#include<windows.h>

#include<GL/glut.h>

#include<stdlib.h>

#include<math.h>

#include<conio.h>

#include<stdio.h>

#include <iostream>

#include <iomanip>

#include <gl/glut.h>

using namespace std;

/\* This program demonstrates PolyMan Following a Biezier Curve. The program renders PolyMan in solid form

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

surfaces of Polyman are rendered by glBegin(GL\_POLYGON). All faces have been loaded clockwise.

The Biezier path is generated with 6 control points along the path from the Biezier function clacbiezu(float, int, float[])

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

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

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

float static theta = 0.0, theta2 = 5;;//global angular value for rotation

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

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

float xctrl[6], yctrl[6], uvalx = 0.0, uvaly = 10.0; // these are for Biezier Control points for the path for the of PolyMan.

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

int fact(int);//calclates factorial

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

void RenderScene(void);//this is a function to draw PolyMan in an opened window

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

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

void settrans3(void);//sets translation matrix of the Ski slope

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

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

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

void SetupRC(void);//sets up the clear color

void TimerFunction(int);//this call back function is call each 30 ms and changes the location,scale and rotation

// of PolyMan.

//Main Program

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

{//set up window title

char header[] = "PolyMan Skis by Chris Stewart";

glutInit(&argc, argv);

// Set up the display mode with a double buffer and a depth buffer and RGB colors

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

SetupRC();

//Initialize window size and position

glutInitWindowSize(540, 440);

glutInitWindowPosition(0, 0);

// Open and Label Window

glutCreateWindow(header);

glutDisplayFunc(RenderScene);

glutTimerFunc(500, TimerFunction, 1);

//Now draw the scene

glutMainLoop();

return 0;

}

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

void RenderScene(void)

{

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

float x1, y1, xdel = 0.25; //These are Biezier Control Points

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

// Set Up AThe Control Points

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

xctrl[1] = 6.0; yctrl[1] = 1.0;//point 1

xctrl[2] = 10.0; yctrl[2] = 6.0;//point 2

xctrl[3] = 4.0; yctrl[3] = 8.0;//point 3

xctrl[4] = 1.0; yctrl[4] = 6.0;//point 4

xctrl[5] = 10.0; yctrl[5] = 0.0;//right end point

int ncontrolpts = 6, i;

//clear the window with the current background color

cout << "in renderscene" << endl;

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

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

glEnable(GL\_DEPTH\_TEST);

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

glLoadIdentity();

//set the current drawing color to white

glColor3f(1.0, 1.0, 1.0);

//set the viewport to the window dimensions

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//set the viewport to the window dimensions

glViewport(0, 0, 540, 440);

//Establish the clipping volume in user coordinates

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

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

//set the drawing color to white

glColor3f(1.0, 1.0, 1.0);

// Draw x-axis and y-axis and place tic marks on each

glBegin(GL\_LINES);

// Set end points of x-axis

glVertex2f(-1.0, 0.0);//x left

glVertex2f(10.0, 0.0);// x right

// Now put tic marks on the axis

for (x1 = -1.0; x1 <= 10.0; x1 += 1.0)

{

glVertex2f(x1, 0.0);

glVertex2f(x1, 0.5);

};

// Set end points of y-axis

glVertex2f(0.0, -1.0);// y bottom

glVertex2f(0.0, 10.0);// y top

//Now put tic marks on the axis

for (y1 = -1.0; y1 <= 10.0; y1 += 1.0)

{

glVertex2f(-0.15, y1);

glVertex2f(0.15, y1);

};

glEnd();

// now draw the CONTROL POINTS

glPointSize(5.0);

//loop through all the points

glBegin(GL\_POINTS);

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); //This draws the Biezier Curve note that x(u), y(u) update by .005 as 0 <= u <= 1

Uval = 0.0;

for (i = 0; i <= 200; i++)

{//calculate the x,y coordinates for this uval

glVertex2f(calcbiezu(Uval, 5, xctrl), calcbiezu(Uval, 5, yctrl));

Uval += 0.005;

}

glEnd();

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

//

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

settrans3();

glColor3f(1.0, 1.0, 1.0);

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);//Draw Ski Slope

glVertex2f(0.0, 8.0);

glVertex2f(8.0, 0.0);

glVertex2f(0.0, 0.0);

glVertex2f(0.0, 8.0);

glEnd();

settrans2();

