SSN COLLEGE OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UCS1712 – GRAPHICS AND MULTIMEDIA LAB

EX NO: 3 –Drawing 2D Primitives –Line – Bresenham's Algorithm

---

Name : Likhitha Verma A
Reg : 185001084
Date : 05/08/2021

1. **AIM :**

   To plot points that make up the line with endpoints (x0,y0) and (xn,yn) using Bresenham's line drawing algorithm for the following case
   (i) |m|< 1 (ii) |m|>=1

   ALGORITHM :

   1. Read two points (X_1 , Y_1) and (X_2 , Y_2) and assign (X_1 , Y_1) to (X, Y)
   2. Compute absolute difference between X and Y coordinates as dx and dy.
   3. If X_2 < X_1, make incx = -1, otherwise incx = 1.
   4. If Y_2 < Y_1, make incy = -1, otherwise incy = 1.
   5. If dx > dy (|m| < 1):
      a. Compute p = 2*dy - dx, inc1 = 2*(dy - dx), inc2 = 2*dy
      b. If p >= 0: compute Y = Y+incy and p = p+inc1, otherwise p = p+inc2
      c. Compute X = X + incx
      d. Plot the point (X, Y) using glVertex2d()
      e. Repeat steps 5b - 5d, dx times
   6. Otherwise :
      a. Compute p = 2*dx - dy, inc1 = 2*(dx - dy), inc2 = 2*dx
      b. If p >= 0: compute X = X+incx and p = p+inc1, otherwise p = p+inc2
      c. Compute Y = Y + incy
      d. Plot the point (X, Y) using glVertex2d()
      e. Repeat steps 6b - 6d, dy times

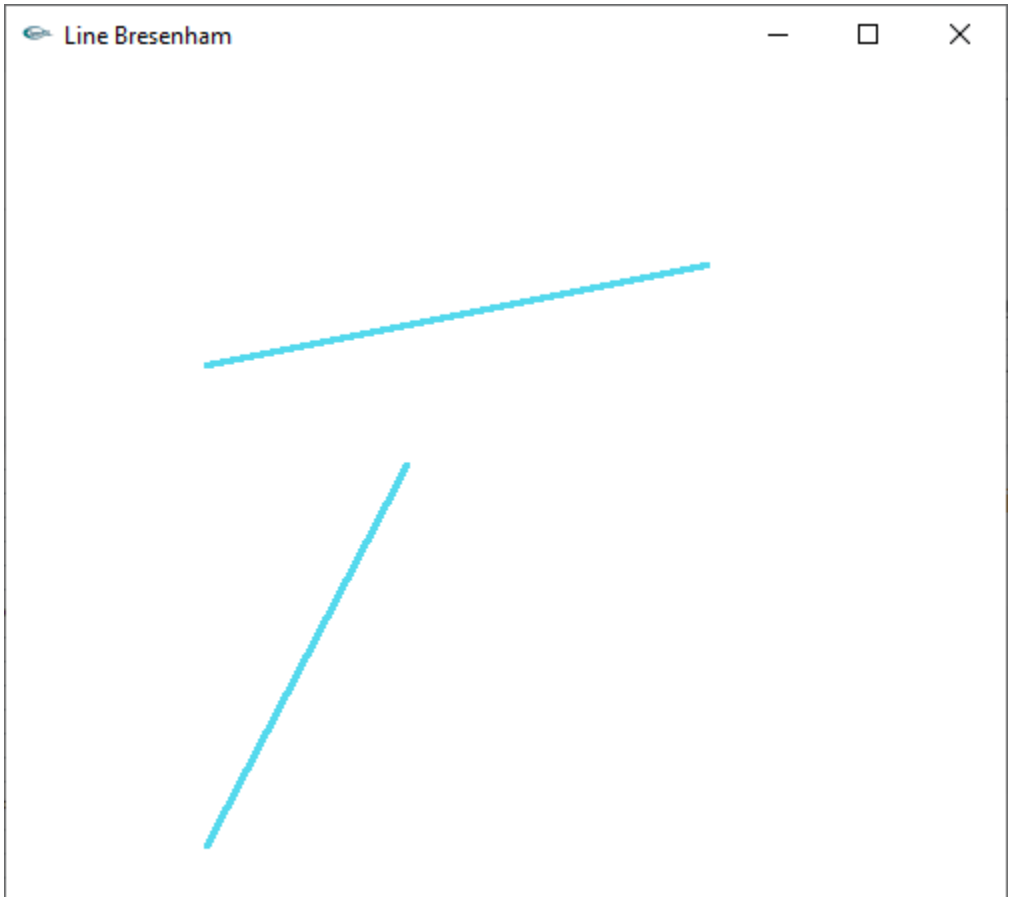
CODE :

```
#include<glut.h>
#include<math.h>
#include<stdio.h>
void myInit() {
      glClearColor(1.0, 1.0, 1.0, 0.0);
```

```
	glColor3f(0.33, 0.85, 0.93);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glPointSize(3);
	gluOrtho2D(0.0, 500.0, 0.0, 500.0);
}

void line(int x1, int y1, int x2, int y2) {
	int dx = abs(x2 - x1);
	int dy = abs(y2 - y1);
	int p, incx, incy, inc1, inc2, x, y;

	incx = 1;
	if (x2 < x1) incx = -1;
	incy = 1;
	if (y2 < y1) incy = -1;
	x = x1; y = y1;

	glBegin(GL_POINTS);
	if (dx > dy) {
		glVertex2d(x, y);
		p = 2 * dy - dx;
		inc1 = 2 * (dy - dx);
		inc2 = 2 * dy;
		for (int i = 0; i < dx; i++) {
			if (p >= 0) {
				y += incy;
				p += inc1;
			}
			else
				p += inc2;
			x += incx;
			glVertex2d(x, y);
		}

	}
	else {
		glVertex2d(x, y);
		p = 2 * dx - dy;
		inc1 = 2 * (dx - dy);
		inc2 = 2 * dx;
		for (int i = 0; i < dy; i++) {
			if (p >= 0) {
```

```
                        x += incx;
                        p += inc1;
                }
                else
                        p += inc2;
                y += incy;
                glVertex2d(x, y);
        }
    }
    glEnd();
}
void myDisplay() {
    glClear(GL_COLOR_BUFFER_BIT);

    // |m| < 1
    line(100, 350, 350, 400);
    // |m| >= 1
    line(100, 110, 200, 300);
    glFlush();
}
int main(int argc, char* argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Line Bresenham");
    glutDisplayFunc(myDisplay);
    myInit();
    glutMainLoop();
    return 1;
}
```

