#include <stdio.h>

#include <GL/glut.h>

#include <math.h>

// Draw chess pattern rotate it and fill it with different colours

typedef struct pixel

{

GLubyte r, g, b;

} pixel;

pixel f\_color, b\_color;

float mat1[20][3];

float ans1[20][3];

float trans1[3][3];

void initial\_co()

{

int i, y, x;

y = 90;

// horizontal lines

for (i = 0; i < 10; i += 2)

{

// first point

mat1[i][0] = 90;

mat1[i][1] = y;

mat1[i][2] = 1;

// second point

mat1[i + 1][0] = 210;

mat1[i + 1][1] = y;

mat1[i + 1][2] = 1;

y += 30;

}

x = 90;

// vertical lines

for (i; i < 20; i += 2)

{

// first point

mat1[i][0] = x;

mat1[i][1] = 90;

mat1[i][2] = 1;

// second point

mat1[i + 1][0] = x;

mat1[i + 1][1] = 210;

mat1[i + 1][2] = 1;

x += 30;

}

}

void rotate\_fig()

{

int i, j, k;

float theta;

theta = 45 \* 3.14 / 180;

/\*----------------translation to origin -------------------------------\*/

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

{

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

{

if (i == j)

trans1[i][j] = 1;

else

trans1[i][j] = 0;

}

}

trans1[2][0] = trans1[2][1] = -150;

/\*

trans1= 1 0 0

0 1 0

tx ty 1

\*/

for (i = 0; i < 20; i++)

{

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

{

ans1[i][j] = 0;

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

ans1[i][j] += mat1[i][k] \* trans1[k][j];

}

}

/\*-----------------------rotation at origin--------------------------------\*/

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

{

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

{

if (i == j)

trans1[i][j] = 1;

else

trans1[i][j] = 0;

}

}

trans1[0][0] = trans1[1][1] = cos(theta);

trans1[0][1] = sin(theta);

trans1[1][0] = -sin(theta);

/\*

trans1= cos sin 0

-sin cos 0

0 0 1

\*/

for (i = 0; i < 20; i++)

{

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

{

mat1[i][j] = 0;

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

mat1[i][j] += ans1[i][k] \* trans1[k][j];

}

}

/\*-----------------------translation back-----------------------------\*/

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

{

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

{

if (i == j)

trans1[i][j] = 1;

else

trans1[i][j] = 0;

}

}

trans1[2][0] = trans1[2][1] = 150;

for (i = 0; i < 20; i++)

{

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

{

ans1[i][j] = 0;

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

ans1[i][j] += mat1[i][k] \* trans1[k][j];

}

}

}

void boundary\_fill(int x, int y)

{

pixel c;

glReadPixels(x, y, 1, 1, GL\_RGB, GL\_UNSIGNED\_BYTE, &c); // values are put into c

// if color not equal to backgroung color and filling color put color

if ((c.r != b\_color.r || c.g != b\_color.g || c.b != b\_color.b) && (c.r !=

f\_color.r || c.g != f\_color.g || c.b != f\_color.b))

{

glColor3ub(f\_color.r, f\_color.g, f\_color.b); // set fill color for pixel

glBegin(GL\_POINTS);

glVertex2d(x, y); // put pixel

glEnd();

glFlush();

boundary\_fill(x + 1, y); // right pixel

boundary\_fill(x - 1, y); // left pixel

boundary\_fill(x, y + 1); // upper pixel

boundary\_fill(x, y - 1); // lower pixel

}

}

void before()

{

int i;

initial\_co();

glBegin(GL\_LINES); // draws the new figure

for (i = 0; i < 20; i += 2)

{

glVertex2f(mat1[i][0], mat1[i][1]);

glVertex2f(mat1[i + 1][0], mat1[i + 1][1]);

}

glEnd();

glFlush();

}

void figure()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

int i;

float factor = 30 \* cos(45 \* 3.14 / 180);

rotate\_fig(); // rotates the figure about the middle point (150,150)

glBegin(GL\_LINES); // draws the new figure

for (i = 0; i < 20; i += 2)

{

glVertex2f(ans1[i][0], ans1[i][1]);

glVertex2f(ans1[i + 1][0], ans1[i + 1][1]);

}

glEnd();

glFlush();

// filling the boxes with colours

// red

boundary\_fill(150, 150 + factor);

// green

f\_color.r = 0;

f\_color.g = 255;

f\_color.b = 0;

boundary\_fill(150, 150 + 3 \* factor);

// blue

f\_color.r = 0;

f\_color.g = 0;

f\_color.b = 255;

boundary\_fill(150, 150 - factor);

// yellow

f\_color.r = 255;

f\_color.g = 255;

f\_color.b = 0;

boundary\_fill(150, 150 - 3 \* factor);

// light blue

f\_color.r = 0;

f\_color.g = 255;

f\_color.b = 255;

boundary\_fill(150 + 2 \* factor, 150 + factor);

// pink

f\_color.r = 255;

f\_color.g = 0;

f\_color.b = 255;

boundary\_fill(150 - 2 \* factor, 150 + factor);

// purple

f\_color.r = 150;

f\_color.g = 0;

f\_color.b = 255;

boundary\_fill(150 + 2 \* factor, 150 - factor);

// light violet

f\_color.r = 150;

f\_color.g = 150;

f\_color.b = 255;

boundary\_fill(150 - 2 \* factor, 150 - factor);

}

void init\_func() // empty function doesnt do anything

{

glClearColor(1.0, 1.0, 1.0, 0.0); // sets the background colour

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0, 0.0, 0.0); // sets the drawing colour

gluOrtho2D(0, 500, 0, 500); // sets the co ordinates

before();

//figure();

glFlush();

}

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

{

glutInit(&argc, argv); // initializing the library

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB); // setting the display mode

glutInitWindowPosition(0, 0); // position of the window

glutInitWindowSize(500, 500); // size of the window

glutCreateWindow("Pattern"); // name of the window

glutDisplayFunc(init\_func); // displays the function

glutMainLoop(); // keeps the program open until closed

return 0;

}