Assignment 4: WRITEUP.pdf

Katrina VanArsdale

February 12, 2023

1 Description

This program implements the Game of Life. The Game of Life is given an initial state and will change depending on how many neighbors each cell has. You can watch as the initial state changes from generation to generation. It's interesting to see how some shapes will change and grow or stay the same or disappear entirely after some generations. This write-up talks about what I learned while making this program.

2 Lessons learned and Thoughts

Going into this assignment I was interested to see what parts would be needed to recreate the Game of Life. It was interesting to be able to see how the game originally looks and see what I could do on my own. It was fun to implement how each cell interacts with its neighbors and to see which ones will die and come to life. First I went through the descriptions of each Universe function and wrote down my basic understanding of it as pseudo-code. Then I started on universe.c and worked on each function. I'm still not completely sure how structs work but I got a better understanding of how to affect the variables defined it.

2.1 uv_populate

At first, my uv_create function didn't work correctly because I didn't notice that there was a bool grid in the struct and I thought that we were supposed to edit the values of u[r][c]. But working through that and allocating the correct amount of space for the universe and the grid helped me better understand pointers.

2.2 uv_census

I also had a hard time wrapping my head around uv_census so I visualized the coordinates of each neighbor on a piece of paper. I have r,c in the middle and the neighbors were in a range from r-1 to r+1 and c-1 to c+1 so I used that to create some for loops and work from there. I ended up having a lot of trouble debugging it because I couldn't figure out why it wasn't calculating the correct amount of neighbors. The problem ended up being that in order to check that I wasn't looking at r,c I did if(i != r && j != c) but that was skipping a lot more than just when i == r and j == c. so I changed it to if(i == r && j == c) then continue else add to neighbors.

2.3 Standard Input

When I moved on to start life.c I couldn't figure out how to have the program take in a standard input as default. The problem was that I didn't understand that stdin could simply be read as a file and that it didn't have to be opened. But once I learned that with the help of the discord, I realized I could just set the file I used to read to stdin and move on. Then I had to figure out how to produce a Malformed input error. At first, my code would just change a single number like "3" to "3 0" and a letter like "f" to "0 0" so it didn't see it as an error. So then I changed it to a while loop so if fscanf is != 2 it'll print an error.

2.4 ncurses

When I first started setting up the ncurses window I thought I could take out two birds with one stone by doing the census and applying the game's rules all the same for loops. But then I realized that if ncurses was silenced then the program wouldn't do anything to the universe. So then I took out all of the ncurse stuff and put it in separate if statements. I also create a function that would update the ncurses window after every generation just to get my head around how to do it and not have my code look so confusing.

3 Conclusion

Overall, I gained a better understanding of how pointers, memory allocation, structs, and functions work. I feel a lot more comfortable making and using functions in C and have a better scope of what I'm able to do in C.