Lab Report

Report No : 01
Report Title : Drawing GL Points, GL Lines
Course Title: Computer Graphics Lab
Course Code: CSE422

Submitted to,
Alvee Ehsan
lecturer
Department of
Daffodil International University

Submitted By,
Name: **Redwan Rahman**
ID: 0242220005101127
Section: 63_E1
Department of CSE
Daffodil International University

Date of Submission: 21/02/2026

Task 1: Drawing Random Pixels

Code:

randomPoints.py > ...

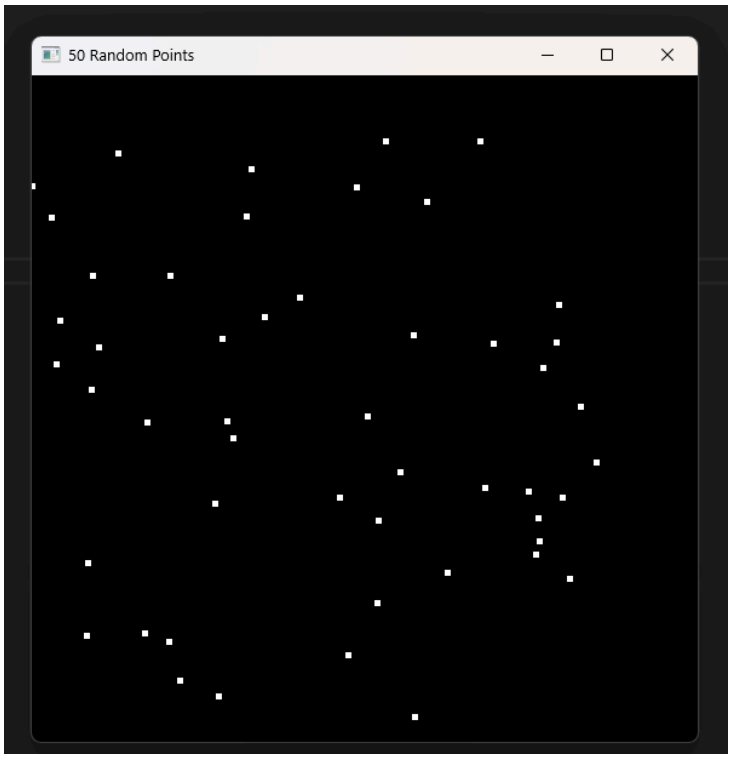
```python
import random
from OpenGL.GL import *
from OpenGL.GLUT import *
points = []
for i in range(50):
    x = random.randint(0, 500)
    y = random.randint(0, 500)
    points.append((x, y))
def iterate():
    glViewport(0, 0, 500, 500)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(0.0, 500, 0.0, 500, 0.0, 1.0)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
def showScreen():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glLoadIdentity()
    iterate()
    glColor3f(1, 1, 1)
    glPointSize(5)
    glBegin(GL_POINTS)
    for x, y in points:
        glVertex2f(x, y)
    glEnd()
    glutSwapBuffers()
glutInit()
glutInitDisplayMode(GLUT_RGBA)
glutInitWindowSize(550, 550)
glutInitWindowPosition(0, 0)
wind = glutCreateWindow(b"50 Random Points")
glutDisplayFunc(showScreen)
glutMainLoop()
```

Implementation Details

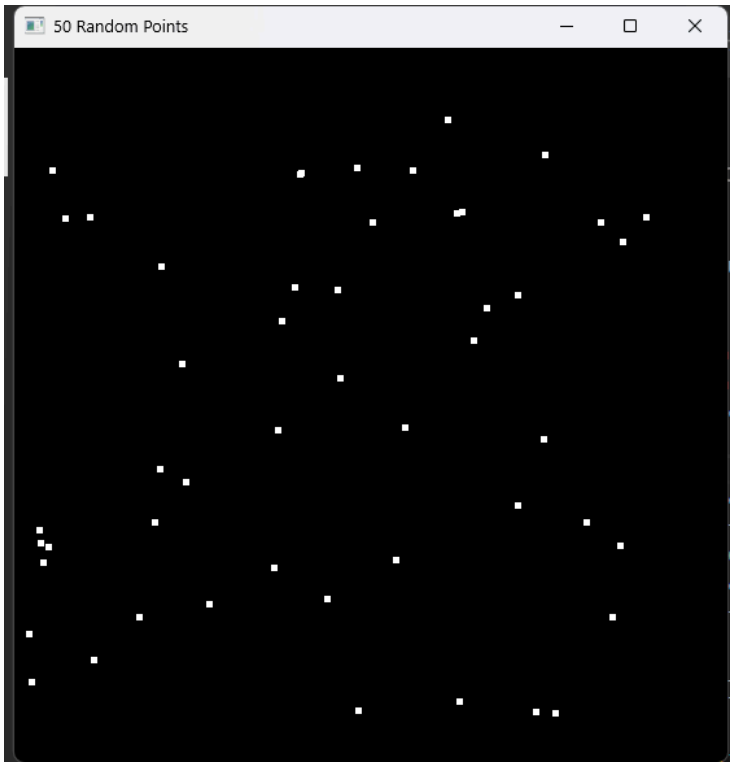
- Primitive Used: GL_POINTS
- Logic: A Python for loop combined with the random.randint(0, 500) function was used to generate an array of 50 (x, y) coordinate pairs.
- Rendering: The point size was increased using glPointSize(5) to make the pixels clearly visible against the black background.

