JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY (JKUAT)

DEPT: COMPUTING & INFORMATION TECHNOLOGY

COURSE UNIT: COMPUTER GRAPHICS

COURSE CODE: ICS 2311

PRESENTED BY:

KIARIE ZAWADI MUTHONI SCT211-0462/2022

HILDA MWANGI SCT211-0026/2022

NEEMA OGAO SCT211-0086/2022

NJOROGE KANYAGIA SCT211-0009/2022

1.Using OpenGL draw a filled polygon with the following dimensions (8,4;2,4;0,8;3,12;7,12;10,8) hint (GL POLYGON function) might be useful

a. Write a function to fill the polygon above in Red (#FF0000.)

b. write program to scale up (scaling) the polygon by a factor of 2

c. Write a procedure to fill the interior of a given polygon with shades of asterisks

OPENGL CODE

```
#include <GL/glut.h>
#include <cmath>
#include <cstdlib>
#include <cstdio>

// Polygon vertices
GLfloat vertices[][2] = {
    {8, 4},
    {2, 4},
    {0, 8},
    {3, 12},
    {7, 12},
    {10, 8}
};

const int numVertices = 6;

// Function to draw the grid
void drawGrid() {
    glColor3f(0.9, 0.9, 0.9);
    for (int i = -5; i <= 25; i++) {
        // Horizontal
        glBegin(GL_LINES);
        glVertex2f(-5, i);
        glVertex2f(25, i);
        glEnd();
```

```
        // Vertical
        glBegin(GL_LINES);
        glVertex2f(i, -5);
        glVertex2f(i, 25);
        glEnd();
    }
}

// Function to draw the axes and labels
void drawAxes() {
    glColor3f(0.0, 0.0, 0.0); // Black for axes

    // Drawing X and Y axes
    glBegin(GL_LINES);
    glVertex2f(0, 0); glVertex2f(25, 0); // X-axis start and end points
    glVertex2f(0, 0); glVertex2f(0, 25); // Y-axis start and end points
    glEnd();

    // Labeling points on the X-axis
    for (int i = 0; i <= 25; i+=2) {
        // Tick marks
        glBegin(GL_LINES);
        glVertex2f(i, -0.2f);
        glVertex2f(i, 0.2f);
        glEnd();

        // Numbering labels
        glRasterPos2f(i - 0.2f, -1.0f);
        char label[5];
        sprintf(label, "%d", i);
        for (char* c = label; *c != '\0'; c++) {
            glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10, *c);
        }
    }
```

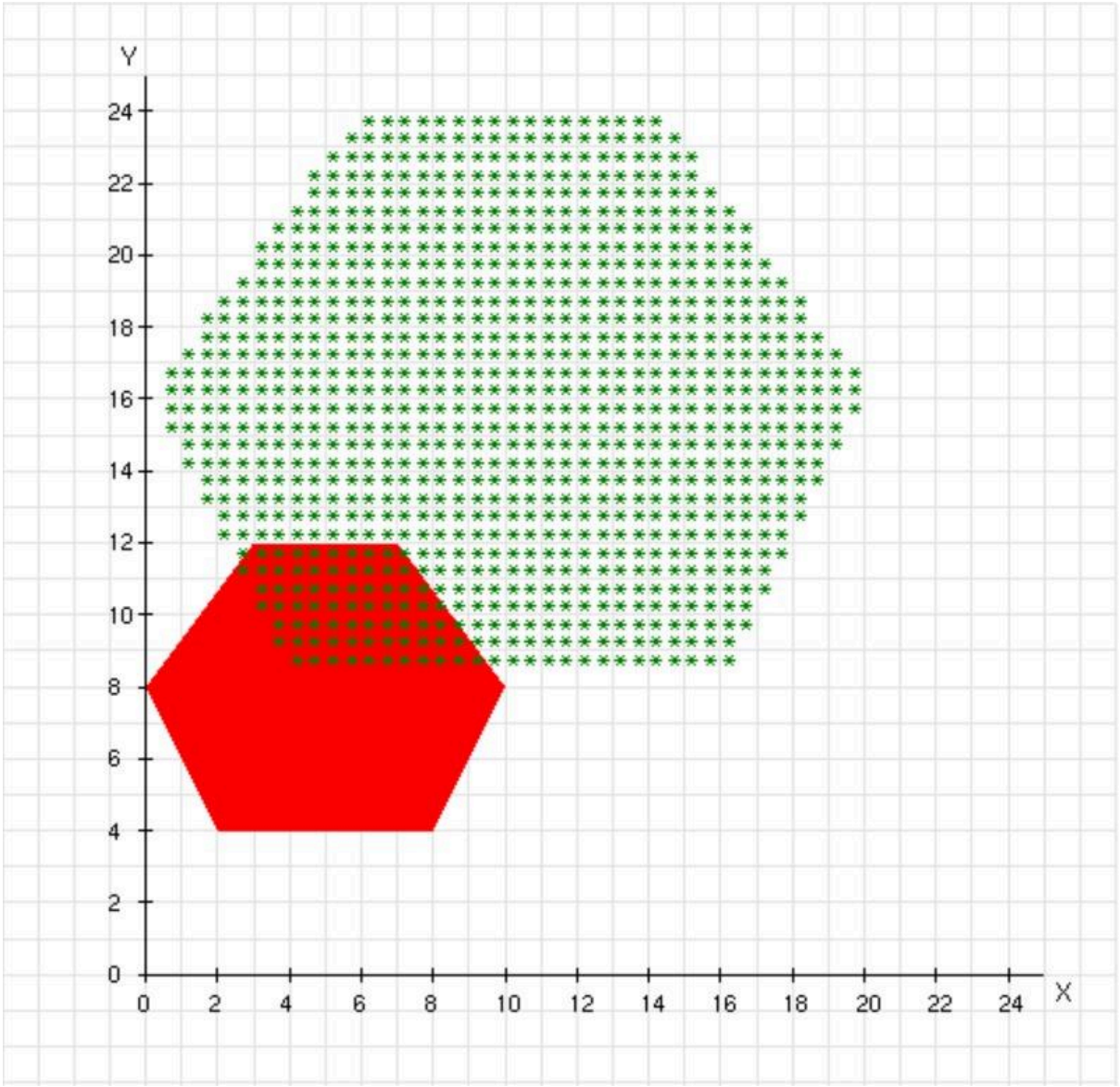
```
    // Labeling points on the Y-axis
    for (int i = 0; i <= 25; i+=2) {
        // Tick marks
        glBegin(GL_LINES);
        glVertex2f(-0.2f, i);
        glVertex2f(0.2f, i);
        glEnd();

        // Numbering labels
        glRasterPos2f(-1.0f, i - 0.2f);
        char label[5];
        sprintf(label, "%d", i);
        for (char* c = label; *c != '\0'; c++) {
            glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10, *c);
        }
    }

    // Axis labels
    glRasterPos2f(25.3f, -0.7f);
    glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, 'X');

    glRasterPos2f(-0.7f, 25.3f);
    glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, 'Y');
}

// Function to draw the initial polygon in red
void drawPolygon(GLfloat vertexArray[][2], int numVertices) {
    glColor3f(1.0, 0.0, 0.0); // Red
    glBegin(GL_POLYGON);
    for (int i = 0; i < numVertices; i++) {
        glVertex2f(vertexArray[i][0], vertexArray[i][1]);
    }
    glEnd();
```

```
}

// Function to fill the polygon with asterisks
void fillWithAsterisks(GLfloat vertexArray[][2], int numVertices) {
    glColor3f(0.0, 0.5, 0.0); // Darker green

    // Determine bounds
    GLfloat minX = vertexArray[0][0], maxX = vertexArray[0][0];
    GLfloat minY = vertexArray[0][1], maxY = vertexArray[0][1];
    for (int i = 1; i < numVertices; i++) {
        if (vertexArray[i][0] < minX) minX = vertexArray[i][0];
        if (vertexArray[i][0] > maxX) maxX = vertexArray[i][0];
        if (vertexArray[i][1] < minY) minY = vertexArray[i][1];
        if (vertexArray[i][1] > maxY) maxY = vertexArray[i][1];
    }

    for (float x = minX + 0.5; x < maxX; x += 0.5) {
