t[3][3] = 0.0;

//loading face 4

s[4][0] = 0.0; t[4][0] = 0.0;

s[4][1] = 0.0; t[4][1] = 1.0;

s[4][2] = 1.0; t[4][2] = 1.0;

s[4][3] = 1.0; t[4][3] = 0.0;

//loading face 5

s[5][0] = 0.0; t[5][0] = 0.0;

s[5][1] = 0.0; t[5][1] = 1.0;

s[5][2] = 1.0; t[5][2] = 1.0;

s[5][3] = 1.0; t[5][3] = 0.0;

for (face = 0; face <= 5; face++)

{// render each face of the cube

// Decide which texture we want bound to this face.

if (face <= 3)glBindTexture(GL\_TEXTURE\_2D, textures[1]);

else glBindTexture(GL\_TEXTURE\_2D, textures[1]);

if (nocolors == 1) glColor3f(1.0, 1.0, 1.0);

else

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

glBegin(GL\_POLYGON);

glNormal3f(nvector1[face][0], nvector1[face][1], nvector1[face][2]);

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

{// Place the texture coordinate on the surface of the cube clamp

// it on this vertex corner. Note as we move around the

// s[i] varies from 0.0 to 1.0 and t[i] varies from 0.0 to 1.0

glTexCoord2f(s[face][i], t[face][i]);

glVertex3f(x[face][i], y[face][i], z[face][i]);

}

glEnd();

}//end of textured face build.

return;

}//end of present2

void present2Model(void)

//Sets the translation matrix for the cube

{

cout << "in settrans2" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

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

glTranslatef(present2DX, present2DY, present2DZ);

glRotatef(12.0, 1.0, 0.0, 0.0);

glRotatef(-12.0, 0.0, 1.0, 0.0);

glScalef(p2SX, p2SY, p2SZ);

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

//glRotatef(theta2, 1.0, 1.0, 1.0);

return;

}

//moon functions

void loadMoon(float moonX[][8], float moonY[][8], float moonZ[][8]) {

//load front

moonX[0][0] = 0.5; moonY[0][0] = 1.5; moonZ[0][0] = 0.5;

moonX[0][1] = -0.5; moonY[0][1] = 1.5; moonZ[0][1] = 0.5;

moonX[0][2] = -1.5; moonY[0][2] = 0.5; moonZ[0][2] = 0.5;

moonX[0][3] = -1.5; moonY[0][3] = -0.5; moonZ[0][3] = 0.5;

moonX[0][4] = -0.5; moonY[0][4] = -1.5; moonZ[0][4] = 0.5;

moonX[0][5] = 0.5; moonY[0][5] = -1.5; moonZ[0][5] = 0.5;

moonX[0][6] = 1.5; moonY[0][6] = -0.5; moonZ[0][6] = 0.5;

moonX[0][7] = 1.5; moonY[0][7] = 0.5; moonZ[0][7] = 0.5;

//load back

moonX[1][0] = 0.5; moonY[1][0] = 1.5; moonZ[1][0] = -0.5;

moonX[1][1] = -0.5; moonY[1][1] = 1.5; moonZ[1][1] = -0.5;

moonX[1][2] = -1.5; moonY[1][2] = 0.5; moonZ[1][2] = -0.5;

moonX[1][3] = -1.5; moonY[1][3] = -0.5; moonZ[1][3] = -0.5;

moonX[1][4] = -0.5; moonY[1][4] = -1.5; moonZ[1][4] = -0.5;

moonX[1][5] = 0.5; moonY[1][5] = -1.5; moonZ[1][5] = -0.5;

moonX[1][6] = 1.5; moonY[1][6] = -0.5; moonZ[1][6] = -0.5;

moonX[1][7] = 1.5; moonY[1][7] = 0.5; moonZ[1][7] = -0.5;

}

void drawMoon(float moonX[][8], float moonY[][8], float moonZ[][8]) {

//back face

glColor3f(0.0, 0, 0);

