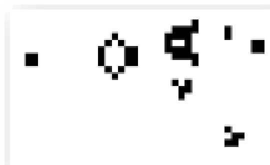
SIMULATION
BIO INSPIRED MODELING

The need of modeling came from the try to understand what we observe from our environment.

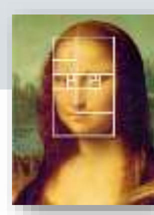
Here are two simple examples which have a great impact in many domains:

- Fibonacci Sequence
- Cellular automata



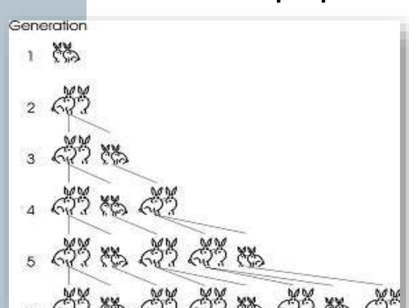
U(N) = U(N-1) + U(N-2)

PART I – LIFE SCIENCE 1st MODEL



A model… a simulation…

For rabbit population growth



Leornardo of Pizza, was the son of Bonaccius (*Filius Bonacci…)*

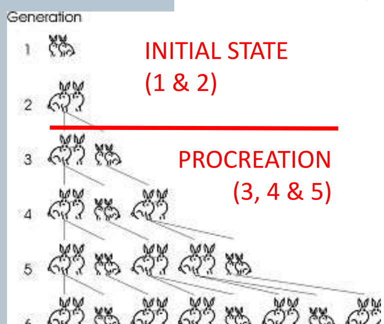
ISIMA

DETAILS OF THIS SIMPLE MODEL

The algorithm describing the model has two parts: initialization and loop. The simulation has a discrete time step of one month.

(1) Take a couple of baby rabbits

(2) Wait 1 month – the couple is now mature for reproduction



(3) Number of couples for month 'N'

=

Number of couples for month N-1

+

Number of couples for month N-2

(4) Display the current # of couples

(5) Return at stage (3)

ISIMA

# MODEL = Simplified Representation of Reality
SIMULATION = The model state changes over time

The algorithm of this simple model is described by the 'Fibonaci' sequence :

U(N) = U(N-1) + U(N-2)

Φ = 1,6180339...
(le rapport idéal ou nombre d'or)

Progression actuelle	Progression précédente	Division	Rapport
1	1	1 / 1	1,0
2	1	2 / 1	2,0
3	2	3 / 2	1,5
5	3	5 / 3	1,6666
8	5	8 / 5	1,600
13	8	13 / 8	1,625
21	13	21 / 13	1,615384
34	21	34 / 21	1,619048
55	34	55 / 34	1,617647
89	55	89 / 55	1,618182
144	89	144 / 89	1,617978
233	144	233 / 144	1,618056

The ratio between two numbers of this sequence converges towards PHI

Known as – the « Gold Number »

« ideal ratio or « divine proportion »

This number is often used by artists and architects who study this number as a criteria for beauty

This number is also found at all scales in nature

ISIMA

USE IN GEORMETRY

ISIMA

A QUICK LOOK AT DIFFERENT PROGRAMMING « PARADIGMS »

- Procedural programming – abstraction with function names
- Functional programming – functions can call themselves
- Object-oriented programming – objects that surround us are described and their interactions make the program (this approach is currently the most wide spread and is inspired by simulation – The first language which introduced this programming style is Simula 67)
- Logic programming – like ProLog for instance – uses a base of facts, a base of rules and an algorithm named « inference engine » which exploites the facts and rules to obtain deductions
- Meta-programming – programs that write other programs.

Set approaches : which use the concept of set in databases to manipulate data sets.

ISIMA

RECURSIVE CODE IN JAVA...

The 'Fib' function calls itself
(Functional programming)

```
public static long fib(int n) {
    if (n <= 1) return n;
    else return fib(n-1) + fib(n-2);
}
```

U(N) = U(N-1) + U(N-2)

ISIMA

COMPLETE CODE...

