Noel Ball
CSE13S

Design Document life.c

Description:

An executable test harness ***life.c*** takes command line arguments to play The Game of Life, with an input file to set the specific matrix size and points including the starting live cells, and the option to enable a toroidal universe, a set number of generations, and the *ncurses* display. A universe starting with dead cells will be populated based on the input file. Each generation changes based on the Game of Life's rules, until the set number of generations is met, and the final generation is output to a file. The universe functions are included in the file ***universe.c***.

Goal:

Run ./life with chosen arguments, and getopt() parses the input argument. A switch-case statement is used to set initial values that are either defaulted or based on the input arguments. Two universes, the current matrix and the next step, are created with dimensions based on the input files first line, set to toroidal if specified, and populated based on the rest of the input files lines indicating coordinates for rows and columns. Each generation is calculated from A and projected onto B, displayed if *ncurses* isn't specified to be silenced, and swapped for the next generation to be calculated.

Pseudo:

Every generation in ***life.c*** is calculated in a for loop until the number of generations is met:

- ➔ *ncurses* displays the current matrix

- ➔ two for loops iterate through every row and col in the current matrix
  - ◆ uv_census() finds the amount of neighboring live cells of each cell
  - ◆ the same cell on the next-step matrix is updated as live or dead
- ➔ after the next-step matrix is complete, it is swapped with the current matrix

uv_create() in *universe.c* allocates memory for:

- ➔ a matrix with malloc(size of Universe struct)
- ➔ its rows with calloc(number of input rows, size of bool *)

and allocates memory by iterating through every row for:

- ➔ its columns with calloc(number of input cols, size of bool)

All the allocated cells are marked as dead cells with *uv_dead_cells(),* and struct values are defined by using the universe pointer to access the struct values with u->.

uv_delete() in *universe.c* uses *free()* while iterating through every row, freeing the memory containing its cols. Then it frees the memory containing the rows, and the universe as a whole, before setting the universe pointer to null.

uv_rows() and *uv_cols()* in *universe.c* returns the rows or cols struct value of the universe by accessing them with u->.

uv_live_cell() and *uv_dead_cell* in *universe.c* check to ensure the cell is within the universe's bounds, then set's the cell to true or false respectively with u->grid[r][c] = bool.

Noel Ball
CSE13S

uv_populate() in *universe.c* uses a bool "bounds" starting off as true, while infile != EOF:

- ➔ if bounds is true { bounds = false, continue; }
- ➔ use *uv_live_cell()* to populate cell with coordinates from infile

"Bounds" is used in order to skip the first line of infile containing the matrix dimension size.

uv_census() in *universe.c* uses two nested for loops iterate through the adjacent coordinates including all combinations of the input row, minus one, or plus one with the input col, minus one, or plus one. The iterating adjacent coordinates are used to calculates the toroidal adjacents:

- ➔ two for loops iterate from the row-1 to row+1 and col-1 to col+1
    - ◆ toroidally adjacent cells are calculated:
    - ◆ if target row or col is < 0 (out of the universe left or top bounds)
        - add universe range to row or col out of bounds to wrap to end
    - ◆ else (only changes when out of the universe right or bottom bounds)
        - mod universe range to row or col to wrap out of bounds to start
    - ◆ if (not toroidal) and (*uv_get_cell(adjacent coordinates)* is true) and (row and col are not target cell) : then it is counted as a live neighbor
    - ◆ if (toroidal) and (*uv_get_cell(toroidally adjacent coordinates)* is true) and (row and col are not target cell) : then it is counted as a live neighbor

uv_print() in *universe.c* iterates through every row and column to:

- ➔ use *uv_get_cell()* to check if cell is live or not
    - ◆ use fprintf() to print "o" for live cells and "." for dead cells to the outfile.