Lab 4

Name : Abdelaziz Mohamed Abdelaziz.

I.D : 19015941.

Code :

Function apply algorithm for DDA draw line in which g_vertices is a global variable contain all vertices to be drawn.

```
void drawLineDDA(float x1, float y1, float x2, float y2) {
    float dx = x2 - x1;
    float dy = y2 - y1;
    int numSteps;
    if (fabs(dx) > fabs(dy)) {
        numSteps = fabs(dx) / 0.1f;
    } else {
        numSteps = fabs(dy) / 0.1f;
    }
    float xStep = dx / numSteps;
    float yStep = dy / numSteps;
    g_vertices.reserve(3 * numSteps);
    g_vertices.push_back(x1);
    g_vertices.push_back(y1);
    g_vertices.push_back(0.0f);
    float x = x1 + xStep;
    float y = y1 + yStep;
    for (int i = 1; i < numSteps; i++) {
         g_vertices.push_back(x);
         g_vertices.push_back(y);
         g_vertices.push_back(0.0f);
         x += xStep;
         y += yStep;
    }
     g_vertices.push_back(x2);
     g_vertices.push_back(y2);
     g_vertices.push_back(0.0f);
}
```

Function apply algorithm for bresenham draw line .

```
}
void drawLineBresenham(int x1, int y1, int x2, int y2) {
    int dx = abs(x2 - x1);
    int dy = abs(y2 - y1);
    int sx = (x1 < x2) ? 1 : -1;
    int sy = (y1 < y2) ? 1 : -1;
    int err = dx - dy;
    int x = x1;
    int y = y1;
    g_vertices.reserve(2 * std::max(dx, dy));
    g_vertices.push_back(x);
    g_vertices.push_back(y);
    g_vertices.push_back(0.0f);
    while (x != x2 || y != y2) {
        int e2 = 2 * err;
        if (e2 > -dy) {
            err -= dy;
            x += sx;
        }
        if (e2 < dx) {
            err += dx;
            y += sy;
        }
        g_vertices.push_back(x);
        g_vertices.push_back(y);
        g_vertices.push_back(0.0f);
    }
}
```

In this function which is used to animate as numvertices in which is total number divide by 3 as each point is consider as 3 coordinates and we loop on it and we make red and blue constant and green change every loop .

```
void drawScene(void)
{
glClear(GL_COLOR_BUFFER_BIT);
float* bufferData = (float*)glMapBuffer(GL_ARRAY_BUFFER, GL_WRITE_ONLY);
float inc = 3.0f / g_vertices.size();
int numVertices = g_vertices.size() / 3;
int startIndex = finish / 3;
for (int i = 0; i < numVertices; i++)
{
    int currentIndex = (startIndex + i) % numVertices;
    bufferData[currentIndex * 3 + g_vertices.size()] = 1;
    bufferData[currentIndex * 3 + g_vertices.size() + 1] = g;
    bufferData[currentIndex * 3 + g_vertices.size() + 2] = 1;
    g += inc;
    if (g >= 1.0f)
    {
        g = 0.0f;
    }
}
    finish += 300;
    if(finish>=g_vertices.size())
    {
        finish = 0;
    }
    glUnmapBuffer(GL_ARRAY_BUFFER);
    glBindVertexArray(vao);
    glPointSize(5.0f);
    glDrawArrays(GL_POINTS,0,g_vertices.size()/3);
    glutSwapBuffers();
}
```

```
void animate(int someValue)
{
    glutPostRedisplay();
    glutTimerFunc(60, animate, 1);
}
```

```
void resize(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 300.0, 0.0, 300.0, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
```