Many languages, can mix programming style. Here the following Java code mixes object-oriented and functional programming

```
public class Fibonacci {
    public static long fib(int n) {
        if (n <= 1) return n;
        else return fib(n-1) + fib(n-2);
    }

    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = 1; i <= N; i++)
            StdOut.println(i + ": " + fib(i));
    }

}
```

Tweeter is written in Scala – a language on top of the JVM which is precisely designed to mix OO programming and functionnal programming

ISIMA

STILL RECURSICE IN JAVASCRIPT

Javascript is the 1er language for web-based development. A kind of modern assembly language generated by advanced frameworks

Fibonacci Number Via Recursion

```
var recursive = function(n) {
    if(n <= 2) {
        return 1;
    } else {
        return this.recursive(n - 1) + this.recursive(n - 2);
    }
};
```

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 ...

ISIMA

PROCEDURAL CODE… (JAVA LIKE)

In the code below, recursion is not used.

The old procedural approach is often preferred for such code (for speed and stack usage)

fibo.java

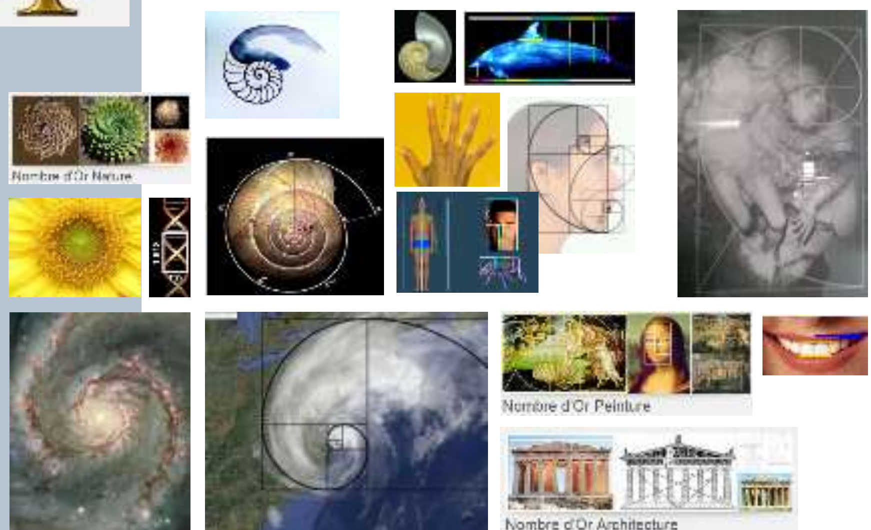
```
void fibonnaci()
{
    int fiboN-2 = 1;
    int fiboN-1 = 1;
    int fiboN   = fiboN-1 + fiboN-2;
    int mois    = 2;
    do
    {
      Stdout.println("Mois : " + N + " = " + fiboN);

      // On passe au mois suivant
      mois    = mois + 1;
      fiboN-2 = fiboN-1;
      fiboN-1 = fiboN;
    } while ( mois < 7 );
}
```

Java

In real life…

$$\varphi = \frac{1+\sqrt{5}}{2} = 1.6180339887\ldots$$



Part II

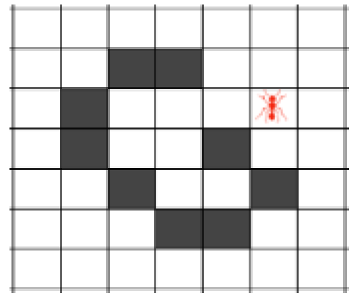
Cellular Automaton

Langton's Ant

- The Langton ant is a **two-dimensional cellular automaton** with a very simple set of rules.
- It was named after **Christopher Langton**, his inventor.
- It is one of the simplest systems for highlighting an example of **emerging behavior**.

12

2D black & white grid



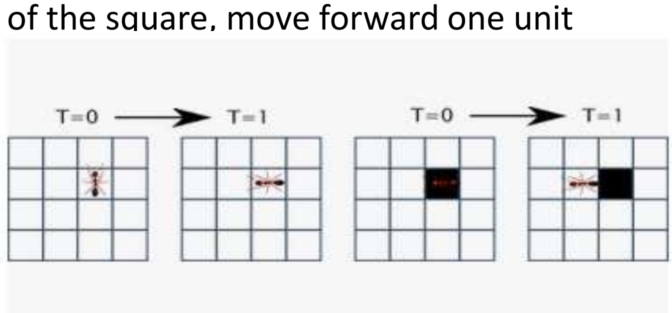
- Squares on a plane are colored variously either **black** or white.
- We arbitrarily identify one square as the "ant".
- The ant can travel in any of the four cardinal directions at each step it takes.
- Despite very simple rules, a complex phenomenon is emerging.