Output:

Run 1:



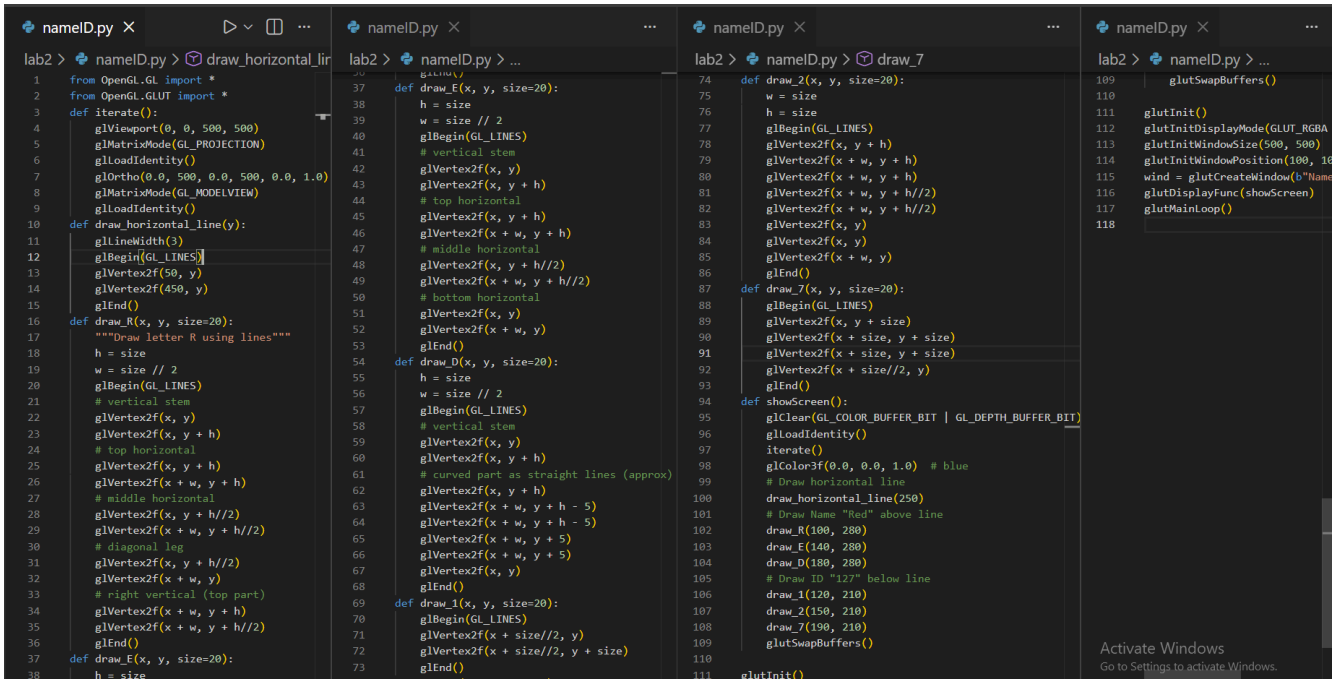
Run 2:



[These executions show the randomness of the generated points]

Task 2: Drawing Name Initials & ID:

Code:



