cse13s asgn4 DESIGN.pdf

Lucas Lee; CruzID: luclee

1/28/2022

1 Program Details

In this program we will make a game of life. The game uses a 2x2 grid as an ADT, and the game follows three rules.

1. live cells stay alive if two or three neighbors are alive.
2. dead cells come to life if exactly three neighbors are alive.
3. all other cells die.

2 Files and Pseudocode

1. universe.c
   (a) creates the 2D universe that the game will be played in. Contains functions:
   (b) uvcreate(rows, cols, toroidal) - creates the universe grid. Use calloc to initialize the array to 0's. if the boolean toroidal is true then return a pointer to Universe *.
      i. set struct rows, cols, toroidal to parameter rows, cols, and toroidal
      ii. set struct of grid to calloc using booleans and parameter rows
      iii. for loop to allocate memory for struct grid at each index in the rows, (allocate memory for 2x2 matrix)
      iv. return struct
   (c) uvdelete(Universe *u) gets rid of the created universe that is passed into the function as a parameter. Use free() here to avoid memory leaks

i. for loop that frees memory as it was allocated in uvcreate()
ii. free grid memory
iii. free struct memory

(d) uvrows(Universe *u) returns number of rows inside of the universe.

(e) uvcols(Universe *u) return the number of columns in the specified universe.

(f) uvlivecell(Universe *u, uint32 r, uint32 c) sets the cell at row r and column c to live. use booleans to mark live and dead cells, true is live, false is dead. if the row and column don't exist in the universe given, then don't change anything.

i. check if given r and c are valid
ii. if not return nothing
iii. otherwise set grid value at row r and col c to true (true = alive)

(g) uvdeadcell(Universe *u, uint32 r, uint32 c) sets the cell at row r and column c to dead. Use false to mark the cell as dead. Do nothing is r and c don't exist in the Universe.

i. check if given r and c are valid
ii. if not return nothing
iii. otherwise set grid value at row r and col c to false (false = dead)

(h) uvgetcell(Universe *u, uint32 r, uint32 c) returns the boolean value at row r and column c. Return false if r and c don't exist.

i. check if cell is valid
ii. if not return false
iii. if it is valid then return the boolean from grid at row r and col c

(i) ucpopulate(Universe *u, FILE *infile) creates universe using the given infile. Line 1 of the file should be the number of rows and columns separated with whitespace. Every other line in the file is the row and column of the live cells in the universe for that file. fscanf() reads row-column pairs for use in the universe.

i. assume the infile is already open and you can start on the next line of the file

ii. from the infile, scan every line until the end of the file (scan returns -1 at file end)

iii. check each line as they are scanned in to make sure that they are valid points

iv. if it is a valid point, then mark it as a live cell

v. close file and return true to signify that it successfully populated using the infile

(j) uvcensus(Universe *u, uint32 r, uint32 c) returns number of adjacent live cells to the one given at row r and column c. If toroidal is set to true, then also consider the other side of the universe grid as adjacent if the live cell is at the edge of the universe. Otherwise do not consider this option.

i. initialize a counter for the census

ii. if it is not toroidal:

A. check each box around the given r and c if it is a valid box

B. this will require a bunch of if statements to check the grid boolean value and the cell validity

iii. if it is toroidal:

A. make new variables to keep track of the above row, below row, left column, right column

B. add total rows and cols += 1 depending on which row or col you are looking for(ex. if current row is 0, below row is 1 and above row is the last row)

C. list of if statements to check each grid position's value

D. each of these increment counter and after all if statments, counter is returned

iv. uvprint(Universe *u, FILE *outfile) prints the universe with the live cells being 'o' and the dead cells being '.'. Use fputc() or fprintf to specify the outfile as the place to print the universe.

A. for each row and column (nested loop):

B. if cell at current row and col = true then print o to outfile.

C. if cell at current row and col = false then print . to outfile.

2. universe.h

(a) header file for universe.c that is going to be included in the file containing the main() function.

3. life.c

   (a) helper function update() that takes two universes for parameters:
      i. loop through the first universe given and change the other universe based off the first universe
         A. if current cell is alive
         B. if census is not 2 or 3 then mark current cell as dead
         C. otherwise the cell is alive
         A. if cell is dead
         B. if census is 3 mark current cell as alive
         C. otherwise mark cell as dead

   (b) helper function swap() that swaps the two working universes
      i. pass in both universes
      ii. set a temp universe equal to universe 1
      iii. universe 1 = universe 2
      iv. universe 2 = temp universe(1)

   (c) contains main() function and runs the game of life simulation code using universe.h.
      i. set variables equal to default values for command line inputs
      ii. while loop to parse command line options
      iii. change default values of each command line input inside each case
      iv. default output is help screen
      v. scan the first line in the infile
      vi. create 2 universes using the first line values and the toroidal value
      vii. base case for if generations is 0 = print universe A, delete both universes, return
      viii. is ncurses is not silenced:
         A. initialize screen and set cursor to false
         B. loop for each generation
         C. clear screen
         D. if current iteration of loop is even, print universe A's display

E. if current iteration of loop is odd, print universe B's display
F. refresh screen and sleep
G. if current iteration of loop is even, update Universe B
H. if current iteration of loop is odd, update Universe A
I. swap the two universes
J. outside generation loop, end ncurses and if generations is even, print A to outfile, if it is odd then print B to outfile
K. delete both universes and return

ix. if ncurses is silenced:
A. for loop from 0 to generations
B. if current iteration of loop is even, update B and swap universes
C. if current iteration of loop is odd, update A and swap universes
D. outside loop, if generations is even, print A to outfile
E. if generations is odd, print B to outfile
F. delete both universes and return

4. Makefile
   (a) use CC = clang, CFLAGS = Wall, Wextra, Wpedantic, Werror
   (b) make, make life, make all must make the life executable file
   (c) make clean removes the files that the compiler generates
   (d) make format formats all c and h files
   (e) (to make life with ncurses we will need to link -lncurses at the end of compile)
5. README.md
   (a) describes how to use the program, run the program, and compile the program. Also includes errors in the code.
6. DESIGN.pdf
   (a) describes the design process, and how to make the desired functions.