        for (float y = minY + 0.5; y < maxY; y += 0.5) {
            int count = 0;
            for (int i = 0, j = numVertices - 1; i < numVertices; j = i++) {
                if (((vertexArray[i][1] > y) != (vertexArray[j][1] > y)) &&
                    (x < (vertexArray[j][0] - vertexArray[i][0]) * (y - vertexArray[i][1]) /
                        (vertexArray[j][1] - vertexArray[i][1]) + vertexArray[i][0])) {
                    count++;
                }
            }
            if (count % 2 == 1) {
                glRasterPos2f(x, y);
                glutBitmapCharacter(GLUT_BITMAP_8_BY_13, '*');
            }
        }
    }
}
```

```
// Display function
void display() {
    glClear(GL_COLOR_BUFFER_BIT);

    drawGrid(); // Grid
    drawAxes(); // Axes and labels
    drawPolygon(vertices, numVertices); // Original polygon

    // Scale polygon by factor of 2
    GLfloat scaledVertices[numVertices][2];
    for (int i = 0; i < numVertices; i++) {
        scaledVertices[i][0] = vertices[i][0] * 2;
        scaledVertices[i][1] = vertices[i][1] * 2;
    }

    // Fill scaled polygon with asterisks
    fillWithAsterisks(scaledVertices, numVertices);

    glFlush();
}

// Initialization
void init() {
    glClearColor(1.0, 1.0, 1.0, 1.0); // White background
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(-5, 30, -5, 30); // Adjusted to fit labels
}

// Main function
int main(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(600, 600);
    glutCreateWindow("Group 23: Transformation and Shapes");
```

```
    init();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

OPENGL OUTPUT



PYTHON CODE

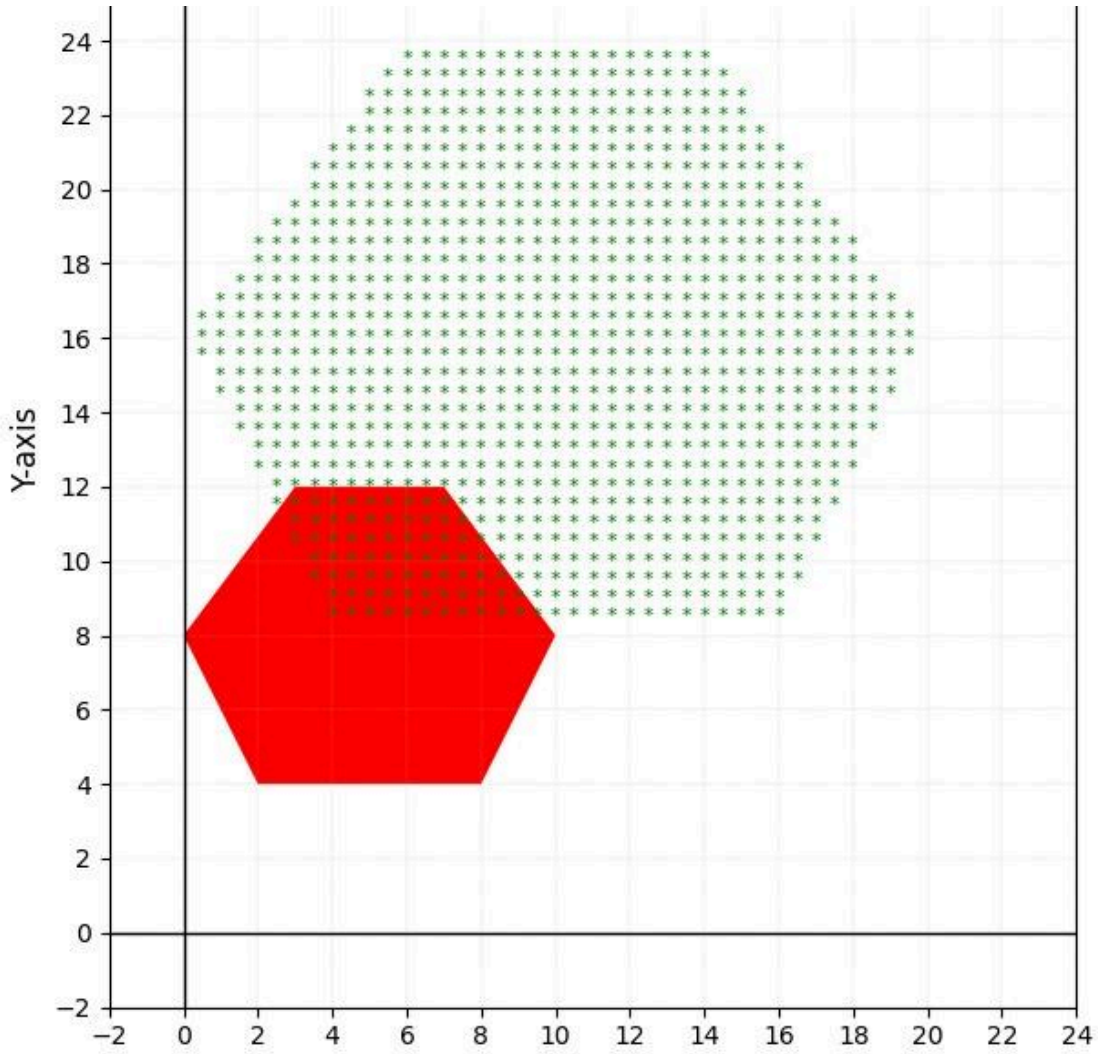
```
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon as MplPolygon
```

