**Assignment: Lab Report 04**

*Course title: Computer Graphics Lab*

*Course code: CSE-304*

*3rd Year 1st Semester Examination 2022*

**Date of Submission**: *15th July 2023*



**Submitted to-**

**Dr. Mohammad Shorif Uddin**

*Professor*

**Dr. Morium Akter**

*Associate Professor*

*Department of Computer Science and Engineering Jahangirnagar University Savar, Dhaka-1342*

| **Sl** | Class Roll | Exam Roll | Name |
| --- | --- | --- | --- |
|  |  |  |  |
| 01 | 370 | 202182 | Rubayed All Islam |
|  |  |  |  |

**Name of the experiment:**

Implementation of

1. flood fill algorithm
2. boundary fill algorithm.
3. Resize to double of a triangle whose vertexes are A(0,0), B(1,1), C(5,2) keeping C fixed.

**Introduction:**

Scan conversion with 2D transformations involves applying geometric transformations, such as translation, rotation, scaling, and shearing, to objects before converting them into pixel representations on a computer screen. Before scan conversion, objects are transformed by applying appropriate transformation matrices to their vertices, which modify their positions, orientations, sizes, or shapes in the 2D space. Translation involves shifting the position of an object along the x and y axes, while rotation rotates the object around a specific point or axis. Scaling modifies the size of an object, making it larger or smaller in the x and y directions, while shearing skews the object along the x or y axis. After applying the transformations, the transformed object is then scan converted using algorithms such as Bresenham line algorithm, midpoint circle algorithm, or polygon fill algorithms. The transformed object is divided into smaller elements such as lines, curves, or polygons, and the scan conversion algorithm determines which pixels within these elements should be activated to accurately represent the transformed shape. The accuracy of scan conversion with 2D transformations depends on the precision of the transformation matrices used and the efficiency of the scan conversion algorithm employed. Scan conversion with 2D transformations is essential in computer graphics for rendering transformed objects on a digital screen, allowing for the generation of complex and dynamically changing visuals

Source Code:

**1A) Flood Fill Algorithm:**

**Code:**

#include<stdio.h>

#include<graphics.h>

void flood(int x,int y,int new\_col,int old\_col)

{

if(getpixel(x,y)==old\_col)

{

putpixel(x,y,new\_col);

flood(x+1,y,new\_col,old\_col);

flood(x-1,y,new\_col,old\_col);

flood(x,y+1,new\_col,old\_col);

flood(x,y-1,new\_col,old\_col);

}

}

int main()

{

int gd,gm=DETECT;

initgraph(&gd,&gm,"");

int top,left,bottom,right;

top=left=50;

bottom=right=300;

rectangle(left,top,right,bottom);

int x=51;

int y=51;

int newcolor=12;

int oldcolor=0;

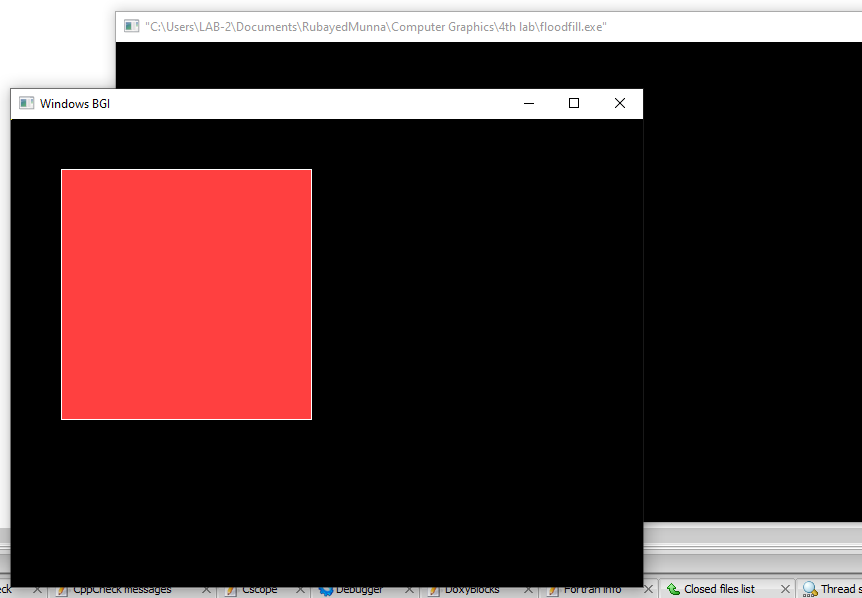
flood(x,y,newcolor,oldcolor);

getch();

return 0;

