

Nicholas Vallejos

CS460

Assignment 2 Write-Up

**Problem Statement:** Implement Sutherland-Hodgman clipping algorithm, boundary fill algorithm, and a window-to-viewport transformation.

### Algorithm Design:

#### Sutherland-Hodgman:

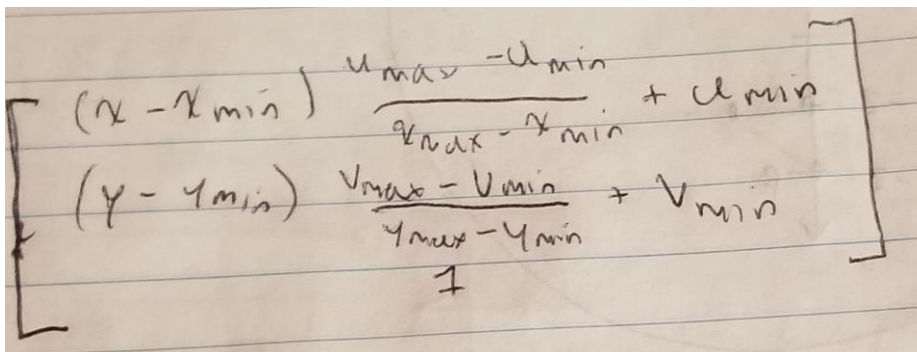
The Sutherland-Hodgman clipping algorithm relies on two separate vectors. One vector holds the vertexes of the clipping window. The second vector holds the vertexes of the polygon to be clipped. The basic idea is to extend each edge of the clipping window infinitely and then perform calculations to determine which polygon vertexes lie within the clipping window with respect to the clipping edge. After iterating over all edges on the clipping window, the polygons vertexes will have been recalculated such that all vertexes lie within the clipping window.

#### Boundary Fill Algorithm:

The boundary fill algorithm takes 4 inputs: x-coordinate, y-coordinate, fill color value, boundary color value. Using these inputs, the algorithm recursively checks 4 neighboring pixels color value. If the color value does not match the fill color value nor the boundary color value, it will set the pixel at (x,y) equal to the fill color value.

#### Window-to-Viewport Algorithm:

The window-to-viewport algorithm follows a simple formula to calculate the new vertexes of the clipped polygon. It requires the world min and max coordinates (minX, minY) and (maxX, maxY). It also requires the viewport min and max coordinates (uMinX, vMinY) and (uMaxX, vMaxY). Using the following transformation matrix M, I can map the clipped polygons vertexes to the correct viewport coordinates:


$$\begin{bmatrix} (x - x_{min}) \frac{u_{max} - u_{min}}{x_{max} - x_{min}} + u_{min} \\ (y - y_{min}) \frac{v_{max} - v_{min}}{y_{max} - y_{min}} + v_{min} \\ 1 \end{bmatrix}$$

### Important notes regarding my code:

- **SutherlandHodgmanClipping()** - Performs the actual clipping algorithm on the polygon
- **boundaryFill()** - Performs the recursive boundary fill algorithm
- **WindowToViewportMapping()** - Performs the window-to-viewport mapping calculation
- **DoMenuSelection()** - Handles menu events.
- **The important global variables are the vectors**
  - **ControlPoints** – original polygon drawn using mouse clicks
  - **ClippingRectanglePoints** – holds the clipping window vertexes
  - **ClippedPolygonPoints** – holds the clipped polygon vertexes (i.e. the output vertexes from the SutherlandHodgmanClipping() function)
  - **ViewportPolygonPoints** – holds the window-to-viewport mapped polygon vertexes calculated in the WindowToViewportMapping()
- **Other important functions:**
  - **Redraw()** - this function will clear the screen and redraw the original polygon or the clipped polygon or the viewport mapped polygon.
  - **Redraw2()** - just a duplicate of the above that doesn't clear the screen. It's a helper function for the passiveMotionFunc()
  - **ComputeIntersection()** - this is a helper function for the SutherlandHodgmanClipping() algorithm. It computes the intersection between a polygon edge and a clipping edge when needed.

