East Delta University

Experiment Name: Implementation of Flood fill algorithm

Name: Pranesh Chowdhury

ID: 202003112

Section: 02

Instructor Name: Tasnimatul Jannah

Course Name: CSE 322 Computer Graphics Lab

Title: Implementation of Flood Fill Algorithm.

Introduction:

Flood fill algorithm is a popular algorithm used in computer graphics, image processing and game development to fill a bound are a with a particular color or pattern.

Discussion:

In this algorithm, A point which is initial point or seed which is inside region is selected. This point is called a seed point. then four connected approaches or eight connected approaches in used to fill with specified color. The flood fill algorithm has many characters similar to boundary fill. But this method is suitable for filling multiple colors and interior is to be filled with one color we use this algorithm.

In this algorithm, we start from a specified interior point (x,y) and reasign all pixel values are currently set to a given interior color with the desired color. Using either 4 connected or 8 connected approaches, we then step through pixel positions until all interior points have been repainted. The flood fill algorithm takes two parameters:

Starts position represents the index of the cell that we want to recolor with all its connected cells that have the names initial color.

New color that the given area will have after applying the flood fill algorithm.

Flood filling Process:

(i) Plot the position of the start points.

(ii) Decide whether you want to go in 4 connect or 8 connect approach.

(iii) Target color replaced by with the white (or 1) color. the target color land on the pixel. It will replace with the choosen color.

(iv) Repeat (iii) untill you have been everywhere within boundaries.

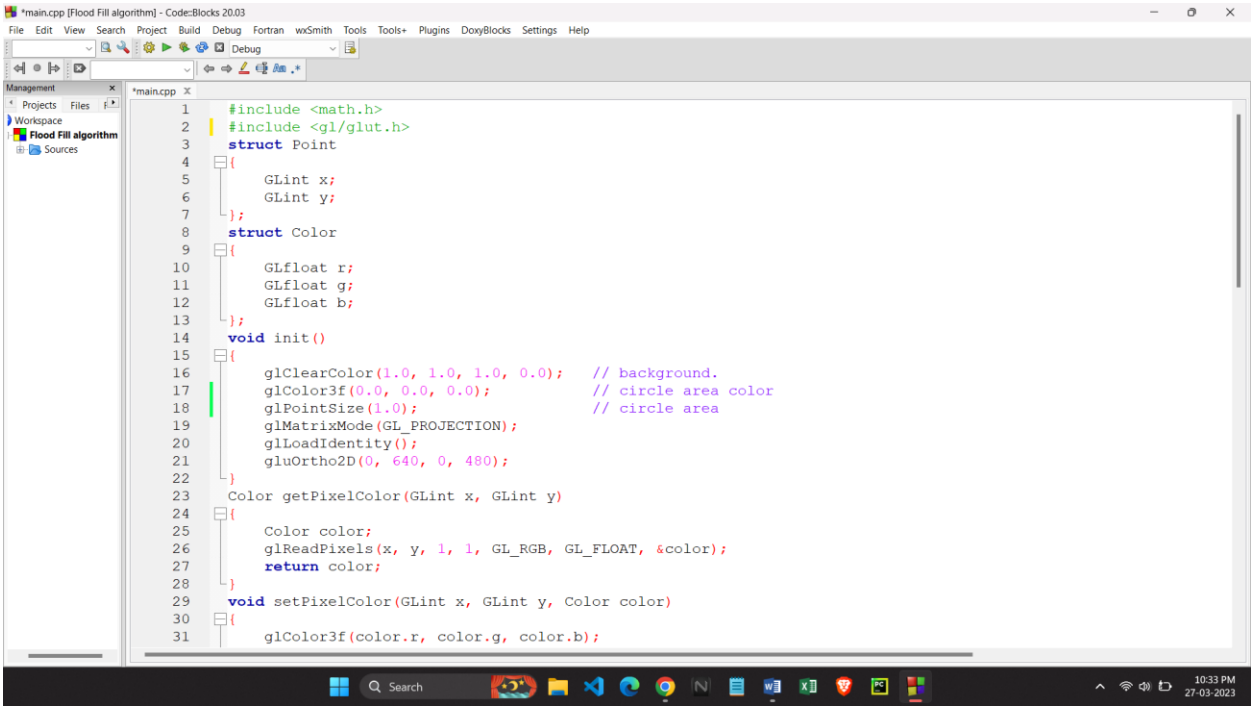
the algorithm don't only color the neighbors of the initial cells, and keep finding the neighbore of the initial cells, and finding the neighbore of each newly disconnected cell. The operation continues until no neighboring cells are disconnected.

Conclusion:

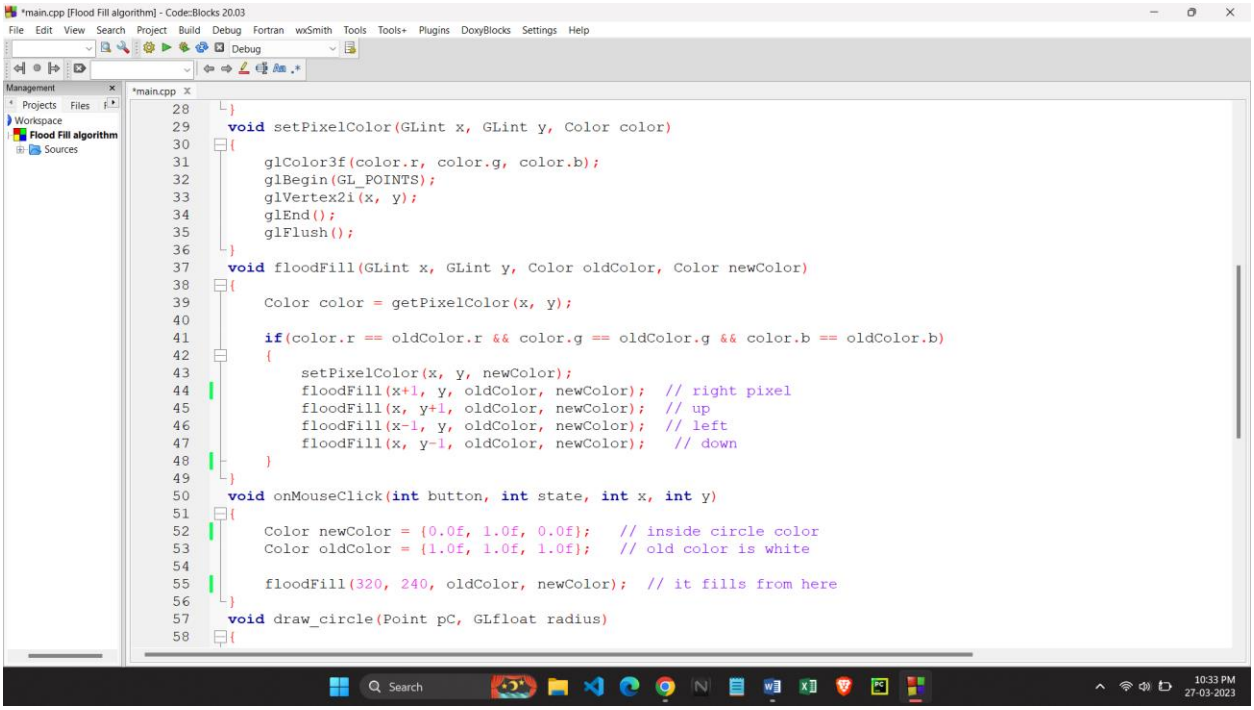
Flood fill algorithm is a effective way to fill closed areas in an image. It can handle irregularly shaped regions, as long as they are completely enclosed. Initial pixel required more knowledge about

Sqnnounding pixels. I didn't ~~for~~ encounter any difficulties during the implementation.

Code:



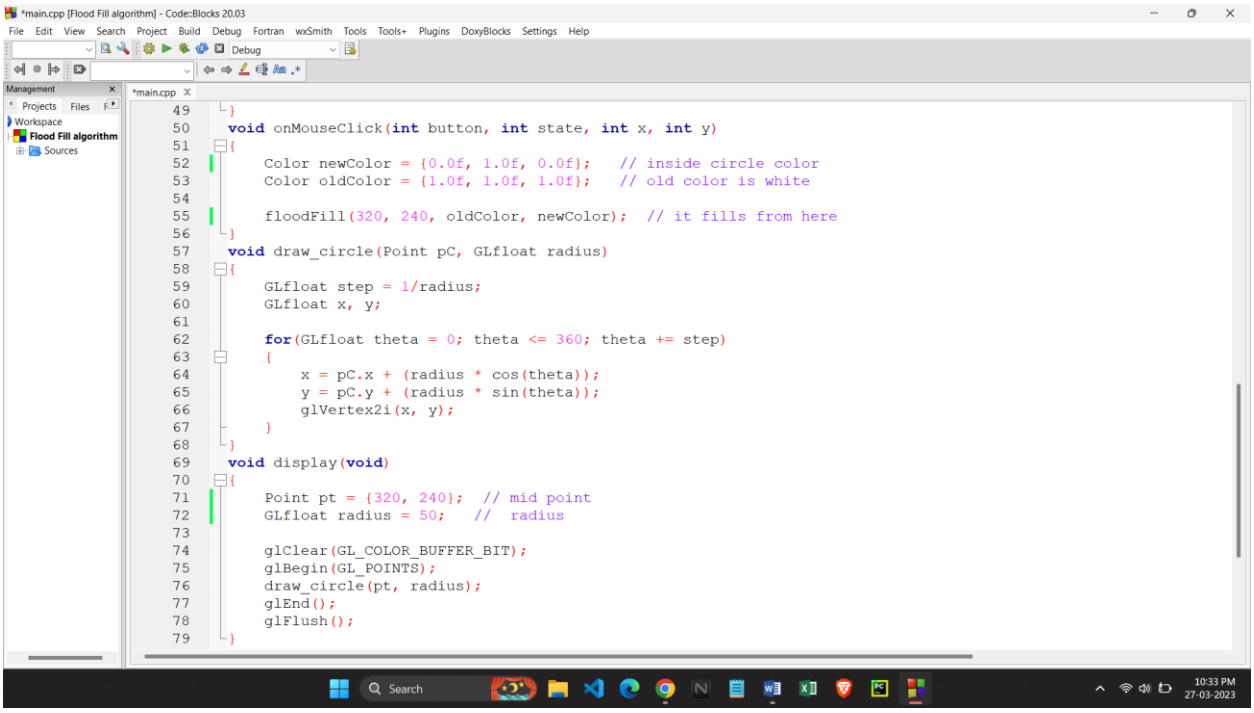
```
#include <math.h>
#include <gl/glut.h>
struct Point
{
    GLint x;
    GLint y;
};
struct Color
{
    GLfloat r;
    GLfloat g;
    GLfloat b;
};
void init()
{
    glClearColor(1.0, 1.0, 1.0, 0.0);   // background.
    glColor3f(0.0, 0.0, 0.0);           // circle area color
    glPointSize(1.0);                   // circle area
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, 640, 0, 480);
}
Color getPixelColor(GLint x, GLint y)
{
    Color color;
    glReadPixels(x, y, 1, 1, GL_RGB, GL_FLOAT, &color);
    return color;
}
void setPixelColor(GLint x, GLint y, Color color)
{
    glColor3f(color.r, color.g, color.b);
```