13

Simple Rules

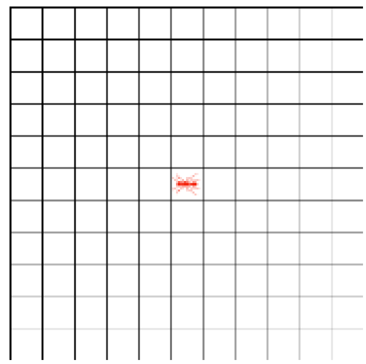
The "ant" moves according to the rules below:

- At a white square, turn 90° right, flip the color of the square, move forward one unit
- At a black square, turn 90° left, flip the color of the square, move forward one unit



14

A look at the first 200 steps



15

Hidden complexity

- These simple rules lead to a **complex behavior**.
- Three distinct modes of behavior are apparent, when starting on a completely white grid.
- We will see emergence as a form of apparition despite an underlying and previously hidden complexity.

16

ISIMA

1. Simplicity

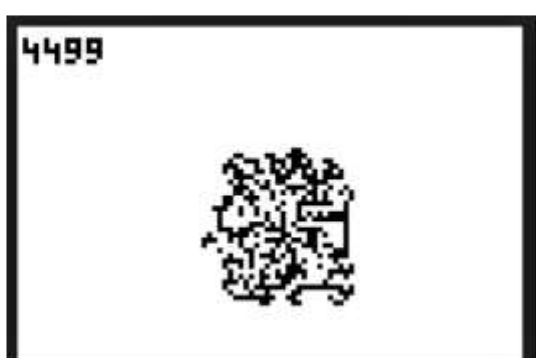
During the first few hundred moves
it creates very simple patterns which are
often symmetric.

17

ISIMA

2. Chaos

After a few hundred moves,
a **big, irregular pattern**, of black and white squares appears. The ant traces a pseudo-random path until around 10,000 steps.

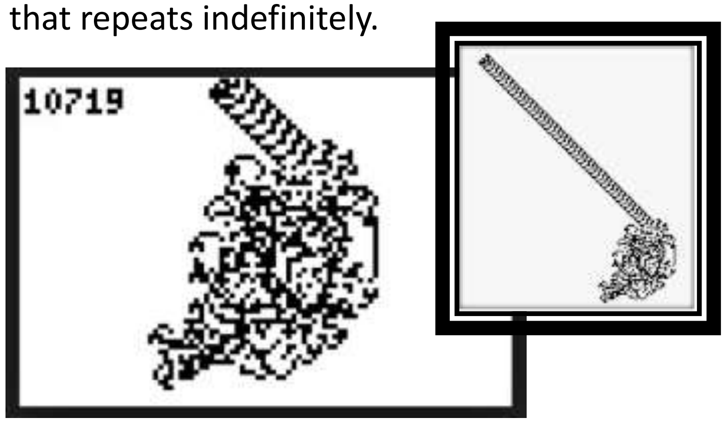


18

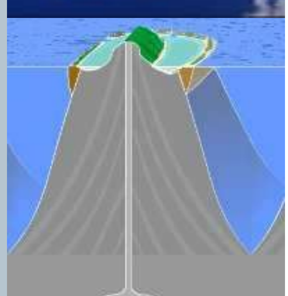
3 . Emergent order

下一节课的实验

Finally the ant starts building a recurrent **"highway" pattern** of 104 steps that repeats indefinitely.



19



Emergence ?

- The **emergence**, evokes the idea of an unveiling, an apparition that takes shape, but that was somewhere already there, under an appearance still elusive.
- This is the case or emerging land (hidden below water) or the case of an iceberg.
- It is the case of continents, or **atoll island** rooted under the surface of the water, but really take shape for our eyes when they come out.
- The last picture could **disappear**…

20

PART II – LIFE SCIENCE : A MORE COMPLEX MODEL

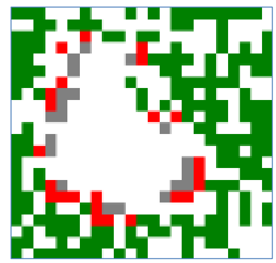
Presentation of anothe 2D Cellular automata

I – Presentation of cellular automata

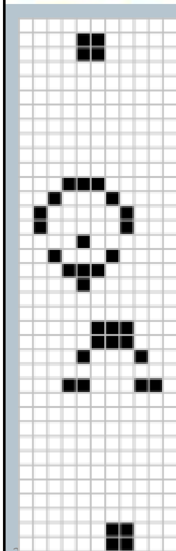
II – Example of a design for a fire simulation

III – Study of results

- Game of life Cellular Automata:
  - Are represented by a grid of cells
  - Evolve during time according to rules depending on the neighborhood of each cell
- Looks like flash codes !?



