CS 460
HW-2
Selin DINC

Problem Statement

- Creating a menu that includes polygon clipping, region filling and window to viewport mapping options.
- Drawing a clipper window in the rectangular shape with dashed lines.
- Drawing a polygon around the clipper window by using the mouse.
- Being able to clip a polygon by drawing the polygon around the clipper window when the user selects the "polygon clipping option". Using Sutherland_Hodgeman algorithm to clip.
- Being able to fill the clipped polygon when the user selects the "region filling option". Using the Boundary Fill algorithm.
- Drawing a viewport and mapping the clipped polygon from the window to the viewport.
- Being able to change the viewport size by using the mouse-click to drag the bottom-right corner of the viewport and displaying the scaling effect of the clipped polygon at the same time.
- Being able to change the window size and map the clipped polygon from the wondow to viewport dynamically by achieving zoom-in and zoom-out effects.
- Being able to move the window horizontally or vertically and map the clipped polygon from the window to the viewport by achieving the panning effect.

Algorithm Design

There are **seventeen** functions for the program; **display**, **boundaryFill4**, **polygonFill**, **clip**, **drawWindow**, **x_intersect**, **y_intersect**, **suthHodgClip**, **drawViewport**, **windowToViewMapCalc**, **windowToViewMap**, **motionForViewPortAndWindow**, **motion**, **make_menu**, **menu** and **mouse**.

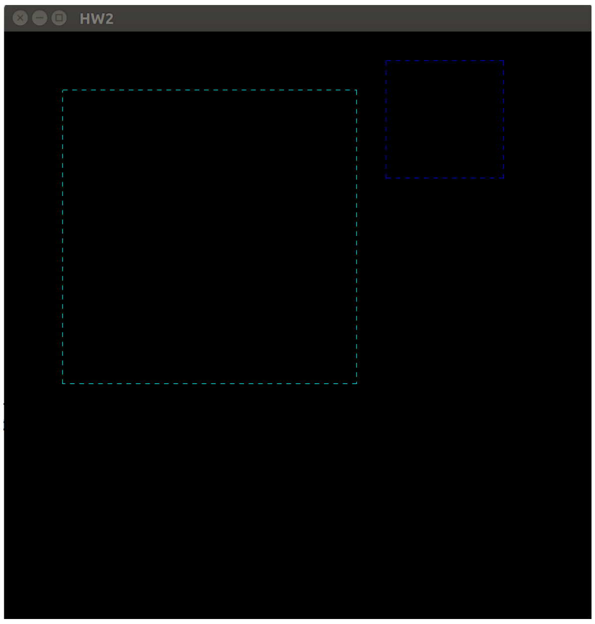


Figure 1: Window and Viewport

Program creates a window that shows a clipper window at the left side of the window and a viewport at the right side of the window. Also window displays a menu when the user clicked the right click of the mouse. Menu contains clip polygon, fill polygon and window-to-viewport map options.

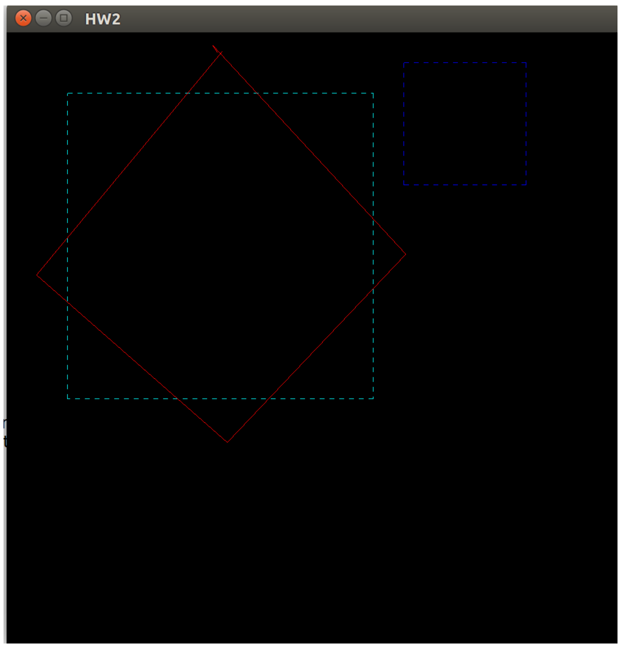
- **display** function displays the clipper window, viewport and mouse movements dynamically on the window by calling the **drawWindow** and **drawViewport** functions.