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glBegin(GL\_POLYGON);

glVertex3f(moonX[1][0], moonY[1][0], moonZ[1][0]);

glVertex3f(moonX[1][1], moonY[1][1], moonZ[1][1]);

glVertex3f(moonX[1][2], moonY[1][2], moonZ[1][2]);

glVertex3f(moonX[1][3], moonY[1][3], moonZ[1][3]);

glVertex3f(moonX[1][4], moonY[1][4], moonZ[1][4]);

glVertex3f(moonX[1][5], moonY[1][5], moonZ[1][5]);

glVertex3f(moonX[1][6], moonY[1][6], moonZ[1][6]);

glVertex3f(moonX[1][7], moonY[1][7], moonZ[1][7]);

glEnd();

glFlush();

glColor3f(1.0, 0.961, 0.769); //setting color to egg white

//front face

glFrontFace(GL\_CCW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

glVertex3f(moonX[0][0], moonY[0][0], moonZ[0][0]);

glVertex3f(moonX[0][1], moonY[0][1], moonZ[0][1]);

glVertex3f(moonX[0][2], moonY[0][2], moonZ[0][2]);

glVertex3f(moonX[0][3], moonY[0][3], moonZ[0][3]);

glVertex3f(moonX[0][4], moonY[0][4], moonZ[0][4]);

glVertex3f(moonX[0][5], moonY[0][5], moonZ[0][5]);

glVertex3f(moonX[0][6], moonY[0][6], moonZ[0][6]);

glVertex3f(moonX[0][7], moonY[0][7], moonZ[0][7]);

glEnd();

glFlush();

//drawing connecting pieces

//top

glColor3f(0.8, 0.761, 0.569); //setting color to egg white

glBegin(GL\_POLYGON);

glVertex3f(moonX[1][1], moonY[1][1], moonZ[1][1]);

glVertex3f(moonX[0][1], moonY[0][1], moonZ[0][1]);

glVertex3f(moonX[0][0], moonY[0][0], moonZ[0][0]);

glVertex3f(moonX[1][0], moonY[1][0], moonZ[1][0]);

glEnd();

glFlush();

//left top

glBegin(GL\_POLYGON);

glVertex3f(moonX[1][2], moonY[1][2], moonZ[1][2]);

glVertex3f(moonX[0][2], moonY[0][2], moonZ[0][2]);

glVertex3f(moonX[0][1], moonY[0][1], moonZ[0][1]);

glVertex3f(moonX[1][1], moonY[1][1], moonZ[1][1]);

glEnd();

glFlush();

//left side

glBegin(GL\_POLYGON);

glVertex3f(moonX[1][3], moonY[1][3], moonZ[1][3]);

glVertex3f(moonX[0][3], moonY[0][3], moonZ[0][3]);

glVertex3f(moonX[0][2], moonY[0][2], moonZ[0][2]);

glVertex3f(moonX[1][2], moonY[1][2], moonZ[1][2]);

glEnd();

glFlush();

//left bottom

glBegin(GL\_POLYGON);

glVertex3f(moonX[1][4], moonY[1][4], moonZ[1][4]);

glVertex3f(moonX[0][4], moonY[0][4], moonZ[0][4]);

glVertex3f(moonX[0][3], moonY[0][3], moonZ[0][3]);

glVertex3f(moonX[1][3], moonY[1][3], moonZ[1][3]);

glEnd();

glFlush();

//bottom

glBegin(GL\_POLYGON);

glVertex3f(moonX[1][5], moonY[1][5], moonZ[1][5]);

glVertex3f(moonX[0][5], moonY[0][5], moonZ[0][5]);

glVertex3f(moonX[0][4], moonY[0][4], moonZ[0][4]);

glVertex3f(moonX[1][4], moonY[1][4], moonZ[1][4]);

glEnd();

glFlush();

//bottom right

glBegin(GL\_POLYGON);

glVertex3f(moonX[1][6], moonY[1][6], moonZ[1][6]);

glVertex3f(moonX[0][6], moonY[0][6], moonZ[0][6]);

glVertex3f(moonX[0][5], moonY[0][5], moonZ[0][5]);

glVertex3f(moonX[1][5], moonY[1][5], moonZ[1][5]);

glEnd();

glFlush();

