La fourmi de Langton

1. Introduction

Après le « jeu de la vie », la fourmi de Langton est un autre type d'automate cellulaire ayant suscité un certain intérêt sur le forum PANORAMIC, suite à la publication d'un programme de Papydall. Nous en donnons ici une version en « Crocodile BASIC ».

2. Présentation de la fourmi de Langton

Des descriptions plus ou moins identiques de l'automate se retrouvent sur de nombreux sites Internet, dont celui de Wikipedia. La version qui suit est celle donnée dans les commentaires du programme de Papydall :

Sur une grille dont les cases peuvent être blanches ou bleues, on met une petite flèche invisible dans la case centrale : ce sera notre fourmi.

L'orientation de la flèche indiquera sa direction.

A chaque tour, la fourmi se déplace selon les règles suivantes :

- *Si la fourmi est sur une case blanche, elle effectue une rotation vers la droite.*
- *Si elle est sur une case bleue, elle effectue une rotation vers la gauche.*
- *La fourmi inverse la couleur de la case sur laquelle elle se trouve (blanc devient bleu et réciproquement).*
- *La fourmi avance d'une case dans la direction de son orientation.*

Les phases de la vie de la fourmi

La fourmi obéit à des règles très simples et pourtant, quand on la simule pendant quelques milliers de tours, il se passe des choses vraiment étonnantes. La fourmi va passer par 3 phases très différentes :

1. *La phase « symétrique »*
2. *La phase « chaotique »*
3. *La phase « autoroute »*

Au début de son évolution, la fourmi se balade dans une zone assez limitée de la grille, en dessinant des configurations régulières et symétriques. Mais à partir de 500 tours, tout change. Elle se met à avoir un comportement chaotique, en élargissant son terrain de jeu et en créant des configurations très irrégulières. Cette phase chaotique dure jusqu'à environ 10000 tours, et là le miracle se produit : la fourmi entame la construction d'une autoroute très régulière qui la conduit à l'infini.

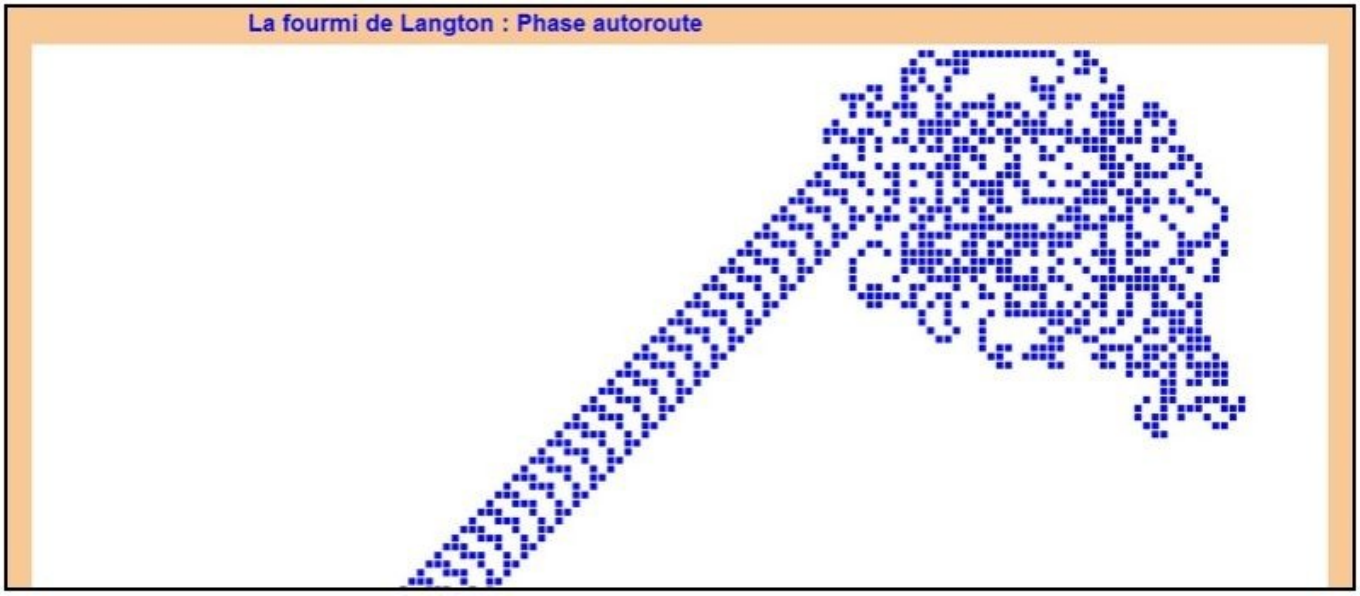
L'autoroute est en fait un motif périodique de 104 pas qui se répète. Personne ne comprend pourquoi elle apparaît ni comment elle peut émerger du désordre qui caractérise la phase chaotique.

Ce qu'il y a de plus perturbant, c'est que même si on part d'une grille dont les cases sont coloriées aléatoirement en blanc ou en bleu, l'autoroute finit toujours par apparaître un jour ou l'autre.

A quoi sert la fourmi ?

La fourmi de Langton est un bel exemple de ce concept un peu flou que tout un tas de monde appelle l'émergence. Il s'agit en gros du fait qu'un système au comportement élémentaire simple peut donner lieu à un comportement global complexe. On retrouve cette idée en informatique, en physique, en biologie ou en sociologie.