This function draw name is used whether it is dda or bresenham and then call the function to draw line.

```
void drawName() {
    if(useDDA)
    {
       // Draw "A"
    drawLineDDA(0.0f, 130.0f, 15.0f, 160.0f);
    drawLineDDA(15.0f, 160.0f, 30.0f, 130.0f);
    drawLineDDA(7.5f, 145.0f, 22.5f, 145.0f);


    // Draw "B"
    drawLineDDA(40.0f, 130.0f, 40.0f, 160.0f);
    drawLineDDA(40.0f, 160.0f, 50.0f,150.0f);
    drawLineDDA(50.0f, 150.0f, 40.0f, 145.0f);
    drawLineDDA(40.0f, 145.0f, 50.0f, 140.0f);
    drawLineDDA(50.0f, 140.0f, 40.0f, 130.0f);


    // Draw "D"
    drawLineDDA(60.0f, 130.0f, 60.0f, 160.0f);
    drawLineDDA(60.0f, 160.0f, 80.0f, 130.0f);
    drawLineDDA(80.0f, 130.0f, 60.0f, 130.0f);

    // Draw "E"
    drawLineDDA(90.0f, 130.0f, 90.0f, 160.0f);
    drawLineDDA(90.0f, 130.0f, 110.0f, 130.0f);
    drawLineDDA(90.0f, 160.0f, 110.0f, 160.0f);
    drawLineDDA(90.0f, 145.0f, 110.0f, 145.0f);
```

```
// Draw "L"
drawLineDDA(120.0f, 130.0f, 120.0f, 160.0f);
drawLineDDA(120.0f, 130.0f, 140.0f, 130.0f);

// Draw "-"
drawLineDDA(150.0f, 140.0f, 160.0f, 140.0f);


// Draw "A"
drawLineDDA(170.0f, 130.0f, 185.0f, 160.0f);
drawLineDDA(185.0f, 160.0f, 200.0f, 130.0f);
drawLineDDA(177.5f, 145.0f, 192.5f, 145.0f);


// Draw "Z"
drawLineDDA(210.0f, 130.0f, 230.0f, 130.0f);
drawLineDDA(210.0f, 130.0f, 230.0f, 160.0f);
drawLineDDA(230.0f, 160.0f, 210.0f, 160.0f);

// Draw "I"
drawLineDDA(240.0f, 130.0f, 260.0f, 130.0f);
drawLineDDA(240.0f, 160.0f, 260.0f, 160.0f);
drawLineDDA(250.0f, 130.0f, 250.0f, 160.0f);


// Draw "Z"
drawLineDDA(270.0f, 130.0f, 290.0f, 130.0f);
drawLineDDA(270.0f, 130.0f, 290.0f, 160.0f);
drawLineDDA(290.0f, 160.0f, 270.0f, 160.0f);
```

```
// Draw "A"
glLineWidth(5.0f);
drawLineBresenham(0.0f, 130.0f, 15.0f, 160.0f);
drawLineBresenham(15.0f, 160.0f, 30.0f, 130.0f);
drawLineBresenham(7.5f, 145.0f, 22.5f, 145.0f);


// Draw "B"
drawLineBresenham(40.0f, 130.0f, 40.0f, 160.0f);
drawLineBresenham(40.0f, 160.0f, 50.0f, 150.0f);
drawLineBresenham(50.0f, 150.0f, 40.0f, 145.0f);
drawLineBresenham(40.0f, 145.0f, 50.0f, 140.0f);
drawLineBresenham(50.0f, 140.0f, 40.0f, 130.0f);


// Draw "D"
drawLineBresenham(60.0f, 130.0f, 60.0f, 160.0f);
drawLineBresenham(60.0f, 160.0f, 80.0f, 130.0f);
drawLineBresenham(80.0f, 130.0f, 60.0f, 130.0f);

// Draw "E"
drawLineBresenham(90.0f, 130.0f, 90.0f, 160.0f);
drawLineBresenham(90.0f, 130.0f, 110.0f, 130.0f);
drawLineBresenham(90.0f, 160.0f, 110.0f, 160.0f);
drawLineBresenham(90.0f, 145.0f, 110.0f, 145.0f);

// Draw "L"
drawLineBresenham(120.0f, 130.0f, 120.0f, 160.0f);
drawLineBresenham(120.0f, 130.0f, 140.0f, 130.0f);
```

```
// Draw "-"
drawLineBresenham(150.0f, 140.0f, 160.0f, 140.0f);


// Draw "A"
drawLineBresenham(170.0f, 130.0f, 185.0f, 160.0f);
drawLineBresenham(185.0f, 160.0f, 200.0f, 130.0f);
drawLineBresenham(177.5f, 145.0f, 192.5f, 145.0f);


// Draw "Z"
drawLineBresenham(210.0f, 130.0f, 230.0f, 130.0f);
drawLineBresenham(210.0f, 130.0f, 230.0f, 160.0f);
drawLineBresenham(230.0f, 160.0f, 210.0f, 160.0f);

// Draw "I"
drawLineBresenham(240.0f, 130.0f, 260.0f, 130.0f);
drawLineBresenham(240.0f, 160.0f, 260.0f, 160.0f);
drawLineBresenham(250.0f, 130.0f, 250.0f, 160.0f);


// Draw "Z"
drawLineBresenham(270.0f, 130.0f, 290.0f, 130.0f);
drawLineBresenham(270.0f, 130.0f, 290.0f, 160.0f);
drawLineBresenham(290.0f, 160.0f, 270.0f, 160.0f);
}
```

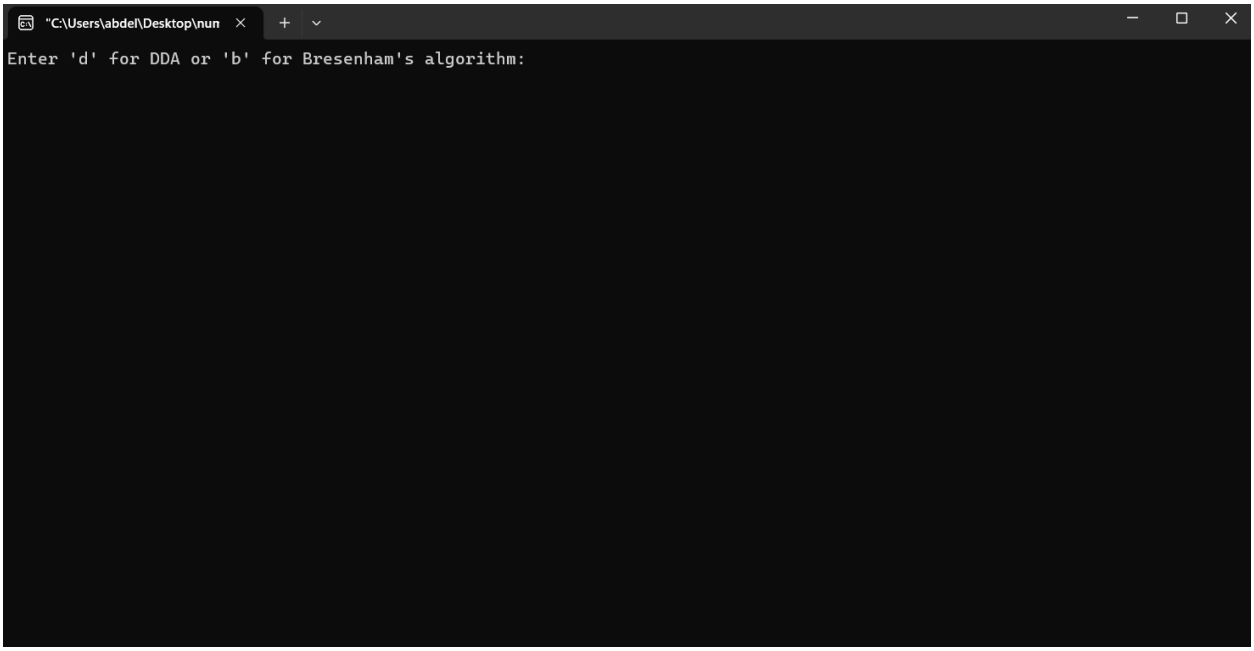
Initialize an empty array color with the same length of global vertices .

```
void  setup(void) {
    drawName();
    float colors[g_vertices.size()];
    glGenVertexArrays(1, &vao);
    glBindVertexArray(vao);
    glGenBuffers(1, &vbo);
    glBindBuffer(GL_ARRAY_BUFFER, vbo);
    glBufferData(GL_ARRAY_BUFFER, g_vertices.size()*sizeof(float)+ g_vertices.size()*sizeof(float), NULL, GL_STATIC_DRAW);
    glBufferSubData(GL_ARRAY_BUFFER, 0, g_vertices.size()*sizeof(float), &g_vertices[0]);
    glBufferSubData(GL_ARRAY_BUFFER, g_vertices.size()*sizeof(float), sizeof(colors), colors);
    glEnableClientState(GL_VERTEX_ARRAY);
    glEnableClientState(GL_COLOR_ARRAY);
    glVertexPointer(3, GL_FLOAT, 0, 0);
    glColorPointer(3, GL_FLOAT, 0, (void *)(g_vertices.size()*sizeof(float)));
    glutTimerFunc(5, animate, 1);
}
```

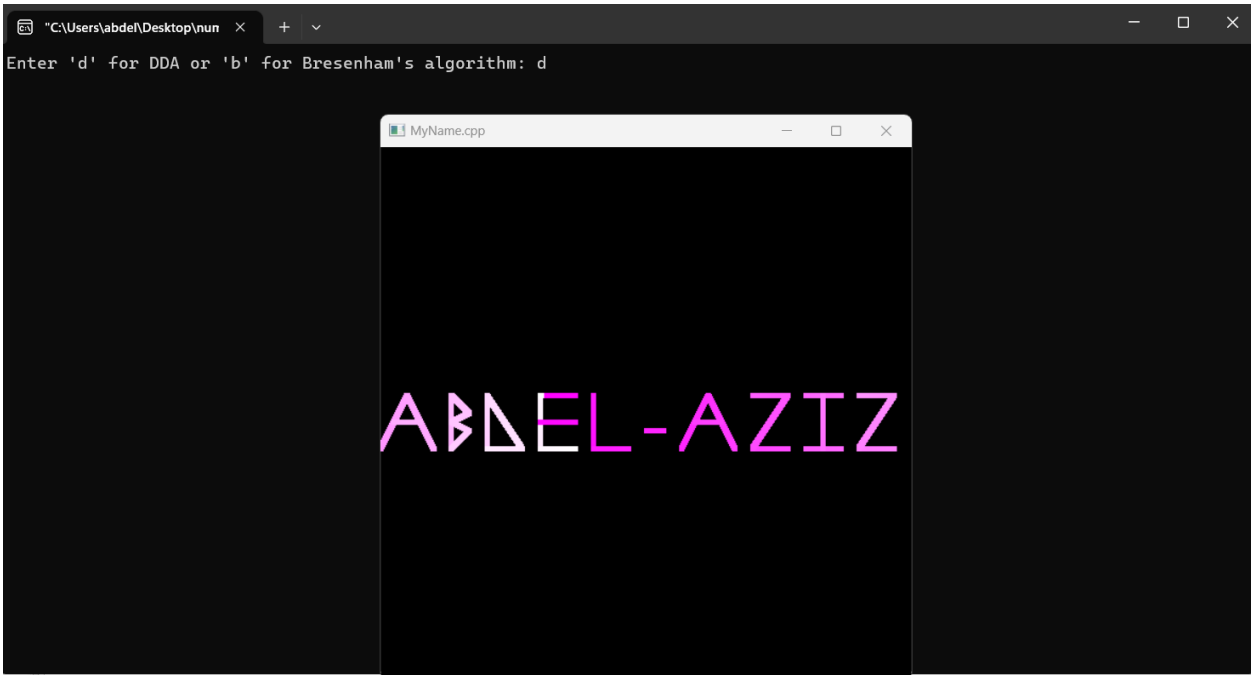
We make Boolean variable and make user whether to draw with dda or bresenham.

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    char choice;
    std::cout << "Enter 'd' for DDA or 'b' for Bresenham's algorithm: ";
    std::cin >> choice;
    if (choice == 'b') {
        useDDA = false;
    }
    glutInitContextVersion(4, 3);
    glutInitContextProfile(GLUT_COMPATIBILITY_PROFILE);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("MyName.cpp");
    glutDisplayFunc(drawScene);
    glutReshapeFunc(resize);
    glewExperimental = GL_TRUE;
    glewInit();
    setup();
    glutMainLoop();
}
```

Sample runs :


"C:\Users\abdel\Desktop\nun
Enter 'd' for DDA or 'b' for Bresenham's algorithm:

DDA:



Bresenham: