Computer Graphics

Lab 2

Name	Ranime Ahmed Elsayed Shehata
ID	21010531

Problem Statement:

In this assignment, you will be working with the materials provided in Lab 2 to enhance your understanding of graphics programming. This task will involve reading data from a text file and using that data to create a visual representation. Please follow the detailed steps below to finish the assignment.

- **Overview of the Assignment**

The main goal of this assignment is to read the vertices from the provided text file, snoopy.txt, and use those vertices to draw a graphical representation. This will help you practice file handling, data extraction, and graphical rendering.

- **Steps to Complete the Assignment**

1. Access the Lab 2 Materials

Begin by reviewing the materials provided in the Lab 2 folder.
Familiarize yourself with the content, as it will provide important context and resources for your assignment.

2. Understanding the snoopy.txt File

Locate the snoopy.txt file within the Lab 2 materials. This file contains the vertices that you will need to read and process.
Open the file to examine its structure. Take note of how the vertices are organized. Shapes are listed in a specific format: the first line specifies the number of vertices and the drawing mode (l: line strip, f: polygon), followed by lines detailing the x and y coordinates, matching the vertex count.

3. Drawing the Vertices

Once you have successfully read the vertices, the next step is to draw them. Here's how you can approach this:
Initialize your drawing environment. Set up the window size and background color as needed.
Loop through the list of shapes you created earlier and draw each shape on the screen.

4. Testing and Debugging

After implementing your drawing code, run your program to see the output. Check if the vertices are displayed correctly.
If you encounter any issues, debug your code by checking:
Whether the file was read correctly.
If the vertices are being processed and drawn as intended.
Any error messages that may help you identify the problem.

---

1. Introduction:

The goal of this assignment was to read vertex data from a file (snoopy.txt) and render it using OpenGL and Python. The task involved understanding the structure of the file, parsing the data, and using OpenGL to draw the shapes described by the vertices. The implementation used the glfw library for window management and OpenGL for rendering. This report explains the process and results of the assignment.

---

2. Understanding the snoopy.txt File:

The snoopy.txt file contains vertex data organized in a specific format. Each shape is defined by:
A header line: <number_of_vertices> <drawing_mode>.
l stands for line strip.
f stands for polygon.
Subsequent lines: <x> <y> coordinates for each vertex.
For example:

```
9 l
12.5 0
11 2
10.5 5
10.1 7
10.1 9
10.5 11
10.5 13
11 15
11.5 18
```

This defines a line strip with 9 vertices.
The file contains multiple shapes, some connected as line strips and others as filled polygons.
These -vertices collectively represent a 2D drawing of "Snoopy."

---

3. Reading the File:

To read the file, Python's file handling capabilities were used. The file was parsed line by line, and the data was stored in a list of tuples. Each tuple contains:
The drawing mode ('l' or 'f').
A list of (x, y) coordinates for the vertices.

Code Snippet:

```
def read_file(filename):
    with open(filename, 'r') as file:
        while True:
            line = file.readline()
            if not line:
                break
            parts = line.strip().split()
            if len(parts) < 2:
                continue
            num_vertices = int(parts[0])
            draw_mode = parts[1]
            vertices = []
            for _ in range(num_vertices):
                vertex_line = file.readline()
                x, y = map(float, vertex_line.strip().split())
                vertices.append((x, y))
            shapes.append((draw_mode, vertices))
```

Code Explanation:

1. Open the file and read it line by line.
2. Extract the number of vertices and drawing mode from the header line.
3. Read the specified number of vertices and store them in a list.
4. Append the drawing mode and vertices as a tuple to the shapes list.

---

4. Drawing the Shapes:

Once the file was read, the shapes were rendered using OpenGL. The glBegin and glEnd functions were used to define the drawing mode (GL_LINE_STRIP for 'l' and GL_POLYGON for 'f'). The vertices were iterated through and drawn on the screen.