//now draw PolyMan

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

glEnd();

glutSwapBuffers();

return;

};//end of render scene

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

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

{

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

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

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

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

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

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

if (dx == 0) {

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

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

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

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

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

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

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

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

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

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

}

else {

//Set the coordinates of the body

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

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

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

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

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

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

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

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

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

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

}

//set the right foot

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

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

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

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

plx[4] = -1.0; ply[4] = -1.0;

//set the left foot

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

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

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

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

pl2x[4] = 1.0; pl2y[4] = -1.0;

//set the Closed mouth

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

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

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

//set the eye

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

pz[0] = 0.5;

pz[1] = -0.5;

return;

} //end of loadicon

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

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

{

//draw Polyman at the transformed position

int i;

cout << "in drawicon" << endl;

glColor3f(1.0, 0.0, 0.0);

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

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

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

}

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

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

}

glEnd();

i = 0;

glFrontFace(GL\_CW);

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

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

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

}

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

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

}

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

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

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

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

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

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

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

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

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

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

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

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

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

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

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

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

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

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

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

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

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

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

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

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

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

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

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

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

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

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

glColor3f(0.0, 0.5, 0.5);

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

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

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

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

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

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

glEnd();

glFrontFace(GL\_CW);

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

//Top

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

glColor3f(0.0, 0.5, 0.5);

glShadeModel(GL\_SMOOTH);

glBegin(GL\_POLYGON);

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

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

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

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

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

glEnd();

glColor3f(1.0, 1.0, 1.0);

//now draw the legs

glBegin(GL\_LINES);

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

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

{

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

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

glEnd();

glBegin(GL\_LINES);

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

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

{

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

}

glEnd();

//Draw the Skis

glColor3f(0.0,0.0,1.0);

glBegin(GL\_LINES);

glVertex3f(plx[4],ply[4],pz[0]);

glVertex3f(pl2x[4], pl2y[4],pz[0]);

glEnd();

glBegin(GL\_LINES);

glVertex3f(plx[4], ply[4], pz[1]);

glVertex3f(pl2x[4], pl2y[4], pz[1]);

glEnd();

//Draw Eye

glBegin(GL\_POINTS);

glColor3f(0.0, 0.0, 0.0);

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

glEnd();

glBegin(GL\_POINTS);

glColor3f(0.0, 0.0, 0.0);

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

//Draw mouth

glColor3f(0.0, 0.0, 0.0);

glEnd();

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

}

else {//ClosedMouth

glBegin(GL\_LINES);

glColor3f(0.0, 0.0, 0.0);

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

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

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

glEnd();

glColor3f(0.0, 0.0, 0.0);

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

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

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

}

glFlush();

return;

} //end of draw icon

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function settrans2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void settrans2(void)

//Sets the translation matrix for the cube

{

cout << "in settrans2" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity(); //Now set the location of the cube(dx, dy) below so it is on the Biezier Curve

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

dx = calcbiezu(uvalx, 5, xctrl);

dy = calcbiezu(uvalx, 5, yctrl);

//Here is where we use the Biezier calculated(dx, dy) to place the cube!

glTranslatef(dx, dy, dz);

glRotatef(180, 0, 1.0, 0.0);

glRotatef(25, 0, 0.0, 1.0);

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

glRotatef(theta2, 1.0, 1.0, 1.0);

return;

}

void settrans3(void)

//Sets the translation matrix for the cube

{

cout << "in settrans2" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity(); //Now set the location of the cube(dx, dy) below so it is on the Biezier Curve

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

//Here is where we use the Biezier calculated(dx, dy) to place the cube!

glTranslatef(0, 0, 0);

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

glRotatef(0.0, 1.0, 1.0, 1.0);

return;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Function SetupRC\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Setup the rendering state

void SetupRC(void)

{// this function sets the clear color of an open window and clears the open window

// Set clear color to green

glClearColor(0.0, 0.0, 0.0, 1.0);

return;

}//end of SetupRC

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Functioner Timer\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TimerFunction(int value)

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

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

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

// of Polyman.

{

uvalx += 0.001;

if (uvalx >= 1.0) uvalx = 1.0;

if ((dx > 5)&&(dy>4)&& (theta>-360))theta -= 1.0;

// Redraw the scene with new coordinates

glutPostRedisplay();

glutTimerFunc(15, TimerFunction, 1);

}

//This is the Biezier Calculation

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;

}

//This is the factorial calculation.

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;

}