Assignment Four Design.pdf

By James Yim

Description of Program:

This program will simulate John Horton Conway's Game Of Life. The rules of the game are: any live cell within two or three live neighbors stays alive, any dead cell with exactly three alive neighbors becomes alive, and lastly all the other cells die due to loneliness or overcrowding. This program creates a universe and within the universe the game is played.

Files to be included in asgn4

- universe.c
    - includes the code and function definitions that make up what a universe is
- universe.h
    - the header file for universe.c. Shows what is needed for each function.
- life.c
    - This is the game that runs Conway's Game Of Life.
- Makefile
    - Tells the compiler how to compile the program
- README.md
    - Describes how to use the program, and how to deal with possible errors

- DESIGN.pdf
    - Describes the design of the program and the process of creating the program

Design & Pseudocode

Universe data type

struct Universe

```
{
        rows, columns, a grid/matrix, toroidal boolean
}
```

uv_create(rows, columns, toroidalBoolean)

```
{
        allocate memory for a Universe
        allocate memory for the grid using a for loop
        initialize struct data features
}
```

uv_delete(Universe)

```
{
        free up memory used by the Universe
}
```

uv_rows(Universe)

```
{
        return the amount of rows in Universe
}
```

uv_cols(Universe)

```
{
        return the amount of columns in Universe
}
```

```
uv_live_cell(Universe, row, column)
{
        sets the cell at position (row, column) to be alive
}


uv_dead_cell(Universe, row, column)
{
        sets the cell at position (row, column) to be dead
}


uv_get_cell(Universe, row, column)
{
        check the cell at position (row, column)
        if (alive)
        {
                return true;
        }
        return false;
}


uv_populate(Universe, inputFile)
{
        read inputFile
        first row of file, set row and column to those two numbers
        for each following line,
        {
                set that cell to be alive
        }
        if (successfully populated)
        {
                return true;
        }
        return false:
}
```

uv_print(Universe, outputFile)

```
{
        for every cell
        {
                check if its alive
                {
                        print("o") to outputFile
                }
                else
                {
                        print(".") to outputFile
                }
        }
}
```

Game of Life

```
main()
set opt variable
set toroidal boolean
set boolean for silence ncurses
set number of generations. Default is 100
set infile file. Default is stdin
set outfile file. Default is stdout

//Organizing user input
getopt() loop with options
        {
        if -t is set:
                set toroidal to be true
        if -s is set:
                set ncurses to silent
        if -n is set:
                set generations to user input
        if -i is set:
                set infile to user input
        if -o is set:
                set outfile to user input
        if -H is set:
                show help page
```

```
//creating universes
set row and col to be first line of infile
create a current universe with (row, col, toroidal boolean)
create a next universe with (row, col, toroidal boolean)
use populate() to populate the current universe with infile
close the infile

//Displaying the game
initiate ncurses screen
set ncurses cursor to false
loop through the amount of generations
{
        clear the ncurses screen
        loop through current universe
        {
                check current cell
                        {
                                if ncurses is enabled
                                        if alive print “o”
                                        if dead print “.”
                                check neighbors using census
                                {
                                        if cell has 2 or 3 neighbors and is alive keep it alive
                                        if cell is dead and has exactly 3 neighbors make it live
                                        otherwise kill all other cells
                                }
                        }
        }
        go to next frame using refresh()
        use usleep(DELAY)
        swap next universe to be the current universe
}
print the last universe using uv_print
free universes
```

Notes about code

- The populate() only reads files of a certain format. The format is: row col on the first line separated by a space. On the next subsequent lines we have coordinates for alive cells. following the same format as the first line.

Error Handling

- When an error happens as a result of a user they are directed to the -H page of ./sorting.
- If a file has incorrect format the program will still run.
- If an input file has a coordinate out of the range of the current universe, an error occurs and the program stops.

Credit

- Attended Eugene's Section on 1/28/22. Helped with how to iterate on a toroidal matrix and how to use ncurses.
- Discord user audrey helped with how to read files from getopt.