Alexandria University

Faculty of Engineering

Computer Graphics

Lab #4 OpenGL

Name: Toka Ashraf Abo Elwafa Ahmed

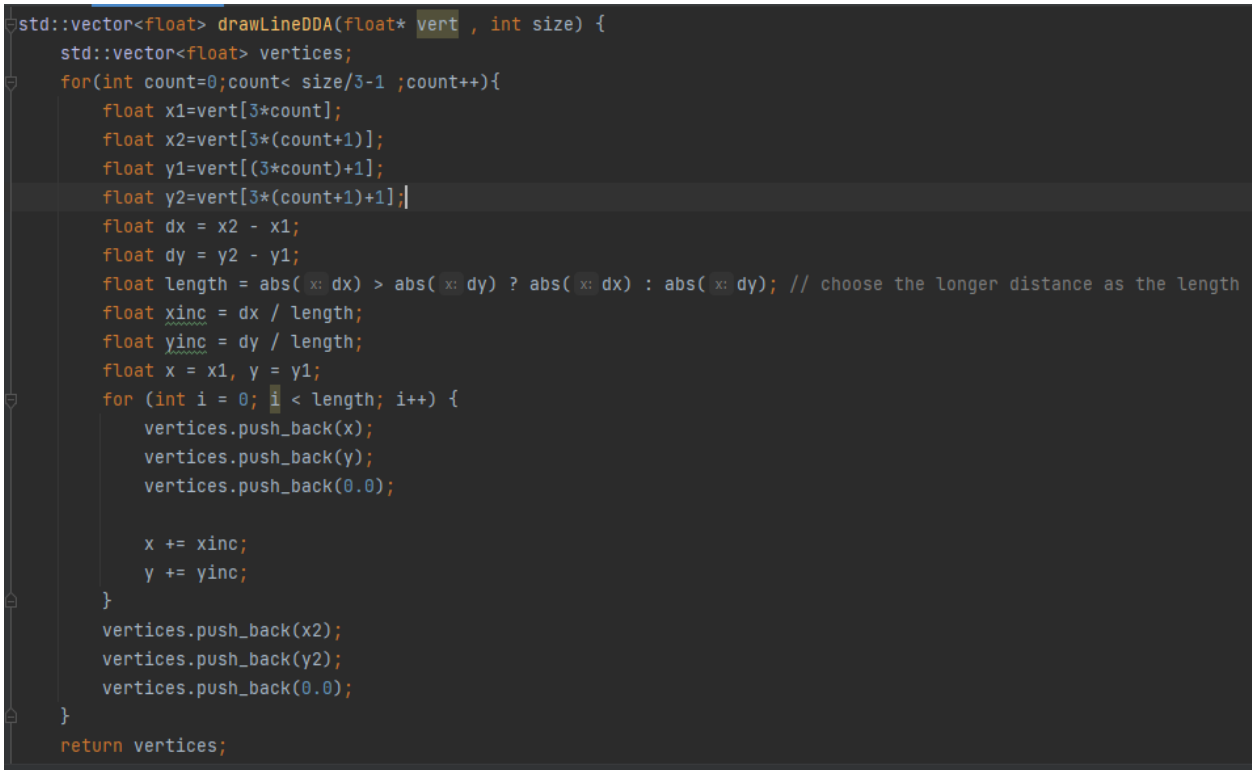
ID: 19015539

➢ Problem Statement:

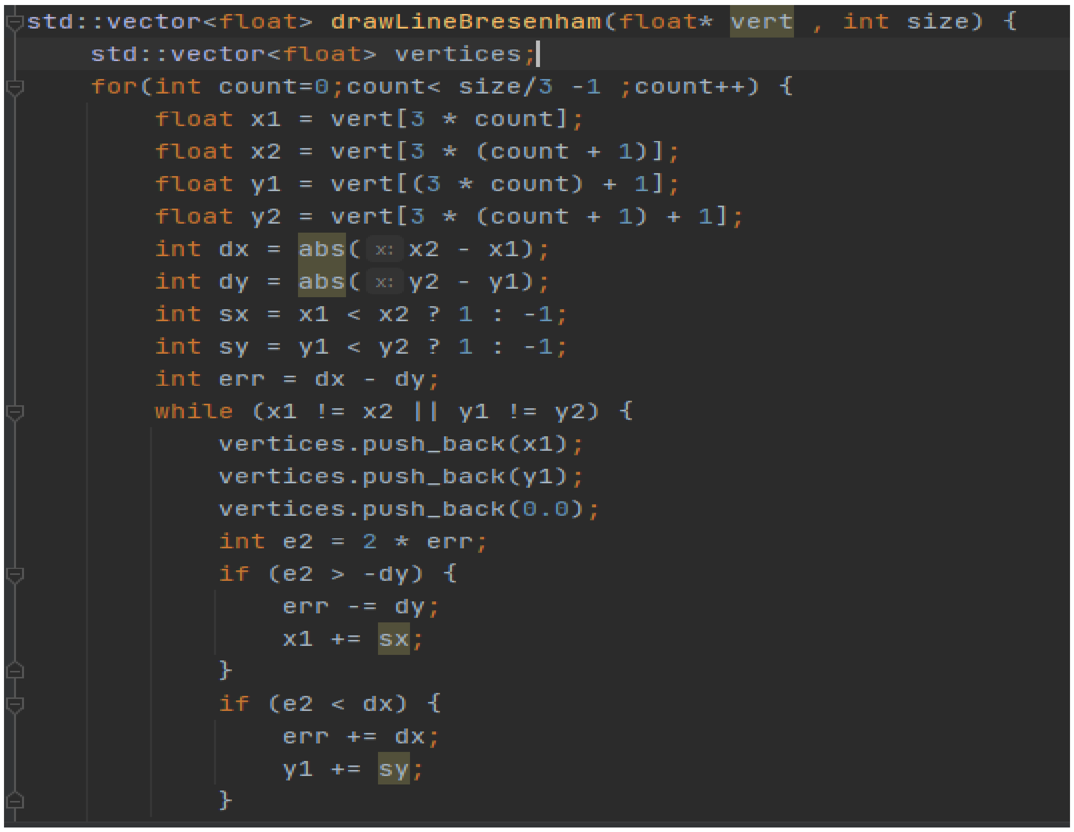
You should implement an OpenGL application that writes your first name using line drawing algorithms (DDA, Bresenham) (your drawing should include all possible slopes). You should use VAOs to store vertices data. set point size to 5.

➢ Code Flow:

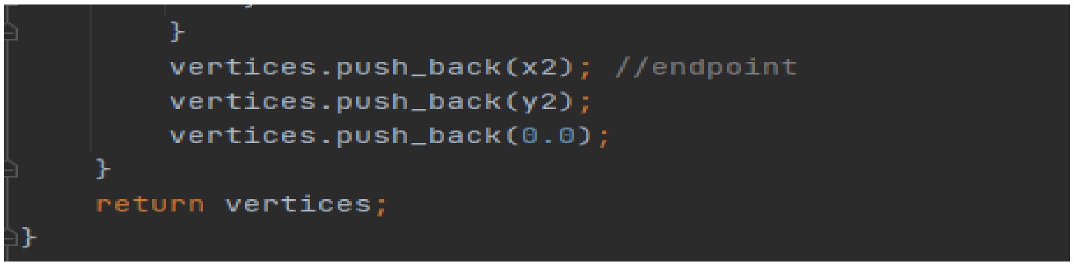
1- DDA and Bresenham algorithms were implemented to take array of vertices and its size then loop around it to draw line using these algorithms between each consecutive two vertices by storing all points from first vertex to its following one in vector and return it to draw each letter.



```
std::vector<float> drawLineDDA(float* vert , int size) {
    std::vector<float> vertices;
    for(int count=0;count< size/3-1 ;count++){
        float x1=vert[3*count];
        float x2=vert[3*(count+1)];
        float y1=vert[(3*count)+1];
        float y2=vert[3*(count+1)+1];
        float dx = x2 - x1;
        float dy = y2 - y1;
        float length = abs( x: dx) > abs( x: dy) ? abs( x: dx) : abs( x: dy); // choose the longer distance as the length
        float xinc = dx / length;
        float yinc = dy / length;
        float x = x1, y = y1;
        for (int i = 0; i < length; i++) {
            vertices.push_back(x);
            vertices.push_back(y);
            vertices.push_back(0.0);

            x += xinc;
            y += yinc;
        }
        vertices.push_back(x2);
        vertices.push_back(y2);
        vertices.push_back(0.0);
    }
    return vertices;
```

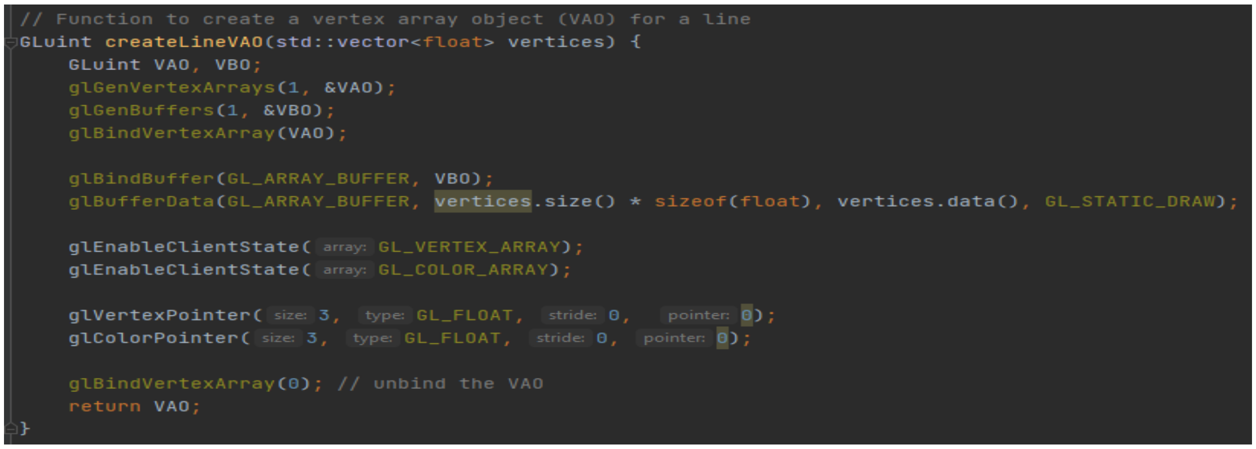


```
std::vector<float> drawLineBresenham(float* vert , int size) {
    std::vector<float> vertices;
    for(int count=0;count< size/3 -1 ;count++) {
        float x1 = vert[3 * count];
        float x2 = vert[3 * (count + 1)];
        float y1 = vert[(3 * count) + 1];
        float y2 = vert[3 * (count + 1) + 1];
        int dx = abs( x: x2 - x1);
        int dy = abs( x: y2 - y1);
        int sx = x1 < x2 ? 1 : -1;
        int sy = y1 < y2 ? 1 : -1;
        int err = dx - dy;
        while (x1 != x2 || y1 != y2) {
            vertices.push_back(x1);
            vertices.push_back(y1);
            vertices.push_back(0.0);
            int e2 = 2 * err;
            if (e2 > -dy) {
                err -= dy;
                x1 += sx;
            }
            if (e2 < dx) {
                err += dx;
                y1 += sy;
            }
```



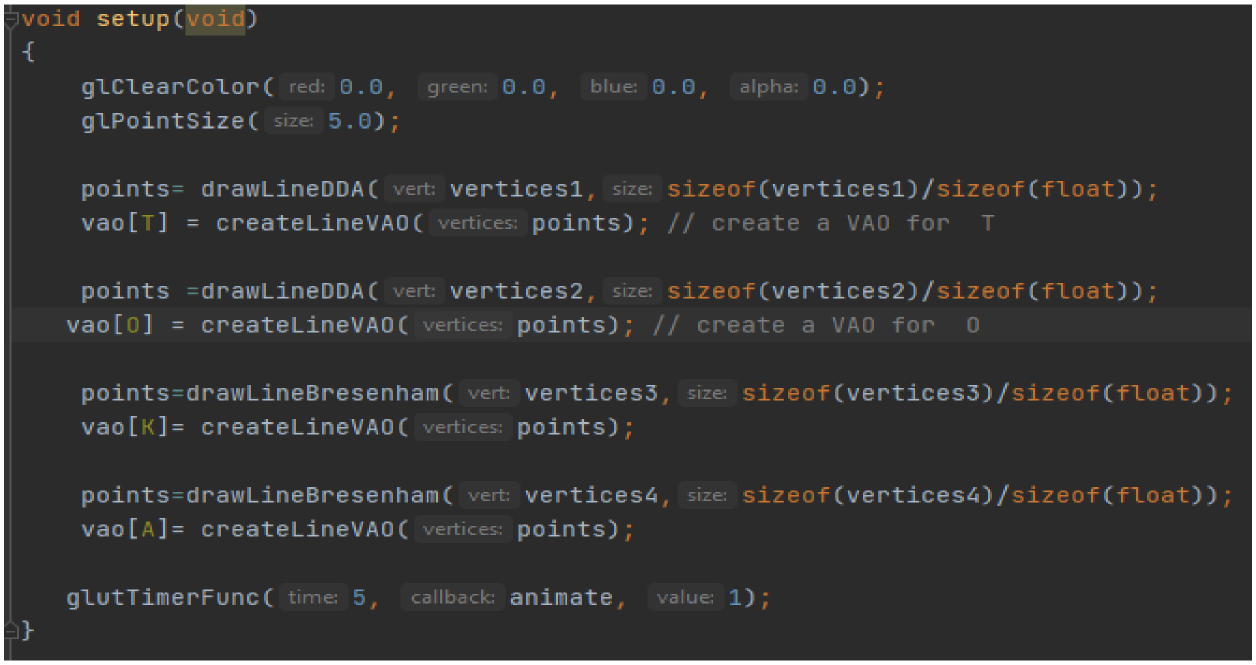
```
        }
        vertices.push_back(x2); //endpoint
        vertices.push_back(y2);
        vertices.push_back(0.0);
    }
    return vertices;
}
```

2- After getting points vector, it will be stored using VBO (vertex buffer object) on the GPU memory.



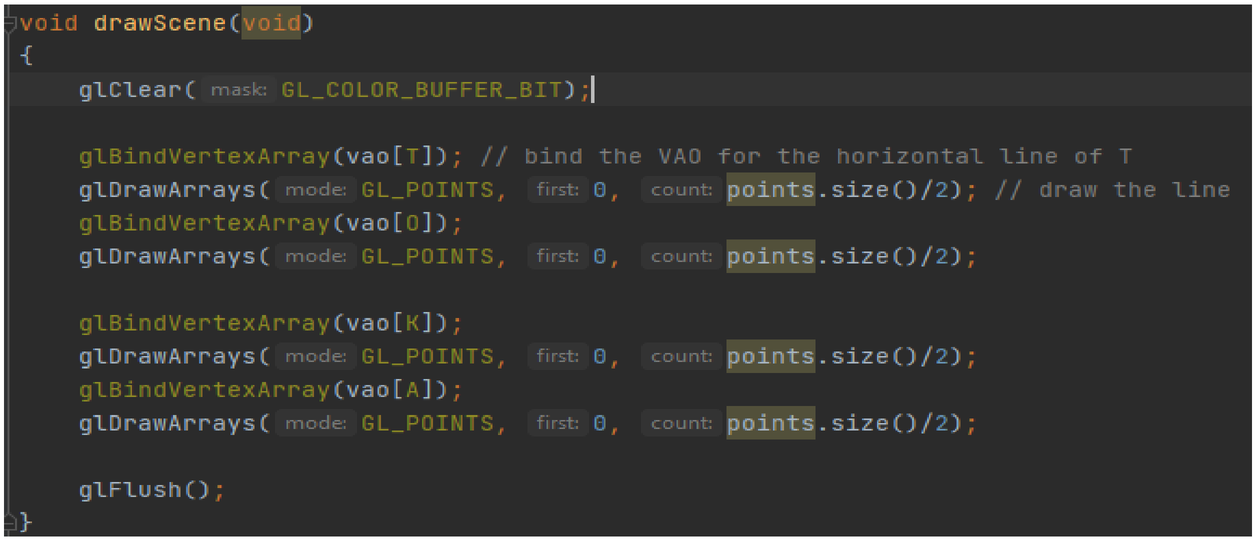
```
// Function to create a vertex array object (VAO) for a line
GLuint createLineVAO(std::vector<float> vertices) {
    GLuint VAO, VBO;
    glGenVertexArrays(1, &VAO);
    glGenBuffers(1, &VBO);
    glBindVertexArray(VAO);

    glBindBuffer(GL_ARRAY_BUFFER, VBO);
    glBufferData(GL_ARRAY_BUFFER, vertices.size() * sizeof(float), vertices.data(), GL_STATIC_DRAW);

    glEnableClientState( array: GL_VERTEX_ARRAY);
    glEnableClientState( array: GL_COLOR_ARRAY);

    glVertexPointer( size: 3, type: GL_FLOAT, stride: 0, pointer: 0);
    glColorPointer( size: 3, type: GL_FLOAT, stride: 0, pointer: 0);

    glBindVertexArray(0); // unbind the VAO
    return VAO;
}
```

3- An array of VAOs of size 4 is made to draw each letter from the four letters using it.



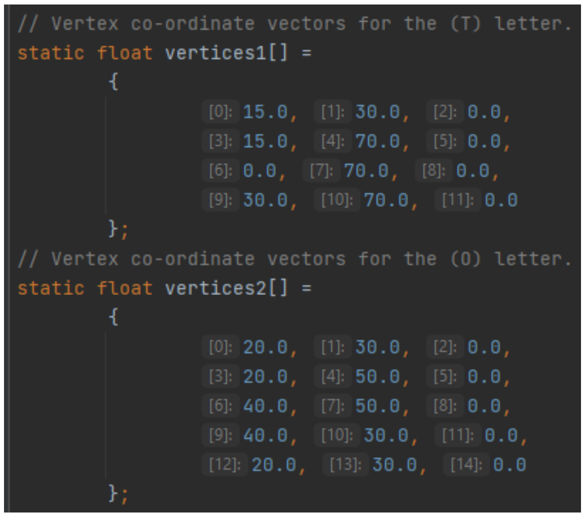
```
void setup(void)
{
    glClearColor( red: 0.0,  green: 0.0,  blue: 0.0,  alpha: 0.0);
    glPointSize( size: 5.0);

    points= drawLineDDA( vert: vertices1, size: sizeof(vertices1)/sizeof(float));
    vao[T] = createLineVAO( vertices: points); // create a VAO for  T

    points =drawLineDDA( vert: vertices2, size: sizeof(vertices2)/sizeof(float));
   vao[O] = createLineVAO( vertices: points); // create a VAO for  O

    points=drawLineBresenham( vert: vertices3, size: sizeof(vertices3)/sizeof(float));
    vao[K]= createLineVAO( vertices: points);

    points=drawLineBresenham( vert: vertices4, size: sizeof(vertices4)/sizeof(float));
    vao[A]= createLineVAO( vertices: points);

   glutTimerFunc( time: 5,  callback: animate,  value: 1);
}
```

4- In drawScene() function, binding the VAO for each letter will take place then drawing the shape using glDrawArrays by taking GL_POINTS as a primitive to draw each point from the vector and also take the size of the vector by 2 as a count of the points that will be drawn.

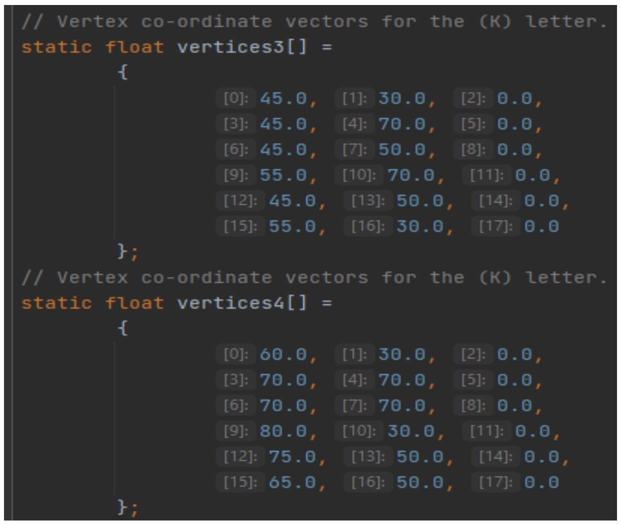


```
void drawScene(void)
{
    glClear( mask: GL_COLOR_BUFFER_BIT);

    glBindVertexArray(vao[T]); // bind the VAO for the horizontal line of T
    glDrawArrays( mode: GL_POINTS,  first: 0,  count: points.size()/2); // draw the line
    glBindVertexArray(vao[O]);
    glDrawArrays( mode: GL_POINTS,  first: 0,  count: points.size()/2);

    glBindVertexArray(vao[K]);
    glDrawArrays( mode: GL_POINTS,  first: 0,  count: points.size()/2);
    glBindVertexArray(vao[A]);
    glDrawArrays( mode: GL_POINTS,  first: 0,  count: points.size()/2);

    glFlush();
}
```

5- Four array of vertices were made to each letter to enter it to the DDA or Bresenham algorithms to get the vector of the points.

