SSN COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Name : Rahul Ram M
Reg .No : 185001121
Date : 17/08/2021

EX NO: 3

Drawing 2D Primitives –Line – Bresenham's Algorithm

1. 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

Aim:

To plot points that make up the line with endpoints (x0,y0) and (xn,yn) using Bresenham's line drawing algorithm.

Algorithm:

1. Find dy and dx and calculate m.
2. Slope m < 1:
    a. Find p = 2*dy - dx
    b. Run a while loop till x == x2
        i. Plot the pixel for the points x and y.
        ii. Increment x value
        iii. If p is less than 0, update p += 2 * dy
        iv. If p is greater than 0, increment y and update p += 2 * dy - 2 * dx.
3. If slope m >= 1:
    a. Find p = 2*dx-dy
    b. Run a while loop till y == y2
        i. Plot the pixel for the points x and y.
        ii. Increment y value
        iii. If p is less than 0, update p += 2 * dx
        iv. If p is greater than 0, increment x and update p += 2 * dx - 2 * dy

Code:

```
#include<windows.h>
#include<gl/glut.h>
#include<cstdlib>
#include<iostream>
using namespace std;

int x1[8], Y1[8], x2[8], y2[8];

void myInit()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glPointSize(1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 800.0, 0.0, 600.0);
    glClear(GL_COLOR_BUFFER_BIT);
}

void putPixel(int x, int y)
{
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}

void Bresenham(int i)
{
    int dx, dy, x, y, p;

    x = x1[i]; y = Y1[i];

    dx = x2[i] - x1[i];
    dy = y2[i] - Y1[i];



    // slope m < 1
    if (dx > dy)
    {
      p = 2 * dy - dx;
        while (x <= x2[i])
```

```
        {
            putPixel(x, y);
            x++;
            if (p < 0)
            {
                p += 2 * dy;
            }
            else
            {
                p += 2 * dy - 2 * dx;
                y++;
            }
        }
    }
    // slope m >= 1
    else
    {
      p = 2 * dx - dy;
        while (y <= y2[i])
        {
            putPixel(x, y);
            y++;
            if (p < 0)
            {
                p += 2 * dx;
            }
            else
            {
                p += 2 * dx - 2 * dy;
                x++;
            }
        }
    }

}

void display(void)
{

    for (int i = 0; i < 2; i++)
    {
        Bresenham(i);
    }
```