Example of cellular automata



The origins

- The game of Life was invented by John Conway in 1970. This became the most famous cellular automata for at least two reasons:
  1. From simple rules, the game generates an artificial life strangely complex and not predictable without simulation. This generated plenty of interesting questions.
  2. The game of life is easy to program, generations of students have coded “Life” programs in a any programming languages.

Mathematical Formalization

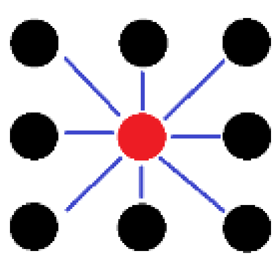
From a mathematical point of view, a cellular automata is a quadruplet *(d, A, N, R)*.

- *d* represents the automata *dimension*.
- Z^d is then its network, a discrete space of dimension *d* ('the automata grid').
- *A* is the automata *alphabet.* It is a finite state which represents all the possible states for a cell.
- *N* is a subset of Z^d representing the *neighborhood* of a cell. It is from this *neighborhood*, that the next state of a cell is computed.
- *R* is the set of local rules that define the behavior of a cell depending on its current state and on the state of the neighboring cells *(N).*

23

Game of Life : Introduction to the Moore neighborhood

- The Moore neighborhood is composed of the 8 direct neighbors of a cell (*N)*



Representation of the Moore neighborhood

Edward Forrest Moore if Professor of Mathematics and Computer Science at the University of Wisconsin-Madison.

24

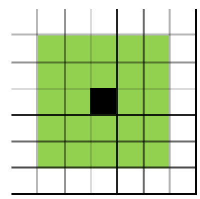
ISIMA

Moore neighborhood at order ‘N’

- The Moore neighborhood at order 2 of a cell is composed of the 24 direct surrounding cells.
- More generally, the Moore neighborhood at order N of a cell is composed of the :

$$(2N+1)^2 - 1$$

surrounding cells at a Tchebychev distance of N



Techebychev distance = N = 2
Moore neighborhood at order 2

25

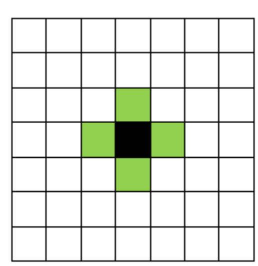
ISIMA

Von Neumann neighborhood

In a cellular automata,
the **von Neumann** neighborhood of a cell is composed of the 4 adjacent cells (vertically and horizontally).

John von Neumann, was born in Budapest in 1903. He died in Washington, D.C. on February the 8, 1957.
He was an Americano-hungarian mathematician and physicist.

Most famous (unfinished) book (1958):
« The brain and the computer »



Von Neumann neighborhood – order 1

26

ISIMA

Presentation of the ‘Game of Life’ rules

- The game of life model is in 2 dimensions:
  - 2 cell states: dead or alive (alphabet)
  - Evolution rules to compute the next CA:
    - If a cell is alive with 2 or 3 living neighboring cells, it will stay alive, otherwise it dies.
    - A dead cell (or empty space in the Network) surrounded by 3 living neighboring cells will come to life.

t *t + 1*

Evolution rules for the game of Life

27

ISIMA

Details of the original rules

John Horton Conway is a Bristish mathematician. Extremely prolific, he studied the theory of finite groups, nodes, numbers, games and coding

R (sometimes noted δ) : set of local original rules

“

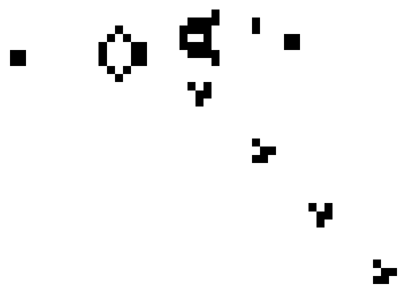
1. Any live cell with fewer than two live neighbours dies, as if caused by under-population.
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies, as if by overcrowding.
4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