}

**Screenshot:**



**1B) Boundary Fill Algorithm:**

**Code:**

#include<graphics.h>

void boundaryFill(int x,int y,int fill\_color,int boundary\_color)

{

if(getpixel(x,y)!=boundary\_color&&getpixel(x,y)!=fill\_color)

{

putpixel(x,y,fill\_color);

boundaryFill(x+1,y,fill\_color,boundary\_color);

boundaryFill(x,y+1,fill\_color,boundary\_color);

boundaryFill(x-1,y,fill\_color,boundary\_color);

boundaryFill(x,y-1,fill\_color,boundary\_color);

}

}

int main()

{

int gd=DETECT,gm;

initgraph(&gd,&gm,"");

int x=250,y=200,radius=100;

circle(x,y,radius);

boundaryFill(x,y,10,15);

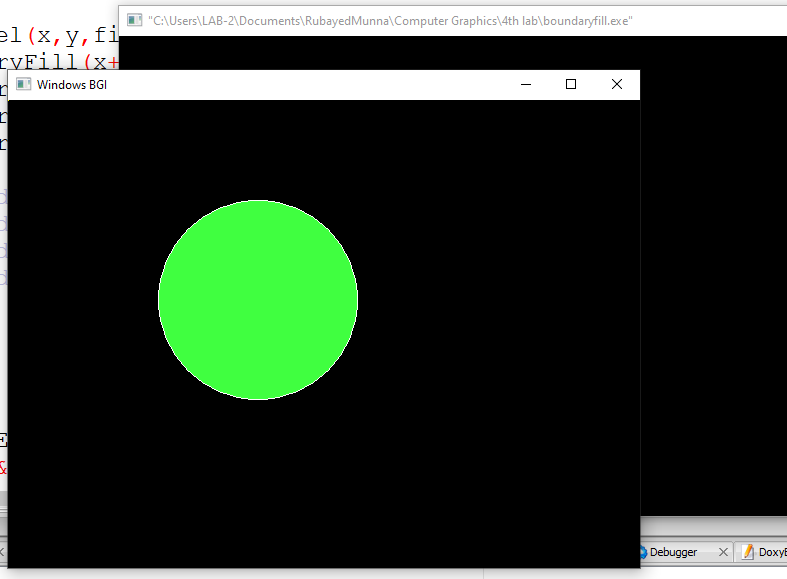
getch();

closegraph();

return 0;

}

**Screenshot:**



**2) Resize to double of a triangle whose vertexes are A (0,0), B (1,1), C (5,2) keeping C fixed.**

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <graphics.h>

void drawTriangle(int x1, int y1, int x2, int y2, int x3, int y3) {

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

}

void resizeTriangle(double scaleFactor, int fixedX, int fixedY,int x1, int y1, int x2, int y2, int x3, int y3,int\* newX1, int\* newY1, int\* newX2, int\* newY2, int\* newX3, int\* newY3) {

\*newX1 = fixedX + (int)((x1 - fixedX) \* scaleFactor);

\*newY1 = fixedY + (int)((y1 - fixedY) \* scaleFactor);

\*newX2 = fixedX + (int)((x2 - fixedX) \* scaleFactor);

\*newY2 = fixedY + (int)((y2 - fixedY) \* scaleFactor);

\*newX3 = fixedX + (int)((x3 - fixedX) \* scaleFactor);

\*newY3 = fixedY + (int)((y3 - fixedY) \* scaleFactor);

}

int main() {

int gd, gm;

gd = DETECT;

initgraph(&gd, &gm, "");

int x1 = 0, y1 = 0;

int x2 = 100, y2 = 100;

int x3 = 500, y3 = 200;

// int x1 = 300, y1 = 300;

// int x2 = 400, y2 = 400;

// int x3 = 800, y3 = 500;

int fixedX = x3;

int fixedY = y3;

drawTriangle(x1, y1, x2, y2, x3, y3);

double scaleFactor = 2.0;

int newX1, newY1, newX2, newY2, newX3, newY3;

resizeTriangle(scaleFactor, fixedX, fixedY, x1, y1, x2, y2, x3, y3,&newX1, &newY1, &newX2, &newY2, &newX3, &newY3);

drawTriangle(newX1, newY1, newX2, newY2, newX3, newY3);

getch();

closegraph();

return 0;

}

**Screenshot:**

