Jeux de la vie

Principe

Le jeux de la vie fonctionne sur un grille (un plateau).

Ce jeux contient des un nombre fixé de case, chaque case est soit activée ou désactivée.

Voici par exemple une capture d'écran de la première étape du jeux de la vie :

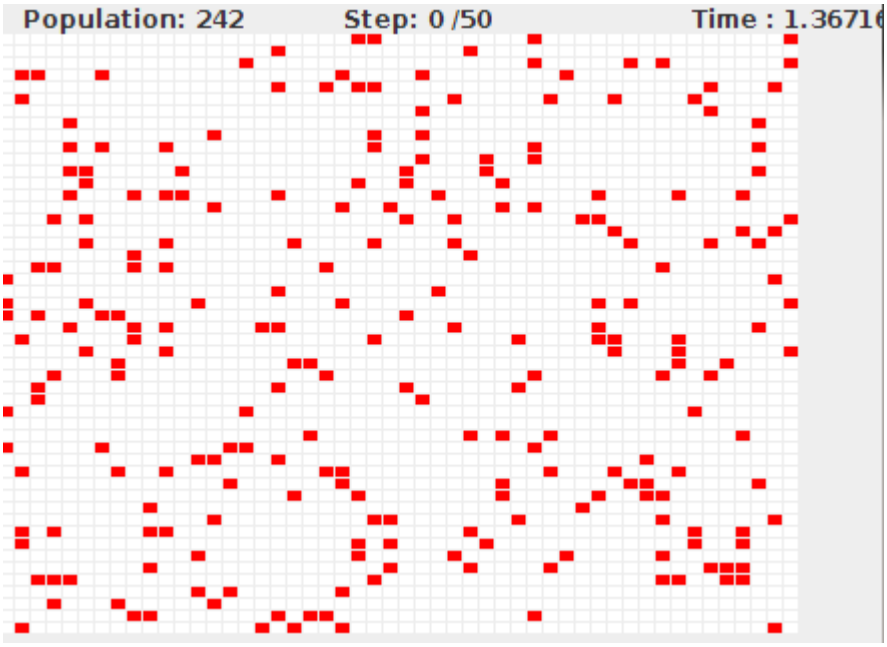


Illustration 1: Capture d'écran du jeu de la vie

Cette grille peut être rempli au départ de manière aléatoire, ou part des cases choisis par l'utilisateur.

Le jeux n'a pas vraiment de but, mais consiste en une succession d'étape pouvant éventuellement amener à un état terminal dans lequel les cases ne bougent plus.

A chaque étape, les cases prennent une nouvelle valeur qui est calculée en fonction de ses voisins.

Si le nombre de voisin est de :

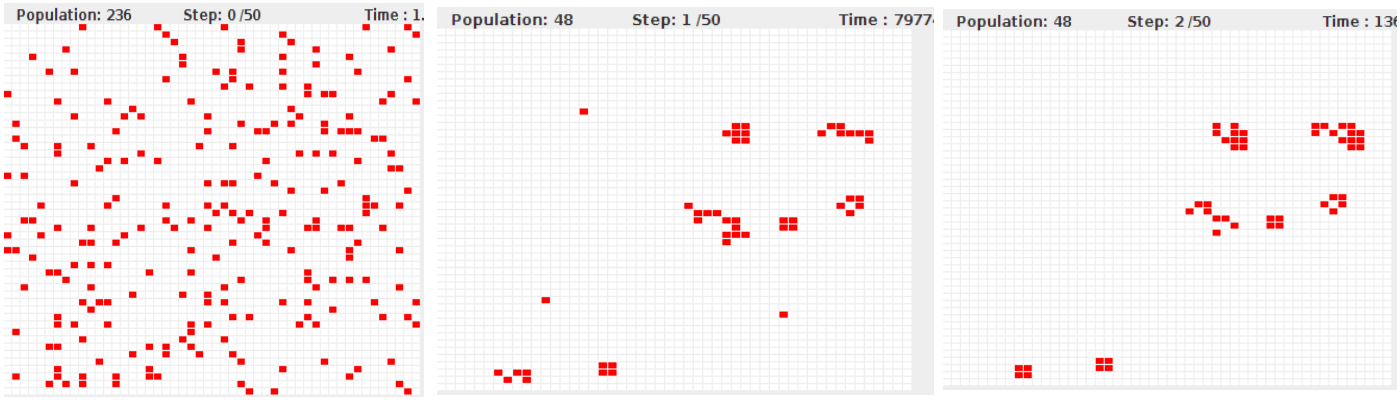
- 3 alors la cas prend vie et devient active (rouge dans la capture d'écran).
- 2 et la case été active au tour précédent, alors elle reste active au nouveau tour.
- < 2 alors la case est désactivée par famine.
- > 3 alors la case est désactivée par surpopulation.

