COMPUTER GRAPHICS

LAB#4

RAMY AHMED EL SAYED

19015649

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.

Bonus part:

Apply simple text gradient animation to drawn lines. See video for better understanding. You can use VAOs, color animation as references.

CODE DESCRIPTION

The code is mainly composed of 3 major parts:

- DDA Function
- Bersenham Function
- Setup Function

Note: Before starting, a vertex array containing the corner points of our letters is constructed and is passed to our line drawing algorithms at the beginning of the setup to generate our points.

Std::vector is used instead of a normal array so that we can dynamically allocate data to it .

DDA

```
void DDA(int X1, int Y1, int X2, int Y2) {
          int I;
          float Length;
          float X, Y, Xinc, Yinc;
          Length = abs(X2 - X1);
          if (abs(Y2 - Y1) > Length)
                    Length = abs(Y2 - Y1);
          Xinc = (X2 - X1) / (Length);
          Yinc = (Y2 - Y1) / (Length);
          X = X1;
          Y = Y1;
          for (I = 0; I < Length; I++) {
                    points.push_back(X);
                    points.push_back(Y);
                    points.push_back(0.0);
                    colors2.push_back(1.0);
                    colors2.push_back(0.0);
                    colors2.push_back(0.0);
                    size += 3;
                    X = X + Xinc;
                    Y = Y + Yinc;
          }
}
```

Simple implementation of the DDA function that follows the lecture pseudo code.

BERSENHAM

```
void Bersenham(int x1, int y1, int x2, int y2) {
                    int dx, dy, i, e;
                    int incx, incy, inc1, inc2;
                    int x, y;

                    dx = x2 - x1;
                    dy = y2 - y1;

                    if (dx < 0) dx = -dx;
                    if (dy < 0) dy = -dy;
                    incx = 1;
                    if (x2 < x1) incx = -1;
                    else if (x2 == x1) incx = 0;
                    incy = 1;
                    if (y2 < y1) incy = -1;
                    else if (y2 == y1) incy = 0;
                    x = x1; y = y1;
                    if (dx > dy) {
                                        points.push_back(x);
                                        points.push_back(y);
                                        points.push_back(0.0);
                                        colors2.push_back(1.0);
                                        colors2.push_back(0.0);
                                        colors2.push_back(0.0);
                                        size += 3;
                                        e = 2 * dy - dx;
                                        inc1 = 2 * (dy - dx);
                                        inc2 = 2 * dy;
                                        for (i = 0; i < dx; i++) {
                                                            if (e >= 0) {
                                                                                y += incy;
                                                                                e += inc1;
                                                            }
                                                            else
                                                                                e += inc2;
                                                            x += incx;
                                                            points.push_back(x);
                                                            points.push_back(y);
                                                            points.push_back(0.0);
                                                            colors2.push_back(1.0);
                                                            colors2.push_back(0.0);
                                                            colors2.push_back(0.0);
                                                            size += 3;
                                        }

                    }
                    else {
                                        points.push_back(x);
                                        points.push_back(y);
                                        points.push_back(0.0);
                                        colors2.push_back(1.0);
                                        colors2.push_back(0.0);
                                        colors2.push_back(0.0);
                                        size += 3;
                                        e = 2 * dx - dy;
                                        inc1 = 2 * (dx - dy);
                                        inc2 = 2 * dx;
                                        for (i = 0; i < dy; i++) {
                                                            if (e >= 0) {
                                                                                x += incx;
                                                                                e += inc1;
                                                            }
                                                            else
                                                                                e += inc2;
                                                            y += incy;
                                                            points.push_back(x);
                                                            points.push_back(y);
                                                            points.push_back(0.0);
                                                            colors2.push_back(1.0);
                                                            colors2.push_back(0.0);
                                                            colors2.push_back(0.0);
                                                            size += 3;
                                        }
                    }
}
```