- **drawWindow** function draws the window by using glLineStipple function.

- **drawViewport** function draws the viewport by using glLineStipple function.

- **clip** function takes the clipper window points and the points that creates the polygon and checks four situations that decides the place of the polygon's points related to the clipper window and then reacts accordingly. Situations that the algorithm checks if the polygon points is; **in-in** then only add the second point to the output list, **out-in** then both the point of intersection of the edge with the clipper boundary and the second point are added to the output list, **in-out** only the point of intersection of the edge with the clip boundary is added to the output list, **out-out** then no points added to the output list.

- **x_intersect** function returns x-value of point of intersectipn of two lines.
- **y_intersect** function returns y-value of point of intersectipn of two lines.



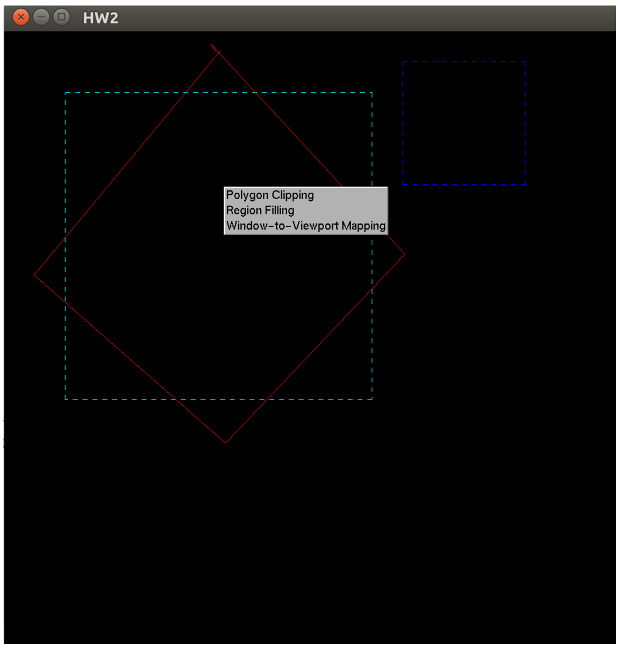


Figure 2,3: Drawing a polygon and opening the Menu

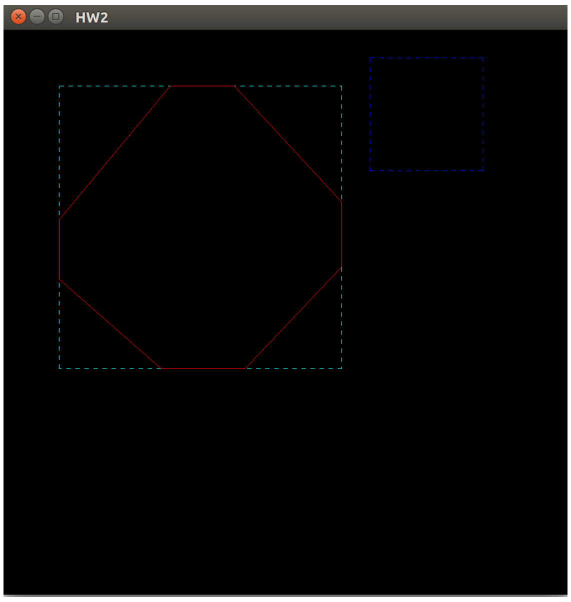


Figure 2: Output of the clipped polygon

- **suthHodgClip** function converts the polygon lines to polygon points the calls the clip function by iterating every side of the clipper window then converts the polygon points to polygon lines.
- **boundaryFill4** function starts at a pixel inside the polygon to be filled and paints the interior proceeding outwards towards the boundary. If the boundary is of one single color, this approach proceeds outwards pixel by pixel until it hits the boundary of the region. This is a recursive algorithm. It takes an interior point(x, y) this point is send by the Mouse left click. Then it creates a size 3 pixel buffer that indicates the RGB colors. Then read the pixel's color. If the pixel's color is not equal to the fill color and the boundary color, then it is painted with the fill color and the function is called for all the neighbours of (x, y). If a point is found to be of fill color or of boundary color, the function does not call its neighbours and returns. This process continues until all points up to the boundary color for the region have been tested.
- **polygonFill** function calls the **boundaryFill4** function.