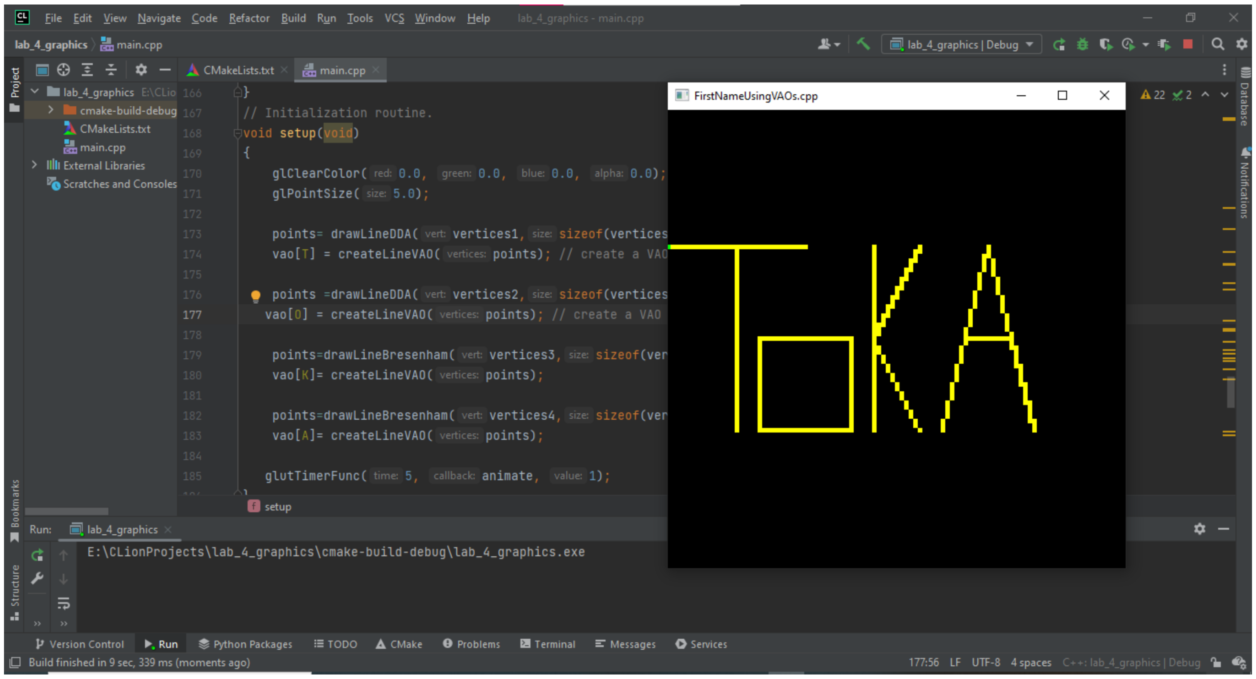


```
// Vertex co-ordinate vectors for the (T) letter.
static float vertices1[] =
        {
                [0]: 15.0,  [1]: 30.0,  [2]: 0.0,
                [3]: 15.0,  [4]: 70.0,  [5]: 0.0,
                [6]: 0.0,  [7]: 70.0,  [8]: 0.0,
                [9]: 30.0,  [10]: 70.0,  [11]: 0.0
        };
// Vertex co-ordinate vectors for the (O) letter.
static float vertices2[] =
        {
                [0]: 20.0,  [1]: 30.0,  [2]: 0.0,
                [3]: 20.0,  [4]: 50.0,  [5]: 0.0,
                [6]: 40.0,  [7]: 50.0,  [8]: 0.0,
                [9]: 40.0,  [10]: 30.0,  [11]: 0.0,
                [12]: 20.0,  [13]: 30.0,  [14]: 0.0
        };
```



```
// Vertex co-ordinate vectors for the (K) letter.
static float vertices3[] =
        {
                [0]: 45.0,  [1]: 30.0,  [2]: 0.0,
                [3]: 45.0,  [4]: 70.0,  [5]: 0.0,
                [6]: 45.0,  [7]: 50.0,  [8]: 0.0,
                [9]: 55.0,  [10]: 70.0,  [11]: 0.0,
                [12]: 45.0,  [13]: 50.0,  [14]: 0.0,
                [15]: 55.0,  [16]: 30.0,  [17]: 0.0
        };
// Vertex co-ordinate vectors for the (K) letter.
static float vertices4[] =
        {
                [0]: 60.0,  [1]: 30.0,  [2]: 0.0,
                [3]: 70.0,  [4]: 70.0,  [5]: 0.0,
                [6]: 70.0,  [7]: 70.0,  [8]: 0.0,
                [9]: 80.0,  [10]: 30.0,  [11]: 0.0,
                [12]: 75.0,  [13]: 50.0,  [14]: 0.0,
                [15]: 65.0,  [16]: 50.0,  [17]: 0.0
        };
```

➢ **Screenshots:**



```
}
// Initialization routine.
void setup(void)
{
    glClearColor( red: 0.0,  green: 0.0,  blue: 0.0,  alpha: 0.0);
    glPointSize( size: 5.0);

    points= drawLineDDA( vert: vertices1, size: sizeof(vertices
    vao[T] = createLineVAO( vertices: points); // create a VAO

    points =drawLineDDA( vert: vertices2, size: sizeof(vertices
   vao[O] = createLineVAO( vertices: points); // create a VAO

    points=drawLineBresenham( vert: vertices3, size: sizeof(ver
    vao[K]= createLineVAO( vertices: points);

    points=drawLineBresenham( vert: vertices4, size: sizeof(ver
    vao[A]= createLineVAO( vertices: points);

   glutTimerFunc( time: 5,  callback: animate,  value: 1);
```

```
E:\CLionProjects\lab_4_graphics\cmake-build-debug\lab_4_graphics.exe
```

Note: this code was made to draw the first two letters (TO) using DDA algorithm and the second two letters (KA) using Bresenham algorithm as it was not determined.