Nick Wright CSC328 Test 1

Code

#include<windows.h>

#include<GL/glut.h>

#include<stdlib.h>

#include<math.h>

#include<conio.h>

#include<stdio.h>

#include <iostream>

#include <iomanip>

using namespace std;

/\* \*\*\*\*\*\*\*\*\*\*\*\*Take Home Test 1 CSC328\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*Nick Wright\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* The purpose of this program is to successfully use the function DrawsAllIcons to draw 3 different icons,

a square, a trapezoid, and a combination of the two, as well as transform them with translation, rotation,

and scaling using the modelview matrix. \*/

/\*-----------------Global Variables------------------\*/

//icon1 global

float icon1Theta = 0, icon1DX = 5.0, icon1DY = 5.0, icon1ScaleX = 1.0, icon1ScaleY = 1.0;

//icon2 global

float icon2Theta = 0, icon2DX = -5.0, icon2DY = -5.0, icon2ScaleX = 1.0, icon2ScaleY = 1.0;

//icon3 global

float icon3Theta = 0, icon3DX = 0.0, icon3DY = 0.0, icon3ScaleX = 1.0, icon3ScaleY = 1.0;

int frame = 1;

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

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

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

void TimerFunction(int);

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

/\* drawsAllIcons has the functionality to draw any number of shapes, with any combination of RGB color values,

drawn using GL\_LINES, GL\_LINE\_STRIP, or GL\_POLYGON.

It also has the functionality to translate, rotate, and scale created icons using the modelview matrix \*/

//creating function drawsAllIcons

void drawsAllIcons(float x[], float y[], int ndraws, int pointsperdraw[], int drawtype[],

float colorR[], float colorG[], float colorB[],

float rotate, float scaleX, float scaleY, float transX, float transY);

//defining the function drawsAll Icons

void drawsAllIcons(float x[], float y[], int ndraws, int pointsperdraw[], int drawtype[],

float colorR[], float colorG[], float colorB[],

float rotate, float scaleX, float scaleY, float transX, float transY)

{

int i;

cout << "in drawsAllIcons" << endl;

//creating global variables for rotate, scale, and translate within the drawsAllIcons functions

float gRotate = 0, gScaleX = 1, gScaleY = 1, gTransX = 0, gTransY = 0;

//creating accumulators for draw functionality

int ppdAcc = 0; //points per draw accumulator

int ndAcc = 0; //number of draws accumulator

int dtAcc = 0; //drawtype accumulator

int colorAcc = 0; //color accumulator

int vertexAcc = 0;//the value for the arrays in the draw loop

//modelView functionalty

//sets the modelviel matrix for the icon

cout << "at ModelView functionality" << endl;

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

//not sure that these calls will work

glTranslatef(transX, transY, 0.0);

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

glScalef(scaleX, scaleY, 0.0);

while (ndAcc < ndraws)

//while the number of draws accumulator (starts at 0),

//is less than or equal to the number of draws, \*go through the draw conditions\*

{

//setting the color value for each new draw in loop

glColor3f(colorR[colorAcc], colorG[colorAcc], colorB[colorAcc]);

//if statements for the draws

if (drawtype[dtAcc] == 1)

{

glBegin(GL\_LINES);

for (i = 0; i < pointsperdraw[ppdAcc]; i++)

{

glVertex2f(x[vertexAcc], y[vertexAcc]);

vertexAcc += 1;

}

glEnd();

}

else if (drawtype[dtAcc] == 2)

{

glBegin(GL\_LINE\_STRIP);

for (i = 0; i < pointsperdraw[ppdAcc]; i++)

{

glVertex2f(x[vertexAcc], y[vertexAcc]);

vertexAcc += 1;

}

glEnd();

}

else if (drawtype[dtAcc] == 3)

{

glShadeModel(GL\_FLAT);

//redraw the polygon

glBegin(GL\_POLYGON);

//the colored polygon must be redrawn to render it

for (i = 0; i < pointsperdraw[ppdAcc]; i++)

{

glVertex2f(x[vertexAcc], y[vertexAcc]);

vertexAcc += 1;

}

glEnd();

}

//accumulate

ndAcc += 1; //add one to the number of draws accumulator

ppdAcc += 1; //add one to the accumulator for points per draw

dtAcc += 1;

colorAcc += 1;

} //end of while loop

return;

} //end of drawsAllIcons

//creating loadIcon1

void loadIcon1(float icon1X[], float icon1Y[]);

//defining loadIcon1

void loadIcon1(float icon1X[], float icon1Y[])

{

//this function will load icon 1 (square)

//setting the coordinates of the square

//square

icon1X[0] = 1.0; icon1Y[0] = 1.0;

icon1X[1] = 1.0; icon1Y[1] = -1.0;

icon1X[2] = -1.0; icon1Y[2] = -1.0;

icon1X[3] = -1.0; icon1Y[3] = 1.0;

icon1X[4] = 1.0; icon1Y[4] = 1.0;

//line through square

icon1X[5] = 0.0; icon1Y[5] = 2.0;

icon1X[6] = 0.0; icon1Y[6] = -2.0;

//coords for fill

icon1X[7] = icon1X[0]; icon1Y[7] = icon1Y[0];

icon1X[8] = icon1X[1]; icon1Y[8] = icon1Y[1];

icon1X[9] = (icon1X[1] + icon1X[2]) / 2; icon1Y[9] = (icon1Y[1] + icon1Y[2]) / 2;

icon1X[10] = (icon1X[3] + icon1X[0]) / 2; icon1Y[10] = (icon1Y[3] + icon1Y[0]) / 2;

return;

} //end of loadIcon1

//creating loadIcon2

void loadIcon2(float icon2X[], float icon2Y[]);

//defining loadIcon2

void loadIcon2(float icon2X[], float icon2Y[])

{

//this function will load icon 2 (trapezoid)

//setting the coordinates of the trapezoid

icon2X[0] = 1.0; icon2Y[0] = 1.0;

icon2X[1] = 1.5; icon2Y[1] = -1.0;

icon2X[2] = -1.5; icon2Y[2] = -1.0;

icon2X[3] = -1.0; icon2Y[3] = 1.0;

icon2X[4] = 1.0; icon2Y[4] = 1.0;

//line through trapezoid

icon2X[5] = 0.0; icon2Y[5] = 2.0;

icon2X[6] = 0.0; icon2Y[6] = -2.0;

//trapezoid fill

icon2X[7] = (icon2X[3] + icon2X[0]) / 2; icon2Y[7] = (icon2Y[3] + icon2Y[0]) / 2;

icon2X[8] = (icon2X[2] + icon2X[1]) / 2; icon2Y[8] = (icon2Y[2] + icon2Y[1]) / 2;

icon2X[9] = icon2X[2]; icon2Y[9] = icon2Y[2];

icon2X[10] = icon2X[3]; icon2Y[10] = icon2Y[3];

return;

}

//creating loadIcon3

void loadIcon3(float icon3X[], float icon3Y[]);

//defining loadIcon3

void loadIcon3(float icon3X[], float icon3Y[])

{

//this function will load the combined square trapezoid icon

//setting the coordinates of the square

//square

icon3X[0] = 1.0; icon3Y[0] = 1.0;

icon3X[1] = 1.0; icon3Y[1] = 0.0;

icon3X[2] = -1.0; icon3Y[2] = 0.0;

icon3X[3] = -1.0; icon3Y[3] = 1.0;

icon3X[4] = 1.0; icon3Y[4] = 1.0;

//square fill

icon3X[5] = icon3X[0]; icon3Y[5] = icon3Y[0];

icon3X[6] = icon3X[1]; icon3Y[6] = icon3Y[1];

icon3X[7] = (icon3X[1] + icon3X[2]) / 2; icon3Y[7] = (icon3Y[1] + icon3Y[2]) / 2;

icon3X[8] = (icon3X[3] + icon3X[0]) / 2; icon3Y[8] = (icon3Y[3] + icon3Y[0]) / 2;

//setting the coordinates of the trapezoid

//trapezoid

icon3X[9] = 1.5; icon3Y[9] = 0.0;

icon3X[10] = 2.0; icon3Y[10] = -1.0;

icon3X[11] = -2.0; icon3Y[11] = -1.0;

icon3X[12] = -1.5; icon3Y[12] = 0.0;

icon3X[13] = 1.5; icon3Y[13] = 0.0;

//trapezoid fill

icon3X[14] = (icon3X[12] + icon3X[9]) / 2; icon3Y[14] = (icon3Y[12] + icon3Y[9]) / 2;

icon3X[15] = (icon3X[11] + icon3X[10]) / 2; icon3Y[15] = (icon3Y[11] + icon3Y[10]) / 2;

icon3X[16] = icon3X[11]; icon3Y[16] = icon3Y[11];

icon3X[17] = icon3X[12]; icon3Y[17] = icon3Y[1];

//line through both shapes

icon3X[18] = 0.0; icon3Y[18] = 3.0;

icon3X[19] = 0.0; icon3Y[19] = -3.0;

return;

}

//Main Program

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

{

//set up the window title

char header[] = "Graphics Test 1 by Nick Wright";

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

initialize the native window system.

