

# Rapport

## Réalisation d'un simulateur d'automates cellulaires

Alexandre AUDA

PRÉSENTÉ À NICE SOPHIA ANTIPOLIS

Présenté le 23 juin 2017

# Table des matières

Rapport .....	1
I- Introduction: .....	3
II- État de l'art:.....	3
2.1- Contexte général:.....	3
2.2- Automate cellulaire uniforme:.....	4
Définition: .....	4
Représentation des automates cellulaire élémentaire:.....	7
Concepts d'uniformité et de non uniformité:.....	8
III- Travail effectué:.....	11
IV- Gestion de projet: .....	26
V- Conclusion:.....	29
VI- Perspectives et réflexions personnelles: .....	30
Annexes: .....	31
A.1- Bibliographie:.....	31

## I- Introduction:

L'objectif de ce TER a été la réalisation d'un simulateur informatique d'automates cellulaires. Ce simulateur a été conçu et développé pour être un outil d'aide à la recherche dans ce domaine.

En effet, s'il existe des simulateurs basiques d'automates cellulaires uniformes, il n'existe pas à l'heure actuelle de simulateurs d'automates cellulaires non-uniformes. Les représentations de tels automates doivent donc être construites à la main, au cas par cas, rendant de plus impossible toutes manipulations dynamiques avec ces derniers à des fins d'études.

C'est dans ce contexte qu'est né ce projet visant à construire un simulateur d'automates cellulaires uniformes et non uniformes. Son utilisation permettra d'interagir avec eux afin d'en étudier les comportements ou pour obtenir des représentations graphiques afin d'illustrer et de valider des modèles théoriques, notamment pour des articles de recherche.

C'est également à ce niveau qu'intervient ma contribution, ayant pour but de fournir aux équipes de recherche un outil les assistant dans leurs travaux.

## II- État de l'art:

### 2.1- Contexte général:

L'objectif de mon projet étant la construction d'un simulateur d'automates cellulaires, il est important de comprendre en premier lieu ce qu'est un automate cellulaire et notamment les concepts d'uniformité et de non uniformité qui est au cœur de ma réalisation.

## 2.2- Automate cellulaire uniforme:

### Définition:

Un automate cellulaire (CA) se représente sous la forme d'un triplet  $(A, r, f)$  où  $A$  est un ensemble fini appelé alphabet,  $r$  un entier positif appelé rayon et  $f: A^{2r+1} \rightarrow A$  est une fonction appelé règle locale de rayon  $r$ . Cette règle locale induit une règle globale  $F: A^{\mathbb{Z}} \rightarrow A^{\mathbb{Z}}$  définit par:

$$\forall x \in A^{\mathbb{Z}}, \forall i \in \mathbb{Z}, F(x) = f(x_{[i-r, i+r]})$$

Source: [these provillard.pdf](#)

Un automate cellulaire représente la modélisation mathématique du comportement d'un ensemble d'entité élémentaire que nous nommerons *cellule*.

Nous appellerons *monde cellulaire*, l'ensemble des cellules considérées.

Chaque cellule du monde cellulaire se trouve dans un *état*. L'alphabet de l'automate cellulaire modélise l'ensemble des états possibles dans lequel une cellule peut se trouver. Ainsi, si l'on considère un automate cellulaire  $(A, r, f)$  tel que  $A = \{0; 1\}$ , chaque cellule pourra se trouver dans un état 0 ou alors dans un état 1.

Nous appellerons également *configuration*, l'état des cellules du monde à un instant  $t$  donné.

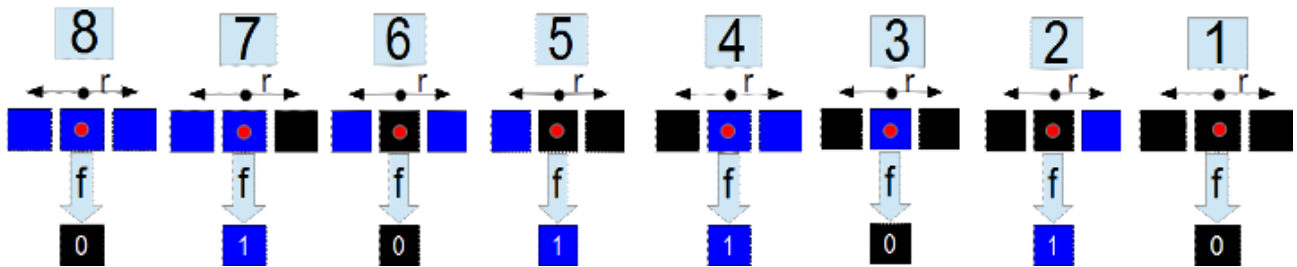


Figure 1: Exemple de configuration à un instant  $t$ , extrait du simulateur réalisé.

De plus, chaque cellule évolue dans le temps en suivant une *règle* définie représenté par la fonction  $f$  de rayon  $r$ . A chaque unité de

temps, l'état des cellules va évoluer en fonction de leur état actuelle et ceux de leurs voisins en suivant la règle définie.

Ainsi, nous pouvons par exemple considérer un automate cellulaire  $(A, r, f)$  tel que  $A = \{0 ; 1\} = \{\text{Noir} ; \text{Bleu}\}$ ,  $r = 1$  et  $f$  la règle définit comme suit:



Sur cet exemple, nous voyons que si une cellule  $c$  est dans un état 1 (couleur bleu) à l'instant  $t$  et que cette même cellule  $c$  est entourée de cellules eux même dans un état 0 (couleur noir) alors nous nous trouvons dans la configuration 3. L'état de la cellule  $c$  évoluera donc à l'instant  $t+1$  dans un état 0.

Nous pouvons donc, étant donné une configuration initiale, en déduire l'évolution de cette configuration dans le temps.

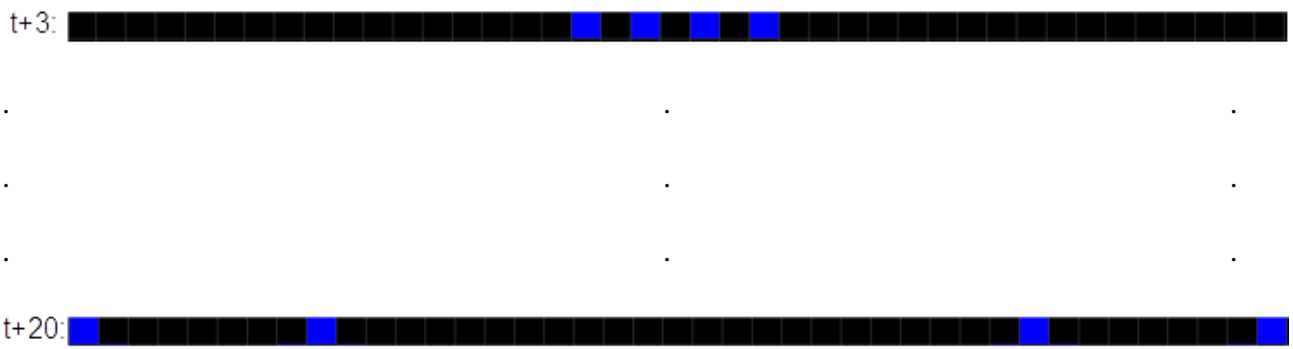
Par exemple, si nous prenons la configuration suivante:

Temps  $t$ :

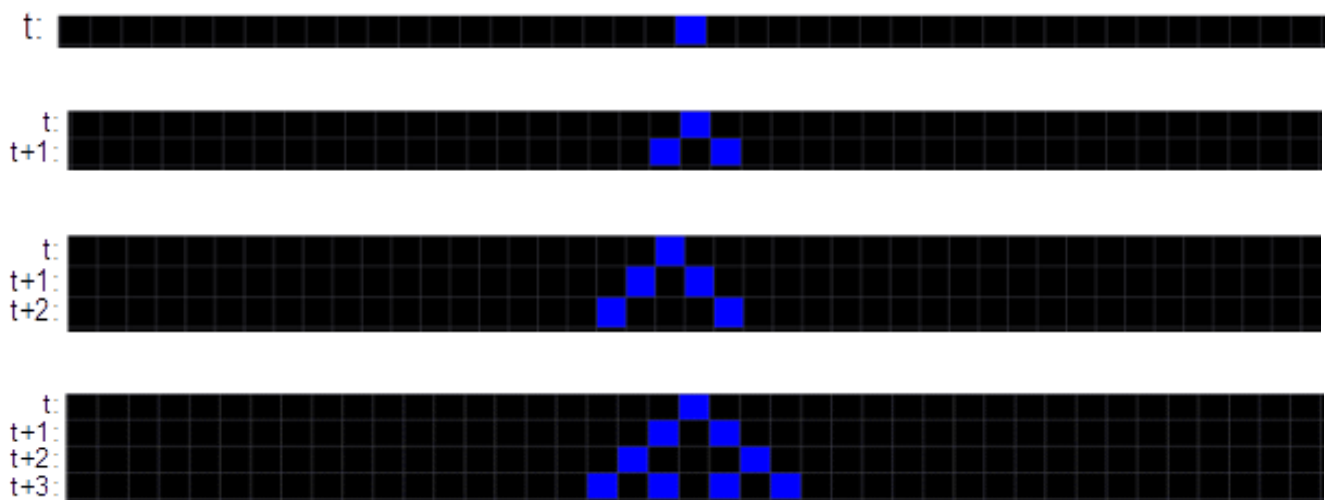


Et que nous prenons la règle présentée ci-dessus, nous pouvons alors en déduire l'évolution de cette configuration au cours du temps:





Ainsi, si nous superposons les configurations obtenues au cours du temps, nous obtenons la construction du *diagramme espace-temps* de l'automate cellulaire.



Nous obtenons donc le diagramme espace-temps suivant après la 20-ième évolution:

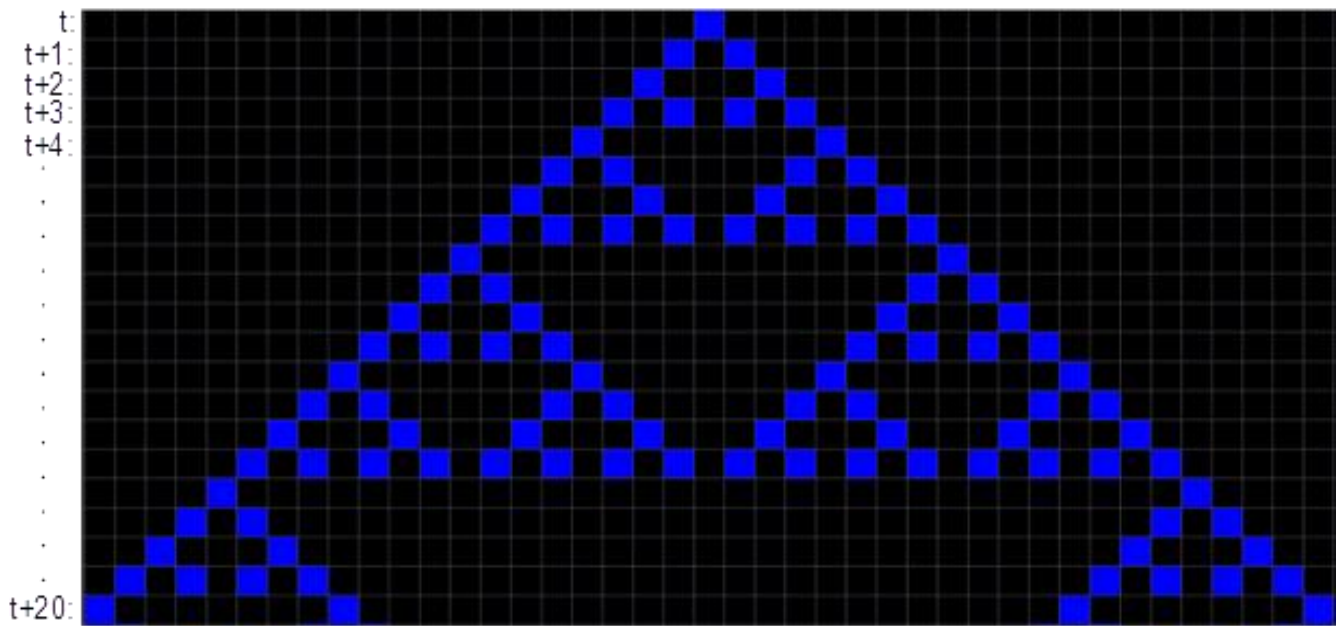


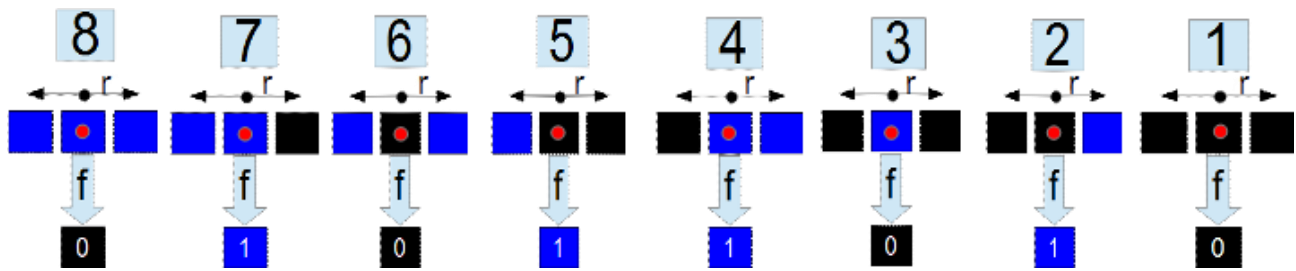
Figure 2: Exemple de diagramme espace-temps, extrait du simulateur réalisé.

Nous remarquons par ailleurs que cette règle d'évolution appliquée à une telle configuration initiale (configuration au temps  $t_0$ ) produit un diagramme espace-temps prenant la forme du [triangle de Sierpinski](#).

L'objectif de notre simulateur est de construire dynamiquement de tel diagramme espace-temps.

### Représentation des automates cellulaire élémentaire:

Aussi, si l'automate cellulaire présente un alphabet tel que  $A = \{0 ; 1\}$ , nous pouvons en déduire une représentation des règles sous forme décimale en traduisant les états obtenus après l'application des règles sur l'ensemble des combinaisons possibles. Nous traduirons par exemple ainsi, la règle définit comme suit:

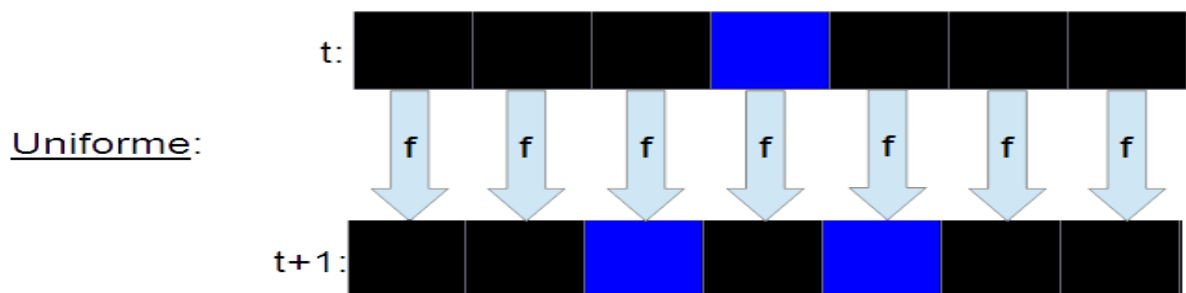


$$(01011010)_2 = (90)_{10}$$

Par conséquent, la règle définit plus haut sera appelé: ***La règle 90***. De tels automates sont appelés des *automates cellulaires élémentaires*.

Concepts d'uniformité et de non uniformité:

De plus, on dit qu'un automate cellulaire est *uniforme* lorsque chaque cellule du monde cellulaire suit **la même règle d'évolution**.





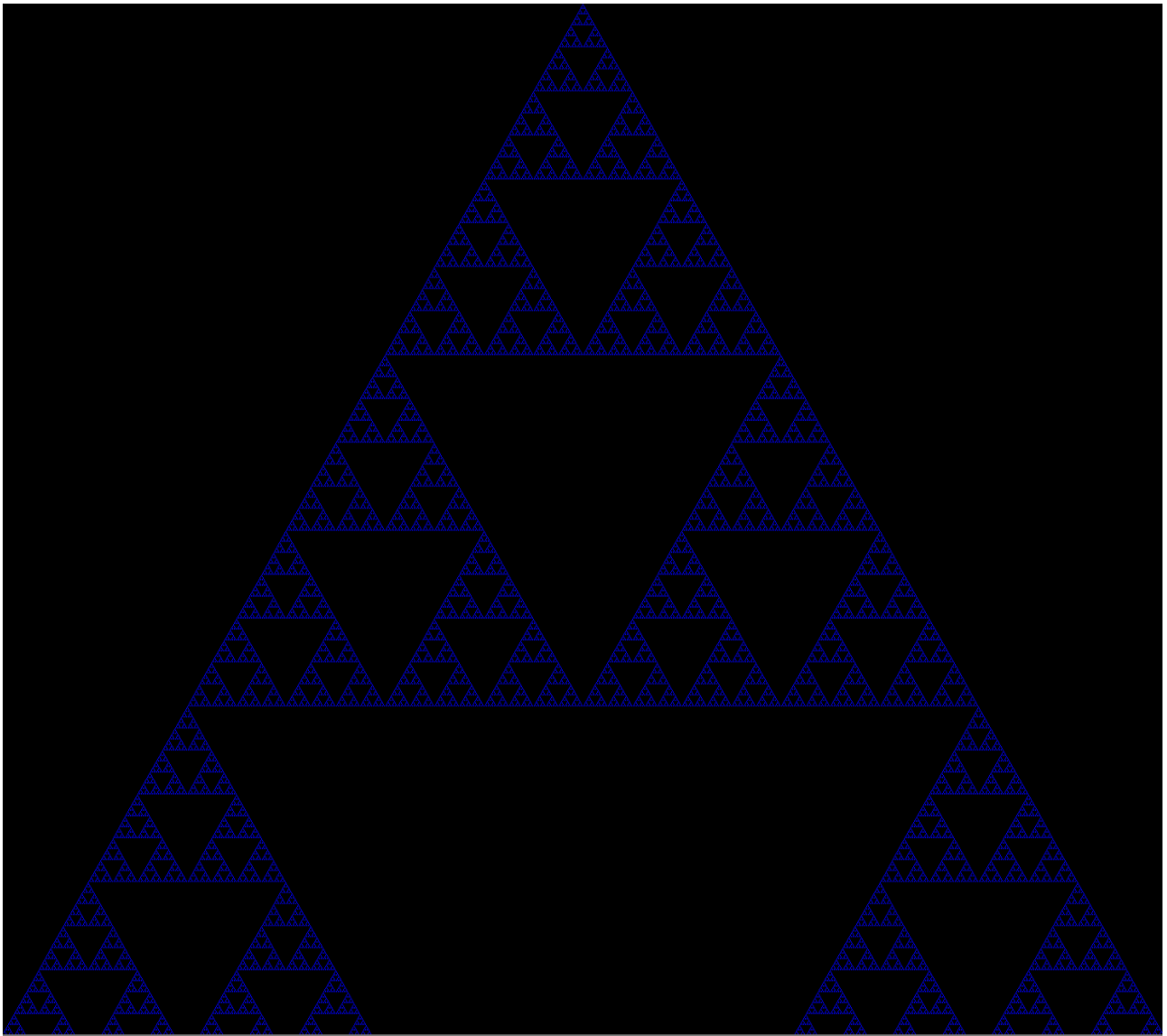
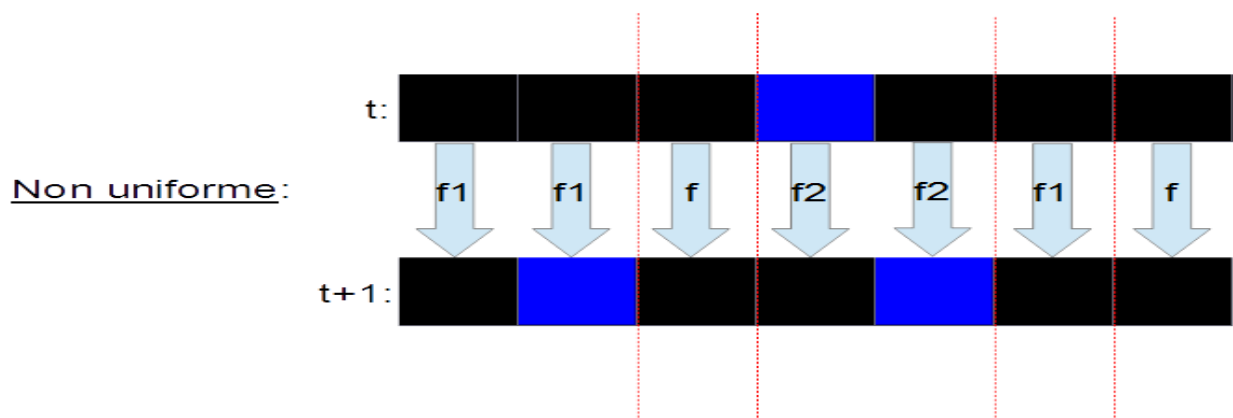
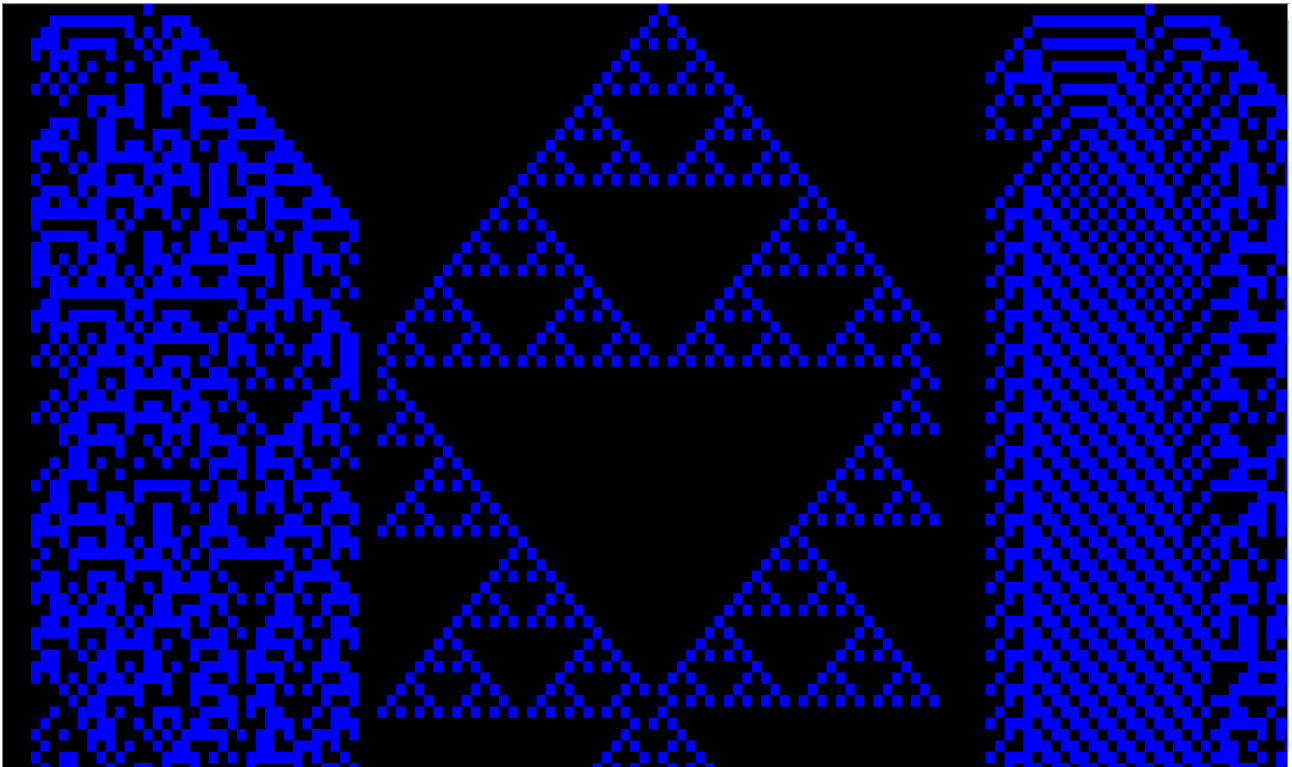


Figure 3: Exemple d'automate cellulaire uniforme, extrait du simulateur réalisé.

Lorsque des cellules du monde cellulaire suivent **des règles différentes**, on parle alors d'automate cellulaire *non uniforme*.



De même que pour les automates cellulaires uniformes, nous sommes en mesure de construire des diagrammes espace-temps :



*Figure 4: Exemple d'automate cellulaire non uniforme, extrait du simulateur réalisé.*

En outre, s'il existe quelques simulateurs d'automates cellulaires uniformes sur le marché (nous pouvons citer entre autres « Golly »), ces derniers ne sont généralement pas conçus pour la recherche scientifique (ce sont généralement des automates en 2 dimensions, avec une taille fixe et non dynamique notamment). De plus, aucun simulateur sur le marché ne propose à l'heure actuelle de construire des automates cellulaires non uniformes.

La principale valeur ajoutée de mon simulateur comparé à ceux existant sur le marché est qu'il comble cette lacune. Ajoutons également que contrairement à la quasi-totalité des simulateurs actuels, le simulateur n'est pas limité en temps dans les diagrammes espace-temps. En effet, le simulateur construit les diagrammes espace-temps de manière procédurale et itérative sans que sa taille ne soit définie au préalable. Ainsi, la seule limite en taille du diagramme espace-temps est l'espace mémoire de

l'ordinateur ce qui permet aux chercheurs d'étudier des comportements de systèmes complexes sur des plages de temps élevées. Notre simulateur est donc particulièrement bien adapté pour un usage de recherche scientifique.

### III- Travail effectué:

Tout d'abord, concernant le langage de programmation pour l'implémentation du simulateur, le langage java a été choisi. En effet, le langage java étant multiplateforme et portable, ce choix semblait pertinent. De plus, le langage java étant très utilisé (c'est le premier langage utilisé dans le monde à l'heure actuelle), le code pourra aisément être repris à des fins d'améliorations.

De même pour les composants graphiques, le module Swing a été préféré pour les même raisons.

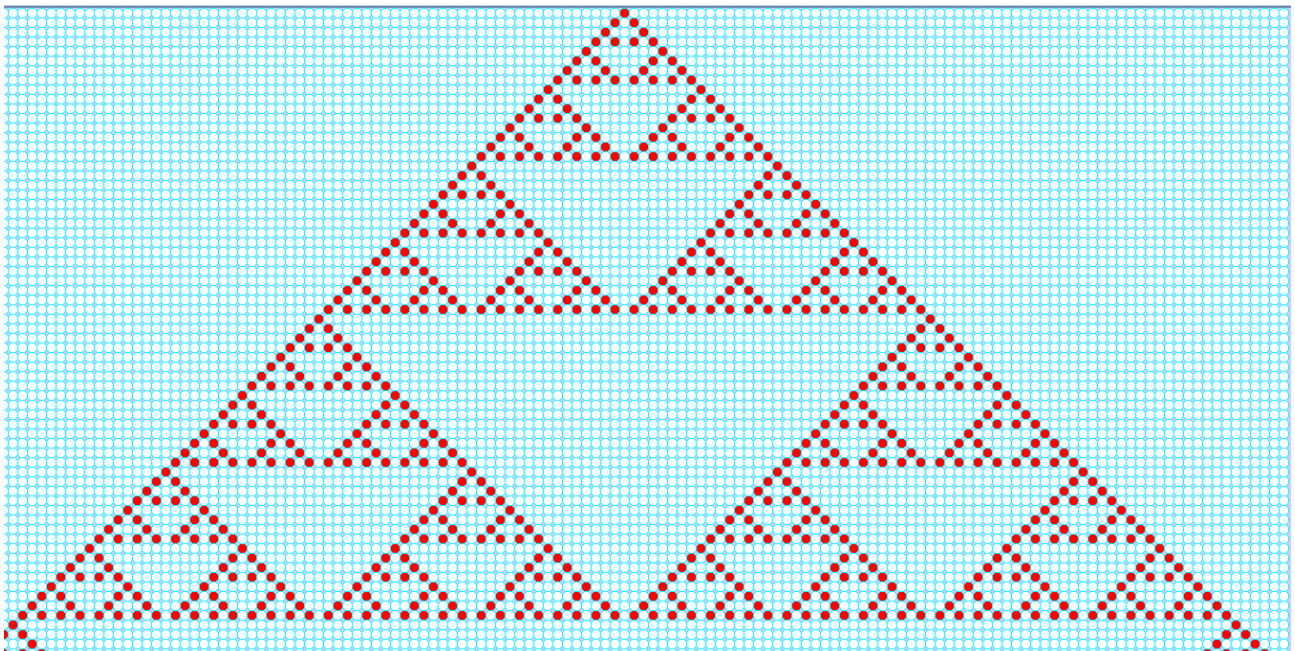
Ainsi, outre l'implémentation complète de l'interface homme-machine (IHM) de l'application, il a également fallu implémenter diverses fonctionnalités tant sur la visualisation dynamique et en temps réel de l'automate cellulaire que sur diverses fonctionnalités annexes telles que la sauvegarde d'un projet ou encore l'exportation du diagramme espace-temps produit sous différents formats (PNG, JPG, GIF, ...), etc... .

Contrairement à beaucoup de simulateur existant, j'ai fais en sorte que l'utilisateur ne soit pas dans l'obligation de spécifier à l'avance le nombre d'étapes en temps du diagramme. Celui-ci est en effet dynamique et se construit en temps réel à une vitesse que l'utilisateur peut spécifier. De cette façon, tant que l'utilisateur ne souhaite pas arrêter la construction du diagramme, ce dernier continuera à se construire.

Cela présente également l'avantage que l'utilisateur peut à tout moment introduire une modification ou une perturbation dans le diagramme, ce qui permet à l'utilisateur d'en visualiser les effets.

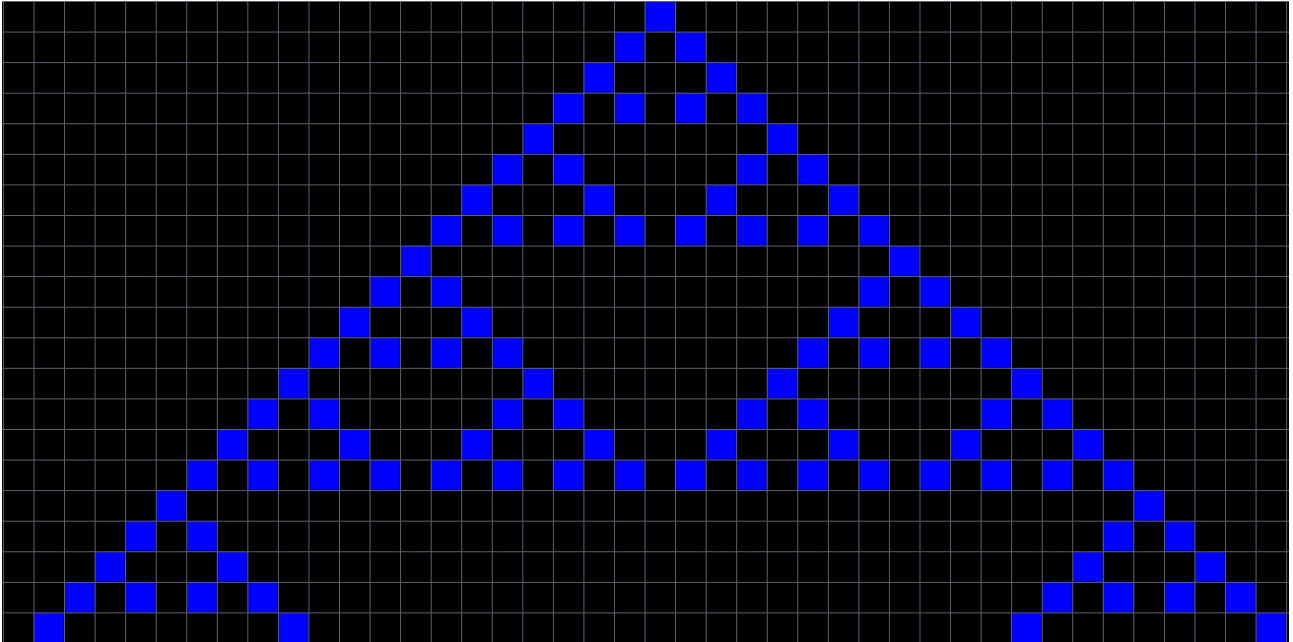
De même, un mode d'incrémentation du diagramme espace-temps par pas modulaire a également été implémenté. Ce mode permet d'incrémenter le diagramme en temps d'une unité de temps ou d'un nombre d'unité de temps élémentaire choisit par l'utilisateur, permettant d'analyser de manière incrémentale l'évolution d'un automate cellulaire.

Nous avons également fait en sorte que l'utilisateur puisse ajuster comme il le souhaite la visualisation d'un automate cellulaire. Ainsi, l'utilisateur est libre de changer la couleur de fond, changer les couleurs correspondantes aux états des cellules, changer la forme des cellules, modifier la taille en espace du diagramme ou encore zoomer comme il l'entend.

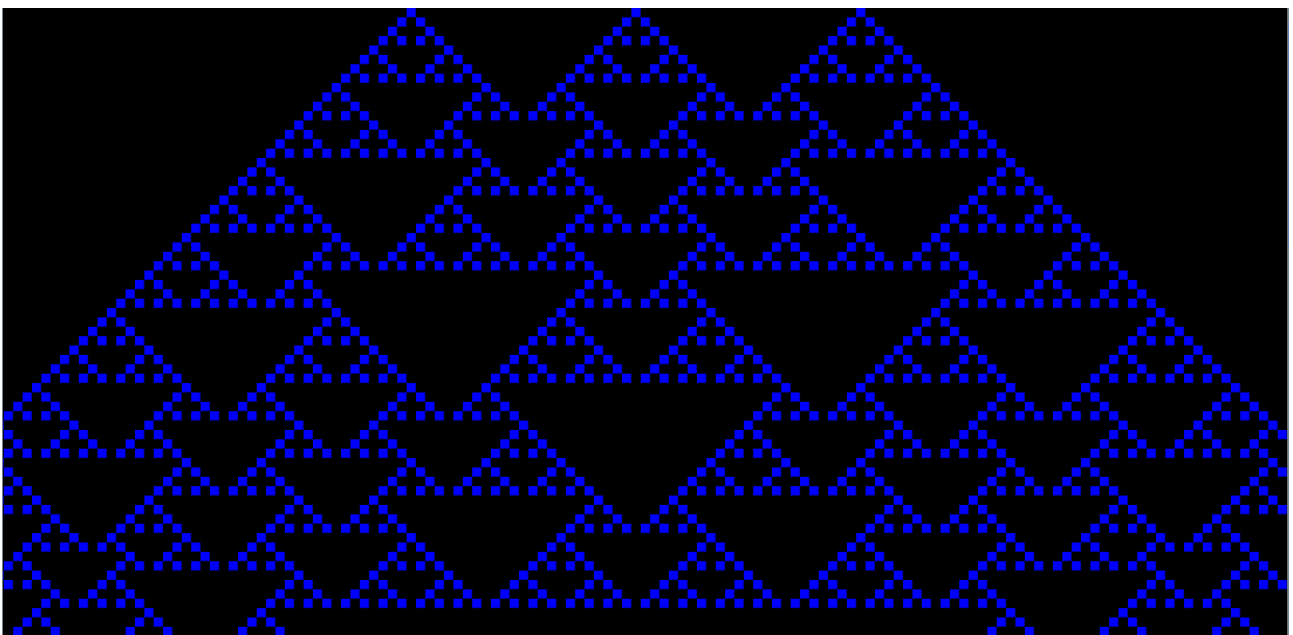


*Figure 5: Automate cellulaire avec paramètres de vue spécifiées, extrait du simulateur réalisé.*

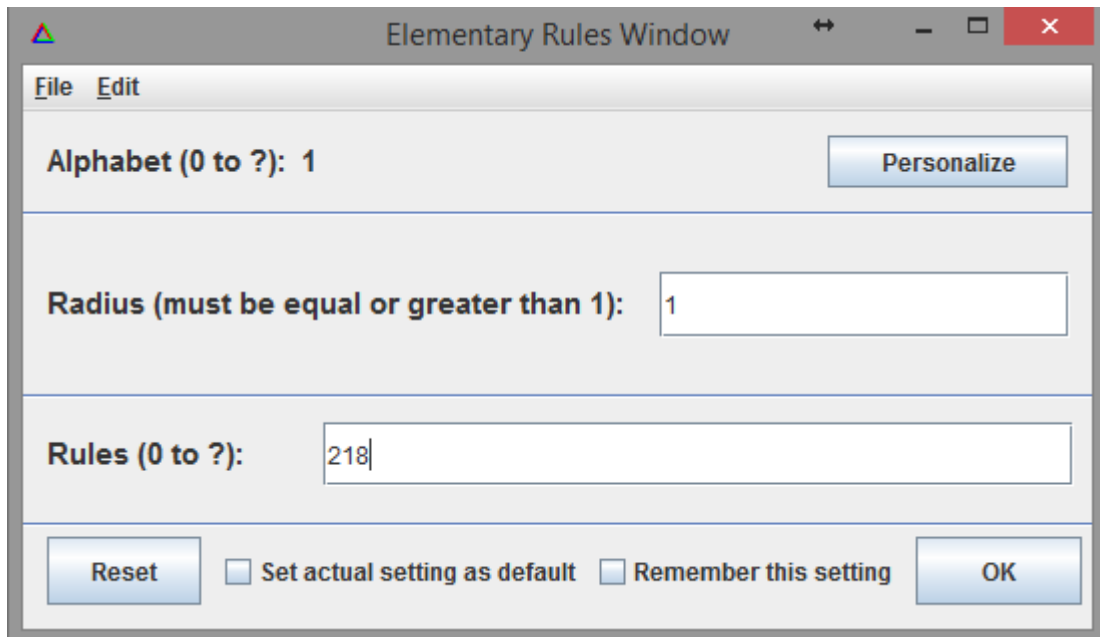
De plus, animés par le même souci d'aider l'utilisateur dans son travail de recherche, l'utilisateur peut également choisir d'afficher une grille dont il peut spécifier à sa guise la couleur et lui permettant de délimiter chaque cellule dans le but de mieux visualiser l'automate.



De même, l'utilisateur peut également choisir l'initialisation par défaut de la configuration initiale et peut à tout moment la modifier manuellement.

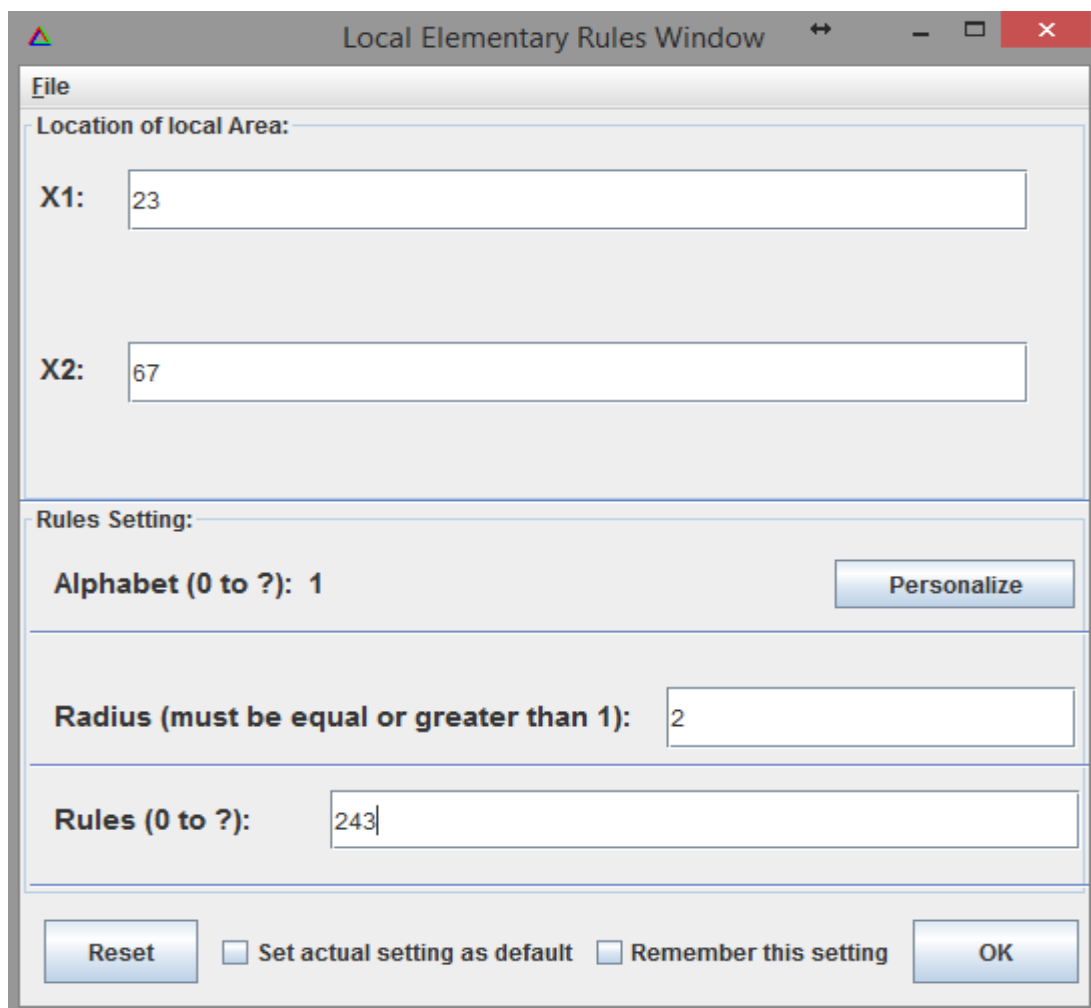


En outre, l'utilisateur est en mesure de choisir des règles d'évolutions que ce soit pour créer des automates uniformes:



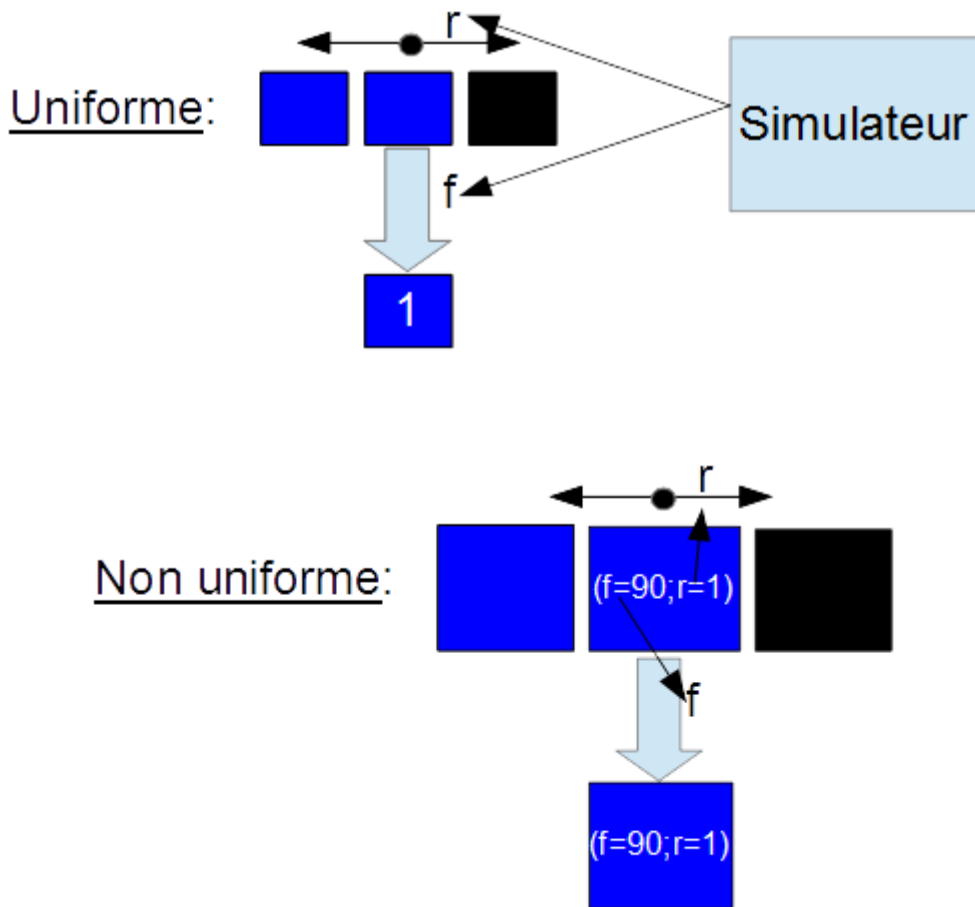
The 'Elementary Rules Window' is a standard Windows-style dialog box with a title bar containing a small icon, the text 'Elementary Rules Window', and standard window controls (maximize, minimize, close). The main area has a menu bar with 'File' and 'Edit'. Below the menu bar, there are three sections: 1. 'Alphabet (0 to ?): 1' with a 'Personalize' button to its right. 2. 'Radius (must be equal or greater than 1):' followed by a text input field containing the value '1'. 3. 'Rules (0 to ?):' followed by a text input field containing the value '218'. At the bottom, there is a row of controls: a 'Reset' button, two unchecked checkboxes labeled 'Set actual setting as default' and 'Remember this setting', and an 'OK' button.

Ou non uniformes:



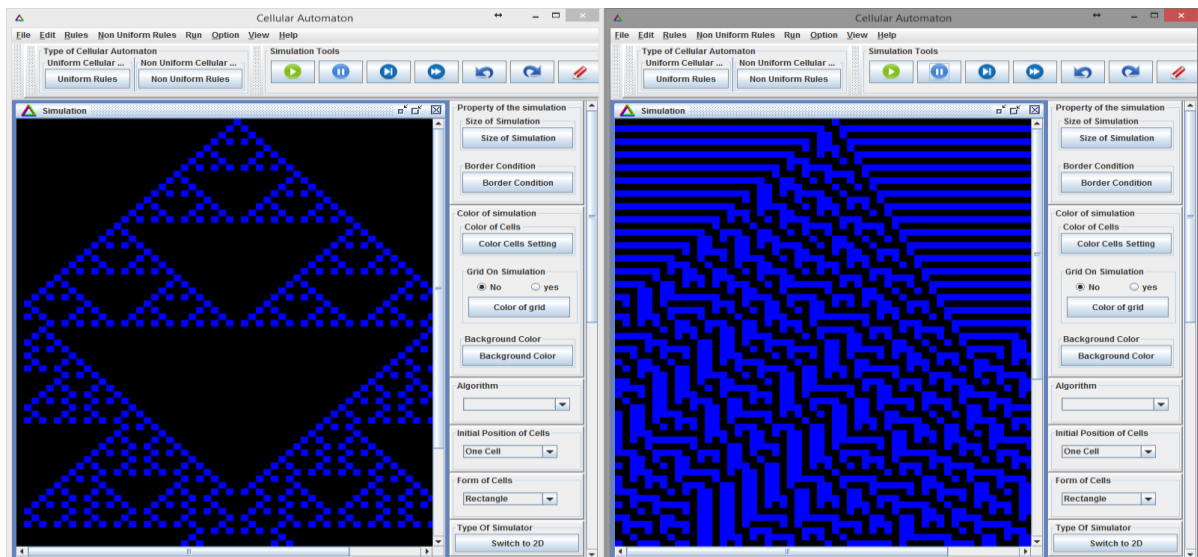
The 'Local Elementary Rules Window' is a standard Windows-style dialog box with a title bar containing a small icon, the text 'Local Elementary Rules Window', and standard window controls (maximize, minimize, close). The main area has a menu bar with 'File'. Below the menu bar, there are three sections: 1. 'Location of local Area:' followed by two text input fields: 'X1:' containing '23' and 'X2:' containing '67'. 2. 'Rules Setting:' followed by three sections: 'Alphabet (0 to ?): 1' with a 'Personalize' button to its right; 'Radius (must be equal or greater than 1):' followed by a text input field containing '2'; and 'Rules (0 to ?):' followed by a text input field containing '243'. At the bottom, there is a row of controls: a 'Reset' button, two unchecked checkboxes labeled 'Set actual setting as default' and 'Remember this setting', and an 'OK' button.

La programmation des automates non uniforme a été réalisée en modifiant l'approche de l'implémentation des règles d'évolutions. En effet, lorsque l'automate devient non uniforme, ce n'est plus le simulateur qui contient l'information concernant la règle d'évolution mais les cellules qui contiennent et héritent de cette information.



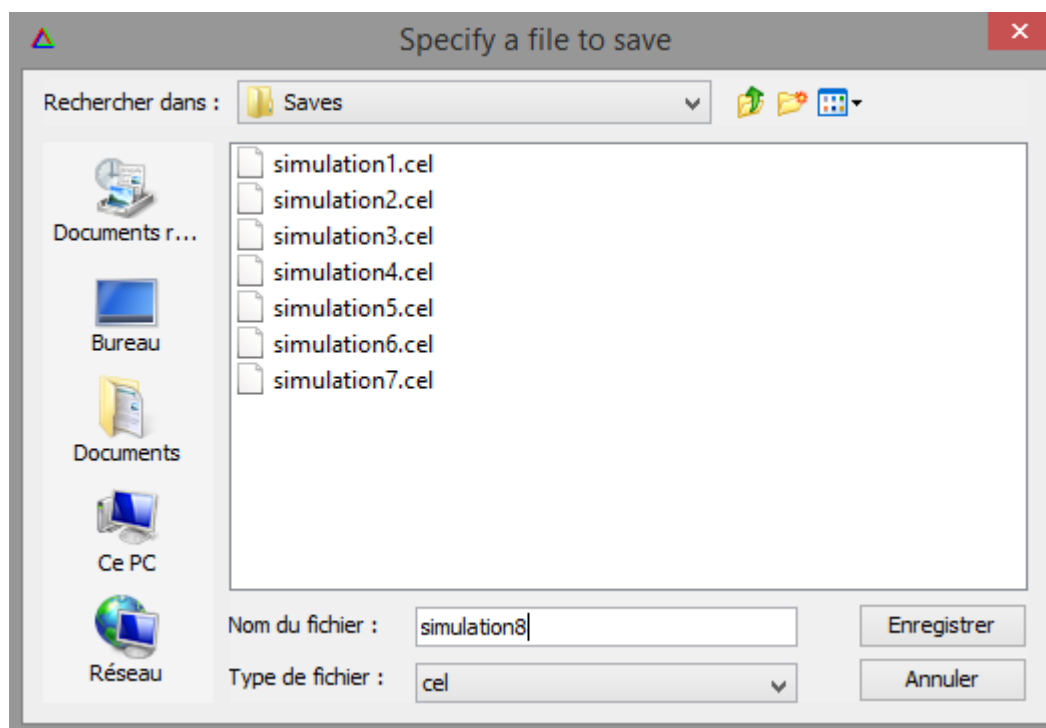
De plus, l'utilisateur peut également faire fonctionner plusieurs simulateurs en même temps de sorte qu'il puisse visualiser plusieurs scénarios en même temps.

Pour cette fonctionnalité, nous avons fait appel au multithreading et à la programmation concurrente.

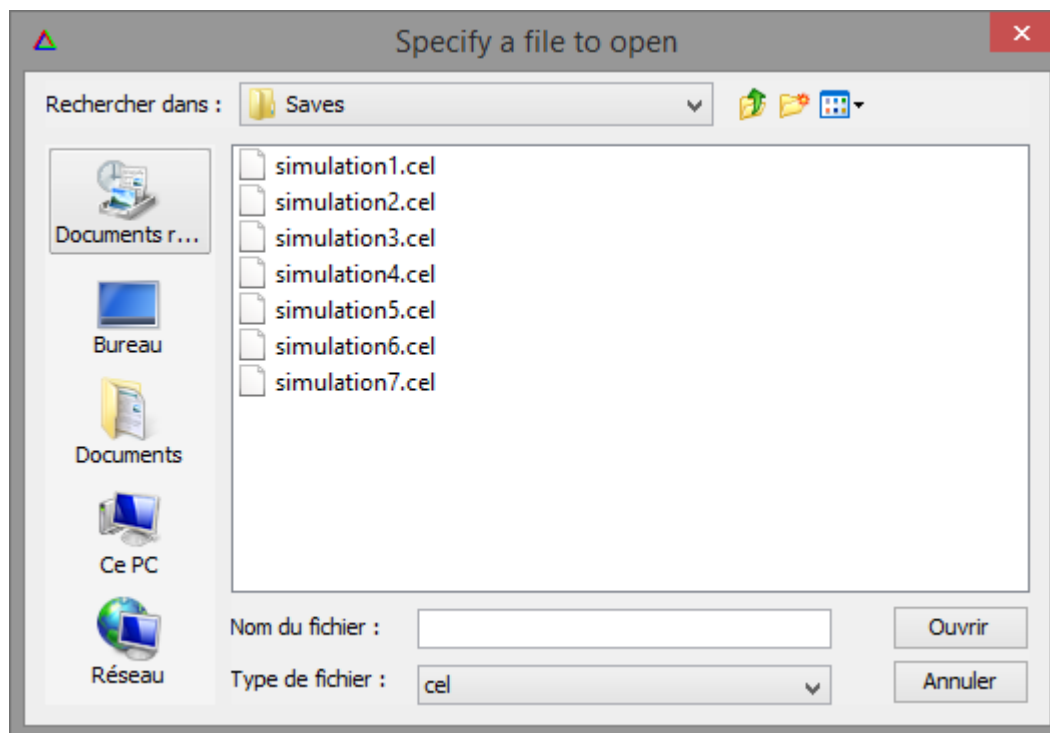


Nous avons également mis à disposition de l'utilisateur, plusieurs outils permettant de l'aider dans sa tâche.

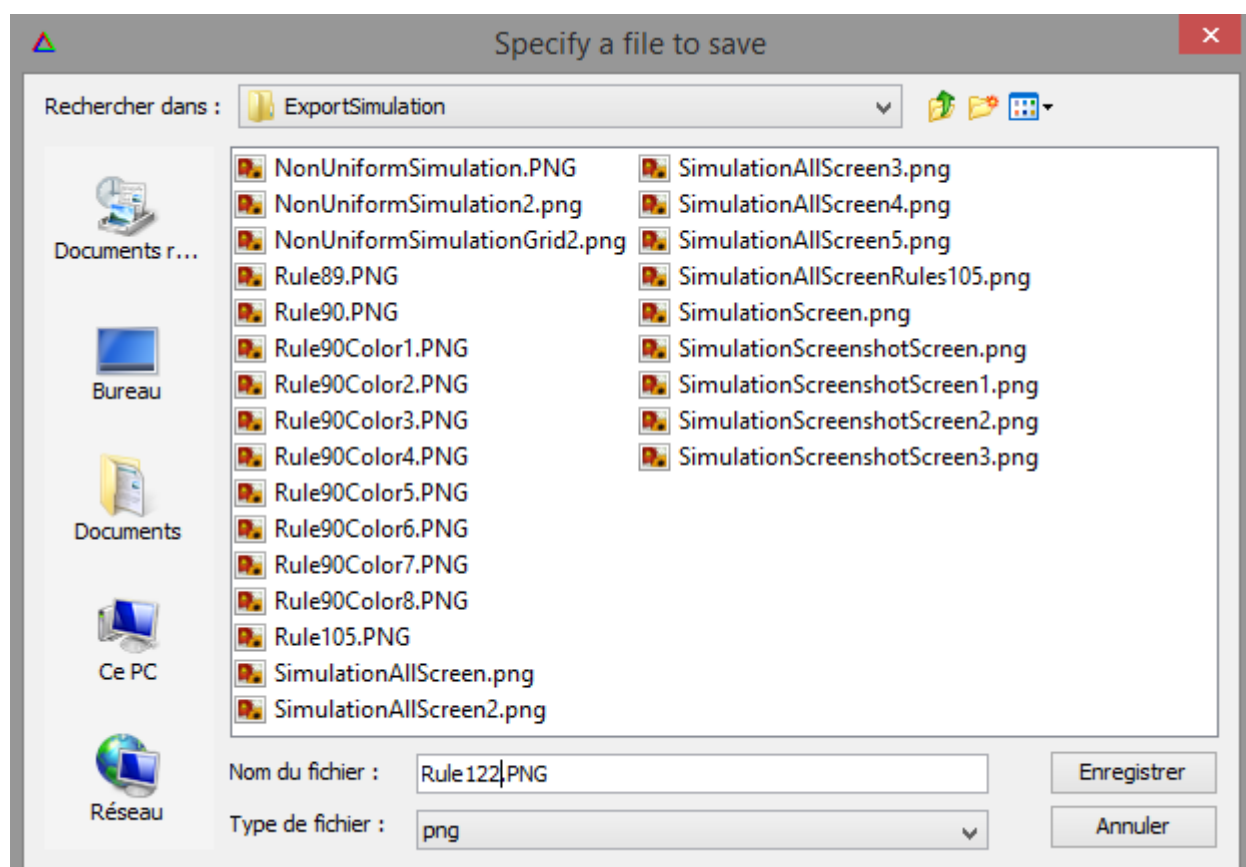
En outre, nous avons implémenté un système permettant de sauvegarder une simulation pour pouvoir la rouvrir par la suite. Les slots de sauvegarde ont en outre leur propre extension, à savoir l'extension « .cel », ce qui permet leur identification de manière aisée.



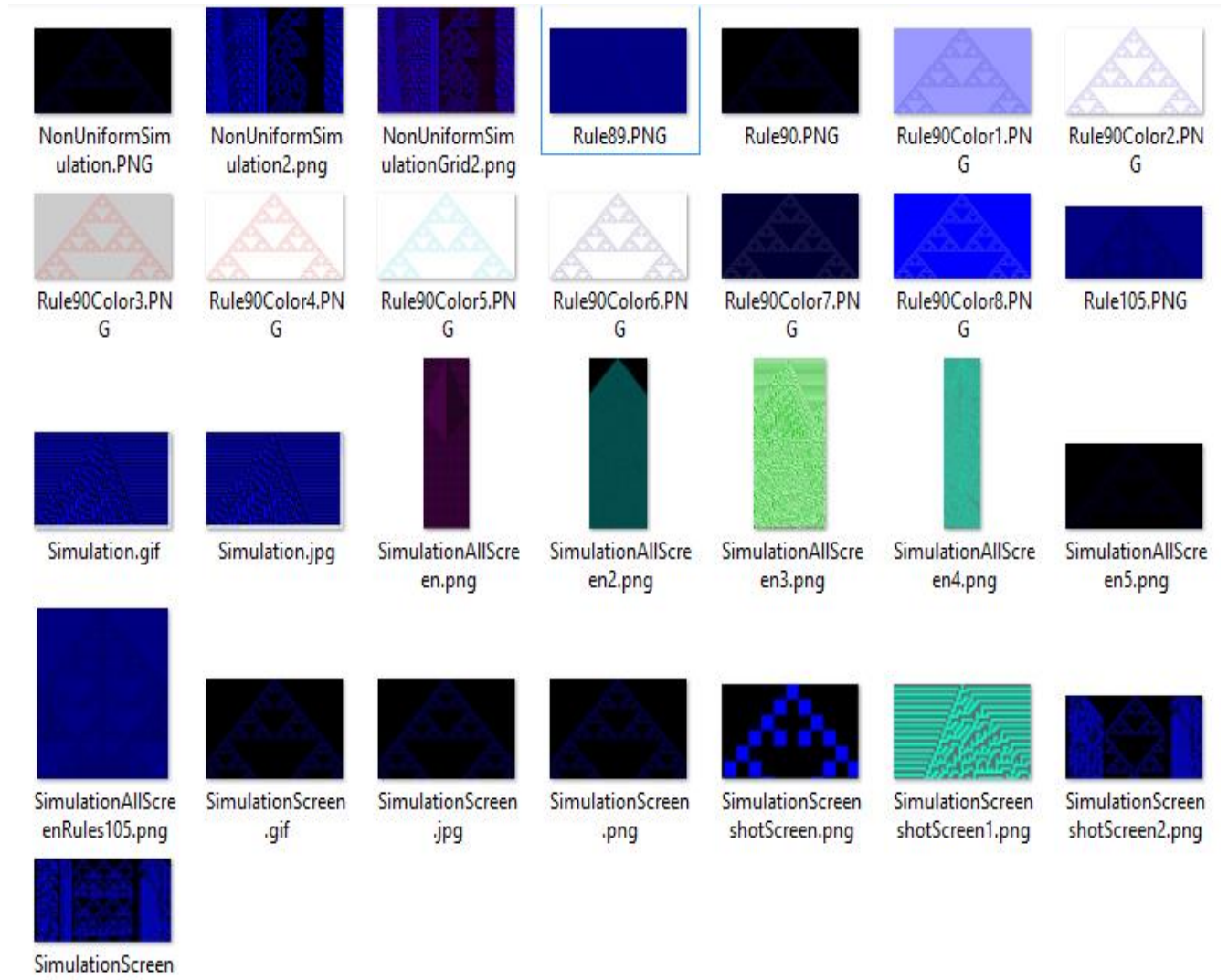




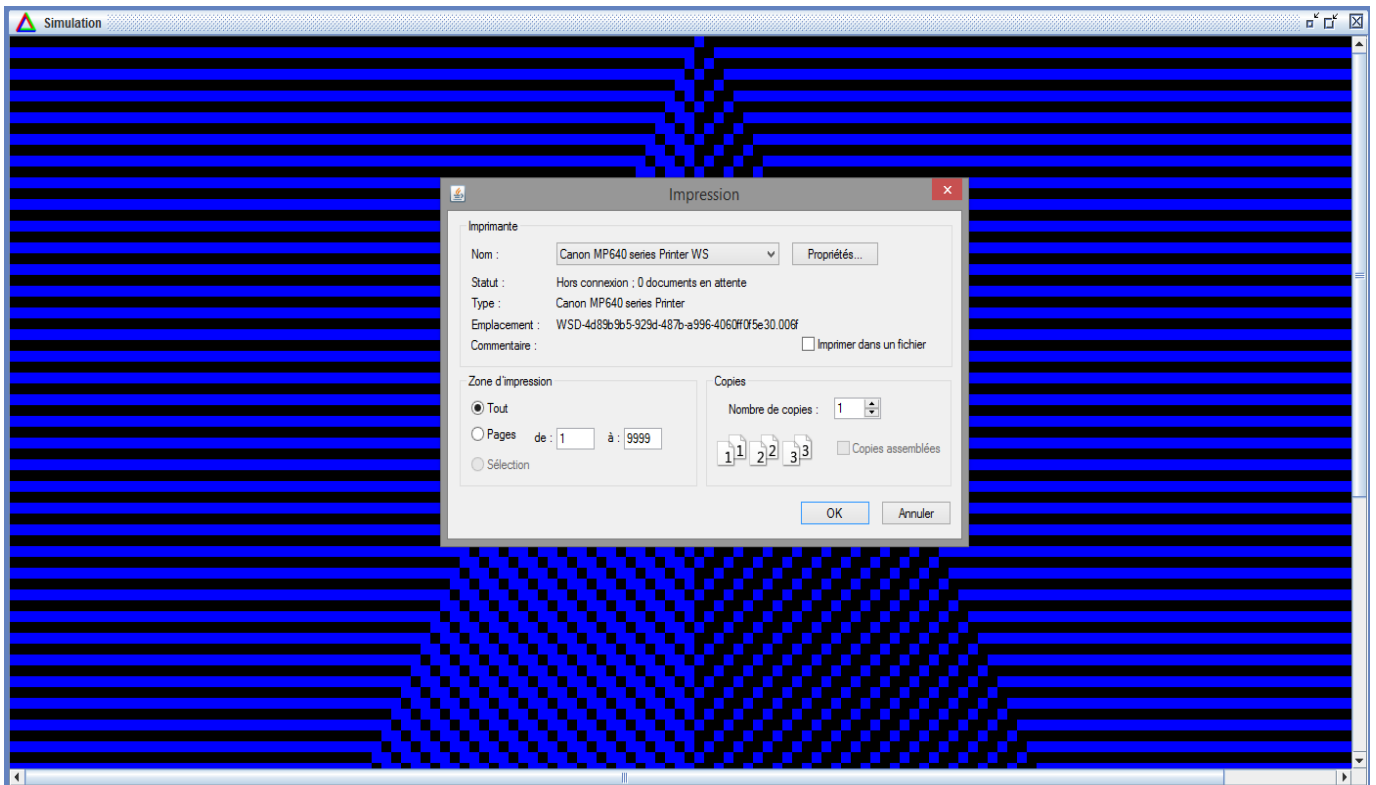
L'utilisateur peut également exporter l'ensemble ou une partie d'un diagramme espace-temps sous divers formats: PNG, JPG, GIF, ... .



Cette fonctionnalité est primordiale dans l'optique où l'outil peut être utilisé pour illustrer des articles de recherche sur les automates cellulaires.



L'utilisateur est même en mesure d'imprimer le simulateur, ainsi que la simulation directement via l'application.



Concernant l’algorithmique, il a également fallu élaborer un algorithme permettant la transformation de la représentation des règles sous forme décimale saisit par l’utilisateur, pour en déduire la règle d’évolution d’une configuration donnée prenant en compte le rayon.

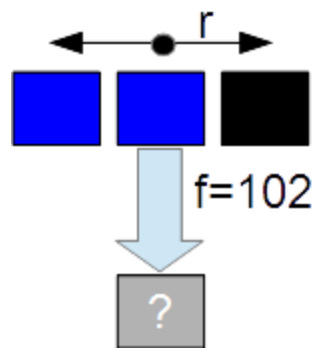
Nous prendrons par la suite les exemples suivant pour illustrer l’algorithme:

### **Exemple 1:**

Radius: 1

Rules: 102

Configuration du voisinage:

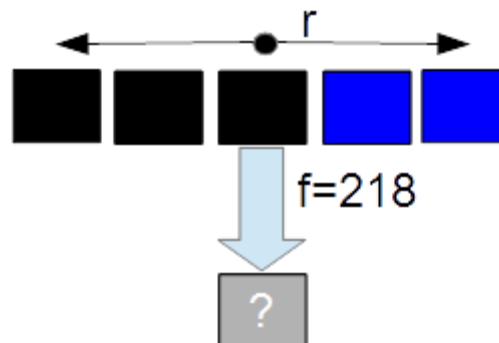


## Exemple 2:

Radius: 2

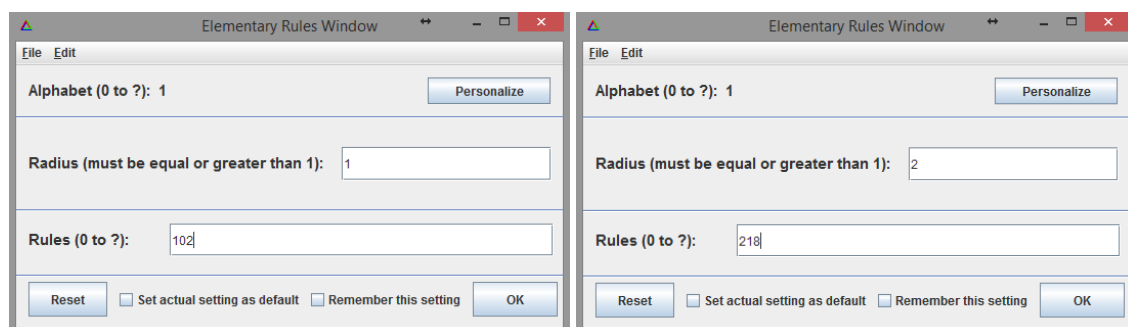
Rules: 218

Configuration du voisinage:



Cet algorithme que j'ai élaboré est décrit ci-dessous:

- L'utilisateur entre une règle sous sa forme décimale, par exemple la règle 102 avec un rayon de 1 ou encore la règle 218 avec un rayon de 2.



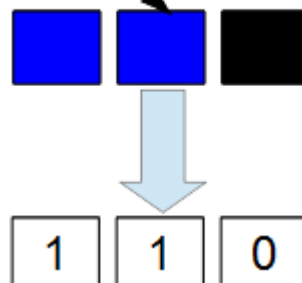
- Nous stockons donc le premier nombre dans une variable que nous appellerons  $n$ . Dans nos exemples, nous aurions  $n=102$  pour le premier exemple et  $n=218$  pour le deuxième.
- Puis pour chaque cellule:
  1. Enregistrer l'état de la cellule et ceux de ses voisins en fonction du rayon spécifié par l'utilisateur.
  2. Transformer la configuration du voisinage de la cellule en nombre binaire. Nous rappelons par ailleurs que nous supposons ici que  $A = \{0 ; 1\}$ , avec  $A$  l'alphabet de la simulation.  
Ainsi, si nous avons par exemple un rayon égale à 1 et une configuration telle que:

Cellule considérée



Alors nous transformons cette configuration en nombre binaire comme suit:

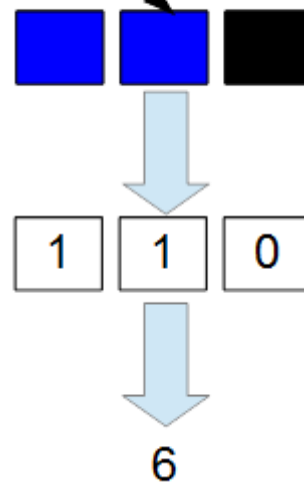
Cellule considérée



Puis nous retransformons ce nombre binaire en nombre décimale ce qui nous donne ici:

(110) en base 2 = (6) en base 10

Cellule considérée



Nous stockons donc 6 dans une variable:  $k = 6$ .

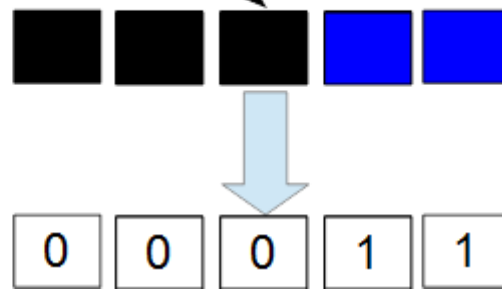
De même, si nous avons par exemple, un rayon égale à 2 et une configuration de la cellule considérée telle que:

Cellule considérée



Alors nous transformons cette configuration en nombre binaire comme suit:

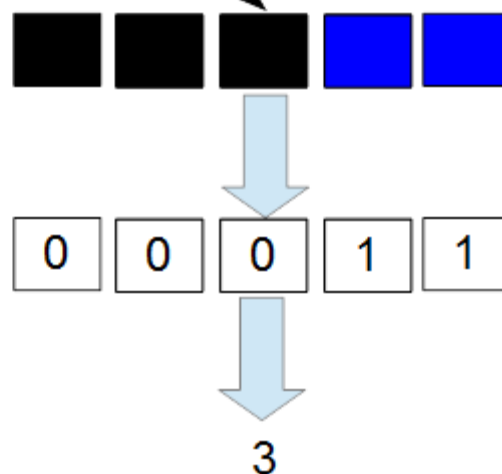
Cellule considérée



Nous retransformons ce nombre binaire en nombre décimale ce qui nous donne ici:

(00011) en base 2 = (3) en base 10

Cellule considérée



Nous stockons donc 3 dans une variable:  $k = 3$ .

3. Puis nous appliquons la formule suivante en utilisant des opérations booléennes pour trouver le prochain état de la cellule considérée:

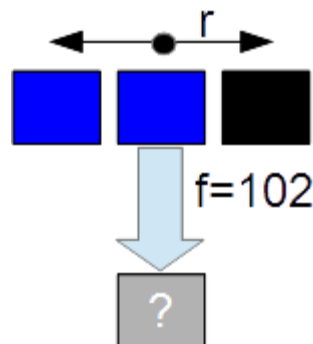
$$(n \& (1 \ll k)) \gg k$$

- Avec  $n$  le numéro de la règle saisi par l'utilisateur.
- $k$  le nombre trouvé et stocké par les étapes précédentes.
- Et  $\ll$  ,  $\gg$  qui sont respectivement le décalage de bits à gauche et le décalage de bits à droites. C'est opération ont de plus l'avantage de détenir d'excellentes performances temporelles.

Ainsi, dans notre premier exemple, nous avons:  
 $n = 102$  et  $k = 6$ .

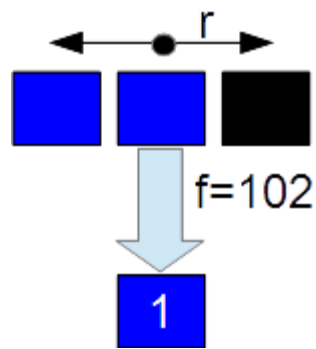
D'où,  
 $(n \& (1 \ll k)) \gg k$   
 $= (102 \& (1 \ll 6)) \gg 6$   
 $= (102 \& (1000000)) \gg 6$   
 $= 1$

Nous trouvons donc que la configuration:



Donne une cellule à l'état 1 pour la règle 102, ce qui est effectivement le cas.





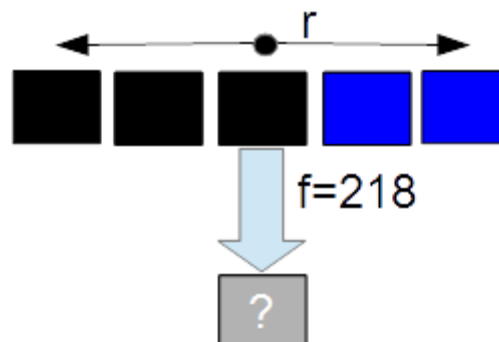
De même, pour notre deuxième exemple, nous avons:

$n = 218$  et  $k = 3$ .