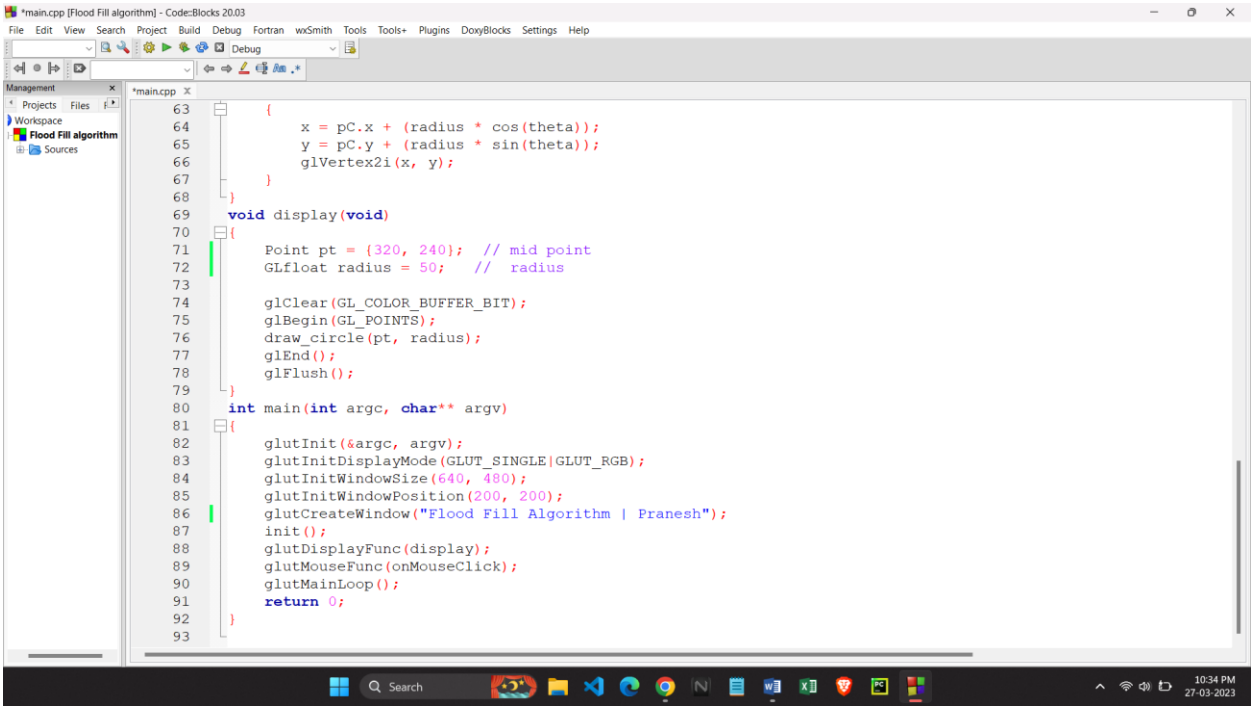


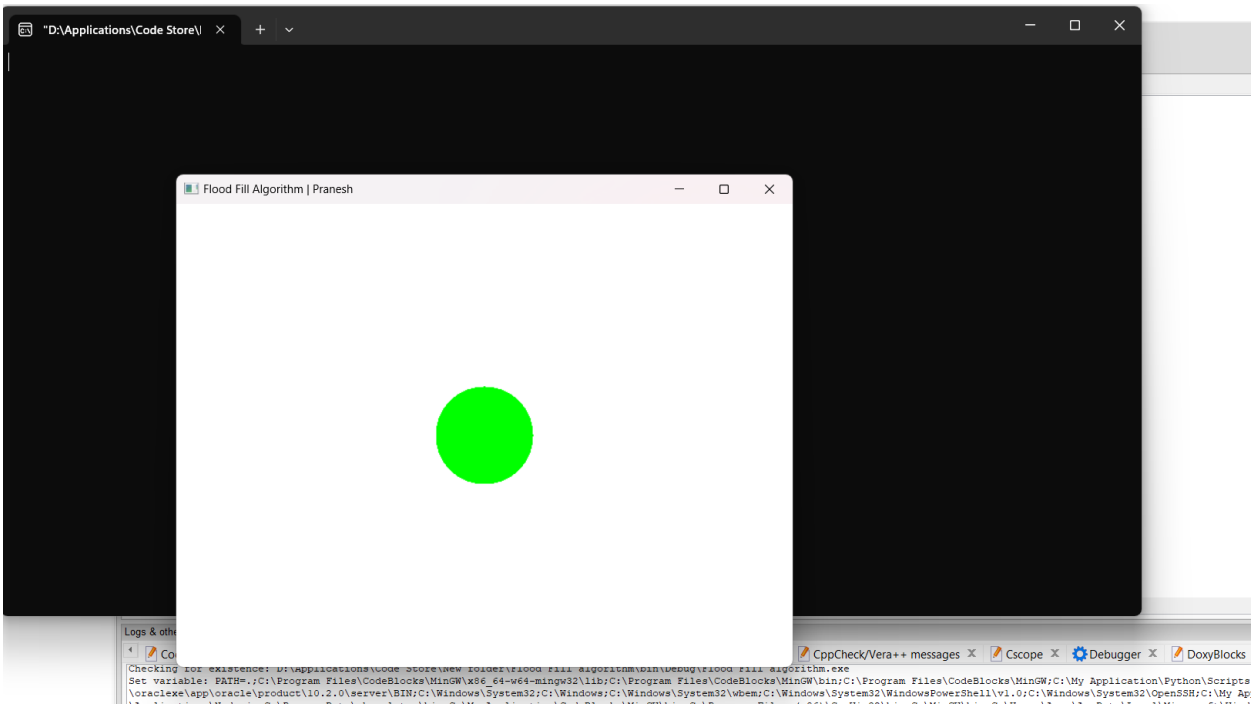
```
}
void setPixelColor(GLint x, GLint y, Color color)
{
    glColor3f(color.r, color.g, color.b);
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
    glFlush();
}
void floodFill(GLint x, GLint y, Color oldColor, Color newColor)
{
    Color color = getPixelColor(x, y);

    if(color.r == oldColor.r && color.g == oldColor.g && color.b == oldColor.b)
    {
        setPixelColor(x, y, newColor);
        floodFill(x+1, y, oldColor, newColor);  // right pixel
        floodFill(x, y+1, oldColor, newColor);  // up
        floodFill(x-1, y, oldColor, newColor);  // left
        floodFill(x, y-1, oldColor, newColor);   // down
    }
}
void onMouseClick(int button, int state, int x, int y)
{
    Color newColor = {0.0f, 1.0f, 0.0f};   // inside circle color
    Color oldColor = {1.0f, 1.0f, 1.0f};   // old color is white

    floodFill(320, 240, oldColor, newColor);  // it fills from here
}
void draw_circle(Point pC, GLfloat radius)
{
```