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

glutInit(&argc, argv);

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

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

//initialize the window size and position

glutInitWindowSize(560, 440);

glutInitWindowPosition(140, 20);

//Initialize background color in window to red

SetupRC();

// Open and Label Window

glutCreateWindow(header);

glutDisplayFunc(RenderScene);

glutTimerFunc(30, TimerFunction, 1);

//now draw the scene

glutMainLoop();

return 0;

}

//Render Scene Function

void RenderScene(void)

{

float xdel = 0.25;

//pattern for icon 1

float icon1X[11], icon1Y[11];

//points per draw on icon 1

int icon1PPD[3] = { 5,2,4 };

//drawType for icon 1

int icon1DT[3] = {2,1,3};

//color floats for icon 1

float icon1ColorR[3] = { 1.0, 1.0, 1.0 };

float icon1ColorG[3] = { 0,0,0 };

float icon1ColorB[3] = { 0,0,0 };

//pattern for icon 2

float icon2X[11], icon2Y[11];

//points per draw on icon 2

int icon2PPD[3] = { 5,2,4 };

//drawtype for icon 2

int icon2DT[3] = { 2,1,3 };

//color floats for icon 2

float icon2ColorR[3] = { 0,0,0 };

float icon2ColorG[3] = { 1.0,1.0,1.0};

float icon2ColorB[3] = { 0,0,0 };

//pattern for icon 3

float icon3X[20], icon3Y[20];

//points per draw on icon 3

int icon3PPD[5] = { 5,4,5,4,2 };

//drawtype for icon 3

int icon3DT[5] = { 2,3,2,3,1 };

//color floats for icon 3

float icon3ColorR[5] = { 1.0,1.0,0,0,0 };

float icon3ColorG[5] = { 0,0,1.0,1.0,0 };

float icon3ColorB[5] = { 0,0,0,0,1.0 };

//clear the window with the current background color

cout << "in renderscene" << endl;

//set the current drawing color to white

glColor3f(1.0, 1.0, 1.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

//set the viewport to the window dimensions

glViewport(0, 0, 540, 440);

//Establish the clipping volume in user coordinates

glOrtho(-9.0, 9.0, -9.0, 9.0, 1.0, -1.0);

//load the icons untransformed

loadIcon1(icon1X, icon1Y);

loadIcon2(icon2X, icon2Y);

loadIcon3(icon3X, icon3Y);

//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);

//calling drawsAllIcons 3 times, once for each icon

//glFlush being performed after each draw

drawsAllIcons(icon1X, icon1Y, 3, icon1PPD, icon1DT, icon1ColorR, icon1ColorG, icon1ColorB,

icon1Theta, icon1ScaleX, icon1ScaleY, icon1DX, icon1DY);

glFlush();

drawsAllIcons(icon2X, icon2Y, 3, icon2PPD, icon2DT, icon2ColorR, icon2ColorG, icon2ColorB,

icon2Theta, icon2ScaleX, icon2ScaleY, icon2DX, icon2DY);

glFlush();

drawsAllIcons(icon3X, icon3Y, 5, icon3PPD, icon3DT, icon3ColorR, icon3ColorG, icon3ColorB,

icon3Theta, icon3ScaleX, icon3ScaleY, icon3DX, icon3DY);

glFlush();

glEnd();

glutSwapBuffers();

return;

}//end of renderscene

//function SetupRC

// Setup the rendering state

void SetupRC(void)

{

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

// Set clear color to blue

glClearColor(0.0, 0.0, 1.0, 1.0);

return;

}//end of setuprc

//function timer

void TimerFunction(int value)

{

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

switch (frame)

{

case 1:

icon1Theta += 5.0;

if (icon1ScaleX < 2.0)

{

icon1ScaleX += 0.1;

icon1ScaleY += 0.1;

}

else

{

icon1ScaleX = 1;

icon1ScaleY = 1;

}

icon2Theta += 5.0;

if (icon2ScaleX < 2.0)

{

icon2ScaleX += 0.1;

icon2ScaleY += 0.1;

}

else

{

icon2ScaleX = 1;

icon2ScaleY = 1;

}

icon3Theta -= 5.0;

break;

}

// Redraw the scene with new coordinates

glutPostRedisplay();

glutTimerFunc(30, TimerFunction, 1);

}

Output