Voici la liste des voisins à vérifié pour connaître la future valeur de la case «ici».

Ainsi les valeurs de « ici » peuvent varié de 0 à 8.

1	2	3
4	ici	5
6	7	8

Exemple :



Ci dessus des captures d'écran montrant les 3 premières étapes du jeux de la vie.

Algorithmes de calcul implémentés

Algorithme linéaire

Cette algorithme parcourt la grille dans son intégralité à chaque étape.

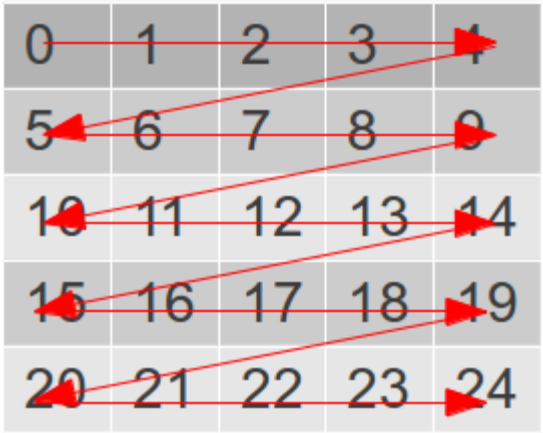
A chaque case analysée, les voisins sont comptés, puis la valeur calculé est sauvegardé dans une nouvelle grille.

Exemple :

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29

Ainsi le calcul pour la valeur de la case 0 nécessite de connaître les valeurs des cases 6, 7 et 1.

Dans cette exemple il est nécessaire de faire 30 calculs les uns a la suite des autres dans un seul thread pour chaque étape.



Algorithme de calcul

Le calcul des cases se fait sequentielement.

Inconvénient

Les calculs se font séquentiellement les uns à la suite des autres.

Algorithmes avec threads

Un thread par case

Principe

Le but étant de faire calculé la prochaine étape de calcul d'une case par un thread.

Section critique

Il n'y a aucune section critique, en effet il n'y a pas de zone partagée. La grille qui est lue n'est pas modifiée mais une autre grille temporaire est créer, dans laquelle on insère les nouvelles données.

Fonctionnement

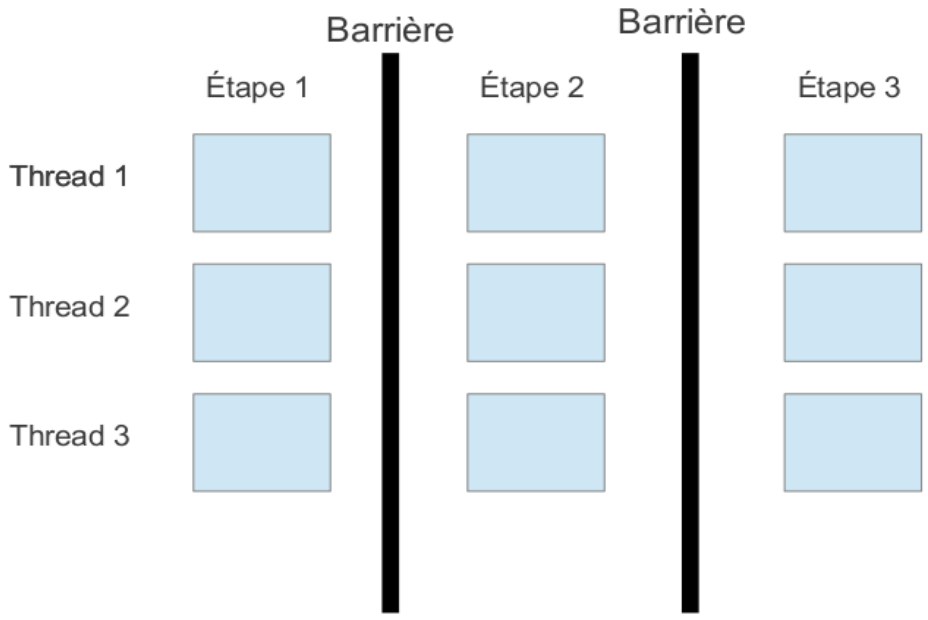
Le problème réside dans le fait que les threads doivent être synchronisés, c'est à dire qu'un thread doit être informé que les valeurs de ses voisins sont à jour et ne correspondent pas aux anciennes valeurs. En effet un thread (une case) serait capable d'être calculé jusqu'à la dernière étape alors que ses cases voisines n'y serait pas encore.

Il est donc nécessaire de « faire attendre » le thread que tous les autres aient calculés leur nouvelle valeur.

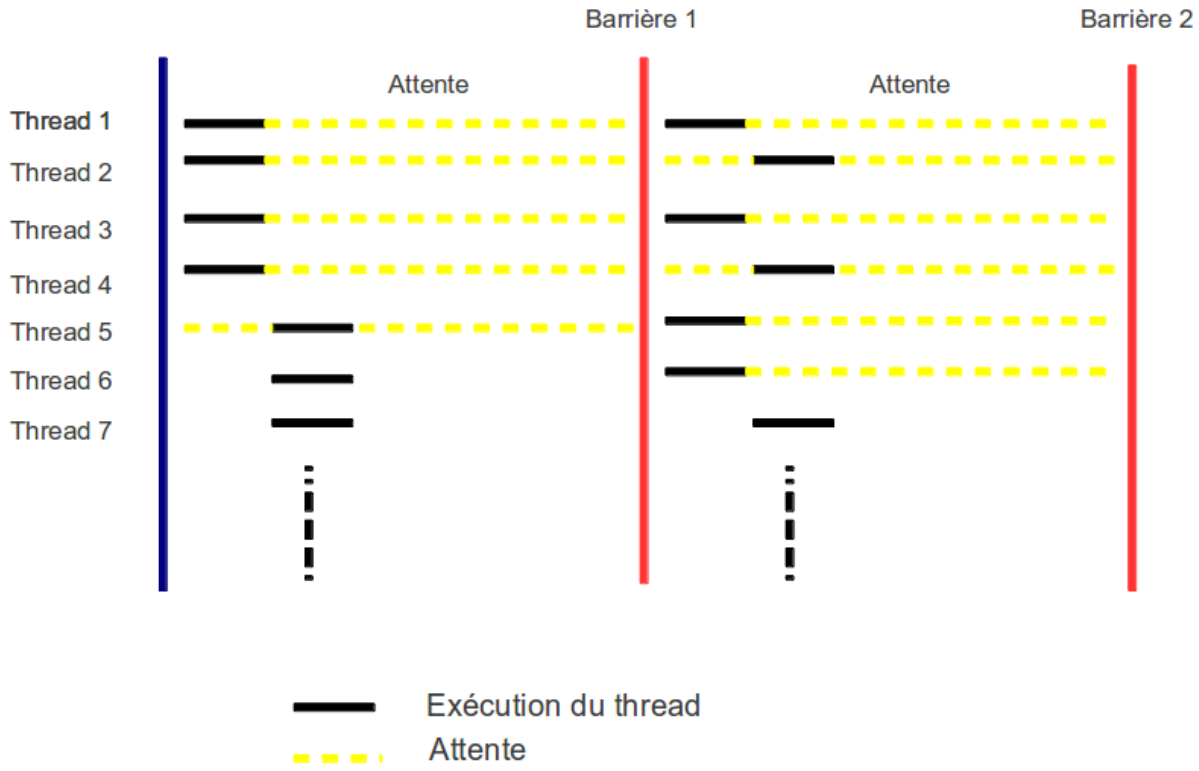
Utilisation des barrières

Afin d'informer les threads que tous les calculs ont été effectué, c'est à dire que toutes les cases de la grille ont été calculées, il est nécessaire d'utiliser deux compteurs. Ces deux compteurs sont initialisés à la valeur de la taille de la grille.