SETUP

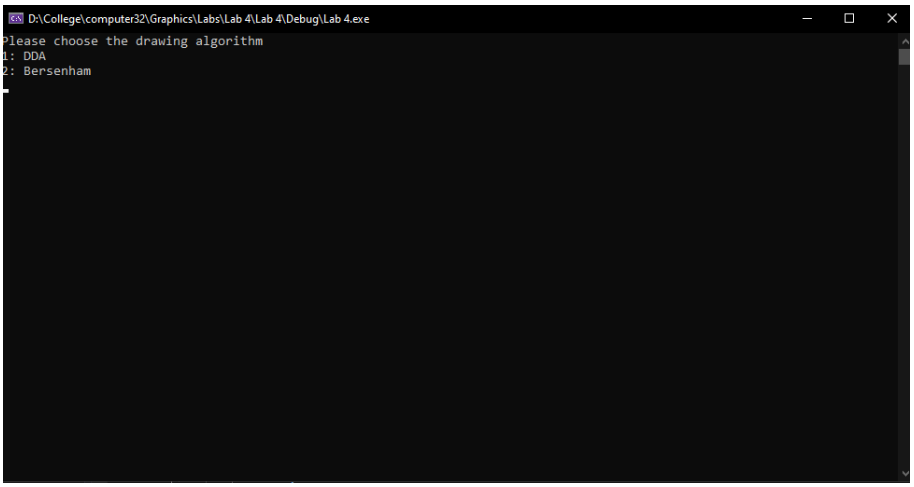
```
void setup(void)
{
	for (int i = 0; i < 60; i += 3) {
		if (i == 15 || i == 24 ||  i == 30 || i == 45) { continue; }
		if(userInput == dda)
			DDA(vertices2[i], vertices2[i + 1], vertices2[i + 3], vertices2[i + 4]);
		else if(userInput == bersen)
			Bersenham(vertices2[i], vertices2[i + 1], vertices2[i + 3], vertices2[i + 4]);
	}
	glClearColor(1.0, 1.0, 1.0, 0.0);

	glGenVertexArrays(2, vao); // Generate VAO ids.

	glBindVertexArray(vao[TRIANGLE]);

	glGenBuffers(1, buffer);

	// Bind vertex buffer and reserve space.
	glBindBuffer(GL_ARRAY_BUFFER, buffer[VERTICES]);
	glBufferData(GL_ARRAY_BUFFER, size * sizeof(float) + size * sizeof(float), NULL, GL_STATIC_DRAW);

	// Copy vertex coordinates data into first half of vertex buffer.
	glBufferSubData(GL_ARRAY_BUFFER, 0, size * sizeof(float), (float *) &points[0]);

	// Copy vertex color data into second half of vertex buffer.
	glBufferSubData(GL_ARRAY_BUFFER, size * sizeof(float), size * sizeof(float), (float*) &colors2[0]);

	// Enable two vertex arrays: co-ordinates and color.
	glEnableClientState(GL_VERTEX_ARRAY);
	glEnableClientState(GL_COLOR_ARRAY);

	// Specify vertex and color pointers to the start of the respective data.
	glVertexPointer(3, GL_FLOAT, 0, 0);
	glColorPointer(3, GL_FLOAT, 0, (void*)(size * sizeof(float)));
	// END bind VAO id vao[TRIANGLE].
	glutTimerFunc(5, animate, 1);
}
```

The user choice is first checked to see whether he chose DDA or LDA, and based on it the corresponding algorithm is chosen.

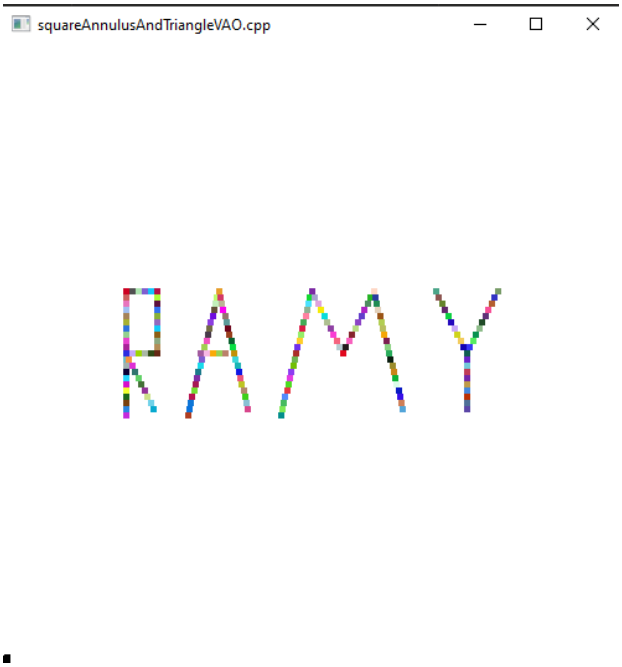
Note: some points in the array are skipped to avoid interleaving connections between the letters.

EXAMPLE OF RUNNING CODE

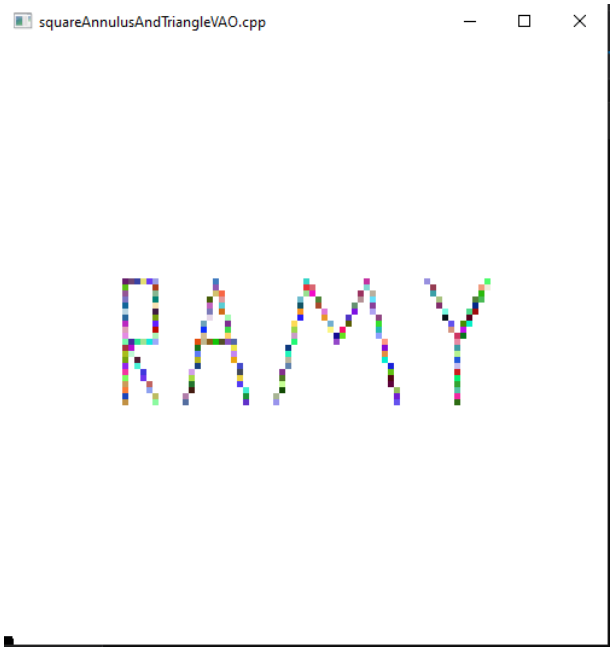
MAIN SCREEN



DDA



BERSENHAM



Note: The animation is done on each point since we used the primitive GL_POINTS

CHALLENGES

The main challenge was dealing with VAOs and vector pointers.