Code Snippet:

```
def draw_shapes():
    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(1.0, 1.0, 1.0)

    for shape in shapes:
        draw_mode, vertices = shape
        if draw_mode == 'l':
            glBegin(GL_LINE_STRIP)
        elif draw_mode == 'f':
            glBegin(GL_POLYGON)
        else:
            continue

        for x, y in vertices:
            glVertex2f(x, y)
        glEnd()
```

Code Explanation:

1. Clear the screen with glClear.
2. Set the drawing color to white using glColor3f.
3. Loop through each shape and use glBegin and glEnd to draw the vertices.
4. Use GL_LINE_STRIP for line strips and GL_POLYGON for polygons.

Orthographic Projection

To ensure the shapes fit within the window, an orthographic projection was set up using glOrtho:

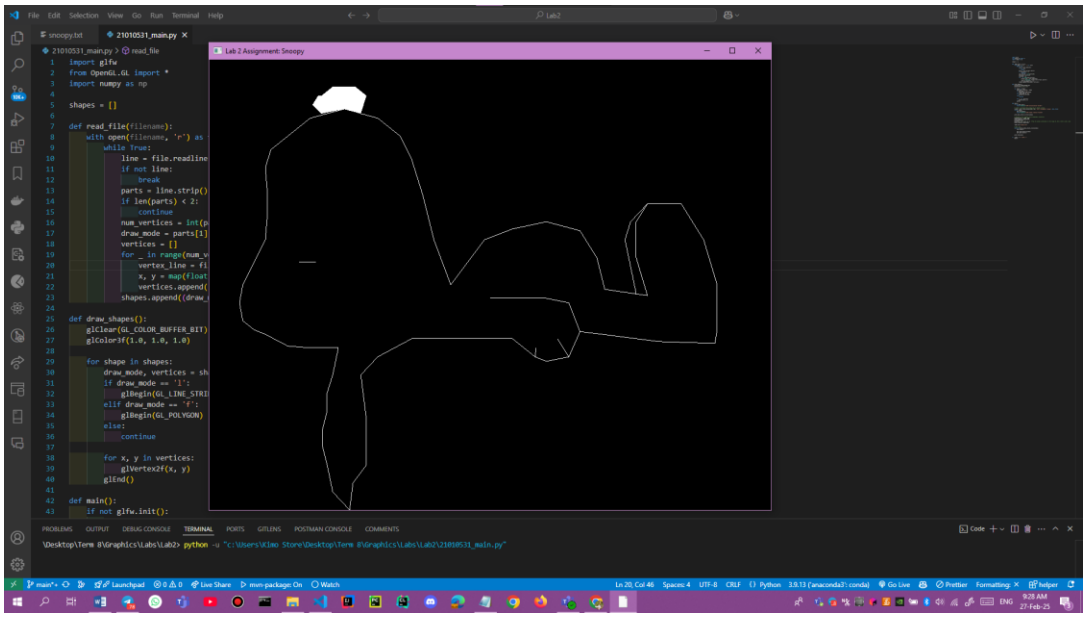
```
glOrtho(0, 50, 0, 50, -1, 1)
```

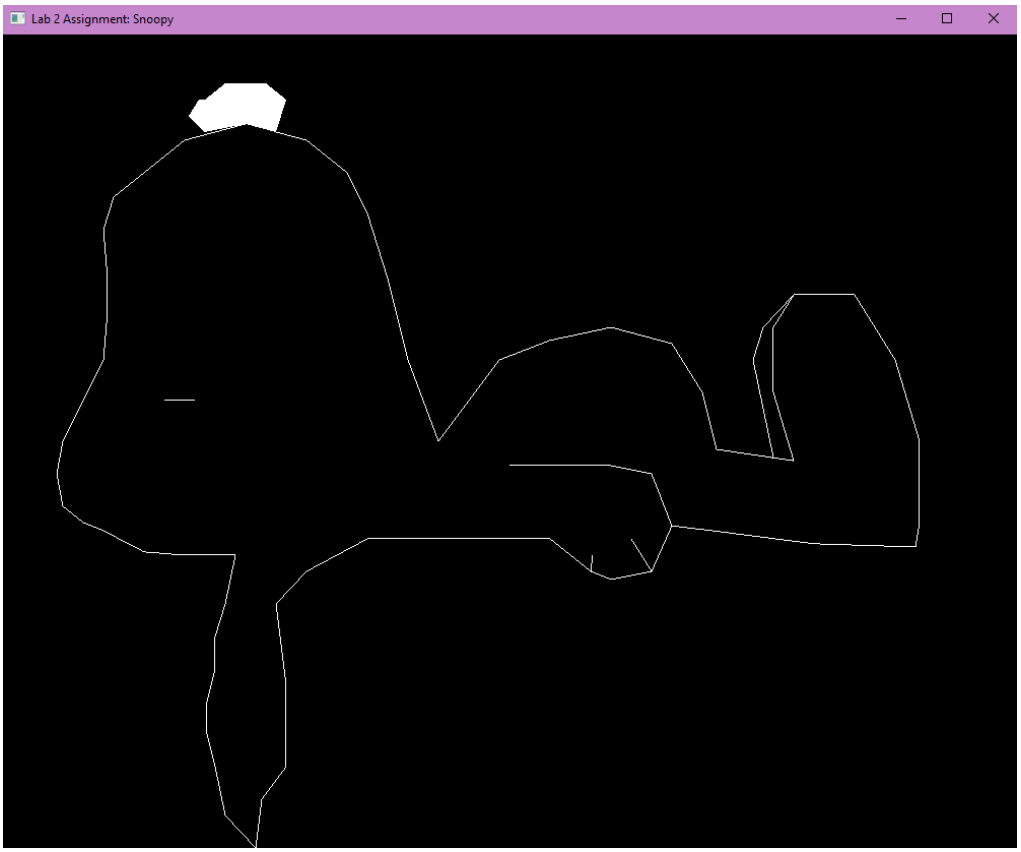
This maps the coordinates in the range [0, 50] for both x and y axes to the window.

---

5. Results:

The program successfully rendered the vertices from snoopy.txt, displaying a 2D drawing of Snoopy. The shapes were drawn as specified, with line strips and polygons rendered correctly.





6. Code Snippet:

```python
import glfw
from OpenGL.GL import *
import numpy as np

shapes = []

def read_file(filename):
    with open(filename, 'r') as file:
        while True:
            line = file.readline()
            if not line:
                break
            parts = line.strip().split()
            if len(parts) < 2:
                continue
            num_vertices = int(parts[0])