//right side

glBegin(GL\_POLYGON);

glVertex3f(moonX[1][7], moonY[1][7], moonZ[1][7]);

glVertex3f(moonX[0][7], moonY[0][7], moonZ[0][7]);

glVertex3f(moonX[0][6], moonY[0][6], moonZ[0][6]);

glVertex3f(moonX[1][6], moonY[1][6], moonZ[1][6]);

glEnd();

glFlush();

//right top

glBegin(GL\_POLYGON);

glVertex3f(moonX[1][0], moonY[1][0], moonZ[1][0]);

glVertex3f(moonX[0][0], moonY[0][0], moonZ[0][0]);

glVertex3f(moonX[0][7], moonY[0][7], moonZ[0][7]);

glVertex3f(moonX[1][7], moonY[1][7], moonZ[1][7]);

glEnd();

glFlush();

}

void moonModel() {

//sets the modelviel matrix for the body

cout << "in moon modelview matrix" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(moonDX, moonDY, moonDZ);

glRotatef(30.0, 1.0, 1.0, 0.0);

//glScalef(moonSX, moonSY, moonSZ);

return;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* biezier curve \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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 LoadCurtain(float xc[], float yc[], float zc[])

{

xc[0] = -14.0; yc[0] = 14.0; zc[0] = 9.0; //upper left corner

xc[1] = -14.0; yc[1] = -14.0; zc[1] = 9.0; //lower left corner

xc[2] = 14.0; yc[2] = -14.0; zc[2] = 9.0; //lower right corner

xc[3] = 14.0; yc[3] = 14.0; zc[3] = 9.0; //upper right corner

return;

}// end of LoadCurtain

void DrawCurtain(float xc[], float yc[], float zc[])

{// This will draw the curtain and apply the texture to it.

// first bind the texture to the polygon

glBindTexture(GL\_TEXTURE\_2D, textures[2]);

// first load the s and t arrays to clamp the texture to the polygon

float s[4], t[4];

s[0] = 0.0; t[0] = 1.0;// bind the upper left of texture to upper left corner of poly

s[1] = 0.0; t[1] = 0.0;//bind the lower left of texture to lower left corner of poly

s[2] = 1.0; t[2] = 0.0;//bind the lower right of texture to lower right corner of poly

s[3] = 1.0; t[3] = 1.0;//bind the upper right of texture to upper right of corner of poly

// now set up the polygon for curtains

glBegin(GL\_POLYGON);

//set normal to curtains for lights

glNormal3f(0.0, 0.0, 1.0);

// now draw curtain polygon with curtain texture bound to it.

for (int i = 0; i <= 3; i++)

{//Place the texture coordinate on the surface of polygon for curtain clamp

// it to this vertex corner.

glTexCoord2f(s[i], t[i]);

glVertex3f(xc[i], yc[i], zc[i]);

}//

glFlush();

glEnd();

return;

}//End of Draw Curtain

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function drawlightsource \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// this function moves the icon around ib the screen

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

void drawlightsource(float lightxyz[])

{// this function draws a light at the position held in the array lightxyz

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTranslatef(lightxyz[0], lightxyz[1], lightxyz[2]);

glColor3f(1.0, 1.0, 1.0);//white light at this position

glutSolidSphere(0.5, 10, 10);

return;

}//end of drawlightsource

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Function Timer\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Setup the timer function

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

void TimerFunction(int value)

// this call back function is call each 30 ms and

// changes the location,scale and rotation

{

//christmas items(they will not move)

treeDX = 0; treeDY = -2; treeDZ = 4.9;

treeSX = 2.5; treeSY = 2.5;

moonDX = -4.0; moonDY = 5.0; moonDZ = -3.0;

present1DX = -5; present1DY = -5.8; present1DZ = 4;

p1SX = 2.5; p1SY = 2.5;

present2DX = 5; present2DY = -6.3; present2DZ = 3;

switch (frame)