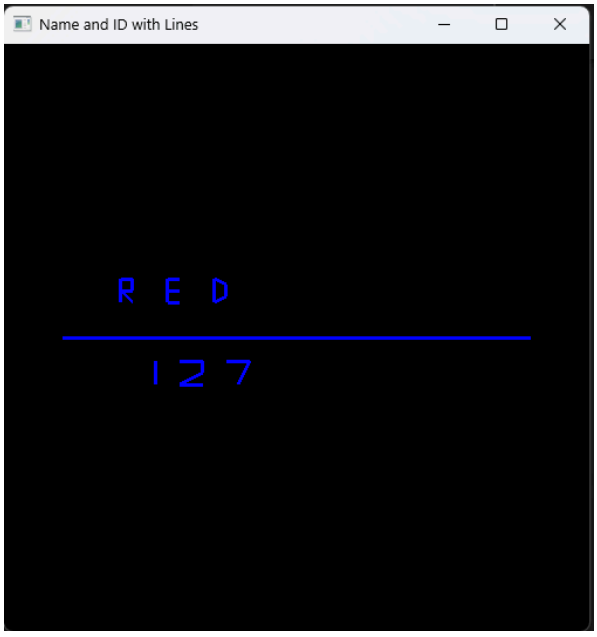
```
from OpenGL.GL import *
from OpenGL.GLUT import *
def iterate():
    glViewport(0, 0, 500, 500)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(0.0, 500, 0.0, 500, 0.0, 1.0)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
def draw_horizontal_line(y):
    glLineWidth(3)
    glBegin(GL_LINES)
    glVertex2f(50, y)
    glVertex2f(450, y)
    glEnd()
def draw_R(x, y, size=20):
    """Draw letter R using lines"""
    h = size
    w = size // 2
    glBegin(GL_LINES)
    # vertical stem
    glVertex2f(x, y)
    glVertex2f(x, y + h)
    # top horizontal
    glVertex2f(x, y + h)
    glVertex2f(x + w, y + h)
    # middle horizontal
    glVertex2f(x, y + h//2)
    glVertex2f(x + w, y + h//2)
    # diagonal leg
    glVertex2f(x, y + h//2)
    glVertex2f(x + w, y)
    # right vertical (top part)
    glVertex2f(x + w, y + h)
    glVertex2f(x + w, y + h//2)
    glEnd()
def draw_E(x, y, size=20):
    h = size
    w = size // 2
    glBegin(GL_LINES)
    # vertical stem
    glVertex2f(x, y)
    glVertex2f(x, y + h)
    # top horizontal
    glVertex2f(x, y + h)
    glVertex2f(x + w, y + h)
    # middle horizontal
    glVertex2f(x, y + h//2)
    glVertex2f(x + w, y + h//2)
    # bottom horizontal
    glVertex2f(x, y)
    glVertex2f(x + w, y)
    glEnd()
def draw_D(x, y, size=20):
    h = size
    w = size // 2
    glBegin(GL_LINES)
    # vertical stem
    glVertex2f(x, y)
    glVertex2f(x, y + h)
    # curved part as straight lines (approx)
    glVertex2f(x, y + h)
    glVertex2f(x + w, y + h - 5)
    glVertex2f(x + w, y + h - 5)
    glVertex2f(x + w, y + 5)
    glVertex2f(x + w, y + 5)
    glVertex2f(x, y)
    glEnd()
def draw_1(x, y, size=20):
    glBegin(GL_LINES)
    glVertex2f(x + size//2, y)
    glVertex2f(x + size//2, y + size)
    glEnd()
def draw_2(x, y, size=20):
    w = size
    h = size
    glBegin(GL_LINES)
    glVertex2f(x, y + h)
    glVertex2f(x + w, y + h)
    glVertex2f(x + w, y + h)
    glVertex2f(x + w, y + h//2)
    glVertex2f(x + w, y + h//2)
    glVertex2f(x, y)
    glVertex2f(x, y)
    glVertex2f(x + w, y)
    glEnd()
def draw_7(x, y, size=20):
    glBegin(GL_LINES)
    glVertex2f(x, y + size)
    glVertex2f(x + size, y + size)
    glVertex2f(x + size, y + size)
    glVertex2f(x + size//2, y)
    glEnd()
def showScreen():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glLoadIdentity()
    iterate()
    glColor3f(0.0, 0.0, 1.0)  # blue
    # Draw horizontal line
    draw_horizontal_line(250)
    # Draw Name "Red" above line
    draw_R(100, 280)
    draw_E(140, 280)
    draw_D(180, 280)
    # Draw ID "127" below line
    draw_1(120, 210)
    draw_2(150, 210)
    draw_7(190, 210)
    glutSwapBuffers()

glutInit()
glutInitDisplayMode(GLUT_RGBA
glutInitWindowSize(500, 500)
glutInitWindowPosition(100, 10
wind = glutCreateWindow(b"Name
glutDisplayFunc(showScreen)
glutMainLoop()
```

Implementation Details

- Primitive Used: GL_LINES
- Logic: Custom functions were created for each specific letter (draw_R, draw_E, draw_D) and number (draw_1, draw_2, draw_7). A horizontal dividing line was drawn across the center of the screen ($y=250$) to separate the name from the ID.
- Coordinates: Precise relative coordinate math was used within each function to ensure the letters scaled and positioned correctly based on a starting (x, y) coordinate and a specified size parameter.