”

28

From elementary « life forms »…

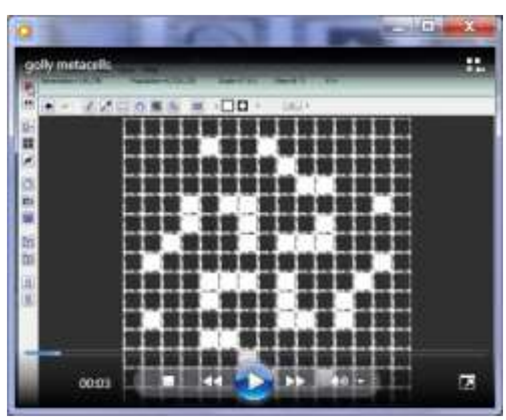
- In the automata below, some configurations appear.
- 2 patterns named « guns » are interacting together and they produce a set of other configurations/patterns, named « gliders »:



29

… to complete Turing machines !

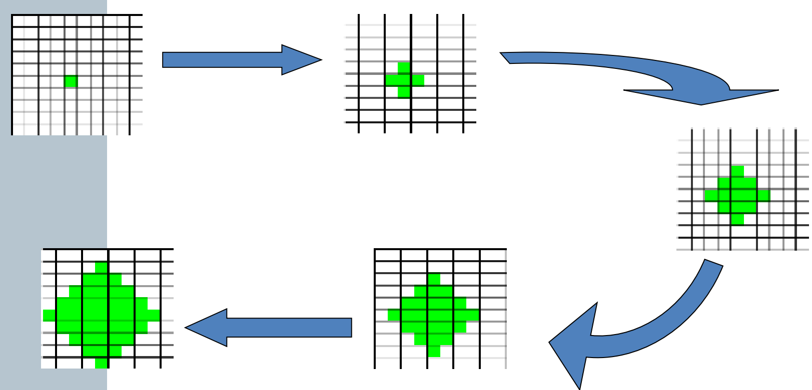
- A **puffer train**, or simply **puffer**, is a finite pattern that moves itself across the "universe", leaving debris behind (in this case guns producing gliders).



30

Basic Cellular Automata (CA)

Simple example : diffusion of a solute in a solvent, 2 states – deterministic CA



2 states :
- Solvent
- Solute

Rule :
- Solvent → Solute if one neighboring state is a solute

31

Automata Cellular common characteristics

Simple rules → complex behavior

- Regular grids of cells
- Finite space of states
- The next state of cells depends on the states of the neighboring cells at the previous discrete time step
- The same set of rules is applied ‘simultaneously’ to all cells

Many characteristics → many classes of automata

32

Cellular Automata classification (1)

Classifications based on the behavior

- **Wolfram (1984) :**
  - 4 classes defined by the behavior (stable, cyclic, chaotic or complex)
- **Eppstein :**
  - derived from the Wolfram classification (mandatory expansion, contraction impossible, impossible expansion, possible expansion and contraction)
- **Langton (1990) :**
  - Derived from the Woflram classification with a definition of the automata space « temperature » allowing the obtaining of the following behaviors : quick disparition rapide, persistant cyclic structures, complex structures behaviors and cahotic behavior).

33

Cellular Automata classification (2)

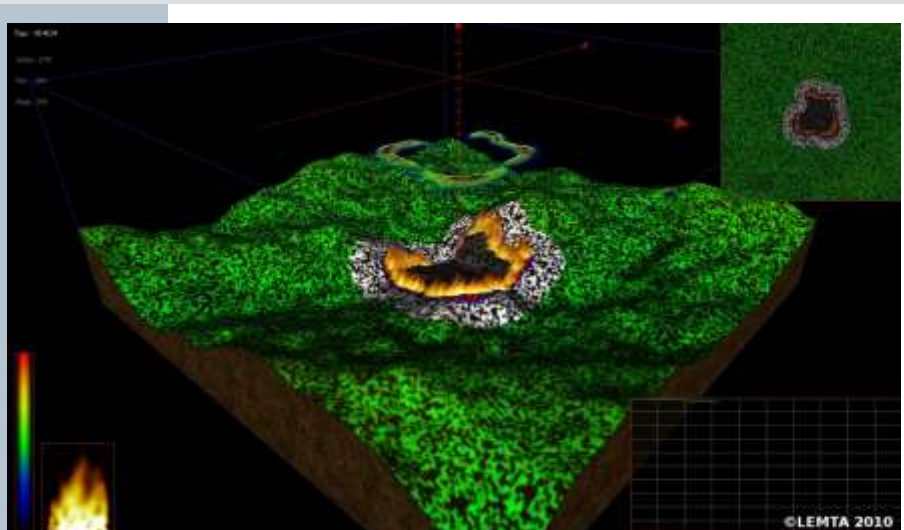
Classifications based on characteristics

- **Deterministic or Probabilistic CA :** does the transition rule imply the use of a random source ?
- **Synchronous or asynchronous CA :** is the update of each cell performed sequentially or simultaneously ?
- **Discrete of Continuous states CA :** is the cell state represented by an integer or by a real number ?
- **Homogeneous or Heterogeneous :** are the same rules applied to all cells at each cycles ?
- **CA with memory :** the CA saves the previous state of each cells (CA with memory)

34

Application example : Fire Simulation

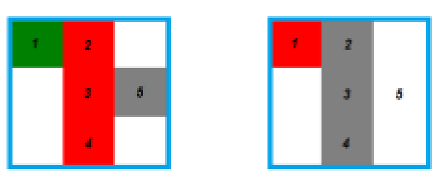
Study of techniques and data structures



35

Basic fire propagation

- Fire propagation model
  - 4 states (empty, forest, ashes, fire)
  - 3 evolution rules:
    - If state = ‘forest’ and a neighboring cell is in fire then the considered cell lights up and goes to the fire state.
    - If state = ‘fire’, then it becomes ‘ashes’
    - If state = ‘ashes’, then it becomes ‘empty’



Sample evolution for the fire propagation model

36

The activity concept (1)

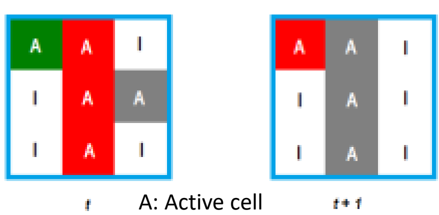
- Identify cells that can possibly change their state between two simulation steps
- Model Activity =

  Number of active cells at a specific time
- Interest : performance gain

37

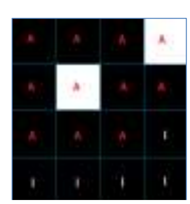
The activity concept (2)

- Activity for the fire model



A: Active cell
I: Idle cell

- Activity for the game of life



A: Active cell
I: Idle cell

38

ISIMA

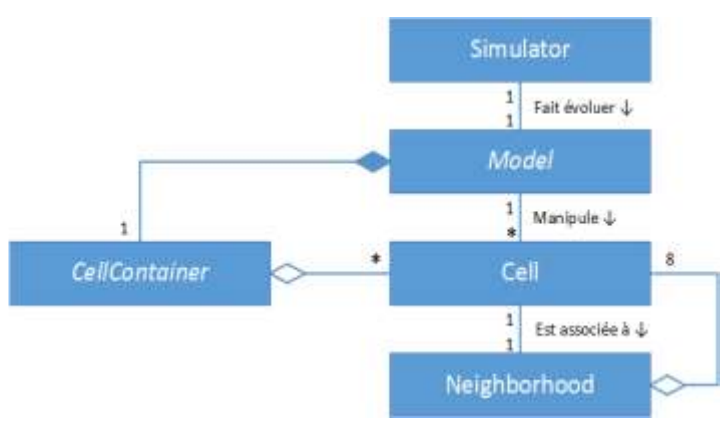
Optimization : having containers for active cells

- **Cell containers** :
  - Store references towards active cells
  - Reduced the size of the studied domain to the size of the container
  - Should facilitate addition and suppression of cells at each time step

- Different types of containers (data structures) can be assessed: dynamic arrays, linked-list, indexed lists, trees, etc.

