EXPERIMENT 4

TITLE: Filling objects using Flood Fill and Boundary Fill.

Flood Fill Algorithm

CODE:

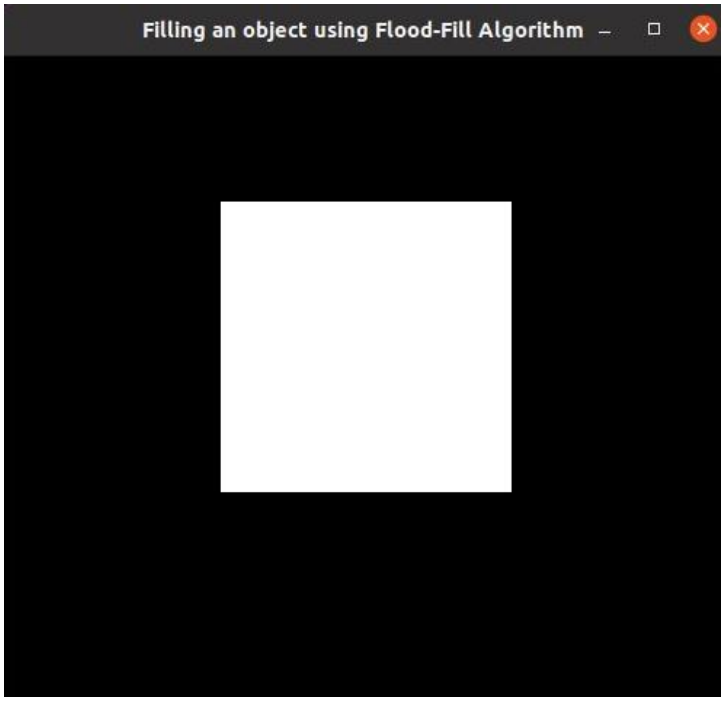
```
#include <GL/glut.h>
int ww = 500, wh = 500;
float bgCol[3] = {0.2, 0.4,0.0};
float intCol[3] = {1.0,0.0,0.0};
float fillCol[3] = {0.4,0.0,0.0};
void setPixel(int pointx, int pointy, float f[3])
{
	glBegin(GL_POINTS);
	glColor3fv(f);
	glVertex2i(pointx,pointy);
	glEnd();
	glFlush();
}
void getPixel(int x, int y, float pixels[3])
{
	glReadPixels(x,y,1.0,1.0,GL_RGB,GL_FLOAT,pixels);
}
void drawPolygon(int x1, int y1, int x2, int y2)
{
	glColor3f(1.0, 1.0, 1.0);
	glBegin(GL_POLYGON);
```

```
        glVertex2i(x1, y1);
        glVertex2i(x1, y2);
        glVertex2i(x2, y2);
        glVertex2i(x2, y1);
        glEnd();
        glFlush();
}
void display()
{
        glClearColor(0.0, 0.0 , 0.0 , 0.0);
        glClear(GL_COLOR_BUFFER_BIT);
        drawPolygon(150,400,350,200);
        glFlush();
}
void floodfill4(int x,int y,float oldcolor[3],float newcolor[3])
{
        float color[3];
        getPixel(x,y,color);
        if(color[0]==oldcolor[0] && (color[1])==oldcolor[1] &&
(color[2])==oldcolor[2])
        {
                setPixel(x,y,newcolor);
                floodfill4(x+1,y,oldcolor,newcolor);
                floodfill4(x-1,y,oldcolor,newcolor);
                floodfill4(x,y+1,oldcolor,newcolor);
                floodfill4(x,y-1,oldcolor,newcolor);
        }
```

```
}
void mouse(int btn, int state, int x, int y)
{
	if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		int xi = x;
		int yi = (wh-y);
		floodfill4(xi,yi,intCol,fillCol);
	}
}
void myinit()
{
	glViewport(0,0,ww,wh);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0.0,(GLdouble)ww,0.0,(GLdouble)wh);
	glMatrixMode(GL_MODELVIEW);
}
int main(int argc, char** argv)
{
	glutInit(&argc,argv);
	glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(ww,wh);
	glutCreateWindow("Filling an object using Flood-Fill Algorithm");
	glutDisplayFunc(display);
	myinit();
```

```
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}
```

OUTPUT:



Boundary Fill Algorithm

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);
	glColor3f(0.0, 0.0, 0.0);
	glPointSize(1.0);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(0, 500, 0, 500);
}
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);
	glBegin(GL_POINTS);
	glVertex2i(x, y);
	glEnd();
	glFlush();
}
void BoundaryFill(int x, int y, Color fillColor, Color boundaryColor)
{
	Color currentColor = getPixelColor(x, y);
	if(currentColor.r != boundaryColor.r && currentColor.g != 
boundaryColor.g && currentColor.b !=boundaryColor.b)
	{
		setPixelColor(x, y, fillColor);
		BoundaryFill(x+1, y, fillColor, boundaryColor);
		BoundaryFill(x-1, y, fillColor, boundaryColor);
		BoundaryFill(x, y+1, fillColor, boundaryColor);
		BoundaryFill(x, y-1, fillColor, boundaryColor);
	}
}
void onMouseClick(int button, int state, int x, int y)
{
	Color fillColor = {1.0f, 0.0f, 1.0f};
	Color boundaryColor = {0.0f, 0.0f, 0.0f};
```

```
	Point p = {51, 301}; //
	BoundaryFill(p.x, p.y, fillColor, boundaryColor);
}
void draw_dda(Point p1, Point p2)
{
	GLfloat dx = p2.x - p1.x;
	GLfloat dy = p2.y - p1.y;
	GLfloat x1 = p1.x;
	GLfloat y1 = p1.y;
	GLfloat step = 0;
	if(abs(dx) > abs(dy))
	{
	step = abs(dx);
	}
	else
	{
		step = abs(dy);
	}
	GLfloat xInc = dx/step;
	GLfloat yInc = dy/step;
	for(float i = 1; i <= step; i++)
	{
		glVertex2i(x1, y1);
		x1 += xInc;
		y1 += yInc;
	}
```

```
}
void draw_square(Point a, GLint length)
{
	Point b = {a.x + length, a.y},
	c = {b.x,b.y+length},
	d = {c.x-length, c.y};
	draw_dda(a, b);
	draw_dda(b, c);
	draw_dda(c, d);
	draw_dda(d, a);
}
void display(void)
{
	Point pt = {50, 300};
	GLfloat length = 150;
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_POINTS);
	draw_square(pt, length);
	glEnd();
	glFlush();
}
int main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutInitWindowSize(500, 500);
```

```
    glutInitWindowPosition(200, 200);
    glutCreateWindow("Filling an object with Boundary Fill Algorithm");
    init();
    glutDisplayFunc(display);
    glutMouseFunc(onMouseClick);
    glutMainLoop();
    return 0;
}
```

OUTPUT: