Design:
Game of Life

Lucais Sanderson

9 February 2023

1 Description of Program

Game of Life implemented in `C`.

2 Pseudocode / Structure:

- Universe Functions

```
*uv_create(rows, cols, toroidal)
    create universe struct from Universe
    create matrix size rows x cols, dynamically allocating the space
    return a pointer to the universe created

uv_delete
    free all dynamically allocated data.
    use for loop similar to uv_create

uv_rows(u*)
    return # of rows in u

uv_cols(u*)
    return # of columns in u

uv_live_cell(u*, r, c)
    set value of cell at index [r][c] to live (true)

uv_dead_cell(u*, r, c)
    set value of cell at index [r][c] to dead (false)

uv_get_cell(u*, r, c)
    return value of cell at index [r][c] (true if alive, false if dead)

uv_populate(*u, FILE *infile)
```

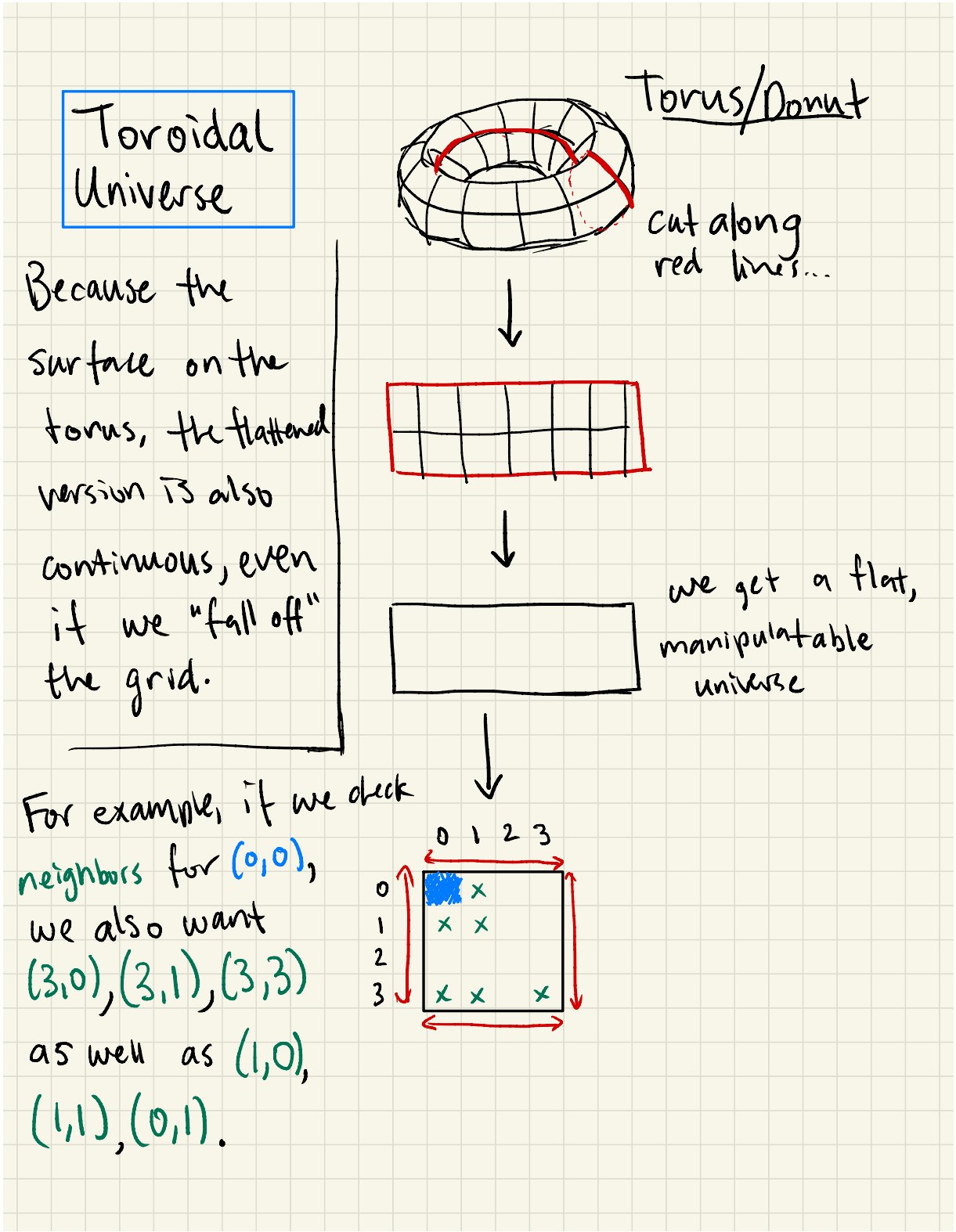
```
    get file stream from either stdin or file
    catch error if invalid input from stream
    check each line and set the row-col pair at that location as "alive"

uv_print(u*, *outfile)
    iterate through each element in universe similar to creating it.
    if the element is "alive" print "o" otherwise print "."
    if process fails, exit 1

uv_census(*u, r, c)
    the below nested loop is used to add or
    subtract from col/row index of the cell of interest.

    iterate dy from 1 to -1
        iterate dx from 1 to -1
            if universe is toroidal
                check current cell value with r,c index:
                // NOT MY CODE
                // below modulus operation derived from tutor Ben on Discord. (See Credit)
                (r + dy + rows_in_u) mod rows_in_u  for row value
                (c + dx + cols_in_u) mod cols_in_u  for column value
                // END OF BORROWED CODE
                    increment neighbor count
            else (not toroidal)
                increment neighbor count if cell alive and not "home" cell
    return neighbor count
```

- life.c

```
check options:
    i: input file
    o: output file
    s: dont display ncurses
    n: number generations

initialize universe A and B

populate A with filein/stdin
IF successful, CONTINUE, ELSE print "Malformed Input"

for number of generations
    for each cell:
        if alive and 2 or 3 live neighbors:
            set cell to alive in B
        else if dead and 3 live neighbors:
```

```
            set cell to alive in B
        else:
            cell dies (in B)
    swap A and B for next generation
    *temp = A
    *A = B
    *B = temp

return A to stdout/outfile
```

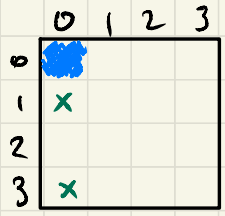
3 Toroidal Visual Representation



To achieve this we use modulus operator.

ie. $(x+m) \bmod m$ ★ From TA Ben on Discord

For our (0,0) example:



Let's just test cell above & below (0,0) keeping column at 0. m is 4, number of rows.

row = 0

row+1 = (1+4)%4 = 1

row-1 = (-1+4)%4 = 3

So we get (1,0) for below cell & (3,0) for above (wraps around) cell, both check out in diagram.

4 Credit

I borrowed code from Tutor Ben from Discord in `uv_census`. Cited within the code as well.
Link to Discord Message
/dev/nvme190n1 (Ben) — 02/07/2023 2:20 PM
"`((x % y) + y) % y`"