Experiment 9

Student Name: Digant Raj **UID: 22BCS10225**
Branch: CSE **Section/Group: IOT 630 B**
Semester: 6 **Date of Performance: 03/04/25**
Subject Name: Computer Graphics Lab **Subject Code: 22CSH-352**

1. **Aim:** Demonstrate the result of window-to-viewport transformation by implementing and visualizing the process.

2. Objective:

Calculate and display the 4-bit region code for line endpoints and determine whether the line lies within the screen boundaries.

Algorithm:

- Initialize Graphics:

  Use initgraph() to initialize the graphics mode.

- Define Window and Viewport Coordinates:

  Set logical window boundaries: wxmin, wxmax, wymin, wymax.

  Set viewport boundaries: vxmin, vxmax, vymin, vymax.

- Draw the Window and Viewport Rectangles:

  Use rectangle() to draw the window and viewport areas.

- Input or Define Line Coordinates in Window:

- Define a line using window coordinates: (wx1, wy1) to (wx2, wy2).

- Draw Line in the Window:

  Use line(wx1, wy1, wx2, wy2) to draw the original line inside the window.

- Calculate Scaling Factors:

  sx = (vxmax - vxmin) / (wxmax - wxmin)

  sy = (vymax - vymin) / (wymax - wymin)

- Apply Transformation to Convert Window Coordinates to Viewport Coordinates:

  vx1 = sx * (wx1 - wxmin) + vxmin

  vy1 = sy * (wy1 - wymin) + vymin

vx2 = sx * (wx2 - wxmin) + vxmin

vy2 = sy * (wy2 - wymin) + vymin

- Draw Transformed Line in Viewport:

  Use line(vx1, vy1, vx2, vy2) to draw the mapped line inside the viewport.

- Label the Window and Viewport:

  Use outtextxy() to add labels.

- End Program:

  Use getch() to wait for user input.

  Close graphics using closegraph().

3. Implementation/Code:

```
#include <graphics.h>
#include <iostream>
#include <conio.h> // Only needed for getch() in Windows

using namespace std;

int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    // Clear screen after initializing graphics
    cleardevice();

    // Window and viewport coordinates
    float wxmin = 10, wxmax = 150, wymin = 10, wymax = 250;
```

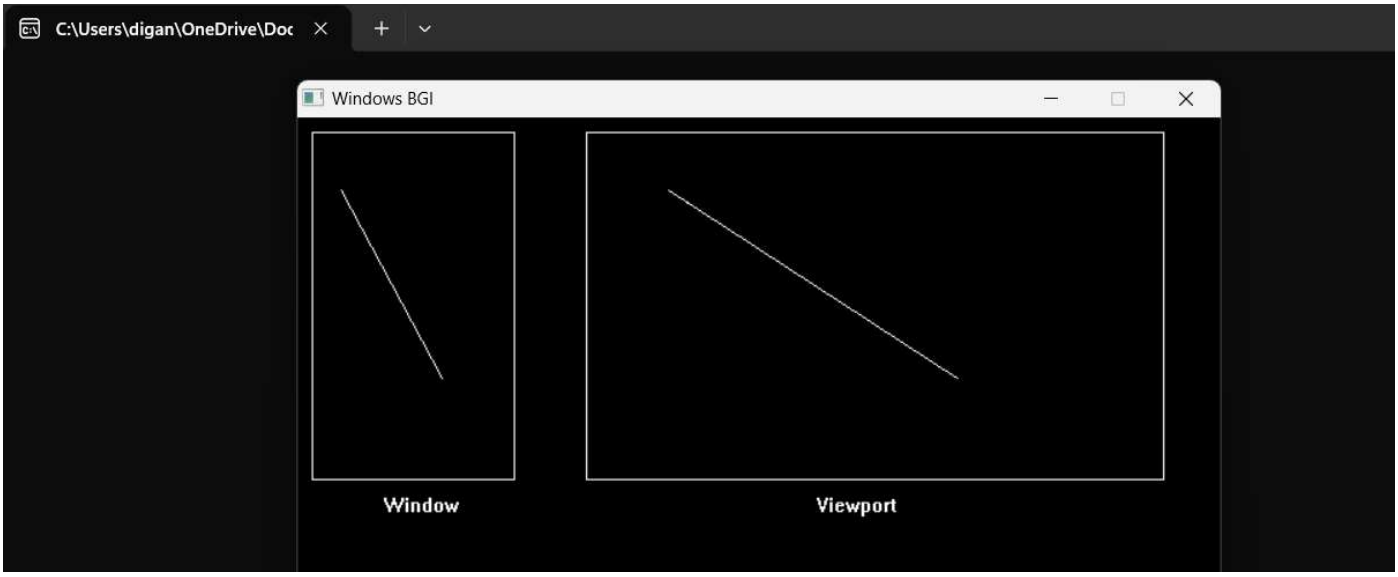
```
float vxmin = 200, vxmax = 600, vymin = 10, vymax = 250;
int wx1 = 30, wy1 = 50, wx2 = 100, wy2 = 180;

// Draw window and viewport
rectangle((int)wxmin, (int)wymin, (int)wxmax, (int)wymax);
rectangle((int)vxmin, (int)vymin, (int)vxmax, (int)vymax);

// Scaling factors
float sx = (vxmax - vxmin) / (wxmax - wxmin);
float sy = (vymax - vymin) / (wymax - wymin);
// Draw original line
line(wx1, wy1, wx2, wy2);
// Transform coordinates
int vx1 = (int)(sx * (wx1 - wxmin) + vxmin);
int vy1 = (int)(sy * (wy1 - wymin) + vymin);
int vx2 = (int)(sx * (wx2 - wxmin) + vxmin);
int vy2 = (int)(sy * (wy2 - wymin) + vymin);
// Draw transformed line in viewport
line(vx1, vy1, vx2, vy2);
// Labels
outtextxy(60, 260, "Window");
outtextxy(360, 260, "Viewport");
getch();
close graph();
```

```
return 0;
}
```

Output: -



4. Learning Outcomes:-

- Understand the concept of window and viewport in computer graphics.
- Learn how to apply scaling for coordinate transformation.
- Gain hands-on experience using `graphics.h` for drawing and mapping.
- Visualize how objects are transformed from logical to screen space.