Nick Wright CSC328 Problem 2

Code

/\*This program demonstrates two important concepts in OpenGL. 1) A frame based annimation engine is demonstrated in

function TimerFunction (note that it runs off of the global variable frame). In this case an icon begins at the right of the stage,f1 rolls to the center, f2 stops f3 rises to top of stage

f4 rotates and scales at top of stage, f5 falls to center of stage, f6 rolls to left off of stage

2) the second concept demonstrated is translation, rotation and scale of an icon. This is done primarily in the function

settrans() and the values for trans, rotat and scale are global in nature. Here the translation variables are dx and dy for movement in

x,y respectively, the scale is scale1 same for x and y and the rotation is theta\*/

#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;

//\*\*\*\*\*\*\*\*\*\*\* 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 theta = 50.0;//global angular value for rotation

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

float dx = 7.0, dy = -3.0;//global movement value for dx and dy/

int frame = 1;

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

void loadicon(float[], float[], float[], float[]); /\* loads the triangle icon \*/

void drawicon(float[], float[], float[], float[]); /\* draws the icon \*/

void settrans(float[][3], float, float, float); /\* sets the transformation matrix for desired scale, rotation, new pos \*/

float xprime(float, float, float[][3]); /\* calculates x' from x and transform \*/

float yprime(float, float, float[][3]); /\* calculates y' from y and transform \*/

void transform(float[], float[], float[], float[],

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

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

void myidle(void);

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

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

//Main Program

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

{//set up window title

char header[] = "Triangle by Nick Wright";

glutInit(&argc, argv);

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

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

//Initialize 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;

}

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

void RenderScene(void)

{

float xdel = 0.25;

float px[4], py[4], plx[2], ply[2]; //CHANGED THE VALUES OF PX AND PY TO 4. IT WAS PREVIOUSLY 5

/\* these variables hold the pattern for the square icon. Note that px,py hold the square, plx, ply hold the line \*/

float pxp[4], pyp[4], plxp[2], plyp[2], t[3][3]; //CHANGED THE VALUES OF PX AND PY TO 4. IT WAS PREVIOUSLY 5

/\* these variables hold the icon after it has been scaled, rotated and translated. like their counterparts pxp,pyp hold the square, plxp,plyp hold the line,

and t is the transformation matrix\*/

//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 volumn in user units

// First clear all translation matricies

glOrtho(-7.0, 7.0, -7.0, 7.0, 1.0, -1.0);

loadicon(px, py, plx, ply);

/\* draw the icon untransformed \*/

settrans(t, scale1, dx, dy);

transform(pxp, pyp, plxp, plyp, t, px, py, plx, ply);

// 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);

//now draw the triangle

drawicon(pxp, pyp, plxp, plyp);

glEnd();

glutSwapBuffers();

return;

};//end of render scene

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

void loadicon(float px[], float py[], float plx[], float ply[])

/\* This procedure loads the triangle icon \*/

{

/\* Setting the coordinates of the triangle \*/

px[0] = 0.0; py[0] = 1.0;

px[1] = 1.0; py[1] = 0.0;

px[2] =-1.0; py[2] = 0.0;

px[3] = 0.0; py[3] = 1.0;

/\* set the line || chnaged the values to match those of the triangle assignment \*/

plx[0] = 0.0; ply[0] = 1.5;

plx[1] = 0.0; ply[1] = -0.5;

return;

}/\* end of load icon \*/

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

void drawicon(float pxp[], float pyp[], float plxp[], float plyp[])

{

/\* this function draws the triangle icon at the transformed position \*/

int i;

cout << "in drawicon" << endl;

glBegin(GL\_LINE\_STRIP);

//move to first point in icon

glVertex2f(pxp[0], pyp[0]);

//now draw the rest of the triangle

for (i = 1; i <= 3; i++) //changed value from 4 to 3

glVertex2f(pxp[i], pyp[i]);

glEnd();

//now draw the line

glBegin(GL\_LINES);

glVertex2f(plxp[0], plyp[0]);

glVertex2f(plxp[1], plyp[1]);

glEnd();

//now fill the triangle which is made by half of the triangle

//set the shading color to red

glColor3f(1.0, 0.0, 0.0);

glShadeModel(GL\_FLAT);

//redraw the polygon

glBegin(GL\_POLYGON);

// note the colored triangle must be redrawn to render it.

