CS 430 – Computer Graphics
Fall 2016
Assignment 2

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

1. Read in data from a file and organize it into structures for drawing
2. Set up a software frame buffer to render to
3. Implement a line drawing algorithm to render the lines into the software frame buffer.
4. Output the software frame buffer as an XPM file.

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

5. Draw lines that are outside of the clipping window by first clipping them to the clipping window.

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 containing your answers 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. Assume the clipping region is a rectangle from (2,2) to (8,8). Compute the new endpoints of the line segment specified by endpoints (10,7), (1,5) using the Cohen-Sutherland algorithm. Show your work. Include the clipped integer locations in your final answer. (8pts)

   X1 = 10
   Y1 = 7

   X0 = 1
   Y0 = 5

   Right Clip:
   y = y0 + (y1 - y0) * (xmax - x0) / (x1 - x0)
   = 5 + (7-5) * (8 -1) / (10-1)
   = 5 + 2*(7/9)
   = 6
   x = xmax = 8

   Left Clip:
   y = y0 + (y1 - y0) * (xmin - x0) / (x1 - x0)
   = 5 + (7-5) * (2-1) / (10 – 1)
   = 5
   x = xmin = 2

   Final Right point (8, 6)
   Final Left point (2,5)

2. Repeat what you were asked to do in the previous problem, but using the Cyrus-Beck Technique. Include the clipped integer locations in your final answer. (8pts)

   P0 = (1, 5) and P1 = (10, 7)

Edge	Ni	PE(i)	P0 – PE(i)	t = Ni (P0 – PE(i))/ - Ni (P1 – P0)
Left X = xmin	(-1,0)	(2, 7)	((1 -2),(5-7)	t = -(1-2)/(10-1)

Right X = xmax	(1,0)	(8,7)	((1-2), (5-7))	t = -(1-2)/ (10-1)

P0 (1 –t) + P1 * t - PE = (2,-7) + t(2,10)

Right point = (8,6)
Left point = (2,5)

3. Use Bernstein Polynomials to define the basis matrix of a Bezier curve with 3 control points and using polynomials of degree 2 (4pts)

Bernstein Polynomials of degree n

$$B_n(y,x) = \frac{1}{(b-a)^n} \sum_{i=0}^{n} \binom{n}{i} (x-a)^i (b-x)^{n-i} y(x_i)$$

where $\binom{n}{i}$ are the binomial coefficients

and $x_i = a + i\frac{b-a}{n} \quad 1 \le i \le n$

Since $[a,b]$ can be transformed to $[0,1]$ by using linear transformation

$$z = \frac{x-a}{b-a} \quad \text{and} \quad x = a + (b-a)z$$

$z \in [0,1]$ for $x \in [a,b]$, substituting z for x

$$B_n(y,z) = \frac{1}{(b-a)^n} \sum_{i=0}^{n} \binom{n}{i} (b-a)^i z^i (b-a)^{n-i} (1-z)^{n-i} y(z_i)$$

$$= \sum_{i=0}^{n} \binom{n}{i} z^i (1-z)^{n-i} y(z_i)$$

where $z_i = \frac{x_i - a}{b-a} = \frac{i}{n} \quad 1 \le i \le n$

Rewritten

$$B_n(y,x) = \sum_{i=0}^{n} \binom{n}{i} x^i (1-x)^{n-i} y\left(\frac{i}{n}\right) = \sum_{i=0}^{n} B_{n,i}\, y\left(\frac{i}{n}\right)$$

$x \in [0,1]$

So $B_{n,i} = \binom{n}{i} x^i (1-x)^{n-i} \quad 1 \le i \le n$

$x \in [0,1]$

4. Given control points at P1=(2,4), P2=(4,2), P3=(5,7), P4=(0,0)
    a. What is the matrix G for the Bezier curve? (1pt)

    b. What is the matrix M for the Bezier curve? (1pt)

    c. Compute Q=GMT (3pts)

    d. What is Q(0.5)? (1pt)

$P1=(2,4)$ $P2=(4,2)$, $P3=(5,7)$, $P4=(0,0)$

$Q(t) = G_B \cdot M_B \cdot T$

$$G_B = [P_1\ P_2\ P_3\ P_4]$$
$$= [2\ 4; 4\ 2; 5\ 7; 0\ 0]$$

$$b_{ik}(t) = \binom{k}{i}(1-t)^{k-i}t^i$$

$$\binom{k}{i} = \frac{k!}{i!(k-i)!}$$

$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$Q = GMT$$
$$= [P_1\ P_2\ P_3 P_4] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

$$Q(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$$
$$Q(t) = (1-t)^3(2,4) + 3t(1-t)^2(4,2) + 3t^2(1-t)(5,7) + t^3(0,0)$$

$$Q(0.5) = (1-0.5)^3(2,4) + 3t(1-0.5)^2(4,2) + 3t^2(1-0.5)(5,7) + 0.5^3$$
$$= 0.5^3(2,4) + 1.5(0.5)^2(4,2) + 0.75(0.5)(5,7) + 0.125$$

$$Q(0.5) = (2 + 18t - 15t^2 - 5t^3,\ 4 + 18t - 3t^2 - 19t^3)$$
$$= (2 + 18(0.5) - 15(0.25) - 5(0.125),\ 4 + 18(0.5) - 3(0.25) - 19(0.1$$
$$= (7.875,\ 14.625)$$

Assignment Details

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 (hw1.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 line segment data (PS)
2. Clip the vertices of each line segment to the frame buffer clipping window using the **Cohen-Sutherland** algorithm.
3. Scan covert (i.e. draw) clipped lines into software frame buffer
4. Output your image (the frame buffer and header information necessary to make an XPM file) via `cout`

Bonus

For ten additional points, allow the user to pass a `[-x]` flag to indicate to use Cyrus-Beck line clipping algorithm. If that flag is omitted, use the Cohen-Sutherland algorithm for line clipping.

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) (26pts)
2. Successfully handles basic single line test cases (4 @ 12pts = 48pts)
3. Successfully handles a multi-line test case (16pts)
4. Program easy to compile and run based on README (10pts)
5. Bonus (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

We have provided several incremental tests for you. We certainly recommend you create some of your own as necessary to test the robustness of your implementation (especially if you have issues going from the 4th test case to the 5th test case).

1. hw2_1.ps – This is a horizontal line that should clip to the left clipping plane
2. hw2_2.ps – This is a horizontal line that should clip to both the left and right clipping planes
3. hw2_3.ps – This is a line that is completely out of the clipping window and therefore nothing should be drawn.
4. hw2_4.ps – This is a diagonal line that should clip to two planes
5. hw2_5.ps – This is a "full-fledged" example with multiple lines, some of which need to be clipped.

Here is the output of a full-fledged test case (note: border is not included in the actual output):

./A2 -f hw2_5.ps > out5.xpm