



Daffodil
International
University

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:

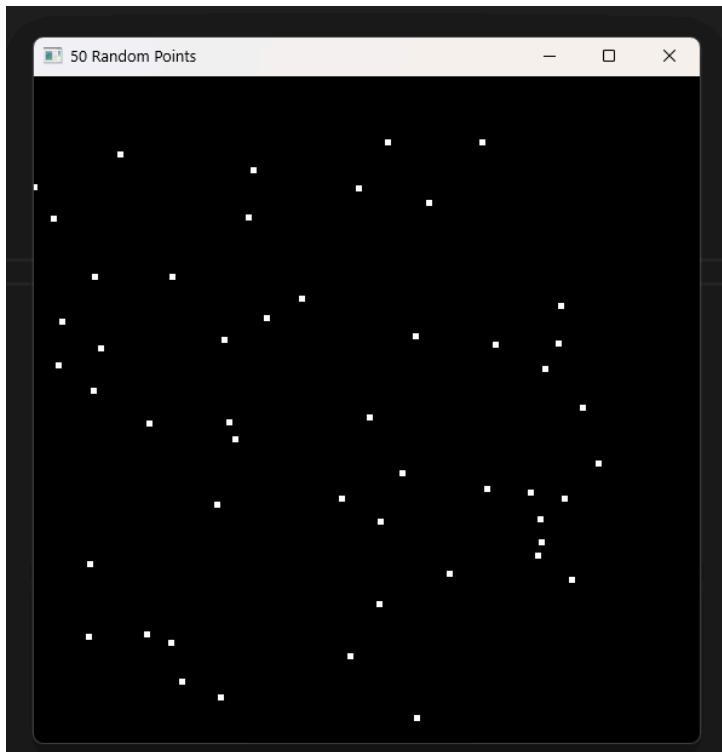
```
> randomPoints.py > ...
import random
from OpenGL.GL import *
from OpenGL.GLUT import *
points = []
for i in range(50):
    x = random.randint(0, 500)
    y = random.randint(0, 500)
    points.append((x, y))
def iterate():
    glViewport(0, 0, 500, 500)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    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

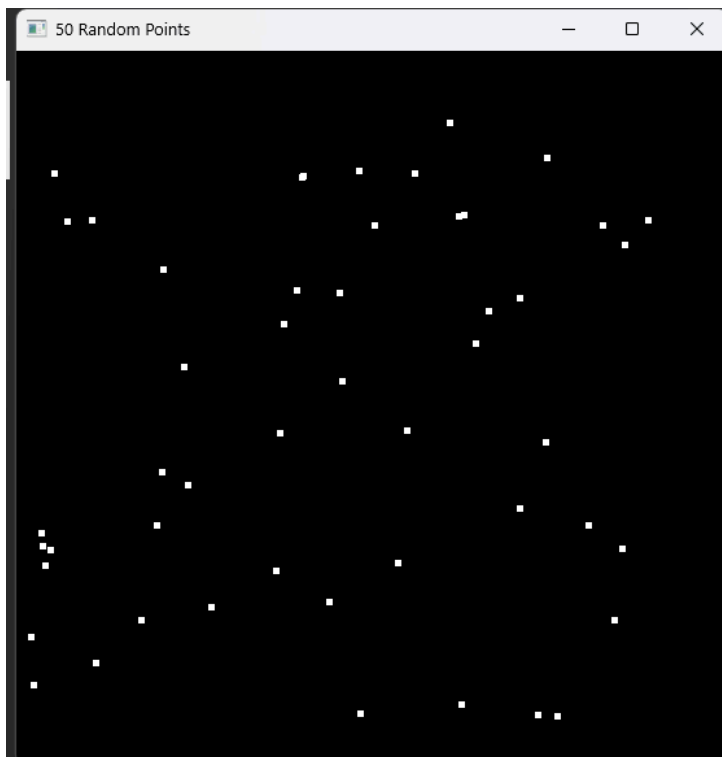
- Primitive Used: GL_POINTS
- Logic: A Python for loop combined with the random.randint(0, 500) function was used to generate an array of 50 (x, y) coordinate pairs.
- Rendering: The point size was increased using glPointSize(5) to make the pixels clearly visible against the black background.

Output:

Run 1:



Run 2:



[These executions show the randomness of the generated points]

Task 2: Drawing Name Initials & ID:

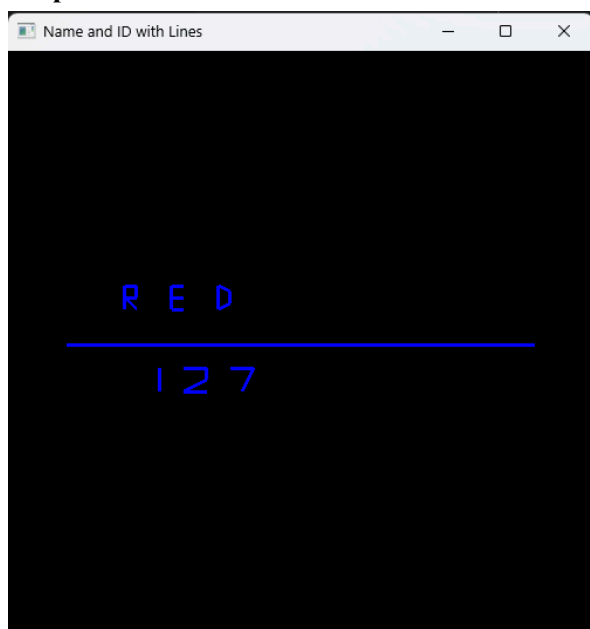
Code:

```
nameID.py X lab2 > nameID.py > draw_horizontal_line(y):
1 from OpenGL.GL import *
2 from OpenGL.GLU import *
3 def iterate():
4     glViewport(0, 0, 500, 500)
5     glMatrixMode(GL_PROJECTION)
6     glLoadIdentity()
7     glOrtho(0.0, 500, 0.0, 500, 0.0, 1.0)
8     glMatrixMode(GL_MODELVIEW)
9     glLoadIdentity()
10 def draw_horizontal_line(y):
11     glLineWidth(3)
12     glBegin(GL_LINES)
13     glVertex2f(50, y)
14     glVertex2f(450, y)
15     glEnd()
16 def draw_R(x, y, size=20):
17     """Draw letter R using lines"""
18     h = size
19     w = size // 2
20     glBegin(GL_LINES)
21     # vertical stem
22     glVertex2f(x, y)
23     glVertex2f(x, y + h)
24     # top horizontal
25     glVertex2f(x, y + h)
26     glVertex2f(x + w, y + h)
27     # middle horizontal
28     glVertex2f(x, y + h/2)
29     glVertex2f(x + w, y + h/2)
30     # diagonal leg
31     glVertex2f(x, y + h/2)
32     glVertex2f(x + w, y)
33     # right vertical (top part)
34     glVertex2f(x + w, y + h)
35     glVertex2f(x + w, y + h/2)
36     glEnd()
37 def draw_E(x, y, size=20):
38     h = size
39     w = size // 2
40     glBegin(GL_LINES)
41     # vertical stem
42     glVertex2f(x, y)
43     glVertex2f(x, y + h)
44     # top horizontal
45     glVertex2f(x, y + h)
46     glVertex2f(x + w, y + h)
47     # middle horizontal
48     glVertex2f(x, y + h/2)
49     glVertex2f(x + w, y + h/2)
50     # bottom horizontal
51     glVertex2f(x, y)
52     glVertex2f(x + w, y)
53     glEnd()
54 def draw_D(x, y, size=20):
55     h = size
56     w = size // 2
57     glBegin(GL_LINES)
58     # vertical stem
59     glVertex2f(x, y)
60     glVertex2f(x, y + h)
61     # curved part as straight lines (approx)
62     glVertex2f(x, y + h)
63     glVertex2f(x + w, y + h - 5)
64     glVertex2f(x + w, y + h - 5)
65     glVertex2f(x + w, y + 5)
66     glVertex2f(x + w, y + 5)
67     glVertex2f(x, y)
68     glEnd()
69 def draw_1(x, y, size=20):
70     glBegin(GL_LINES)
71     glVertex2f(x + size//2, y)
72     glVertex2f(x + size//2, y + size)
73     glEnd()
74 def draw_2(x, y, size=20):
75     w = size
76     h = size
77     glBegin(GL_LINES)
78     glVertex2f(x, y + h)
79     glVertex2f(x + w, y + h)
80     glVertex2f(x + w, y + h/2)
81     glVertex2f(x + w, y + h/2)
82     glVertex2f(x + w, y + h/2)
83     glVertex2f(x, y)
84     glVertex2f(x + w, y)
85     glVertex2f(x + w, y)
86     glEnd()
87 def draw_7(x, y, size=20):
88     glBegin(GL_LINES)
89     glVertex2f(x, y + size)
90     glVertex2f(x + size, y + size)
91     glVertex2f(x + size, y + size)
92     glVertex2f(x + size//2, y)
93     glEnd()
94 def showScreen():
95     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
96     glLoadIdentity()
97     iterate()
98     glColor3f(0.0, 0.0, 1.0) # blue
99     # Draw horizontal line
100    draw_horizontal_line(250)
101    # Draw Name "Red" above line
102    draw_R(100, 280)
103    draw_E(140, 280)
104    draw_D(180, 280)
105    # Draw ID "127" below line
106    draw_1(120, 210)
107    draw_2(150, 210)
108    draw_7(190, 210)
109    glutSwapBuffers()
110    glutInit()
111    glutInitDisplayMode(GLUT_RGBA)
112    glutInitWindowSize(500, 500)
113    glutInitWindowPosition(100, 100)
114    wind = glutCreateWindow(b"Name and ID with Lines")
115    glutDisplayFunc(showScreen)
116    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:

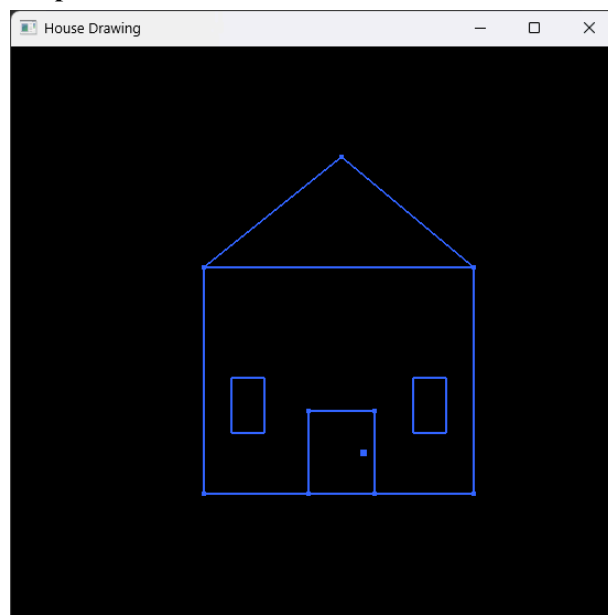
Code:

```
lab2 > house.py > showScreen
1 from OpenGL.GL import *
2 from OpenGL.GLUT import *
3 def iterate():
4     glViewport(0, 0, 500, 500)
5     glMatrixMode(GL_PROJECTION)
6     glLoadIdentity()
7     glOrtho(0.0, 500, 0.0, 500, 0.0, 1.0)
8     glMatrixMode(GL_MODELVIEW)
9     glLoadIdentity()
10    def draw_line(x1, y1, x2, y2):
11        glVertex2f(x1, y1)
12        glVertex2f(x2, y2)
13    def showScreen():
14        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
15        glLoadIdentity()
16        iterate()
17        glColor3f(0.2, 0.4, 1.0)
18        glLineWidth(2.5)
19        # --- HOUSE LINES ---
20        glBegin(GL_LINES)
21        # Front wall
22        draw_line(175, 145, 175, 350) # Left vertical
23        draw_line(175, 350, 420, 350) # Top horizontal
24        draw_line(420, 350, 420, 145) # Right vertical
25        draw_line(175, 145, 420, 145) # Bottom horizontal
26        # Roof
27        draw_line(175, 350, 300, 450) # Left slope
28        draw_line(300, 450, 420, 350) # Right slope
29        # Door
30        draw_line(270, 145, 270, 220) # Left
31        draw_line(330, 145, 330, 220) # Right
32        draw_line(270, 220, 330, 220) # Top
33        # Windows
34        # Left window
35        draw_line(200, 200, 230, 200) # Top
36        draw_line(200, 200, 230, 250) # Left
37        draw_line(200, 250, 230, 250) # Bottom
38        draw_line(230, 200, 230, 250) # Right
39
40    def showScreen():
41        draw_line(270, 145, 270, 220) # Left
42        draw_line(330, 145, 330, 220) # Right
43        draw_line(270, 220, 330, 220) # Top
44        # Windows
45        # Left window
46        draw_line(200, 200, 230, 200) # Top
47        draw_line(200, 200, 230, 250) # Left
48        draw_line(200, 250, 230, 250) # Bottom
49        draw_line(230, 200, 230, 250) # Right
50        glEnd()
51        # --- CORNER POINTS ---
52        glPointSize(4)
53        glBegin(GL_POINTS)
54        points = [
55            (175, 145), (420, 145), (175, 350), (420, 350), (300, 450),
56            (270, 145), (330, 145), (270, 220), (330, 220)
57        ]
58        for pt in points:
59            glVertex2f(*pt)
60        glEnd()
61        #doorknob
62        glPointSize(6)
63        glBegin(GL_POINTS)
64        glVertex2f(320, 182)
65        glEnd()
66        glutSwapBuffers()
67    glutInit()
68    glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE)
69    glutInitWindowSize(550, 550)
70    glutInitWindowPosition(0, 0)
71    wind = glutCreateWindow(b"House Drawing")
72    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.