Le premier compteur indique que le thread peut commencer le traitement, le second indique qu'il termine le traitement. Ces compteurs forment des barrières que l'on ne peut franchir que lorsque tous les threads ont terminés leur traitement.



Représentation dans une architecture multi-processeur (4 processeurs simulés)



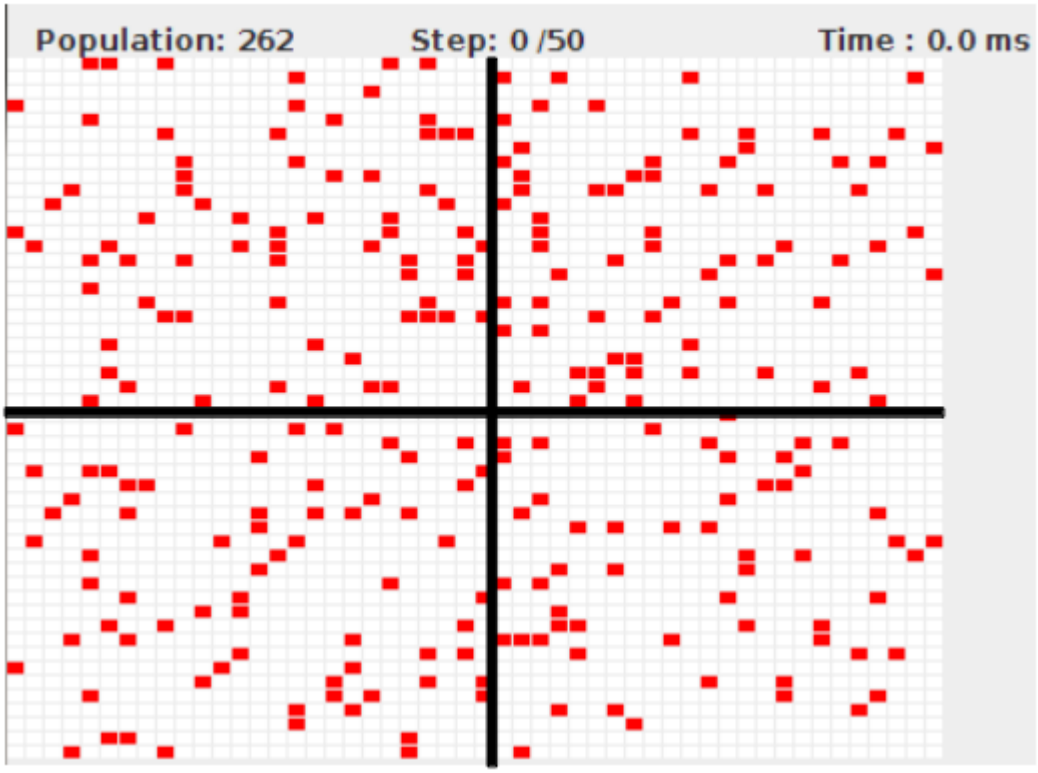
Division de la zone de jeux en 4

Algorithme

La zone de jeux est divisée en 4.

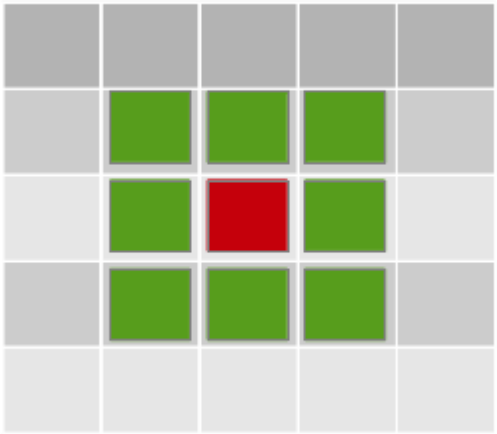
On créer 4 threads qui seront chargé de chaque partie de la zone. Tout ceci est gérer par 4 barrières, une pour chaque zone.

Cette méthode est le mélange entre un calcul séquentiel et un calcul par thread. En fait un thread fait un calcul séquentiel sur une partie de la zone de jeux.



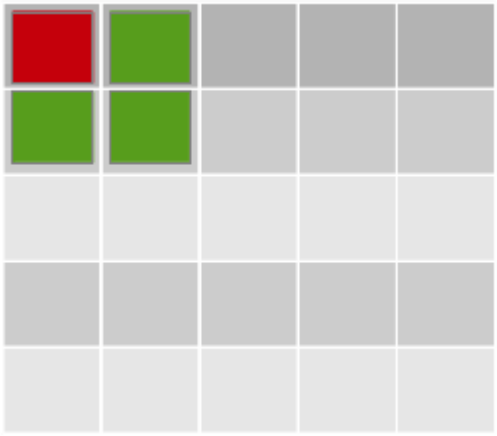
Algorithme des voisin

Introduction



Ceci constitue une amélioration de l'algorithme précédent, en effet, pour calculer la valeur de la case rouge , il suffit d'être sûr que les cases vertes (ses voisins), soient calculés. Chaque case est dirigé par un thread.

Ainsi à l'étape N, pour calculer l'étape N+1 de la case rouge, il faut que les valeurs des cases vertes soient connus et qu'il soit possible de dire que la valeur est valable.



Explication par l'exemple (grille 5x5)

Voici la première itération, on cherche à calculer la valeur de la case rouge.

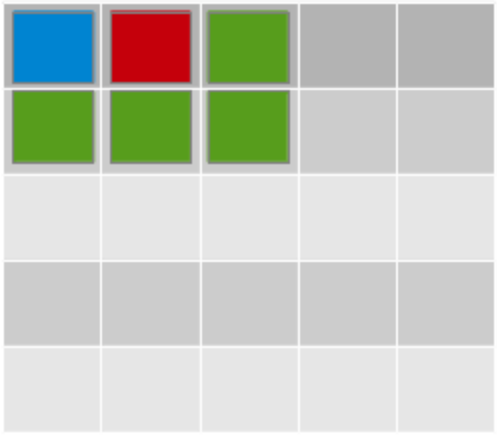
Mais pour calculer sa valeur, il faut que les valeurs des

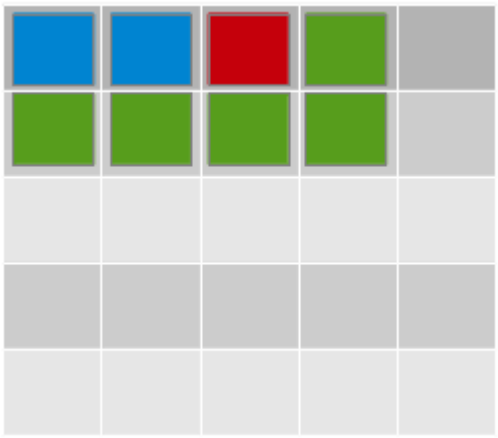
cases vertes soient connus. Pour calculer ces valeurs il faut que les valeurs de ces cases soient aussi connues.

Légende : rouge : case en cour de calcul

vert : case l'étape N à connaitre

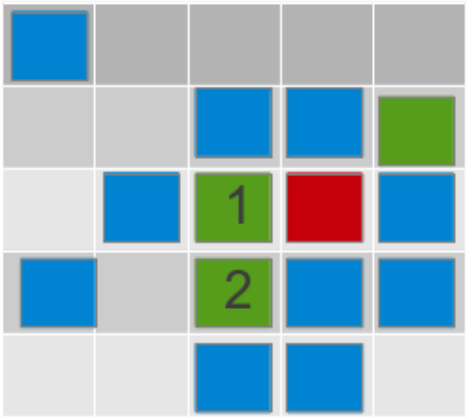
bleue : case à l'étape N+1





Dans cette exemple, nous supposons que les threads s'exécutent séquentiellement.

Maintenant supposons que nous soyons à cette étape :



On remarque qu'il manque trois cases à calculé pour terminé le calcul de la case rouge.

Les deux cases vertes 1 et 2 doivent être connus.