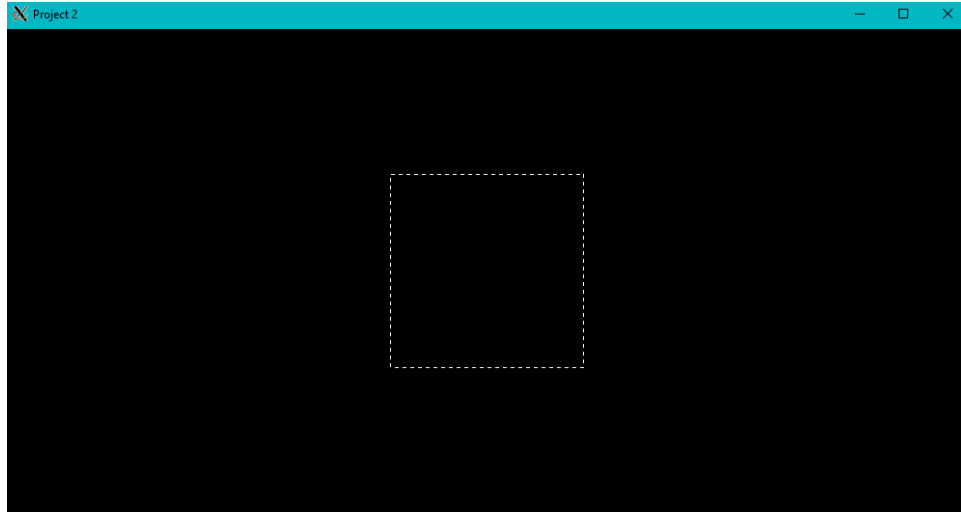
### Features that I couldn't quite implement:

- Zoom-in/Zoom-out
- Panning

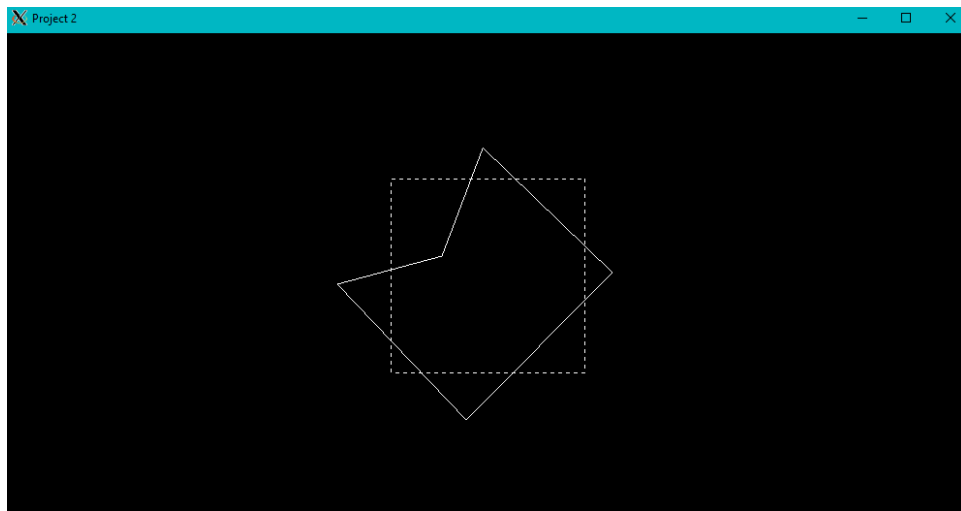
### Miscellaneous:

- Please review my README.md doc for compiling and running.

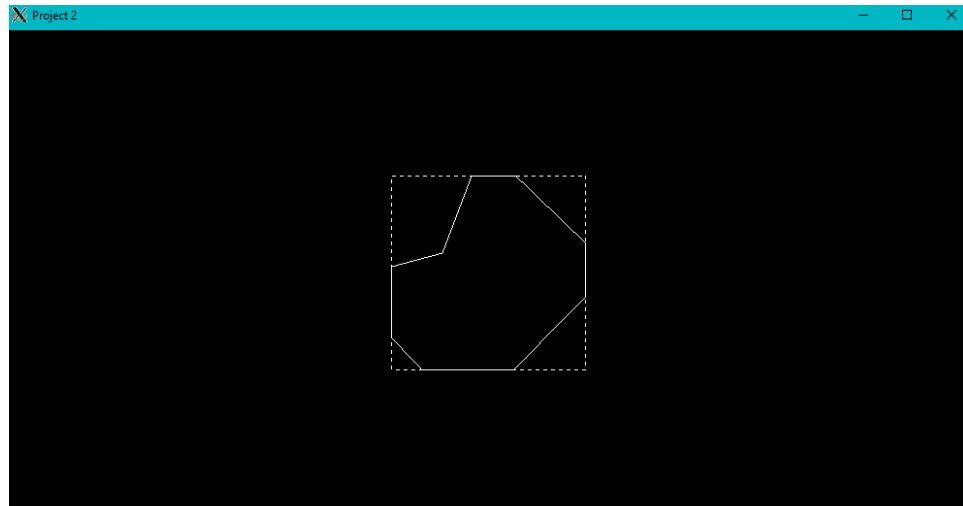
### Pictures:



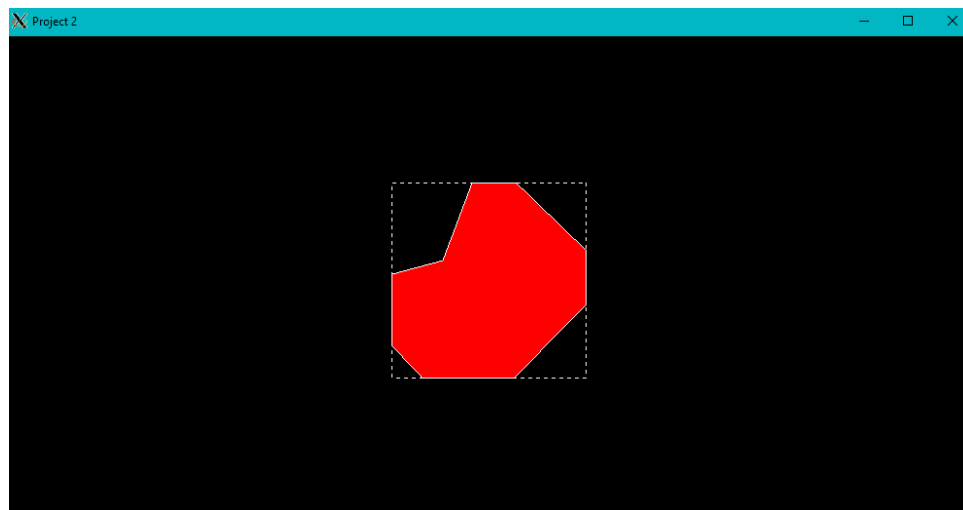
**Figure 1. Result of creating the clipping box from the menu selection:**



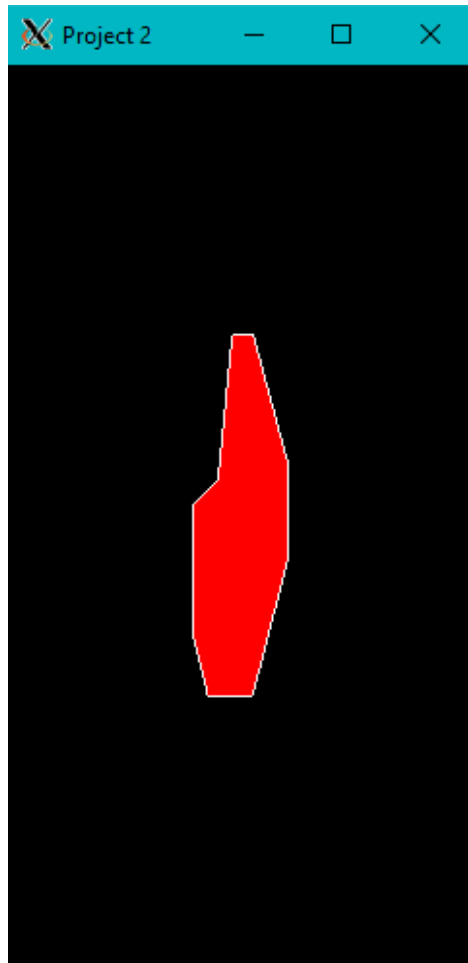
**Figure 2. Polygon Drawn over the clipping box using dynamic line drawing.**



**Figure 3. Result of calling SutherlandHodgmanClipping() function.**



**Figure 4. Calling Region Fill and clicking on inside of clipped polygon.**



**Figure 5. After calling WindowToViewportMapping() function and scaling up by dragging viewport window.**