Assignment 4 The Game of Life

Katrina VanArsdale

February 9, 2023

1 Description of Program

This program implements the Game of Life into C. The game has three rules: Any live cell with two or three live neighbors survives, any dead cell with exactly three live neighbors becomes a live cell, all other cells die, either due to loneliness or overcrowding. The program creates a universe for the game to play in and implements the rules. The universe will use a ADT to create a 2D grid of cells.

2 Files In Directory

- universe.c
  This includes the implementation of the Universe ADT.
- universe.h
  This is the interface for the Universe ADT.
- life.c
  This contains main() and any other functions to complete this Game of Life.
- Makefile
  This file compiles all of the files and creates .o files for every .c file. It also cleans up all those files afterward and can clang format them.
- README.md
  This markdown file will describe how to use my program and Makefile. It also lists and explains the command line options that my program accepts.
- WRITEUP.pdf
  This file will include things I learned about sorting algorithms and graphs showing the differences between them.
- DESIGN.pdf
  This file describes the design for this program with psuedocode. This is the file you are reading.

3 Psuedocode

3.1 Universe

- Define functions
- Typedef struct Universe
  - Define rows and columns
  - Define a boolean point grid
  - Define a boolean called toroidal
- UV create
  - Takes in rows and cols and bool toroidal

  - if toroidal is true then the universe is toroidal
  - use calloc to dynamically allocate memory for universe
  - use calloc to dynamically allocate memory for row pointers
  - set struct variables to given values
  - Returns a pointer to the universe
- UV delete
  - Free the inside of the universe
  - Free the outside
  - Use valgrind to check for memory leaks
- UV rows
  - Define how many rows are needed for the specified Universe
- UV cols
  - Define how many columns are needed for the specified Universe
- UV live cell
  - Mark cell at row r and column c as live (True)
  - If out of bounds do nothing
- UV dead cell
  - Mark cell at row r and column c as dead (False)
  - If out of bounds do nothing
- UV get cell
  - Returns the value of the cell at row r and column c.
  - If out of bounds return False
- UV populate
  - The first line will be skipped by main()
  - Use fscanf() to read all the row-column pairs that contain live cells.
  - use uv live cell to set cell r,c to live
  - Return false if a pair is out of bounds and the universe failed to populate
- UV census
  - If the universe is toroidal then all neighbors are valid
  - loop through r-1 to r+1 and c-1 to c+1
  - check if i and j are within bounds if not toroidal
  - make sure it's not counting given cell r,c as a neighbor
  - calculate the row and column for each neighbor and apply modular arithmetic
- UV print
  - Print universe to outfile
  - live cell is 'o' and dead cell is '.'
  - loop through ever row and column
  - check if cell is true or false
  - Use fprintf() to print into outfile
  - fprintf newline after every row

3.2 Life

- Define command line options
- Use getopt to parse options
- Use fscanf to read the first line from given text file
- Use those values to create two universes called A and B
- Use uv populate to populate universe A
- Set up ncurses screen
- For each generation up to default number of generations or given number
- If ncurses isn't disabled clear the screen and display universe A
- refresh screen then sleep for 50000 microseconds
- Take census of each cell and set that cell in universe B to live or dead depending on rules
- Set A to B and display the next generation
- close the screen with endwin()
- Use uv print to output universe A
-

4 Credits

- I looked at tutorials point to figure out how to read the text files