            draw_mode = parts[1]
            vertices = []
            for _ in range(num_vertices):
                vertex_line = file.readline()
                x, y = map(float, vertex_line.strip().split())
                vertices.append((x, y))
            shapes.append((draw_mode, vertices))

def draw_shapes():
    glClear(GL_COLOR_BUFFER_BIT)
    glColor3f(1.0, 1.0, 1.0)

    for shape in shapes:
        draw_mode, vertices = shape
        if draw_mode == 'l':
            glBegin(GL_LINE_STRIP)
        elif draw_mode == 'f':
            glBegin(GL_POLYGON)
        else:
            continue

        for x, y in vertices:
            glVertex2f(x, y)
        glEnd()

def main():
    if not glfw.init():
        raise Exception("GLFW initialization failed")

    # Create a windowed mode window and its OpenGL context
    window = glfw.create_window(1000, 800, "Lab 2 Assignment: Snoopy", None, None)
    if not window:
        glfw.terminate()
        raise Exception("GLFW window creation failed")

    glfw.make_context_current(window)

    # Setting up the viewport and orthographic projection
    glViewport(0, 0, 1000, 800)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(0, 50, 0, 50, -1, 1)  # Map the window coordinates to the range [0, 50] in both x and y axes
    glMatrixMode(GL_MODELVIEW)

    read_file("snoopy.txt")

    # Main loop
    while not glfw.window_should_close(window):
        draw_shapes()

        glfw.swap_buffers(window)
        glfw.poll_events()

    glfw.terminate()

if __name__ == "__main__":
    main()
```