

## CS770/870 Assignment 2

Due date: Tuesday, September 20th, 2021, before midnight.

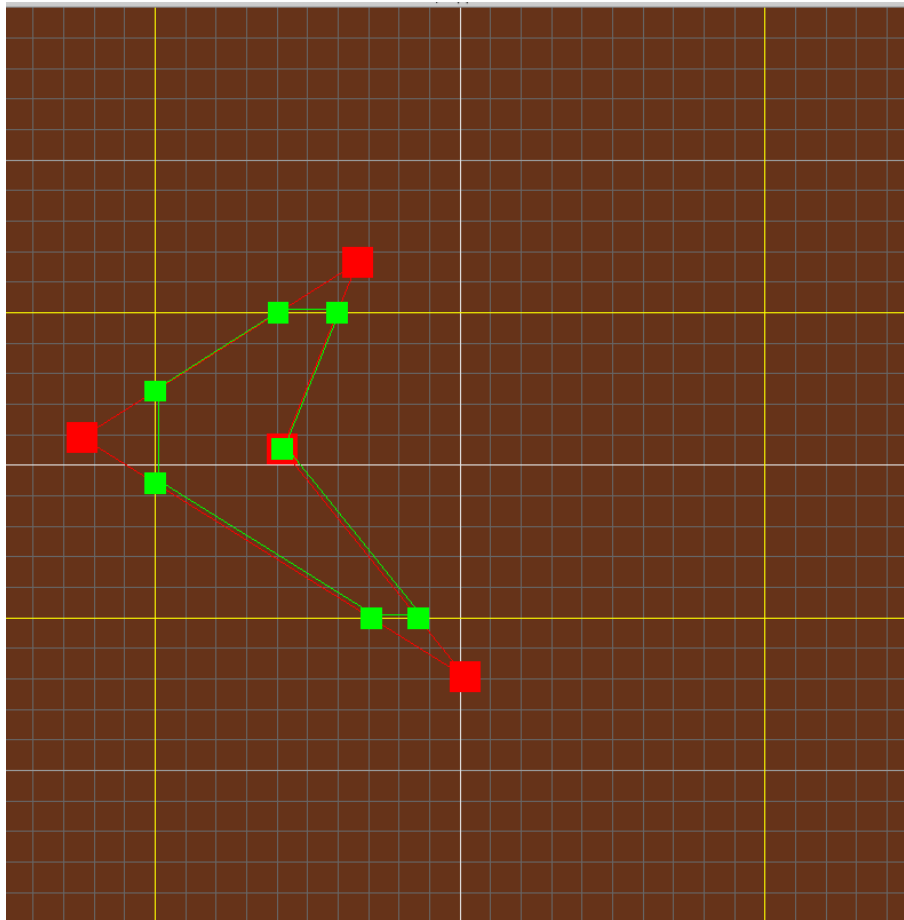
Lateness Penalty: Wed: 5% Thu: 10% Fri: 20% Sat/Sun/Mon: 50% Tue: 100%

### Goals

In this assignment, you will gain more experience with the [GLFW](#) event model and 2D graphics. You will implement both the Sutherland-Hodgman polygon clipping algorithm, and a pixel-rasterizing algorithm.

