UNIVERSIDAD
Panamericana

CONWAY'S GAME OF LIFE

COM 139: SIMULATION & VISUALIZATION [1]

1 INTRODUCTION

The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970. The game is a zero-player game, meaning that its evolution is determined by its initial state, requiring no further input. One interacts with the Game of Life by creating an initial configuration and observing how it evolves.
In late 1940, John von Neumann defined life as a creation which can reproduce itself and simulate a Turing machine. Von Neumann was thinking about an engineering solution which would use electromagnetic components floating randomly in liquid or gas. This turned out not to be realistic with the technology available at the time. Stanislaw Ulam invented cellular automata, which were intended to simulate von Neumann's theoretical electromagnetic constructions. Ulam discussed using computers to simulate his cellular automata in a two-dimensional lattice in several papers. In parallel, Von Neumann attempted to construct Ulam's cellular automaton. Although successful, he was busy with other projects and left some details unfinished. His construction was complicated because it tried to simulate his own engineering design. Over time, simpler life constructions were provided by other researchers, and published in papers and books.
Motivated by questions in mathematical logic and in part by work on simulation games by Ulam, among others, John Conway began doing experiments in 1968 with a variety of different 2D cellular automaton rules. Conway's initial goal was to define an interesting and unpredictable cell automaton. Thus, he wanted some configurations to last for a long time before dying, other configurations to go on forever without allowing cycles, etc. It was a significant challenge and an open problem for years before experts on cell automatons managed to prove that, indeed, Conway's Game of Life admitted of a configuration which was alive in the sense of satisfying Von Neumann's two general requirements. While the definitions before Conway's Life were proof-oriented, Conway's construction aimed at simplicity without a priori providing proof the automaton was alive.
Conway chose his rules carefully, after considerable experimentation, to meet these criteria:

- There should be no explosive growth.
- There should exist small initial patterns with chaotic, unpredictable outcomes.
- There should be potential for von Neumann universal constructors.
- The rules should be as simple as possible, whilst adhering to the above constraints.

The game made its first public appearance in the October 1970 issue of Scientific American, in Martin Gardner's *Mathematical Games* column. Theoretically, Conway's Life has the power of a universal Turing machine: anything that can be computed algorithmically can be computed within Life. Gardner wrote, "Because of Life's analogies with the rise, fall and alterations of a society of living organisms, it belongs to a growing class of what are called 'simulation games' (games that resemble real life processes)."
Since its publication, Conway's Game of Life has attracted much interest, because of the surprising ways

[1] *Engeneering Faculty, Universidad Panamericana, Guadalajara, México*

in which the patterns can evolve. Life provides an example of emergence and self-organization. Scholars in various fields, such as computer science, physics, biology, biochemistry, economics, mathematics, philosophy, and generative sciences have made use of the way that complex patterns can emerge from the implementation of the game's simple rules. The game can also serve as a didactic analogy, used to convey the somewhat counter-intuitive notion that design and organization can spontaneously emerge in the absence of a designer. For example, cognitive scientist Daniel Dennett has used the analogy of Conway's Life "universe" extensively to illustrate the possible evolution of complex philosophical constructs, such as consciousness and free will, from the relatively simple set of deterministic physical laws, which might govern our universe.
The popularity of Conway's Game of Life was helped by its coming into being just in time for a new generation of inexpensive computer access which was being released into the market. The game could be run for hours on these machines, which would otherwise have remained unused at night. In this respect, it foreshadowed the later popularity of computer-generated fractals. For many, Life was simply a programming challenge: a fun way to use otherwise wasted CPU cycles. For some, however, Life had more philosophical connotations. It developed a cult following through the 1970s and beyond; current developments have gone so far as to create theoretic emulations of computer systems within the confines of a Life board.

2 RULES

The universe of the Game of Life is an infinite, two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, alive or dead. Every cell interacts with its eight neighbours, which are the cells that are horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

- Any live cell with fewer than two live neighbours dies, as if by underpopulation.
- Any live cell with two or three live neighbours lives on to the next generation.
- Any live cell with more than three live neighbours dies, as if by overpopulation.
- Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

These rules, which compare the behavior of the automaton to real life, can be condensed into the following:

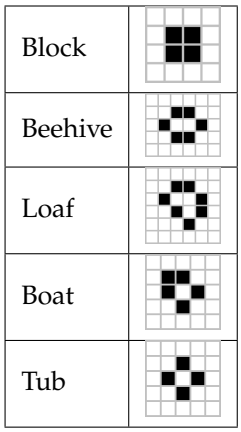
- Any live cell with two or three neighbors survives.
- Any dead cell with three live neighbors becomes a live cell.
- All other live cells die in the next generation. Similarly, all other dead cells stay dead.

The initial pattern constitutes the seed of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed; births and deaths occur simultaneously, and the discrete moment at which this happens is sometimes called a tick. Each generation is a pure function of the preceding one. The rules continue to be applied repeatedly to create further generations.

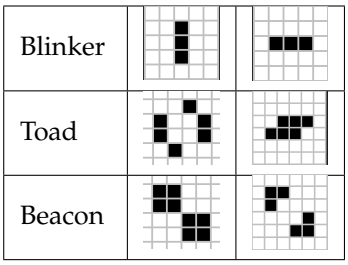
3 CONFIGURATIONS

Many different types of patterns occur in the Game of Life, which are classified according to their behaviour. Common pattern types include: still lifes, which do not change from one generation to the next; oscillators, which return to their initial state after a finite number of generations; and spaceships, which translate themselves across the grid. Here are some examples.

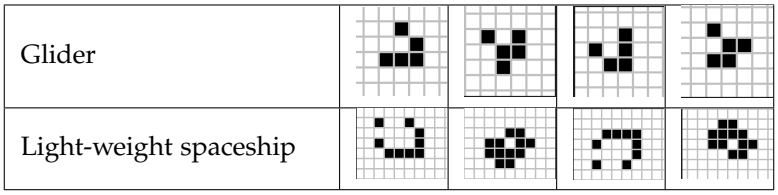
3.1 Still lifes



3.2 Oscilators (period 2)



3.3 Spaceships



4 TASKS

30% Based on the code provided implement the basic rules of the Game of Life

10% Make your program take as input a number that will define the size of the universe

20% Your program should take a file as input that will describe an initial configuration to run your simulation

40% At any moment in time count the number of Blocks, Beehives, Loafs, Boats, Tubs, Blinkers, Toads, Beacons, Gliders and Light-weight spaceships alive, any other entity that does not fit into the standards defined here should be excluded from the count. At the end of the simulation produce an output file with such information. Showing also the percentage of existance of each of these entities.

Your code should run for at least 200 generations, but it should be configurable with an input parameter as well.
The input file format should be as follows:

```
100 200    #Width Height
200        #Generations
20 3       #Cell (20, 3) is alive
20 4       #Cell (20, 4) is alive
...
```

.
The output file should look like this:

```
Simulation at 2021-02-17
Universe size 200 x 200

```

```
Iteration: 1
------------------------------
|           |  Count |  Percent |
|-----------+--------+----------|
|Block      |  12    |  44      |
|Beehive    |  0     |  0       |
|Loaf       |  3     |  11      |
|Boat       |  2     |  7       |
|Tub        |  0     |  0       |
|Blinker    |  4     |  14      |
|Toad       |  3     |  11      |
|Beacon     |  0     |  0       |
|Glider     |  2     |  7       |
|LG sp ship |  1     |  3       |
|-----------+--------+----------|
|Total      |  27    |          |
------------------------------

Iteration: 2
------------------------------
|           |  Count |  Percent |
|-----------+--------+----------|
|Block      |  12    |  44      |
...
```

Variations in the format of either is allowed as long as they are justified and documented.

5 TIPS

This is a deterministic model, so after 200 steps the results should always be the same, so based on your input I will know the real values.
As suggested in class, start simple and then start building up from there. There is an online simulation (bitstorm.org/gameoflife/) that you can use as reference to make sure that you are in the right track. I would suggest using the initial configurations that are there as default to do your testing and debugging. Once you have it working with these configurations, go crazy! You can either look online for some crazy initial configurations or you can come up with yours! Extra points for creativity on solutions!