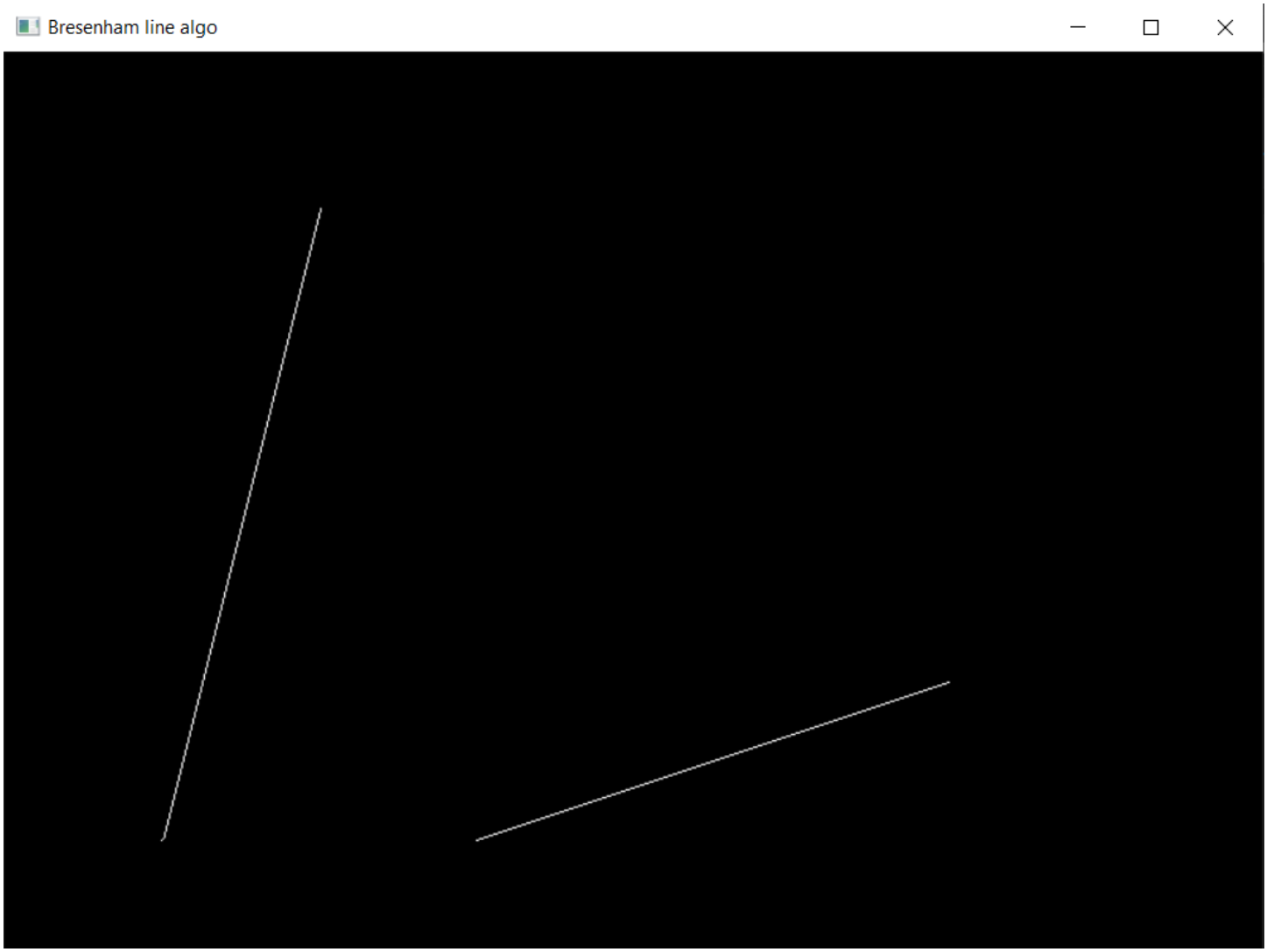
```
    glFlush();
}

int main(int argc, char** argv)
{
    for (int i = 0; i < 2; i++)
    {
        cout << "x1 : ";
        cin >> x1[i];
        cout << "y1 : ";
        cin >> Y1[i];
        cout << "x2 : ";
        cin >> x2[i];
        cout << "y2 : ";
        cin >> y2[i];
    }

    glutInit(&argc, argv);                 // Initialize GLUT
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 600);   // Set the window's initial width & height
    glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
    glutCreateWindow("Bresenham line algo"); // Create a window with the given title
    myInit();
    glutDisplayFunc(display); // Register display callback handler for window re-paint
    glutMainLoop();           // Enter the infinitely event-processing loop
    return 0;
}

// 100 100 200 500 300 100 600 200
```

Output Screenshot:



1. m >= 1 (100, 100) & (200, 500)
2. m < 1 (300, 100) & (600, 200)

Result:

Lines are drawn by using the given endpoints for all the cases and its subdivisions and by implementing the Bresenham algorithm.

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

Aim:

To write any alphabet using sleeping, slanting and standing lines with the help of Bresenham's line drawing algorithm.

Algorithm:

1. Using the Bresenham algorithm for slanting lines.
2. Creating functions for sleeping and standing lines where y value and x value will be incremented respectively at each iteration while keeping the other value constant.
3. For writing the alphabet, read input from the user about which line to use and give coordinates to draw the line.
4. Give as many lines and its type to write the alphabet.

Code:

```
#include<windows.h>
#include<gl/glut.h>
#include<cstdlib>
#include<iostream>
using namespace std;

int x1[8], Y1[8], x2[8], y2[8], type[8];
int lines = 0;

void myInit()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glPointSize(1);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0, 800.0, 0.0, 600.0);
    glClear(GL_COLOR_BUFFER_BIT);
}

void putPixel(int x, int y)
{
    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();
}
```

```
void slantingMPositive(int i)
{
    int dx, dy, x, y, p;

    x = x1[i]; y = Y1[i];

    dx = x2[i] - x1[i];
    dy = y2[i] - Y1[i];



    // slope m < 1
    if (dx > dy)
    {
        p = 2 * dy - dx;
        while (x <= x2[i])
        {
            putPixel(x, y);
            x++;
            if (p < 0)
            {
                p += 2 * dy;
            }
            else
            {
                p += 2 * dy - 2 * dx;
                y++;
            }
        }
    }
    // slope m >= 1
    else
    {
        p = 2 * dx - dy;
        while (y <= y2[i])
        {
            putPixel(x, y);
            y++;
            if (p < 0)
            {
                p += 2 * dx;
            }
```

```
            else
            {
                p += 2 * dx - 2 * dy;
                x++;
            }
        }
    }

}

void slantingMNegative(int i)
{
    int dx, dy, x, y, p;

    x = x1[i]; y = Y1[i];

    dx = x2[i] - x1[i];
    dy = Y1[i] - y2[i];

    // slope m < 1
    if (dx > dy)
    {
        p = 2 * dy - dx;
        while (x <= x2[i])
        {
            putPixel(x, y);
            x++;
            if (p < 0)
            {
                p += 2 * dy;
            }
            else
            {
                p += 2 * dy - 2 * dx;
                y--;
            }
        }
    }
    // slope m >= 1
    else
    {
        p = 2 * dx - dy;
        while (y >= y2[i])
```

```
        {
            putPixel(x, y);
            y--;
            if (p < 0)
            {
                p += 2 * dx;
            }
            else
            {
                p += 2 * dx - 2 * dy;
                x++;

            }
        }
    }
}

void sleeping(int i)
{
    int x, y;
    y = Y1[i]; x = x1[i];
    while (x <= x2[i])
    {
        putPixel(x, y);
        x++;
    }
}

void standing(int i)
{
    int x, y;
    x = x1[i]; y = Y1[i];
    while (y <= y2[i])
    {
        putPixel(x, y);
        y++;
    }
}

void display(void)
{

    for (int i = 0; i < lines; i++)
```

```
    {
        switch (type[i])
        {
        case 1: slantingMPositive(i); break;
        case 2: slantingMNegative(i); break;
        case 3: standing(i); break;
        case 4: sleeping(i); break;
        default: break;
        }
    }

    glFlush();
}

void printMenu()
{
    cout << "1. Slanting line (m+ve)" << "\n";
    cout << "2. Slanting line (m-ve)" << "\n";
    cout << "3. Standing line" << "\n";
    cout << "4. Sleeping line" << "\n";
    cout << "-1. exit" << "\n";
    cout << "Choose : " << "\n";
}

int main(int argc, char** argv)
{
    int input;
    printMenu();
    cin >> input;

    while(input != -1)
    {
        type[lines] = input;
        cout << "x1 : ";
        cin >> x1[lines];
        cout << "y1 : ";
        cin >> Y1[lines];
        cout << "x2 : ";
        cin >> x2[lines];
        cout << "y2 : ";
        cin >> y2[lines++];
        printMenu();
        cin >> input;
```

```
    }

    glutInit(&argc, argv);                 // Initialize GLUT
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 600);   // Set the window's initial width & height
    glutInitWindowPosition(50, 50); // Position the window's initial top-left corner
    glutCreateWindow("Alphabets"); // Create a window with the given title
    myInit();
    glutDisplayFunc(display); // Register display callback handler for window re-paint
    glutMainLoop();           // Enter the infinitely event-processing loop
    return 0;
}

// 1 100 100 200 400 2 200 400 300 100 4 150 250 250 250 -1
```

Output Screenshot:

1. Left slanting line (m+ve) - (100, 100) & (200, 400)
2. Right slanting line (m-ve) - (200, 400) & (300, 100)
3. Sleeping line - (150, 250) & (250, 250)

Result:

Alphabet A is drawn using two slanting lines and a sleeping line successfully with the help of Bresenham's line drawing algorithm.