//my shot at filling in the triangle

glVertex2f(pxp[0],pyp[0]);

//first point of fill starts where point 0 of the triangle is

glVertex2f(pxp[1],pyp[1]);

//second point of fill starts where point 1 of the trianle is

glVertex2f((pxp[1] + pxp[2]) / 2.0, (pyp[1] + pyp[2]) / 2.0);

//third point of fill is halfway between x values of p1 and p2, y value is the same as y value of p1 and p2

//on second thought i think i will still have the y value be divided by 2

//glVertex2f();

//drawing fill back to first / final point

return;

}//end of draw icon

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function settrans \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void settrans(float t[][3], float scale1, float dx, float dy)

{

cout << "in settrans" << endl;

int i, j;

float ts, ct, st;

double theta1;

/\* setup identiy matrix \*/

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

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

{

t[i][j] = 0.0;

if (i == j)t[i][j] = 1.0;

}

/\* set scale parameters \*/

if (scale1 != -9.0)

{

t[0][0] = scale1;

t[1][1] = scale1;

}

if (theta != -9.0)

{

theta1 = (3.1416 / 180.0)\*theta;

ct = cos(theta1);

st = sin(theta1);

ts = t[0][0];

t[0][0] = ts \* ct;

t[0][1] = ts \* st;

ts = t[1][1];

t[1][0] = -ts \* st;

t[1][1] = ts \* ct;

}

/\* translate the figure \*/

if ((dx + dy) != -18.0)

{

t[2][0] = dx;

t[2][1] = dy;

}

return;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function xprime \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float xprime(float x1, float y1, float t[][3])

{//this function multiplies the x vector by the transformation matrix

float xp1;

xp1 = x1 \* t[0][0] + y1 \* t[1][0] + t[2][0];

return xp1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function yprime \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float yprime(float x1, float y1, float t[][3])

{//this function multiplies the y vector by the transformation matrix

float yp1;

yp1 = x1 \* t[0][1] + y1 \* t[1][1] + t[2][1];

return yp1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* function transform \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void transform(float pxp[], float pyp[], float plxp[], float plyp[],

float t[][3], float px[], float py[],

float plx[], float ply[])

{

int i;

cout << "intransform" << endl;

/\*\*\*\*\*\*\* transform the figure \*/

for (i = 0; i <= 3; i++) //changed to 3

{

pxp[i] = xprime(px[i], py[i], t);

pyp[i] = yprime(px[i], py[i], t);

}

/\* transform the line \*/

for (i = 0; i <= 1; i++)

{

plxp[i] = xprime(plx[i], ply[i], t);

plyp[i] = yprime(plx[i], ply[i], t);

}

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, 1.0, 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

// of the square.

{

static float swc = 0.1, sdx = 0.1, sdy = 0.1;

switch (frame)

{

case 1: //frame 1 square starts at right and rolls the square to middle

theta += 5.0;

dx -= 0.15;

if (dx <= 0.0) {

dx = 0.0;

frame = 2;

}

break;

case 2:// frame 2 the square rises to y=3.0

dy += 0.2;

if (dy > 5.0)

{

dy = 5.0;

frame = 3;

}

break;

case 3:// frame 3 square rotates at x=0.0,y=3.0

theta += 5.0;

if (scale1 < 2.0)scale1 += 0.1;

else

scale1 = 1.0;

if (theta >= 720.0)

{

frame = 4;

theta = 0.0;

scale1 = 1.0;

}

break;

case 4: // frame 4 square moves down to x=0.0, y=-3.0

dy -= 0.2;

if (dy <= -3.0)

{

dy = -3.0;

frame = 5;

}

break;

case 5:// frame 5 square rolls off stage to left

dx -= 0.15;

theta += 5.0;

if (dx <= -6.5)dx = -6.5;

break;

}

// Redraw the scene with new coordinates

glutPostRedisplay();

glutTimerFunc(33, TimerFunction, 1);

}