Lab 4

Computer Graphics

Name	ID
Ahmed Adel Abudef	19015264

Table Of Contents:

I. Snippets

```
void dda(float x1, float y1, float x2, float y2, std::vector<float> &points) {
    // Calculate the change in x and y
    float dx = x2 - x1;
    float dy = y2 - y1;
    // Calculate the number of steps
    int steps = std::max(abs( x: dx), abs( x: dy));
    // Calculate the increments for x and y
    float x_inc = dx / steps;
    float y_inc = dy / steps;
    // Draw the line
    for (int i = 0; i < steps; i++) {
        points.push_back(x1);
        points.push_back(y1);
        x1 += x_inc;
        y1 += y_inc;
    }
}
```

```
void bresenham(float x1, float y1, float x2, float y2, std::vector<float> &points) {
    // Calculate the change in x and y
    int dx = abs( x: x2 - x1);
    int dy = abs( x: y2 - y1);
    // Determine the direction of the x and y axis
    int sx = (x1 < x2) ? 1 : -1;
    int sy = (y1 < y2) ? 1 : -1;
    // Calculate the error term
    int err = dx - dy;
    // Draw the line
    while (x1 != x2 || y1 != y2) {
        points.push_back(x1);
        points.push_back(y1);
        int e2 = 2 * err;
        if (e2 > -dy) {
            err -= dy;
            x1 += sx;
        }
        if (e2 < dx) {
            err += dx;
            y1 += sy;
        }
    }
}
```

```
//A
{ x: 100.0f,  y: 150.0f},
{ x: 125.0f,  y: 250.0f},
{ x: 125.0f,  y: 250.0f},
{ x: 150.0f,  y: 150.0f},
{ x: 135.0f,  y: 200.0f},
{ x: 115.0f,  y: 200.0f},
//H
{ x: 175.0f,  y: 250.0f},
{ x: 175.0f,  y: 150.0f},
{ x: 225.0f,  y: 250.0f},
{ x: 225.0f,  y: 150.0f},
{ x: 175.0f,  y: 200.0f},
{ x: 225.0f,  y: 200.0f},
//M
{ x: 250.0f,  y: 150.0f},
{ x: 250.0f,  y: 250.0f},
{ x: 250.0f,  y: 250.0f},
{ x: 275.0f,  y: 150.0f},
{ x: 275.0f,  y: 150.0f},
{ x: 300.0f,  y: 250.0f},
{ x: 300.0f,  y: 250.0f},
{ x: 300.0f,  y: 150.0f},
//E
{ x: 325.0f,  y: 150.0f},
{ x: 325.0f,  y: 250.0f},
{ x: 325.0f,  y: 250.0f},
{ x: 375.0f,  y: 250.0f},
{ x: 325.0f,  y: 200.0f},
{ x: 375.0f,  y: 200.0f},
{ x: 325.0f,  y: 150.0f},
{ x: 375.0f,  y: 150.0f},
//D
{ x: 400.0f,  y: 150.0f},
{ x: 400.0f,  y: 250.0f},
{ x: 400.0f,  y: 250.0f},
{ x: 450.0f,  y: 225.0f},
{ x: 450.0f,  y: 225.0f},
{ x: 450.0f,  y: 175.0f},
{ x: 450.0f,  y: 175.0f},
{ x: 400.0f,  y: 150.0f},
```

```
    // Initialize the points vector
    std::vector<float> points;
    // Iterate over the lines and add their points to the points vector
    for (int i = 0; i < lines.size() - 1; i = i + 2) {
        // Use DDA algorithm for the first 4 lines
        if (user_choice == 1) {
            dda( x1: lines[i].first,  y1: lines[i].second,  x2: lines[i + 1].first,  y2: lines[i + 1].second,  &: poin
        }
        // Use Bresenham algorithm for the last 6 lines
        else{
            bresenham( x1: lines[i].first,  y1: lines[i].second,  x2: lines[i + 1].first,  y2: lines[i + 1].second,  &
        }
    }


    for (int j = 0; j < points.size() - 1; j = j + 2) {
        glBegin( mode: GL_POINTS);
        glVertex3f( x: points[j],  y: points[j + 1],  z: 0.0f);
        glEnd();
    }

    // Create a vertex array object (VAO)
    unsigned int vao;
    glGenVertexArrays( n: 1,  arrays: &vao);
    glBindVertexArray( array: vao);
    // Create a vertex buffer object (VBO)
    unsigned int vbo;
    glGenBuffers( n: 1,  buffers: &vbo);
    glBindBuffer( target: GL_ARRAY_BUFFER,  buffer: vbo);
// Copy the points into the VBO
    glBufferData( target: GL_ARRAY_BUFFER,  size: points.size() * sizeof(float),  data: points.data(),  usage: GL_STATIC_
// Specify the vertex attributes
    glEnableVertexAttribArray( index: 0);
    glVertexAttribPointer( index: 0,  size: 2,  type: GL_FLOAT,  normalized: GL_FALSE,  stride: 0,  pointer: NULL);
    glEnableVertexAttribArray( index: 1);

// Swap buffers
    glutSwapBuffers();
    glBindVertexArray( array: 0);
    glFlush();
```

```cpp
void setup(void) {
    glClearColor( red: 1.0,  green: 1.0,  blue: 1.0,  alpha: 0.0);
}

// Reshape function
void reshape(int w, int h) {
    glViewport( x: 0,  y: 0,  width: w,  height: h);
    glMatrixMode( mode: GL_PROJECTION);
    glLoadIdentity();
    glOrtho( left: 0.0,  right: 800,  bottom: 0.0,  top: 700,  zNear: -1.0,  zFar: 1.0);
    glMatrixMode( mode: GL_MODELVIEW);
    glLoadIdentity();
}

int main(int argc, char **argv) {
    glutInit( pargc: &argc, argv);
// Initialize GLUT and GLEW
    glutInitContextVersion( majorVersion: 4,  minorVersion: 3);
    glutInitContextProfile( profile: GLUT_COMPATIBILITY_PROFILE);
    std::cout << "Which line type do you want\n1) DDA\n2) Bresenham\n>> ";
    std::cin >> user_choice;
    glutInitDisplayMode( displayMode: GLUT_SINGLE | GLUT_RGBA);
    glutInitWindowSize( width: 800,  height: 700);
    glutInitWindowPosition( x: 100,  y: 100);
    glutCreateWindow( title: "name.cpp");
    glutDisplayFunc( callback: display);
    glutReshapeFunc( callback: reshape);
    glewExperimental = GL_TRUE;
    glewInit();
    setup();
    glutMainLoop();
}
```

II. Sample Runs