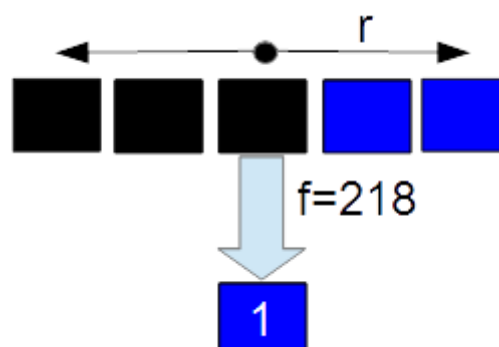
D'où,

$$\begin{aligned} & (n \& (1 \ll k)) \gg k \\ &= (218 \& (1 \ll 3)) \gg 3 \\ &= 1 \end{aligned}$$

Nous trouvons donc que la configuration:



Donne une cellule à l'état 1 pour la règle 218, ce qui est effectivement le cas.



## IV- Gestion de projet:

Concernant la gestion de projet, j'ai utilisé de nombreux modules et logiciels d'aide à la gestion et à la maintenabilité du code.

Tout d'abord, le projet utilise l'outil « Maven » pour permettre une gestion facilitée des dépendances du projet.

Nous avons également utilisé Git comme logiciel de gestion de versions. Cela m'a permis de créer de nombreuses branches à chaque amélioration majeure du code et de garder un historique de chacune d'elle. Aussi, un repository (dépôt) Github dédié au projet a été créé pour permettre une consultation libre et une éventuelle amélioration future du projet. Ce repository se trouve à l'adresse suivante :

<https://github.com/alexandreauda/CellularAutomaton>

Concernant la gestion des différentes améliorations et des fonctionnalités à implémenter, j'ai utilisé le logiciel Redmine. Ce logiciel est un logiciel de management de projet et d'outil de suivi de bugs. Ce logiciel se place comme un concurrent libre de BugZilla, Mantis ou encore JIRA. En outre, j'ai utilisé ce logiciel sous sa forme de méthodologie agile grâce à un tableau de bord Kaban mis à disposition par le logiciel. Par conséquent, chaque fonctionnalité, bug à corriger, ainsi que chaque évolution future du code été consignée sous la forme de tickets sur le tableau de bord Kaban. Chaque ticket représentait une tâche dont j'assignais une priorité, ainsi que le type du ticket. Cela me permettait donc de suivre et de prioriser efficacement les tâches à effectuer.

Nouveau (20)	En cours (7)	Résolu (33)	Commentaire (0)
Feature (Evolution) #8 Cr��r la fen��tre du "Direction of evolution setting"	Feature (Evolution) #59 mettre les commentaires des cellule dans les cellules	Feature (Evolution) #37 Pouvoir choisir la form des cells	
Feature (Evolution) #14 Refactorer l'IHM de la fen��tre principal	Feature (Evolution) #60 Faire un setToolTipsText (ou autre) sur les cellules de mani��re �� pouvoir visualiser la couleur ...	Feature (Evolution) #41 Pouvoir choisir la couleur des cellules + Grid + BackGround	
Feature (Evolution) #20 Rajouter le bouton "Bookmarks" dans la fen��tre ElementaryRulesWindow	Feature (Evolution) #43 Bouton Refresh -> Bouton qui recalcul la simulation �� partir du curseur.	Feature (Evolution) #63 Mettre une autre couleur par default pour la couleur de la grille	
Feature (Evolution) #21 Impl��menter les boutons de la fen��tre ElementaryRulesWindow.	Feature (Evolution) #54 Mettre des infos Bulles sur les boutons	Feature (Evolution) #58 Mettre la simulation en pause lorsque l'on resize	
Feature (Evolution) #24 Faire la fen��tre qui apparait lorsqu'on click sur le bouton "Border Condition Setting"	Feature (Evolution) #30 Impl��menter des boutons (dans le MenuBar par exemple) qui cr��e une nouvelle fen��tre de simulation...	Feature (Evolution) #61 Faire que l'on puisse initialiser la grille �� la vol�� avec du drag&drop. Si on cli gauche est que...	
Feature (Evolution) #25 Impl��menter les diff��rents types de configuration au bord.	Feature (Evolution) #51 cr��er un bouton qui lauch la simulation de n ��tapes dans le temps avec n qui peut ��tre choisi par...	Feature (Evolution) #40 Bouton qui "efface" (set l'��tat de toute les cellule �� 0) la simulation ->Logo une gomme ou un ef...	
Feature (Evolution) #26 Impl��menter la direction d'��volution de la simulation.	Feature (Evolution) #50 cr��er un bouton qui lauch la simulation d'une ��tape dans le temps	Feature (Evolution) #44 Faire un syst��me pour choisir le nom et l'endroit de l'exportation jpg; png ou gif	
Feature (Evolution) #28 Export canvas to pdf.		Feature (Evolution) #35 Faire que la taille du Screen et de la simulation s'incr��mente tant que l'on ne stop pas la simul...	
Feature (Evolution) #34 Cr��er une fen��tre o�� l'on pourrait y sauvegarder une configuration initial en faisant un drag & d...		Feature (Evolution) #17 R��gler le probl��me des dimension du simulateur et du resize du simulateur.	
Feature (Evolution) #38 Zoom -> agrandir les Cells avec			

Concernant le code    proprement parl  , j'ai mis en place un syst  me de loggers pour les r  cup  rations des exceptions    l'aide du module log4J. A la diff  rence des rattrapages d'exceptions classiques, les loggers permettent d'enregistrer la date o   survient une erreur, de leurs assigner un niveau de gravit   et de d  finir le contenu de l'erreur. Gr  ce aux loggers, j'  tais en mesure de suivre pr  cis  ment les erreurs et donc de d  bugger plus facilement et plus rapidement. De plus, les erreurs r  cup  r  es par les loggers   t   stock  es dans une base de donn  es pr  alablement mis en place. Cela nous a permis de garder un historique des erreurs afin de suivre l'  volution du d  bogage dans le temps. En parall  le, nous utilisons un plug-in   clipse appel   Sonar-Lint qui permettait d'analyser en temps r  el la qualit   du code   crit, et proposait de respecter les bonnes pratiques d'impl  mentation.

En ce qui concerne la proc  dure d'impl  mentation, le projet s'articulait autour de plusieurs   tapes telles que d  crites ci-apr  s : La premi  re   tape a consist      comprendre puis    analyser les aspects th  oriques du probl  me des automates cellulaires en vue

d'une implémentation future. La deuxième étape fût de construire des algorithmes répondant aux critères demandés tout en réfléchissant à des solutions aux problèmes liés à l'implémentation (optimisation en temps de calcul, optimisation en espace mémoire, lisibilité du code, etc...). Je me suis ensuite penché sur le maquettage de l'IHM de manière à répondre au mieux au besoin des utilisateurs puis je me suis attaché plus spécifiquement à l'implémentation du cœur de l'application ainsi que l'implémentation de son IHM. A ce stade, je me suis alors penché sur l'implémentation du cœur logique de l'application, ainsi que la programmation des différentes fonctionnalités en respectant les niveaux de priorité des tickets inscrit dans Redmine. De plus, une attention particulière fût portée afin de tester et vérifier le bon fonctionnement des différentes fonctionnalités implémentées à chaque étape du processus d'implémentation. Ainsi, lorsqu'un dysfonctionnement était trouvé ou que la fonctionnalité méritait une amélioration, celui-ci était immédiatement transformé en ticket par l'intermédiaire de Redmine et un niveau de priorité lui était donné comme décrit plus haut.

## V- Conclusion:

Nous avons introduit dans ce présent rapport, différents concepts liés aux systèmes dynamiques et plus précisément liés à l'étude des automates cellulaires, comme les concepts d'uniformité et de non uniformité des automates. Nous avons été amenés lors de ce projet à modéliser puis à concevoir des algorithmes permettant de simuler de tels concepts, de même que la construction d'une interface homme-machine (IHM), ainsi que diverses fonctionnalités répondant aux critères et aux besoins d'utilisateurs finaux.

Aussi, ce projet fut fort enrichissant et ceci, à plusieurs niveaux. Outre la conception et la réalisation d'une interface homme-machine, ce projet nous a également permis de réfléchir à la transposition d'un modèle théorique en un modèle permettant son implémentation ainsi que son optimisation.

De plus, ce projet nous a permis d'utiliser divers logiciels de gestion de source et de gestion de projet, ainsi que l'occasion de pouvoir les mettre en œuvre sur un projet conséquent et donc, de pouvoir en mesurer l'importance.

Par ailleurs, cette expérience fut l'occasion d'entrevoir les mécanismes de la gestion d'un projet d'envergure.

Ce TER fut également propice à l'expérimentation d'une implémentation logiciel devant répondre à des besoins spécifiques des utilisateurs de l'application.

Enfin, ce projet fut l'occasion d'apporter des fonctionnalités qui ne sont pas à l'heure actuelle présentes sur le marché telle que la construction d'automates cellulaires non uniforme.

## VI- Perspectives et réflexions personnelles:

Les automates cellulaires sont un vaste sujet qui fait l'objet d'études encore aujourd'hui. En effet, le modèle des automates cellulaires est simple dans sa définition mais le comportement macroscopique de ces derniers peut être complexe.

Dans ce contexte, ce sujet laisse un large champ de perspectives quant à l'amélioration d'un simulateur d'automates cellulaires. Ainsi, nous pourrions continuer l'implémentation de notre simulateur pour y intégrer de nouvelles fonctionnalités comme la prise en compte de nouvelles conditions au bord, améliorer l'IHM pour une expérience toujours plus agréable pour l'utilisateur, optimiser le code pour améliorer les temps de calculs, ou encore définir une règle par un langage de scripts que pourrait apprendre l'utilisateur.

Ce projet est donc très ouvert et les perspectives qu'offre ce projet sont nombreuses.

## Annexes:

### A.1- Bibliographie:

[http://www.i3s.unice.fr/~provilla/resources/pdf/these\\_provillard.pdf](http://www.i3s.unice.fr/~provilla/resources/pdf/these_provillard.pdf)

<http://mathworld.wolfram.com/ElementaryCellularAutomaton.html>

<http://mathworld.wolfram.com/AdditiveCellularAutomaton.html>

[https://en.wikipedia.org/wiki/Cellular\\_automaton](https://en.wikipedia.org/wiki/Cellular_automaton)

[https://en.wikipedia.org/wiki/Sierpinski\\_triangle](https://en.wikipedia.org/wiki/Sierpinski_triangle)