OUTPUT :



RESULT :

Thus plotted lines for the given cases using Bresenham's line drawing algorithm successfully.

2. AIM :

To write a C++ program using OPENGL to write any Alphabet (using sleeping, slanting, standing lines) with the help of Bresenham's line drawing algorithm.

ALGORITHM :

1. Read two points (X_1 , Y_1) and (X_2 , Y_2) and assign (X_1 , Y_1) to (X, Y)
2. Compute absolute difference between X and Y coordinates as dx and dy.
3. If $X_2 < X_1$, make incx = -1, otherwise incx = 1.
4. If $Y_2 < Y_1$, make incy = -1, otherwise incy = 1.
5. If dx > dy (|m| < 1):
    a. Compute p = 2*dy - dx, inc1 = 2*(dy - dx), inc2 = 2*dy
    b. If p >= 0: compute Y = Y+incy and p = p+inc1, otherwise p = p+inc2
    c. Compute X = X + incx
    d. Plot the point (X, Y) using glVertex2d()
    e. Repeat steps 5b - 5d, dx times

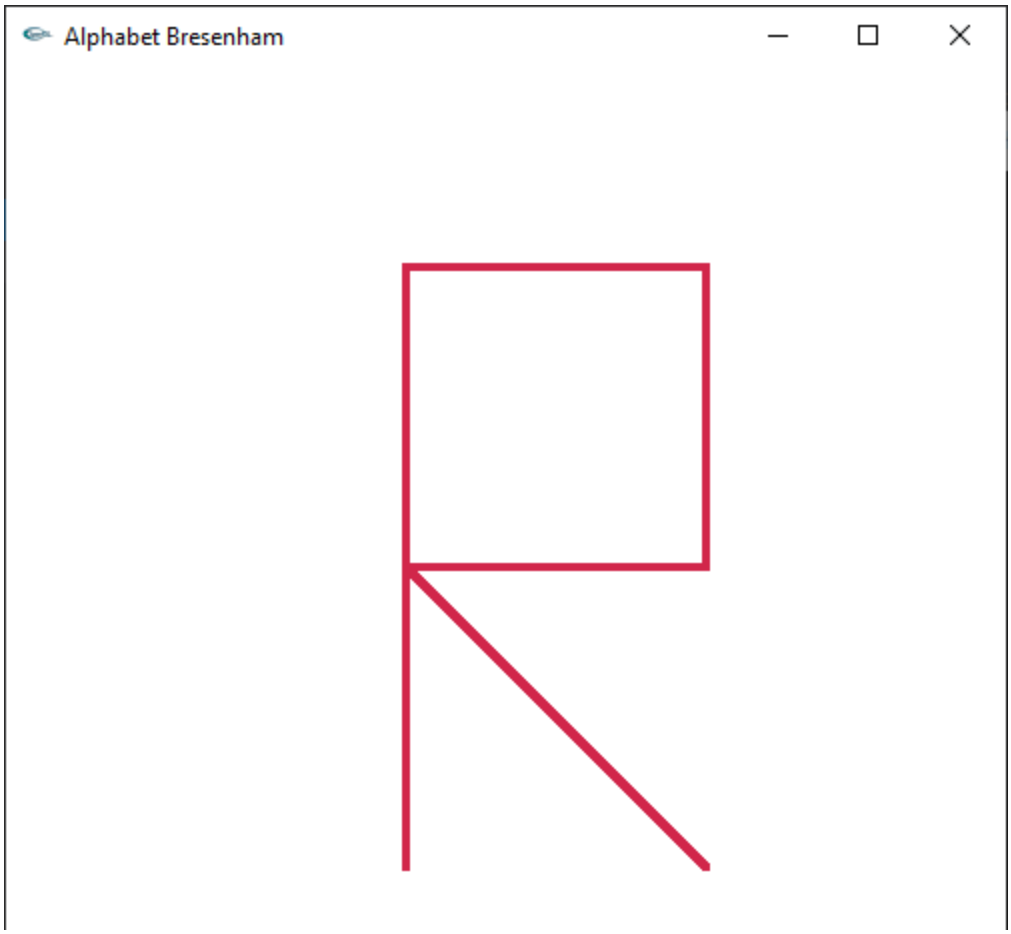
6. Otherwise :
    a. Compute p = 2*dx - dy, inc1 = 2*(dx - dy), inc2 = 2*dx
    b. If p >= 0: compute X = X+incx and p = p+inc1, otherwise p = p+inc2
    c. Compute Y = Y + incy
    d. Plot the point (X, Y) using glVertex2d()
    e. Repeat steps 6b - 6d, dy times
7. Repeat steps 1 - 6 for lines in a alphabet with different coordinates

CODE :

```
#include<glut.h>
#include<math.h>
#include<stdio.h>
void myInit() {
	glClearColor(1.0, 1.0, 1.0, 0.0);
	glColor3f(0.82, 0.15, 0.29);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glPointSize(4);
	gluOrtho2D(0.0, 500.0, 0.0, 500.0);
}
void line(int x1, int y1, int x2, int y2) {
	int dx = abs(x2 - x1);
	int dy = abs(y2 - y1);
	int p, incx, incy, inc1, inc2, x, y;

	incx = 1;
	if (x2 < x1) incx = -1;
	incy = 1;
	if (y2 < y1) incy = -1;
	x = x1; y = y1;

	glBegin(GL_POINTS);
	if (dx > dy) {
		glVertex2d(x, y);
		p = 2 * dy - dx;
		inc1 = 2 * (dy - dx);
		inc2 = 2 * dy;
		for (int i = 0; i < dx; i++) {
			if (p >= 0) {
				y += incy;
				p += inc1;
			}
```

```
                else
                        p += inc2;
                x += incx;
                glVertex2d(x, y);
        }
    }
    else {
        glVertex2d(x, y);
        p = 2 * dx - dy;
        inc1 = 2 * (dx - dy);
        inc2 = 2 * dx;
        for (int i = 0; i < dy; i++) {
            if (p >= 0) {
                x += incx;
                p += inc1;
            }
            else
                p += inc2;
            y += incy;
            glVertex2d(x, y);
        }
    }
    glEnd();
}
void myDisplay() {
    glClear(GL_COLOR_BUFFER_BIT);
    line(200, 100, 200, 400);
    line(200, 400, 350, 400);
    line(350, 400, 350, 250);
    line(200, 250, 350, 250);
    line(200, 250, 350, 100);
    glFlush();
}
int main(int argc, char* argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Alphabet Bresenham");
    glutDisplayFunc(myDisplay);
    myInit();
    glutMainLoop();
    return 1;
}
```

OUTPUT :



RESULT :

Thus implemented a C++ program using openGL to draw an alphabet R successfully.