{

case 1:

//case 1, polywoman moves on a biezier curve

uval += 0.01;

if (uval >= 1.0) uval = 1.0;

if (PWbodyDX >= -3.5) {

turnTheta -= 5.0;

mTurnTheta -= 5.0;

l1TurnTheta -= 5.0;

l2TurnTheta -= 5.0;

}

if (PWbodyDX <= -5.5 && PWbodyDY >= -12) {

turnTheta += 8.0;

mTurnTheta += 8.0;

l1TurnTheta += 8.0;

l2TurnTheta += 8.0;

}

//break condition is when polywoman reaches her designated spot

if (PWbodyDX == -3.5 && PWbodyDY == -7) {

turnTheta = 180.0;

mTurnTheta = 180.0;

l1TurnTheta = 180.0;

l2TurnTheta = 180.0;

frame = 2;

//break;

}

break;

case 2:

//frame 2 polyman comes on the screen, flying in front of the moon, then re-exits the stage on the right

//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 > 6) {

leg1DY -= 0.1; leg2DY += 0.1;

}

else {

leg1DY += 0.1; leg2DY -= 0.1;

}

//use body position to change frame

if (bodyDX >= 16)

{

//polyman

bodyDX = 16;

bodyDY = -7;

mouthDX = 16;

mouthDY = -7;

leg1DX = 16;

leg2DX = 16;

leg1DY = -7.0;

leg2DY = -7.0;

PMTurnTheta = 0;

PMmTurn = 0;

PMleg1Turn = 0;

PMleg2Turn = 0;

frame = 3;

}

break;

case 3:

//frame 3 polyman comes back in from the left

//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 > -7) {

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 = -7.0;

leg2DY = -7.0;

frame = 4;

}

break;

case 4:

//frame 4 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 = 5;

}

break;

case 5:

//frame 5 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 = 6;

bodyTheta = 0.0;

leg1Theta = 0.0;

leg2Theta = 0.0;

mouthTheta = 0.0;

}

break;

case 6:

//frame 6 polyman lands back down on the ground (y = -7.0) 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 <= -7.0)

{

bodyDY = -7.0;

mouthDY = -7.0;

leg1DY = -7.0;

leg2DY = -7.0;

frame = 7;

}

break;

case 7:

//frame 7 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 = 9;

break;

}

else if (rockTheta == 30)

{

frame = 8;

}

break;

case 8:

//frame 8 polywoman rocks forwards

rockTheta -= 5.0;

PWmouthTheta -= 5.0;

PWleg1Theta -= 5.0;

PWleg2Theta -= 5.0;

if (rockTheta == -30)

{

frame = 7;

}

break;

case 9:

//frame 9 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 = 10;

}

break;

case 10:

//frame 10 polyman walks off of the stage to the left, polywoman follows the beizier curve again

uval -= 0.01;

if (uval >= 1.0) uval = 1.0;

if (PWbodyDX <= -3.5) {

turnTheta -= 5.0;

mTurnTheta -= 5.0;

l1TurnTheta -= 5.0;

l2TurnTheta -= 5.0;

}

if (PWbodyDX >= -5.5 && PWbodyDY <= -12) {

turnTheta += 8.0;

mTurnTheta += 8.0;

l1TurnTheta += 8.0;

l2TurnTheta += 8.0;

}

//polyman

bodyDX -= 0.15;

mouthDX -= 0.15;

leg1DX -= 0.15;

leg2DX -= 0.15;

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

if (leg1DY > -7) {

leg1DY -= 0.1; leg2DY += 0.1;

}

else {

leg1DY += 0.1; leg2DY -= 0.1;

}

if (bodyDX <= -16.0)

{

//polyman

bodyDX = -16;

mouthDX = -16;

leg1DX = -16;

leg2DX = -16;

leg1DY = -7.0;

leg2DY = -7.0;

break;

}

}

glutPostRedisplay();

glutTimerFunc(33, TimerFunction, 1);

}

Output

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence