Mathew Partin

CMPEN 351

Final Project Design

In order to complete this project, I will need to achieve several tasks. I will be reading in mazes from a text file. I will store several (15-30) maze text files and pick randomly between them at run time. In order to read the mazes in I will read character by character, and store the data into an array in memory (in a two-dimensional fashion). When reading in the mazes, I will treat the following characters as a “wall”: |, +, - (These characters are representations coming from an online maze generator that I found). I will store all of these characters into memory in the same way. I have not decided how I will be storing this data as of yet. Perhaps it could be an integer flag, or it could be the character W (for wall). I’m not sure it will really matter. Any other space will be a blank space, the start or the end. Again, I have not decided how I will store these into memory as of yet, perhaps just a different character flag. Perhaps B, S, E respectively. Once that is complete I will have the data for the maze in memory. The next step will be to display the maze on the bitmap, which will fairly straightforward. I will make boxes of a certain size, maybe 16 pixels by 16 pixels (almost certainly will change those numbers). First, I will make the background of the bitmap white, then when I read a character, I will check if it is a wall. If it is, I will draw a black box, and move my location to the next spot on the bitmap display. This will leave me with a white background, black walls, and white paths in the maze.

After I have loaded the data from memory, and drawn the maze, I now can begin to solve the maze. I will be solving the maze with a depth first algorithm. In order to accomplish this, I will prioritize checking in the following order: right, bottom, left, top. The way my algorithm will work, I will start…at the start. Then I will check the nodes immediately adjacent to the current node. The current node will have a special denotation in memory, perhaps C (for currently being checked). Once I find a node that is not a wall, I will immediately change my current node to that one. I will then change its denotation in memory to C, without touching the previous nodes denotation. I will recursively repeat this process. In the case that I check all four directions and find only walls or nodes that are currently being checked, or nodes that are finished (which I will discuss momentarily) then I know that I have reached a dead end, and therefore, that path cannot be the solution. Therefore, I will mark that node as finished, perhaps with an F. I will then return to the previous node, and perform a check again. When I have reached the end, there will be a direct path formed to the finish from the start. The entire time I am performing this process, I will be updating the display to represent this information. When I am marking a node as currently being checked, I am going to display that as yellow. When I am marking a node as finished, I am going to display that as red. (On the bitmap).

In order to traverse through the input buffer for the input file, I will be forced to create a calcAddress similar to the one involved in the Simon lab. This will allow me to treat the one-Dimensional buffer as a two Dimensional Array that will be easy to work with.

I will be testing my code at several stages of the project. I will test that the input is working correctly. I will attempt to test each function individually, but in the event that that inevitably fails, I will perform tests at several key points of the project. I will test that the files are properly being read in by simply checking the memory. Then, I will test the initial drawing of the maze and make sure that it matches the pertinent maze. The next major stage I will test at is when I begin traversing, I will test first moving to the next node, and then marking the previous nodes status, and then beginning the backtracking. The final major stage that I will perform tests at is the final path tracing once the maze is solved. Of course, I will perform incremental tests between these, but these tests are the pivotal ones.