SSN COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Name : Rahul Ram M
Reg .No : 185001121
Date : 09/09/2021

EX NO: 6b

Window to Viewport Mapping

Aim:

To create an object for the window. To create a view port of size smaller than the window. To apply Window to viewport transformation of the object.

Algorithm:

1. Initialize 2 integer arrays. One for storing x coordinates and the other for storing y coordinates.
2. Read xw_min, xw_max, yw_min, yw_max, xv_min, xv_max, yv_min and yv_max from the user for the min and max, window and viewport size respectively.
3. Find sx = (xv_max - xv_min) / (xw_max - xw_min) and sy = (yv_max - yv_min) / (yw_max - yw_min).
4. For each vertex on the object:
    a. Change the x coordinate as xv_min + (x[i] - xw_min) * sx.
    b. Change the y coordinate as yv_min + (y[i] - yw_min) * sy
5. This will change the size of the object with respect to the viewport.
6. Now create two windows - one based on the input given for the window and the other for the viewport.
7. Now draw the objects in both the windows based on the original and the modified input.

Code:

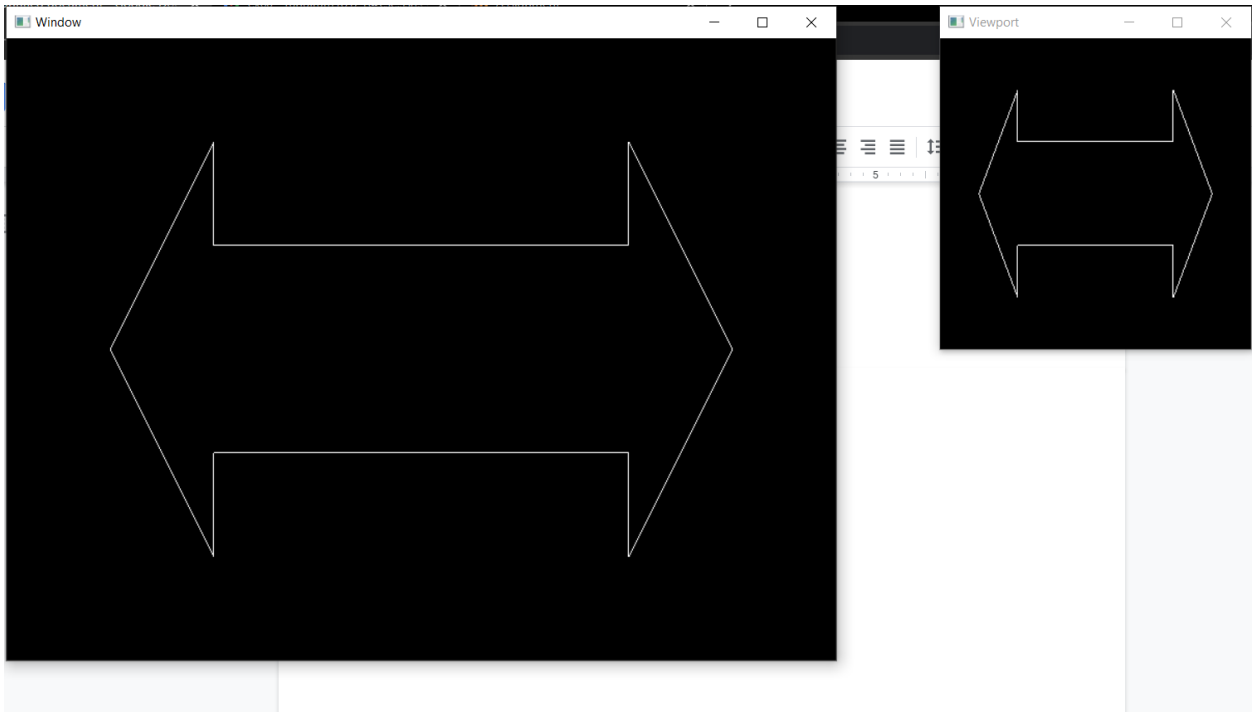
```
#include <windows.h>
#include <gl/glut.h>
#include <math.h>
#include <iostream>
#include <vector>

using namespace std;

float x[10] = { 100, 200, 200, 600, 600, 700, 600, 600, 200, 200 };
float y[10] = { 300, 100, 200, 200, 100, 300, 500, 400, 400, 500 };

float X[10];
float Y[10];

float xw_min, yw_min, xw_max, yw_max, xv_min, xv_max, yv_min, yv_max;

void myInit(int flag)
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glPointSize(1);
    glMatrixMode(GL_PROJECTION);
    if (flag == 0)
        gluOrtho2D(xw_min, xw_max, yw_min, yw_max);
    else
        gluOrtho2D(xv_min, xv_max, yv_min, yv_max);
}


void displayWindow(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINE_LOOP);

    for (int i = 0; i < 10; i++)
    {
        glVertex2f(x[i], y[i]);
```

```
    }
    glEnd();

    glFlush();
}

void displayViewport(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINE_LOOP);

    for (int i = 0; i < 10; i++)
    {
        glVertex2f(X[i], Y[i]);
    }
    glEnd();

    glFlush();
}

void windowToViewPortMapping()
{
    float sx = (xv_max - xv_min) / (xw_max - xw_min);
    float sy = (yv_max - yv_min) / (yw_max - yw_min);

    for (int i = 0; i < 10; i++)
    {
        X[i] = xv_min + (x[i] - xw_min) * sx;
        Y[i] = yv_min + (y[i] - yw_min) * sy;
    }
}

int main(int argc, char** argv)
{
    cout << "Window:\n";
    cout << "X min : "; cin >> xw_min;
    cout << "Y min : "; cin >> yw_min;
    cout << "X max : "; cin >> xw_max;
```

```
    cout << "Y max : "; cin >> yw_max;

    cout << "Viewport:\n";
    cout << "X min : "; cin >> xv_min;
    cout << "Y min : "; cin >> yv_min;
    cout << "X max : "; cin >> xv_max;
    cout << "Y max : "; cin >> yv_max;

    glutInit(&argc, argv);                  // Initialize GLUT
    glutInitWindowSize(xw_max, yw_max);   // Set the window's initial
width & height
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Window");
    myInit(0);
    glutDisplayFunc(displayWindow);

    windowToViewPortMapping();
    glutInitWindowSize(xv_max, yv_max);   // Set the window's initial
width & height
    glutInitWindowPosition(xw_max+150, 50);
    glutCreateWindow("Viewport");
    myInit(1);
    glutDisplayFunc(displayViewport);

    glutMainLoop();
    return 0;
}

// 0 0 800 600 0 0 300 300
```

Input/Output Screenshot:

```
Window:
X min : 0
Y min : 0
X max : 800
Y max : 600
Viewport:
X min : 0
Y min : 0
X max : 300
Y max : 300
```



Result:

Thus window to viewport mapping is acheived using c++ and opengl.