William Santosa
wsantosa@ucsc.edu
31 January 2021

CSE13S Winter 2021
Assignment 3 : The Game of Life
Design Document

DESCRIPTION

John Conway is a famous mathematician most well known for his invention called the "Game of Life." The game of life is a zero-player "cellular automaton" which consists of a collection of cells that can live, die, or multiply. These cells act in generations and are interacted upon by creating an initial configuration and observing how they evolve. In this lab, we aim to implement Conway's Game of Life in C using the following documents.

Files:

- life.c
  - Contains implementation of the Game of Life
- universe.c
  - Contains implementations of the Universe ADT
- universe.h
  - Contain definitions. Supplied for us.
- Makefile
  - Runs program (All, clean, format)
- README.md
  - Information
- DESIGN.pdf
  - Describes purpose, covers the layout, clear description of program parts, pseudo code

TOP LEVEL DESIGN / PSEUDOCODE

life.c

Scan for options (t, s, n, i, o)
Case for each option
    t sets toroidal of u to true
    s sets variable silence to true
    n changes the number of generations to specified value
    i changes inputfile
    o changes outputfile
Scan valid rows and columns
Scan file line by line to rows and columns

```
For generations{
        For rows{
                For columns{
                        Check conditions for game of life
                        Set to live cell and print in ncurses if it meets it
                }
        }
        Swap a and b
        Delay
        Refresh
}

Print board
Delete universes
Close files
Return 0;
```

universe.c

```
#include <stdio.h>
#include <stdlib.h>
#include “universe.h”

// Included within doc
struct Universe {
   int rows;
   int cols;
   bool **grid;
   bool toroidal;
}

Universe *uv_create(int rows, int cols, bool toroidal){
   Initialize universe “u” struct with rows, columns, and toroidal as parameters
   Set rows, columns, and torodial of “u” to respective parameters
   Return “u”
}

void uv_delete(Universe *u){
   For loop to all universe memory allocations
      Free location specified
   Free the universe itself
}
```

```
int uv_rows(Universe *u){
    Return u pointer rows
}

int uv_cols(Universe *u){
    Return u pointer columns
}

void uv_live_cell(Universe *u, int r, int c){
    Set “u” point grid[row][column] = true
}

void uv_dead_cell(Universe *u, int r, int c){
    Set “u” point grid[row][column] = false
}

bool uv_get_cell(Universe *u, int r, int c){
    Return “u” point grid[row][column]
}

bool uv_populate(Universe *u, FILE *infile){
    fopen(FILE)
    For loop all cells
        fscanf(FILE, o if live, . if dead)
    fclose(FILE)
    Return true if successful, false if fail
}

int uv_census(Universe *u, int r, int c){
    int neighborsAlive
    For (u pointer grid[r plus one and minus one ][c])
        If alive neighborsAlive++
    For (u pointer grid[r][c plus one minus one])
        If alive and toroidal neighborsAlive++
}

void uv_print(Universe *u, FILE *outfile){
    fopen(FILE)
    For loop all cells
        fprintf(FILE, o if live, . if dead)
    fclose(FILE)
}
```

NOTE : Will definitely be altered (Listed below)

DESIGN PROCESS / MODIFICATIONS

- Added comments
- Had to use getopt
- Added breaks
- Created function Toroidal to return toroidal value
    - Changed name to toroidal
    - Changed name to convert
- New function toro to return u -> toroidal
- Did not need to open and close file for print, I did that in life.c
- Changed toroidal checks
- Census doesn't include the grid[r][c] one, had to change