|  |  |
| --- | --- |
|  | **2015** |
|  | **Department of Computer Science and Statistics Patan Multiple Campus**  Labin Ojha David Silwal Deependra Chaudhary Binod Shyama |

|  |
| --- |
| **Yet another Simulation of**  **Conway’s Game of life** |
|  |

**Abstract**

# Simulation Report

# for

# MATLAB SIMULATION Competition LOCUS2015

# on



The goal of our project is to design a visualization of the Game of Life as described by John Horton Conway. It is one of the best examples in science of how a few simple rules can result in incredibly complex behavior. It’s also incredibly cool and gorgeous to watch. The Game of Life, actually not a game, but more of an algorithm is meant to show what happens to organisms when they are placed in close proximity to each other. Upon giving the Game initial conditions, each successive ‘generation’ (iteration) shows the evolution of the organisms.

**Introduction**



The Game of Life, invented by John Conway in 1970, is an example of a zero-player “game” known as a cellular automaton. The game consists of a two-dimensional world extending infinitely in all directions, divided into “cells.” Each cell is either “dead” or “alive” at a given “generation”. The game consists of a set of rules that describe how the cells evolve from generation to generation. These rules calculate the state of a cell in the next generation as a function of the states of its neighboring cells in the current generation. In a 2-D world, a cell’s neighbors are those 8 cells vertically, horizontally, or diagonally adjacent to that cell. Conway’s set of rules are summarized as:

1. A live cell with fewer than two live neighbors dies.

2. A live cell with more than three live neighbors also dies.

3. A live cell with exactly two or three live neighbors lives.

4. A dead cell with exactly three live neighbors becomes alive.

In this matlab, we will be implementing Conway’s Game of Life, with the minor restriction that our 2-D world is finite.

**History**

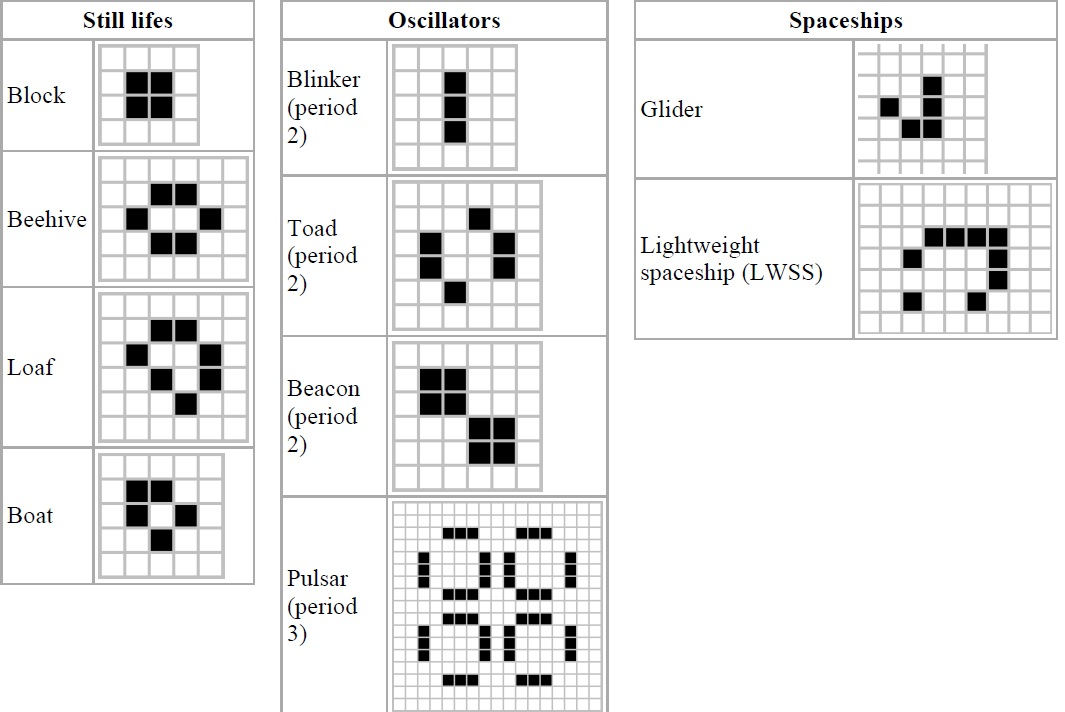


The British mathematician John Conway, who is currently at Princeton University, invented the Game of Life in the late 1960s. He chose rules that produced the most unpredictable behavior.

**Examples of Patterns**



Many different types of patterns occur in the Game of Life, including still life, oscillators, and patterns that translate themselves across the board ("spaceships"). Some frequently occurring examples of these three classes are shown below, with live cells shown in black and dead cells shown in white.



**Implementation in Matlab**



The Game of Life, which is implemented in MATLAB, fairly straightforward and the other an attempt to make it as small as possible.

Game of Life, is implemented as follows:

* Use an n x n grid.
* Initial random distribution of cells.
* Periodic boundary conditions to make it "Infinite".
* Uses matrices to represent the grid.
* Loop over the life time- number of iterations.
  + Count neighbors for each element (cell) in the matrix.
* Uses the "spy" or "pcolor" function to plot (re-plot) the matrix every iteration.

**Source Code**



function gameOfLife(gridSize, runGenerations, delay, name)

% Default pattern to load

if (isempty(name))

name = 'gosper';

end

format bank;

% Initialize a sparse matrix of dimensions gridSize by gridSize

M = sparse(gridSize, gridSize);

origX = round(gridSize/2);

origY = round(gridSize/2);

%Insert initial pattern

[P, nameLife] = lexicon\_read(name);

[offsetX, offsetY] = size(P);

M([origX+1:(origX+offsetX)], [origY+1:(origY+offsetY)]) = P([1:end], [1:end]);

% Initialize variables

gen = 0; % generations counter

counter = 0;

alive = 0;

fig = figure;

fig.NumberTitle = 'off';

fig.Name = ' Game of Life ';

% Loop to generate life successions

while (counter <= runGenerations)

% Visualize the sparse matrix in a 2D grid with 1 and 0

% representing different colours

colormap(bone);

spy(M);

title(['Name of Pattern: ', nameLife ' | Generations: ', num2str(gen) ...

' | Cells: ', num2str(alive) ' | Density: ', num2str(alive\*100/gridSize^2) '%']);

drawnow;

pause(delay);

% Counting neighbours for each cell

n = size(M, 1);

offset1 = [2:n 1]; % The world

offset2 = [n 1:n-1]; % is continuous

%offset1 = [1 1:n-1]; % The world

%offset2 = [2:n n]; % is bounded

N = M(:, offset1) + M(:, offset2) + M(offset1, :) + M(offset2, :) + M(offset1, offset1) + M(offset2, offset2) + M(offset1, offset2) + M(offset2, offset1);

% Elements in the map matrix 'M' is set to 1(i.e alive) only if sum

% of neighbouring elements is either 2 or 3

M = (M & (N == 2) | (N == 3));

% Counter variables

counter = counter + 1;

gen = gen + 1;

alive = nnz(M);

end

% Cleanup

clear,clc;

end % gameOfLife

**RESOURCES**



**Computer:**

Type: ACPI x64-based PC (Mobile)

Motherboard: Acer Aspire 4830TG

CPU: DualCore Intel Core i5-2450M, 2500 MHz (25 x 100) 64-bit

Video Adapter: Intel(R) HD Graphics 3000 (1890230 KB)

Video Adapter: NVIDIA GeForce GT 540M (1 GB)

Memory: 4.00 GB

**Operating system:**

Windows 8.1 Pro 64-bit Operating system

**Computational package / Software used:**

MATLAB R2014b for the programming and simulation

MS - Office 2007 to prepare companion Docs

**Help and references used:**

http://ocw.mit.edu/courses/mathematics/18-s997-introduction-to-matlab-programming-fall-2011/conway-game-of-life/conway-game-of-life-implementation/

http://en.wikipedia.org/wiki/Conway%27s\_Game\_of\_Life/

http://www.argentum.freeserve.co.uk/lex\_home.htm

http://blogs.mathworks.com/cleve/2012/09/03/game-of-life-part-1-the-rule/

**CONCLUSION**



The Game of Life was successfully implemented in MATLAB.