*main.cpp [Flood Fill algorithm] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects Files
Workspace
Flood Fill algorithm
Sources
*main.cpp
49 }
50 void onMouseClick(int button, int state, int x, int y)
51 {
52 Color newColor = {0.0f, 1.0f, 0.0f}; // inside circle color
53 Color oldColor = {1.0f, 1.0f, 1.0f}; // old color is white
54
55 floodFill(320, 240, oldColor, newColor); // it fills from here
56 }
57 void draw_circle(Point pC, GLfloat radius)
58 {
59 GLfloat step = 1/radius;
60 GLfloat x, y;
61
62 for(GLfloat theta = 0; theta <= 360; theta += step)
63 {
64 x = pC.x + (radius * cos(theta));
65 y = pC.y + (radius * sin(theta));
66 glVertex2i(x, y);
67 }
68 }
69 void display(void)
70 {
71 Point pt = {320, 240}; // mid point
72 GLfloat radius = 50; // radius
73
74 glClear(GL_COLOR_BUFFER_BIT);
75 glBegin(GL_POINTS);
76 draw_circle(pt, radius);
77 glEnd();
78 glFlush();
79 }
Search
10:33 PM
27-03-2023


*main.cpp [Flood Fill algorithm] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects Files
Workspace
Flood Fill algorithm
Sources
*main.cpp
63 {
64 x = pC.x + (radius * cos(theta));
65 y = pC.y + (radius * sin(theta));
66 glVertex2i(x, y);
67 }
68 }
69 void display(void)
70 {
71 Point pt = {320, 240}; // mid point
72 GLfloat radius = 50; // radius
73
74 glClear(GL_COLOR_BUFFER_BIT);
75 glBegin(GL_POINTS);
76 draw_circle(pt, radius);
77 glEnd();
78 glFlush();
79 }
80 int main(int argc, char** argv)
81 {
82 glutInit(&argc, argv);
83 glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
84 glutInitWindowSize(640, 480);
85 glutInitWindowPosition(200, 200);
86 glutCreateWindow("Flood Fill Algorithm | Pranesh");
87 init();
88 glutDisplayFunc(display);
89 glutMouseFunc(onMouseClick);
90 glutMainLoop();
91 return 0;
92 }
93
Search
10:34 PM
27-03-2023


"D:\Applications\Code Store\
Flood Fill Algorithm | Pranesh
CppCheck/Vera++ messages
Cscope
Debugger
DoxyBlocks