```
from matplotlib.path import Path
import numpy as np

# Define the original polygon vertices
original_vertices = [
    (8, 4),
    (2, 4),
    (0, 8),
    (3, 12),
    (7, 12),
    (10, 8)
]

# Function to scale a polygon by a factor
def scale_polygon(vertices, factor):
    """
    Scales each vertex of the polygon by a given factor.

    vertices: list of (x, y) tuples
    factor: scale factor (float)

    :return: list of scaled (x, y) tuples
    """
    return [(x * factor, y * factor) for x, y in vertices]

# Function to draw and fill a polygon with a color
def draw_filled_polygon(ax, vertices, color):
    """
    Draws a filled polygon on the given axes.

    ax: matplotlib axes
    vertices: list of (x, y) tuples
    color: fill color
    """
```

```
    polygon = MplPolygon(vertices, closed=True, facecolor=color, linewidth=1.5)
    ax.add_patch(polygon)

# Function to fill a polygon with asterisks
def fill_with_asterisks(ax, vertices, color='green'):
    """
    Fills the interior of a polygon with asterisks (*).
    ax: matplotlib axes
    vertices: list of (x, y) tuples
    color: color of the asterisks
    """
    # Create a Path from the polygon vertices to test point inclusion
    path = Path(vertices)

    # Get bounding box of the polygon
    min_x = min(x for x, y in vertices)
    max_x = max(x for x, y in vertices)
    min_y = min(y for x, y in vertices)
    max_y = max(y for x, y in vertices)

    # Loop through grid of points and check if inside the polygon
    for x in np.arange(min_x, max_x, 0.5):
        for y in np.arange(min_y, max_y, 0.5):
            if path.contains_point((x, y)):
                ax.text(x, y, '*', fontsize=8, color=color, ha='center', va='center')

# Main plot setup
fig, ax = plt.subplots(figsize=(10, 8))
ax.set_title("Original and Scaled Polygon with Asterisk Fill", fontsize=14)
ax.set_facecolor('white')  # Set background to white
ax.set_aspect('equal')     # Keep aspect ratio equal for X and Y

# Set the coordinate space to show both polygons
ax.set_xlim(-2, 24)
```

```
ax.set_ylim(-2, 28)

# (a) Draw and fill the original polygon in Red
draw_filled_polygon(ax, original_vertices, '#FF0000')  # Red fill

# (b) Scale up the polygon by factor of 2
scaled_vertices = scale_polygon(original_vertices, 2)

# (c) Fill the scaled polygon with green asterisks
fill_with_asterisks(ax, scaled_vertices, color='green')

# Draw x and y axes using black lines
ax.axhline(0, color='black', linewidth=1)  # X-axis
ax.axvline(0, color='black', linewidth=1)  # Y-axis

# Add axis labels
ax.set_xlabel("X-axis", fontsize=12)
ax.set_ylabel("Y-axis", fontsize=12)

# Add numerical tick marks for axes
ax.set_xticks(np.arange(-2, 25, 2))
ax.set_yticks(np.arange(-2, 29, 2))

# Show the result
plt.grid(True, which='both', linestyle='--', linewidth=0.3, alpha=0.5)  # Optional: light grid
plt.show()
```

PYTHON OUTPUT


Y-axis
24
22
20
18
16
14
12
10
8
6
4
2
0
−2
−2
0
2
4
6
8
10
12
14
16
18
20
22
24