Discrete Event Simulation Lab # 5
Cellular Automata and Memoization
Multi-Agent Simulation

2D Cellular Automata

They have been invented by John Conway a Cambridge mathematician. A cellular space is populated with living cells (in black). Each cell has 8 neighbours (Moore neighbourhood). To extend this neighbourhood concept to cells on the border or the edge of a cellular space, we can consider that this space is folded in a Torus like universe (figure 1)

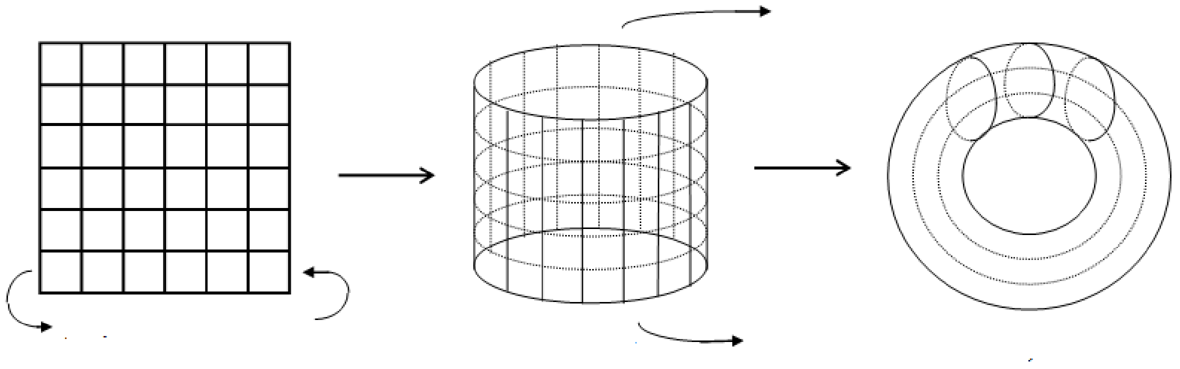


Figure 2: Torus like universe

The Game of Life rules:

"For a cell space that is 'populated':

1. *Each cell with one or no neighbours dies, as if by loneliness.*
2. *Each cell with four or more neighbours dies, as if by overpopulation.*
3. *Each cell with two or three neighbours survives.*

For a cell space that is 'empty' or 'unpopulated'

4. *Each cell with three neighbours becomes populated."*

1) **Implement a game of life** in "text" mode. Test it with a glider. use 2 cell boards: one at time 't' is considered as the reference board and a future board that is built for time 't+1'. Check with a glider configuration on a 10 x 10 cell space. First without torus neighbourhood and next with a torus universe. For both variants show you result to the lab supervisor.

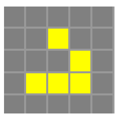
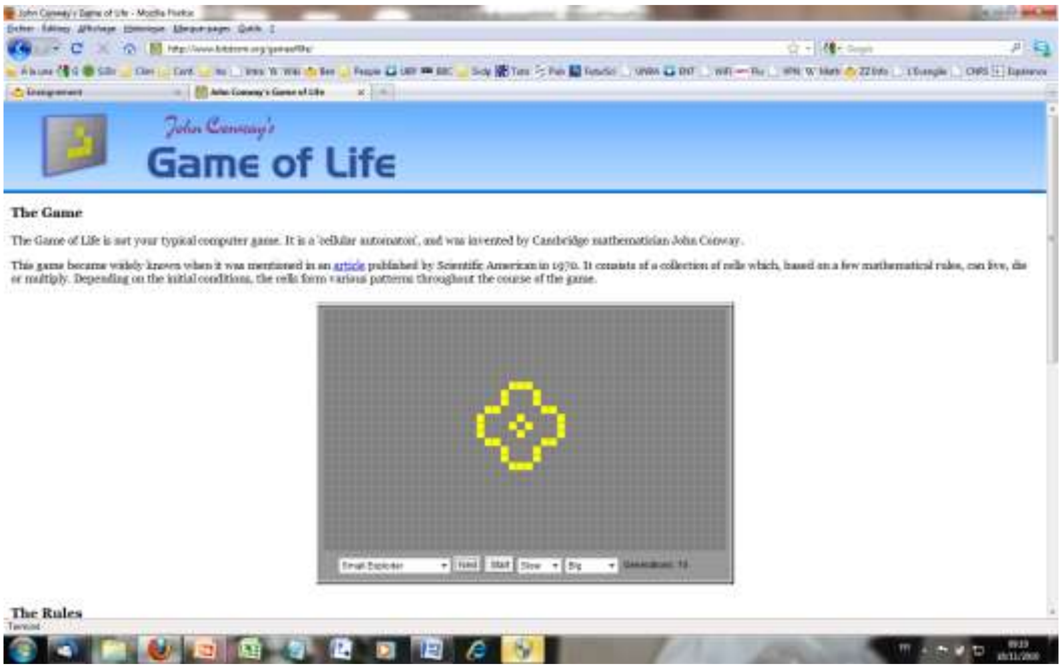


Figure 2: A cell configuration named "glider"

```
$
. . . . . . . . . . .
. X . . . . . . . . .
. . X . . . . . . . .
X X X . . . . . . . .

…



$
```

Figure 3 – Sample display in text mode under Linux

2) Explore possibilities online (Ex: http://www.bitstorm.org/gameoflife/ or other site) – this can help for debugging your implementation.



The Game

The Game of Life is not your typical computer game. It is a 'cellular automaton', and was invented by Cambridge mathematician John Conway.

This game became widely known when it was mentioned in an article published by Scientific American in 1970. It consists of a collection of cells which, based on a few mathematical rules, can live, die or multiply. Depending on the initial conditions, the cells form various patterns throughout the course of the game.

The Rules

Figure 3 – Web site with the game of life implemented in an applet

3) **Check the best implementations with memoization and the hashlife algorithm : Golly**

- Look on the Internet what is behind the hashlife algorithm.

- See an implementation of the Game of Life and some variants which uses this concept:

  http://drgoulu.com/2009/03/29/la-resurrection-du-jeu-de-la-vie/

4) **Implementation of the "memoization" concept.**

First look on the internet to find what is this concept, then implement a C program which computes the factorial of a number using this optimization technique and compare it to a classical and to a recursive implementation.

For the first 4 questions – no need to write a report. The final question lead to a larger development with a full report (50% of the final grade).

5) Design and object-oriented implementation of a basic multi-agent simulation.

Using your preferred language and dev. environment, implement a multi-agent system of your own with the following constraints.

1. Make your own choice for 2 competing categories of agents (predator preys – or agents competing for a resource), define your rules and neighbourhood for each category (agent moving and neighbourhood vision etc.)
2. Agents will evolve in a cellular 2 D space with a torus like folding (start with a 10 x 10 space for debugging).
3. Agent behaviour are stochastic (use a fine pseudo-random generator).
4. The software should propose a textual output to give the main focus to the artificial intelligence (rather than spending too much time on GUI development) – A GUI can be achieved in a second time only if time permits.
5. Follow a professional coding style and propose an online documentation (using for instance the « Doxygen » tool or an equivalent documentation tool – Javadoc for Java etc.).
6. Use (or discover) the Software Engineering tools you need (such as: a software profiler, debugger, version control systems, UML computer aided software engineering tools,...).
7. You can use (discover) GitHub/Lab an easy way & site to collaborate with others (Fork, send / pull requests and manage all your Git repositories (public and private).
8. A written report will be produced at the end of this last (and big) practical work. It will explain your design, some results and it will also present how you employed some soft. eng. tools.

Sample display in text mode :

```
t = 0     . . . . . . . . . .     t = 1     . . . . . . . . . .

          . L . . R . . l . L               . . L . . . . R . L

          . . . . . . . . . .               . . . . . . . . . .

          . . . . . . . . . .               . . .

          L L . . . . . R . .

          . . . . . . . . . .

          . . . . . l L . . .
```

You can check and download a (free) English e-book with articles giving MAS examples on D. Hill’s website. (http://pubp.univ-bpclermont.fr/public/pdf/Activity_Based_Modeling_and_Simulation.pdf.zip)