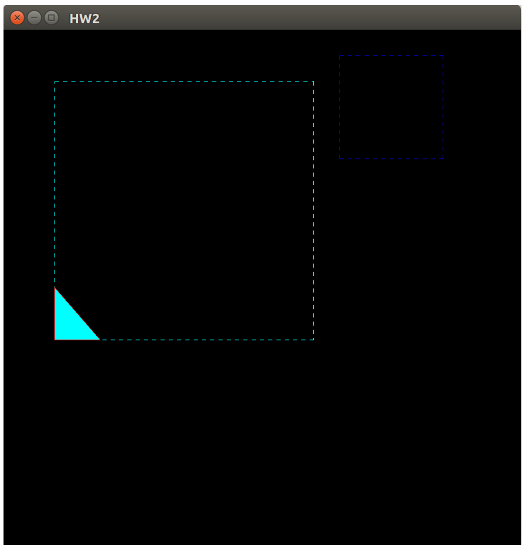
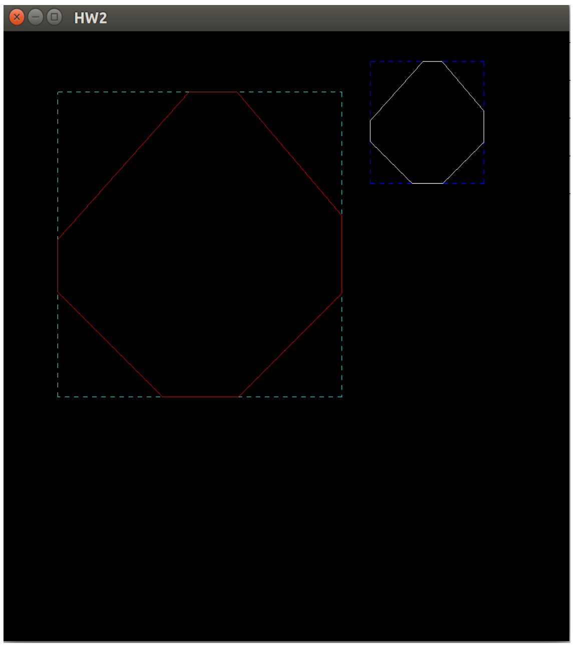


Figure 4: Fill the clipped polygon

- **windowToViewMapCalc** function calculates the new points of the polygon in the viewport.

- **windowToViewMap** function calls the **windowToViewMapCalc** function for all the lines of the polygon.



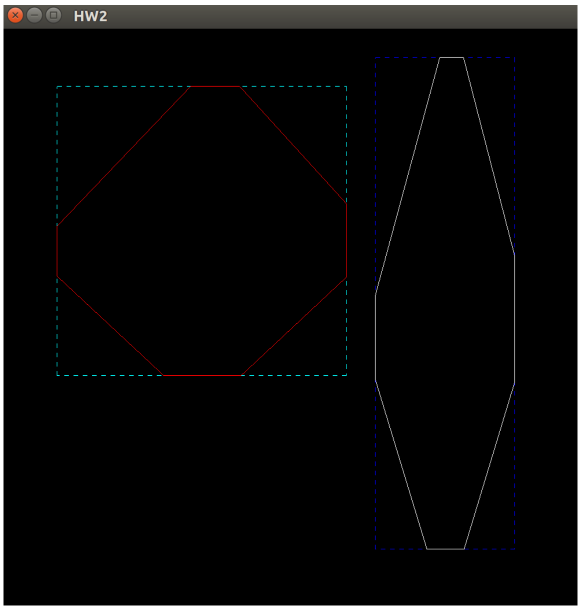


Figure 5,6: Window-to-viewport map and drag the bottom right corner of the viewport

- **Motion** function checks if the left button clicked and makes the motion possible by using glutPostRedispay function of openGL.
- **motionForViewPortAndWindow** function checks if the bottom-right of clipper window clicked, bottom-right of the viewport clicked or any boundary lines from both the clipper window and viewport clicked. Then according to the situation it makes possible the simultaneous movement between window and viewport possible by calling the **windowToViewMap** function.