Output:



Task 3: House Building:

Code:

lab2 > house.py > showScreen

```python
from OpenGL.GL import *
from OpenGL.GLUT import *
def iterate():
    glViewport(0, 0, 500, 500)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    glOrtho(0.0, 500, 0.0, 500, 0.0, 1.0)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
def draw_line(x1, y1, x2, y2):
    glVertex2f(x1, y1)
    glVertex2f(x2, y2)
def showScreen():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glLoadIdentity()
    iterate()
    glColor3f(0.2, 0.4, 1.0)
    glLineWidth(2.5)
    # --- HOUSE LINES ---
    glBegin(GL_LINES)
    # Front wall
    draw_line(175, 145, 175, 350)   # Left vertical
    draw_line(175, 350, 420, 350)   # Top horizontal
    draw_line(420, 350, 420, 145)   # Right vertical
    draw_line(175, 145, 420, 145)   # Bottom horizontal
    # Roof
    draw_line(175, 350, 300, 450)   # Left slope
    draw_line(300, 450, 420, 350)   # Right slope
    # Door
    draw_line(270, 145, 270, 220)   # Left
    draw_line(330, 145, 330, 220)   # Right
    draw_line(270, 220, 330, 220)   # Top
    # Windows
    # Left window
    draw_line(200, 200, 230, 200)   # Top
    draw_line(200, 200, 200, 250)   # Left
    draw_line(200, 250, 230, 250)   # Bottom
    draw_line(230, 200, 230, 250)   # Right
```

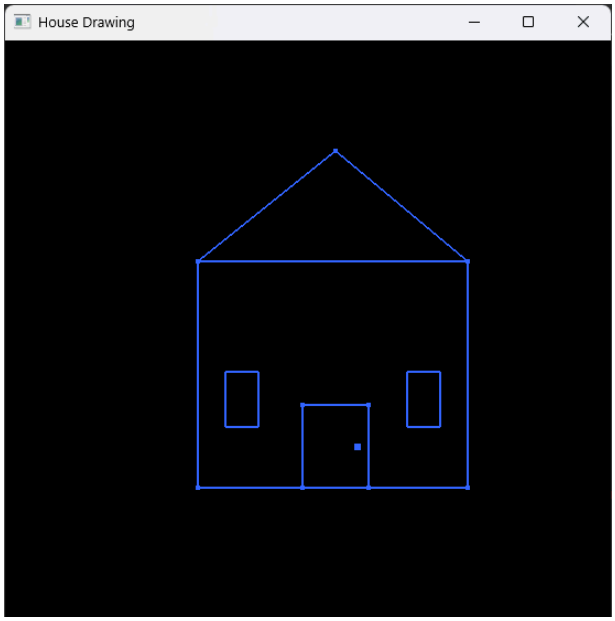
lab2 > house.py > showScreen

```python
def showScreen():
    draw_line(270, 145, 270, 220)   # Left
    draw_line(330, 145, 330, 220)   # Right
    draw_line(270, 220, 330, 220)   # Top
    # Windows
    # Left window
    draw_line(200, 200, 230, 200)   # Top
    draw_line(200, 200, 200, 250)   # Left
    draw_line(200, 250, 230, 250)   # Bottom
    draw_line(230, 200, 230, 250)   # Right
    # Right window
    draw_line(365, 200, 395, 200)   # Top
    draw_line(365, 200, 365, 250)   # Left
    draw_line(365, 250, 395, 250)   # Bottom
    draw_line(395, 200, 395, 250)   # Right
    glEnd()
    # --- CORNER POINTS ---
    glPointSize(4)
    glBegin(GL_POINTS)
    points = [
        (175, 145), (420, 145), (175, 350), (420, 350), (300, 450),
        (270, 145), (330, 145), (270, 220), (330, 220)
    ]
    for pt in points:
        glVertex2f(*pt)
    glEnd()
    #doorKnob
    glPointSize(6)
    glBegin(GL_POINTS)
    glVertex2f(320, 182)
    glEnd()
    glutSwapBuffers()
glutInit()
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE)
glutInitWindowSize(550, 550)
glutInitWindowPosition(0, 0)
wind = glutCreateWindow(b"House Drawing")
glutDisplayFunc(showScreen)
```

Implementation Details

- Primitives Used: GL_LINES (for structure), GL_POINTS (for accents).
- Logic: A custom draw_line(x1, y1, x2, y2) helper function was created to keep the showScreen function clean. GL_LINES were mapped out to construct the front walls, roof slopes, door, and windows.
- Accents: GL_POINTS were strategically placed at the vertices (corners) of the house structure to highlight the joints. A slightly larger point (glPointSize(6)) was used to represent the doorknob.

Output:



All source files were committed to a GitHub repository during and after the lab session. The repository records the progression of development including in-lab commits and subsequent home modifications.
Repository URL: https://github.com/Red1-Rahman/Computer-Graphics-Lab-DIU

Conclusion:

This lab successfully demonstrated foundational 2D rendering using OpenGL state machine concepts. Manipulating GL_POINTS and GL_LINES within explicit coordinate grids established the technical groundwork for more complex graphics in future sessions.