ISIMA

Excerpt of the Simulation Class Diagram



Main classes of the Simulation with an emphasis on activity

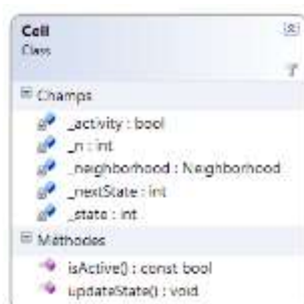
Some problems linked the simulation of cellular automata

- 2 design problems with a particular focus:
  - Modification of a cell state during an iteration (must consider the whole automata state at the previous simulation time)
  - Adding a cell in the container while navigating in this container

41

The Cell class

- A cell :
  - A current state
  - A temporary (future) state

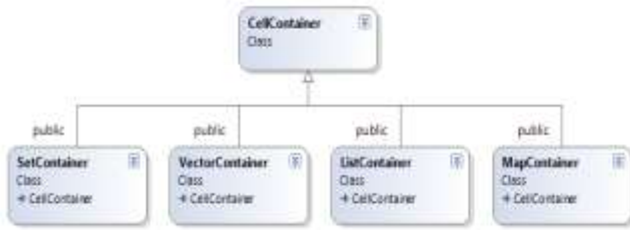


The Cell class

42

ISIMA

The CellContainer class



The CellContainer class

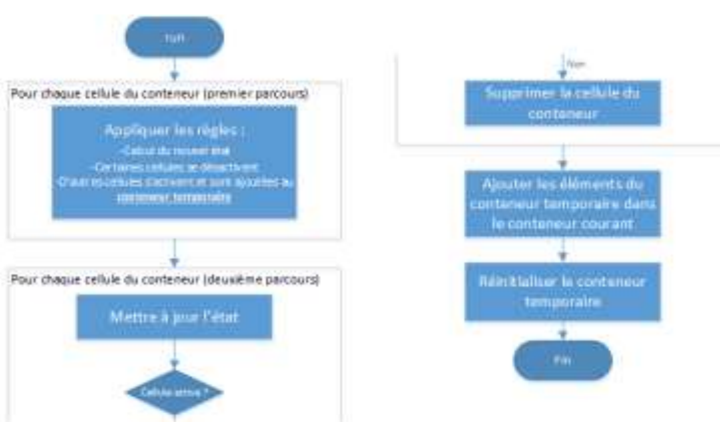
- 2 sub-containers
  - The current container
  - A temporary container (future state)

The ListContainer Class

43

ISIMA

Functioning of an iteration



Organigram (in French)

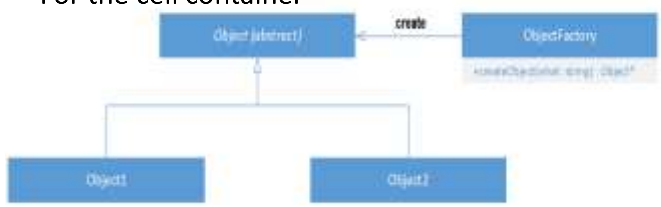
44

ISIMA

Use of the Factory design pattern (1)

Purpose :

- Respect modularity
- Dynamic instantiation of an object of a particular subclass
- Implemented 2 times:
  - For the model
  - For the cell container

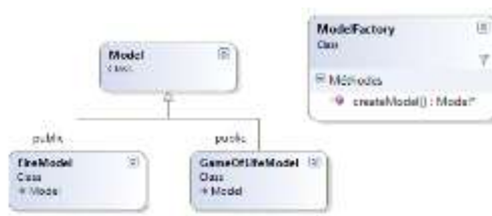


The Factory Design Pattern

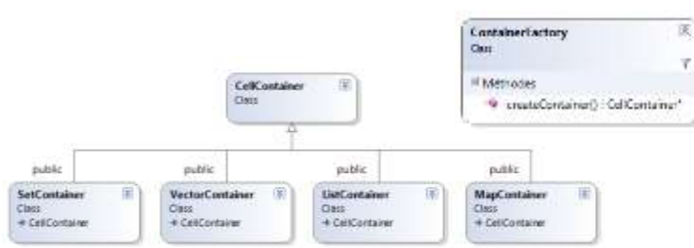
45

ISIMA

Use of the Factory design pattern (2)