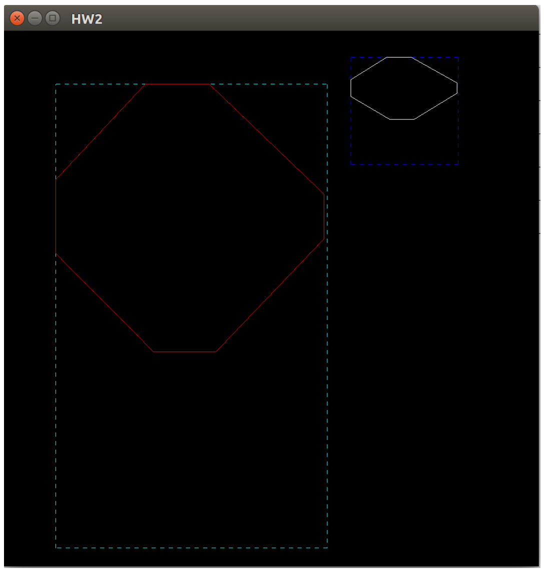


Figure 7: Drag the bottom-right of the clipper window

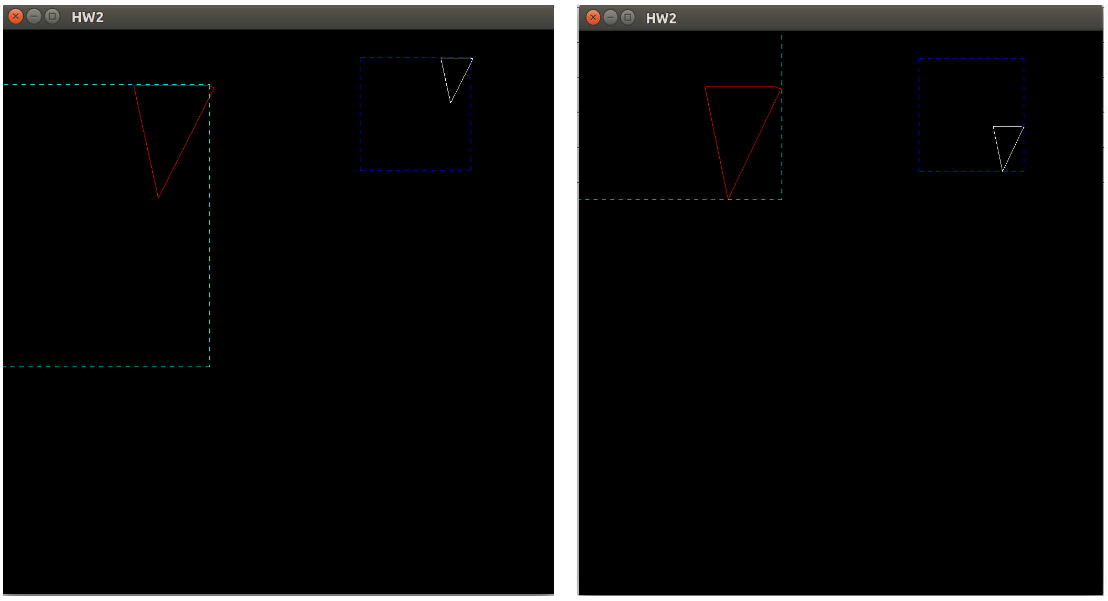


Figure 8,9: Move the clipper window horizontally and vertically

- **Mouse** function sets the line points and then pushes to a vector that consist lines with each different mouse button click.
- **Menu** function sets the menu options.
- **Make_menu** function creates the menu.

How To Run

```
g++ main.cpp -o a.out -framework OpenGL -framework GLUT
```

Note!

If you run my code on MacOS it has an y-axis offset on the Mouse. However, in Ubuntu everything works properly without any offset. I couldn't fix the offset problem in MacOS.