MAT 494 — SpTp: Computer Graphics, Spring 2005

Homework #3, Due on Friday, February 4th

In this homework you will be extending your implementation of DDA and the Bresenham midpoint line rasterization techniques. Remember your program must compile and execute or you receive a 0.

Project Requirements:

1. Provide a method to draw many lines (up to 128) on the screen. (Hint: You could implement an array of lines by creating point and line classes).
2. Allow the user to delete the "top" line using the Delete key. (Hint: You may need to figure out the "value" of the delete key)
3. Allow the user to select a 'box' region using the right mouse button. When the user pressed the button down, store the coordinates of one corner of the box. When the user lets go of the button, store the opposing corner of the box. Sort these corners such that the lower x and y value form one pair of points, while the maximum x and y values are the other pair. Use GL_POLYGON to draw the four vertices defining the box in grey. AFTER the lines are drawn. This will draw a box 'on top' of the given lines, covering them. We'll use this box to clip against on assignment 4.
4. Allow the user to choose the line method at the command line by implementing a *-l* option. Use *-l g* for gl lines, *-l d* for dda lines and *-l m* for bresenham lines. Make the default mode be gl, if the user does not give a value at the command line.
5. For timing, use the *time* command in unix. For example, *time ./line* will execute the line program and print some timing statistcs.
6. Implement a *-n #* option which will draw a line from 0,0 to 500,500 a # number of times. This will allow for us to get meaningful results from the *time* command. Use *int atoi(char *)* in stdlib.h to convert from a character array to an integer. At the end of drawing for # greater than 1, the program should exit. Thus, no user input is required other than at the command line, and the program will exit itself. The default should be 1.
7. Record the times for executing each of the three algorithms using *-n 100000* and report them in your README.
8. Provide a README which discusses the difference in time between the three line drawing algorithms. Postulate on the reason for these differences. Did the results match your intuition? How might these results change on different systems?

Grading Information:

1. README Description, Questions and Answers: 50
2. User Interface/Implementation: 50