The ModelFactory Class



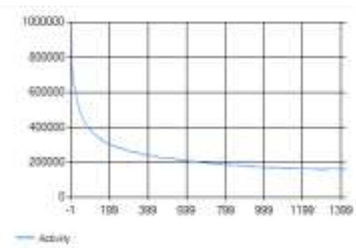
The ContainerFactory Class

46

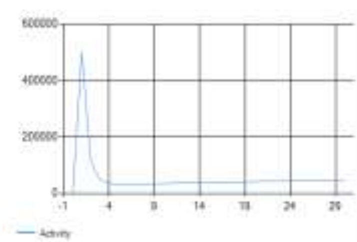
ISIMA

Evolution of the Activity on two models (1)

- Game of life
- Grid with 1 000 000 cells



Initialization with 25% of living cells



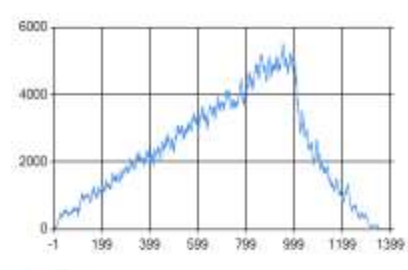
Initialization with 8% of living cells

47

ISIMA

Evolution of the Activity on two models (2)

- The fire model
- Grid of 1 000 000 cells



Initial status with 48% of « forest » cells

48

ISIMA

Study of performances depending on containers (1)

- Game of life

Modèle : Jeu de la vie
Taille : 50
Taux de remplissage : 25
Nombre d'itérations : 200

	Durées (ms)			Moyenne
ListContainer	11 424	11 364	11 596	11 461
VectorContainer	10 491	10 032	10 484	10 336
MapContainer	8 925	8 695	9 222	8 947
SetContainer	7 869	7 199	7 354	7 474

Profiling (1) with only 2500 cells

Modèle : Jeu de la vie
Taille : 1000
Taux de remplissage : 25
Nombre d'itérations : 1

	Durées (ms)			Moyenne
ListContainer	40 161	39 690	38 905	39 585
VectorContainer	43 759	39 541	41 318	41 539
MapContainer	45 115	36 383	42 376	41 291
SetContainer	38 219	38 719	43 231	40 056

Profiling (2) with 1 000 000 cells

49

ISIMA

Study of performances depending on containers (2)

- Fire propagation model

Modèle : Modèle de feu
Taille : 50
Taux de remplissage : 40
Nombre d'itérations : 10

	Durées (ms)					Moyenne
ListContainer	1 181	1 363	1 028	1 419	1 328	1 264
VectorContainer	1 376	772	733	831	1 370	1 016
MapContainer	601	1 216	851	883	896	890
SetContainer	600	651	826	766	490	667

Profiling (1) with only 2500 cells

Modèle : Modèle de feu
Taille : 1000
Taux de remplissage : 40
Nombre d'itérations : 100

	Durées (ms)			Moyenne
ListContainer	5 599	5 939	5 433	5 657
VectorContainer	3 784	3 466	3 131	3 460
MapContainer	5 159	4 983	4 929	5 024
SetContainer	4 558	4 527	4 527	4 537

Profiling (2) with 1 000 000 cells

50

ISIMA

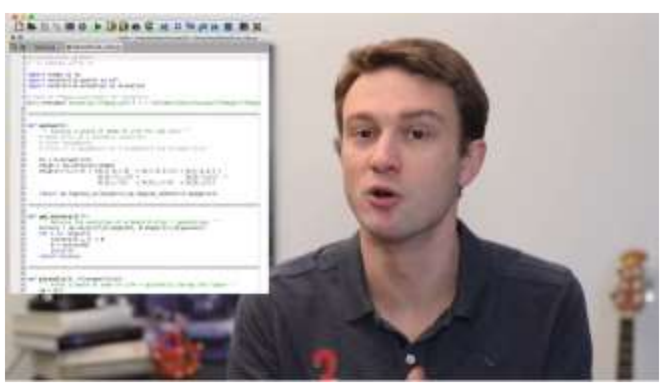
Results and discussion on the ex.

- Modular code allowing the study of different cellular models
- Modularity achieved thanks to the Factory pattern
- The activity level influences on performances
- Lower influence depending on the choice of container
- Possible perspectives : implementation of other kinds of models and containers

51

ISIMA

Astonishing Science…



Dr. David Louapre - chercheur en physique connu pour son travail de vulgarisation scientifique sur sa chaîne internet « Science étonnante » for French speaking students.

52

ISIMA

See more online…

cubes.io - 3d cellular automata on the web!

53