### Tasks

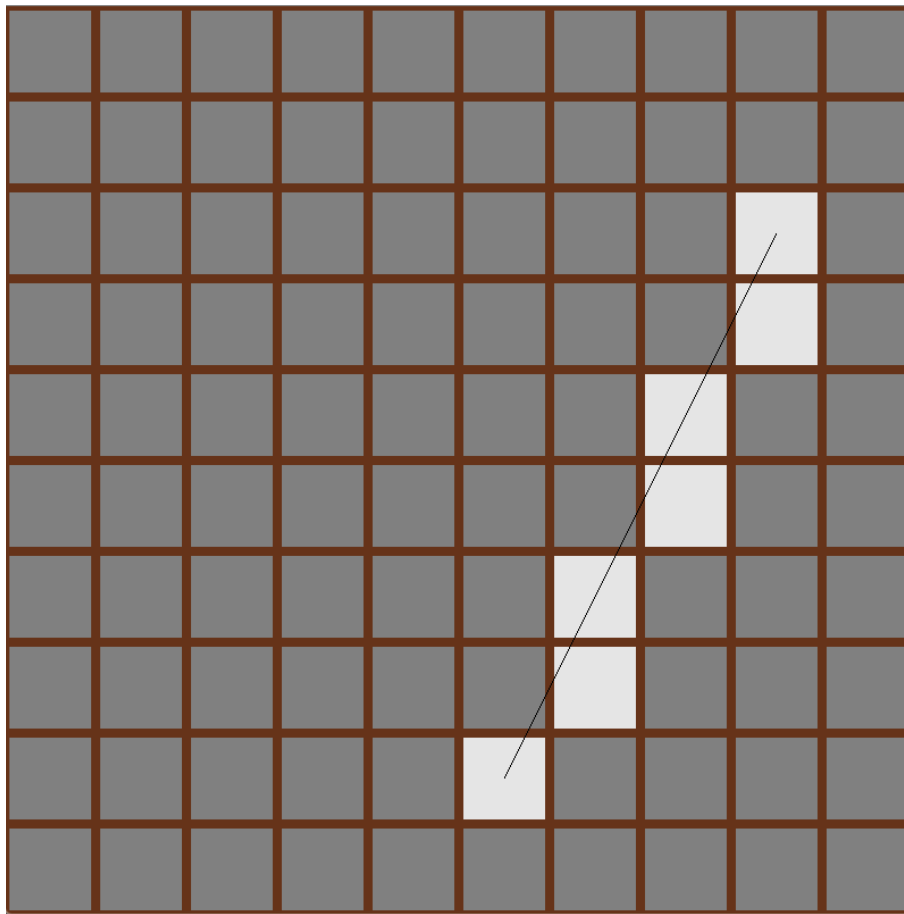
1. [50 points] Write a program that demonstrates the Sutherland-Hodgman clipping algorithm. It should implement the following behavior:



**Figure 1:** .

- There is a clipping window from  $x_{\text{left}}=-1$  to  $x_{\text{right}}=+1$ , and  $y_{\text{bottom}}=-0.5$  to  $y_{\text{top}}=+0.5$ . This window should be drawn with yellow lines that extend past the sides of the window.
  - Each click in an empty region will add another  $(x, y)$  vertex to the current polygon. The polygon is stored as a sequence of  $(x, y)$  points, and each click adds a point to the end of this sequence.
  - Each polygon vertex is drawn as a  $0.1 \times 0.1$  red square, and polygon edges are drawn as red line segments.
  - A click on an existing vertex starts dragging it.
  - Whenever the polygon is changed (by adding a new vertex, or dragging an existing one), a clipped polygon should be obtained. This polygon should be drawn with green lines and green  $0.05 \times 0.05$  vertices.
2. [50 points] Implement a program that demonstrates a line rasterization algorithm. It should

implement the following behavior:



**Figure 2:** .

- The window contains a 10×10 grid of grey squares, each one 0.9×0.9 .
- When the user clicks somewhere, and drags the mouse, the program will draw a black line segment that begins at the mouse-down pixel, and ends at the pixel where the mouse currently is.
- When the user releases the mouse, the line should remain, but should not be updated.
- The endpoints of the line segment should be at the centers of the corresponding pixels.
- *Rasterize*: each pixel that the line is closest to should be drawn in white instead of gray.

### **New! Improved!**

I'm supplying starting code. The code includes these changes:

- The `Mesh` class now has an `update_vertex` method, which lets you modify the coordinates of one vertex. You can now re-use `Mesh` objects, such as ones that are drawn with `GL_LINE_STRIPs`. You no longer have to re-allocate and delete `Mesh` objects.
- The `add_vertex` function (which adds an x y z point to an array of coordinates, with an associated element index), has been moved to the `util.cpp` file.

## Submitting Your Work

When you are done, go to [mycourses.unh.edu](https://mycourses.unh.edu), find the course, and the assignment. Then click [Submit](#), and upload both `clipper.cpp` and `rasterizer.cpp`.