CS 430 – Computer Graphics
Fall 2016
Assignment 4 – Filling Polygons

From your previous assignment(s) in this assignment you should be able to:

1. Read in data from a file pertaining to creating polygons, and organize that data into structures for drawing polygons.
2. Set up a software frame buffer to render to
3. Clip polygons to a clipping window.
4. Draw clipped polygons using a line drawing algorithm to render the lines into the software frame buffer.
5. Output the software frame buffer as an XPM file.

In addition, in this assignment you should be able to:

6. Fill polygons

Make sure you give yourself adequate time. The programming components in particular can be quite time consuming.

As a reminder you may use the programming language of your choice, though I recommend C/C++ and also make sure that your program can run on the Drexel tux cluster to insure its system independence.

Submission Guidelines

1. Assignments must be submitted via Bd Learn
2. Submit a single compressed file (zip, tar, etc..) containing:
    a. A PDF file with your solutions to the theory questions.
    b. A README text file (**not** Word or PDF) that explains
        i. Features of your program
        ii. Language and OS used
        iii. Compiler or interpreter used
        iv. Name of file containing main()
        v. How to compile/link your program
    c. Your source files and any necessary makefiles, scripts files, etc… to compile and run your program.

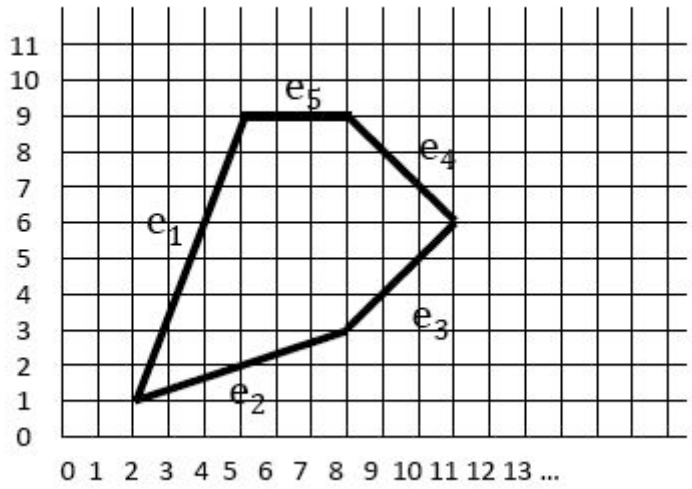
Theory Question(s):

1. On the image below
    a. Where are extrema points for scan-line 6 (assuming we're scanning from bottom to top)? (2pts)

       The extrema points are points that scanline intersects with polygon lines so in this case, they are (4,6) and (11,6)

    b. Describe how scan-line 6 will draw as it goes from x=0 to x=13. For each sequential value of x describe if the pixel should be filled using the parity bit-flip technique. (5pts)

       The scanline first check the intersects with the polygon line on y = 6. So in this case it will be (4, 6) and (11,6). Sort the x in increasing order and draw from lower x to upper max. If the x is within the two x values, use bit flip, meaning 0 is true and should draw until it is false 1. The pixels should be (4,6), (5, 6), (7,6) (8,6), (9,6), (10, 6) , (11,6)



Assignment Details

The only addition in this assignment is that you will fill polygons from the prior assignment using

Write a program that accepts the following command arguments. Defaults are in parenthesis. You should be able to process any subset of the options in any arbitrary order.

a. [-f] The next argument is the input "Postscript" file (hw4_1.ps)

For now we will hard-code the size of your frame buffer to be 500x500 (that is 500 pixels wide by 500 pixels high).

Your program should print output images in the XPM file format to stdout (`cout`) such that it can be piped to a file. All pixels should be initialized to white. Draw your objects in black.

Your general program flow should be:

1. Read in polygon specifying data (PS)
2. Clip the polygons to the frame buffer clipping window
3. Draw the **filled** polygons by scan filling the clipped polygons into software frame buffer
4. Output your image (the frame buffer and header information necessary to make an XPM file) via `cout`

Grading Scheme:

In order to get any credit at all you must be able to generate at least read in a PS file and write out a valid XPM file

1. Theory Question(s) (7pts)
2. Can read in PS file and output a valid XPM file (33pts)
3. Successfully handles basic single polygon test cases (5 @ 7pts = 35pts)
4. Successfully handles a two-polygon test cases (4 @ 3pts = 12pts)
5. Successfully handles a many-polygon test case (3pts)
6. Program easy to compile and run based on README (10pts)

Common deductions:

1. Nothing drawn (-100pts)
2. Missing files (including readme) (-5pts each)
3. Cannot compile/run out-of-the-box on TUX (-10pts)

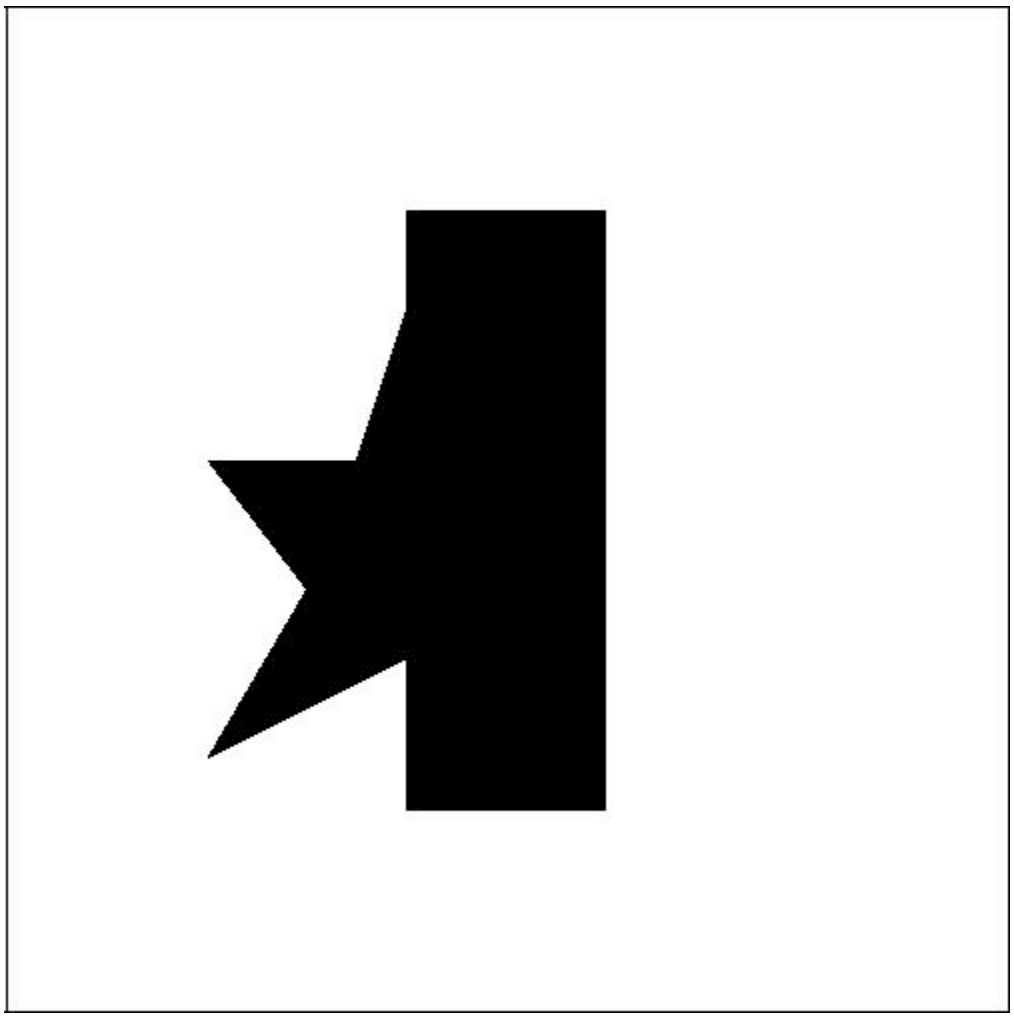
Provided Tests

The provided tests are essentially filled versions of the previous assignment:

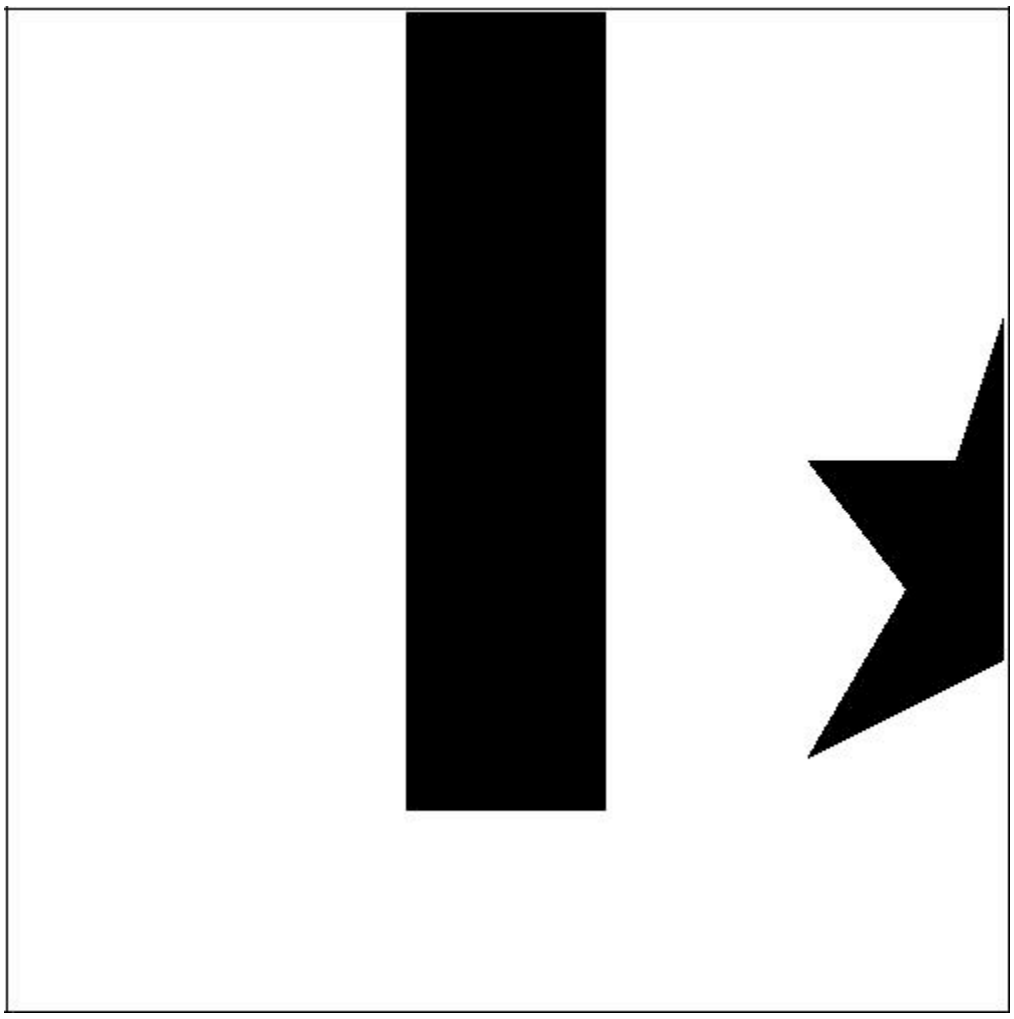
1. hw4_1.ps – Filled star
2. hw4_2.ps – Filled rectangle clipped to the top
3. hw4_3.ps – Rectangle totally out of the clipping window
4. hw4_4.ps – Filled star clipped to the right edge
5. hw4_5.ps – Filled star clipped to the top and right edges
6. hw4_6.ps – Star and rectangle drawn together (non-overlapping)
7. hw4_7.ps – Star and rectangle drawn together (overlapping)
8. hw4_8.ps – Star and rectangle drawn together with clipping
9. hw4_9.ps – An example with a bunch of polygons

Here's some of the more interesting results:

./A4 -f hw4_7.ps > out7.xpm



./A4 -f hw4_8.ps > out8.xpm



./A4 -f hw4_9.ps > out9.xpm