Voici un exemple de résultat obtenu avec le programme de Papydall :



3. Calcul du déplacement

A chaque itération la fourmi se trouve sur une case blanche ou bleue et possède une orientation *r* que nous coderons par une valeur numérique : 0 = Nord, 1 = Est, 2 = Sud, 3 = Ouest. Les règles données précédemment nous permettent de calculer la nouvelle orientation de la fourmi, selon le tableau suivant :

Case actuelle	Orientation (r)	Nouvelle orientation (r')	Formule
Blanche	↑ (0) → (1) ↓ (2) ← (3)	→ (1) ↓ (2) ← (3) ↑ (0)	$r' = (r + 1) \bmod 4$
Bleue	↑ (0) → (1) ↓ (2) ← (3)	← (3) ↑ (0) → (1) ↓ (2)	$r' = (r + 3) \bmod 4$

Si nous mémorisons les cases dans un tableau d'entiers **a** tel que **a(i,j) = 0** pour une case blanche et **a(i,j) = 1** pour une case bleue, les 2 formules précédentes se combinent en une seule :

```
r = (r + 2 * a(i,j) + 1) mod 4
```

Connaissant l'orientation on déduit facilement la nouvelle position de la fourmi ; ce calcul se fait simplement dans un bloc **select** … **end_select**.

(i-1, j)

↑

(i, j-1) ← **(i, j)** → **(i, j+1)**

↓

(i+1, j)

Le changement de couleur de la case peut se faire avec l'opérateur logique **xor** (OU exclusif) :

```
a(i, j) = a(i, j) xor 1
```

Nous aurions aussi pu utiliser :

```
a(i, j) = - not a(i, j)
```

En effet **not 0** donne -1 en « Crocodile BASIC » (chaque bit 0 est transformé en 1, mais le premier bit est le bit de signe, égal à 1 pour un nombre négatif).

4. Programmation en « Crocodile BASIC »

4.1. Le programme : `fourmi.bas`

Ce programme est inclus dans le logiciel *« Crocodile BASIC »* (sous-répertoire **`exemples\autocell`**).

Le programme démarre en mode « Création » ; nous pouvons alors définir la configuration de départ à l'aide des commandes, calquées sur le programme du « jeu de la vie ». En pressant la barre d'espace nous passons en mode « Génération ».

La figure suivante représente le fonctionnement du programme en mode « Génération ». La fenêtre affiche :

- la grille avec la population
- les coordonnées (Row, Col) des cases affichées
- l'état de la sauvegarde automatique des images (ON / OFF)
- la valeur du zoom
- le pas de défilement de la grille, exprimé en nombre de cases (sous forme 2^n, de 1 à 1024)
- le numéro de l'itération affichée

Le programme montre le parcours de la fourmi (case rouge) ; si la sauvegarde automatique est active (AutoSave ON), chaque image est enregistrée dans un fichier PNG ayant un nom de la forme **`00000xxx.png`** où **`xxx`** est le numéro de l'image (si des fichiers de mêmes noms sont présents ils seront écrasés). La touche « Echap » permet de quitter le programme.

4.2. Techniques de programmation

4.2.1. Constantes et variables globales

Le programme démarre en définissant un certain nombre de constantes et de variables globales ; la plupart ont été décrites dans le précédent article sur le « jeu de la vie ». Notons en particulier la taille `n` de la grille (en nombre de cases) et le facteur `fg` qui représente la taille de la partie affichée, en multiple de 240 pixels. Pour la figure précédente nous avons fixé `fg = 2` pour limiter la taille de l'image.

La grille est stockée dans un tableau d'entiers de 1 octet chacun, d'où la déclaration : `dim a%*1(n, n)`

4.2.2. Sous-programmes

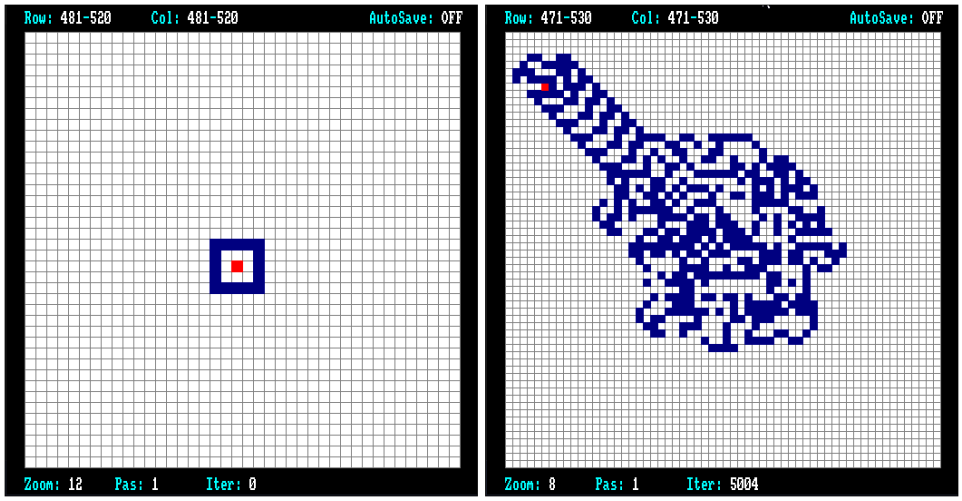
L'initialisation des variables graphiques est effectuée par le sous-programme `init` selon les techniques déjà utilisées pour le « jeu de la vie ».

- Le sous-programme `deplace` applique l'algorithme décrit au paragraphe 3.
- Le sous-programme `printparam` affiche les paramètres du tracé, de part et d'autre de la grille.
- Le sous-programme `plotcell` dessine les cases bleues ou rouge (fourmi)
- Le sous-programme `display` affiche l'ensemble du graphique.
- Les autres sous-programmes sont adaptés de ceux utilisés pour le « jeu de la vie ».

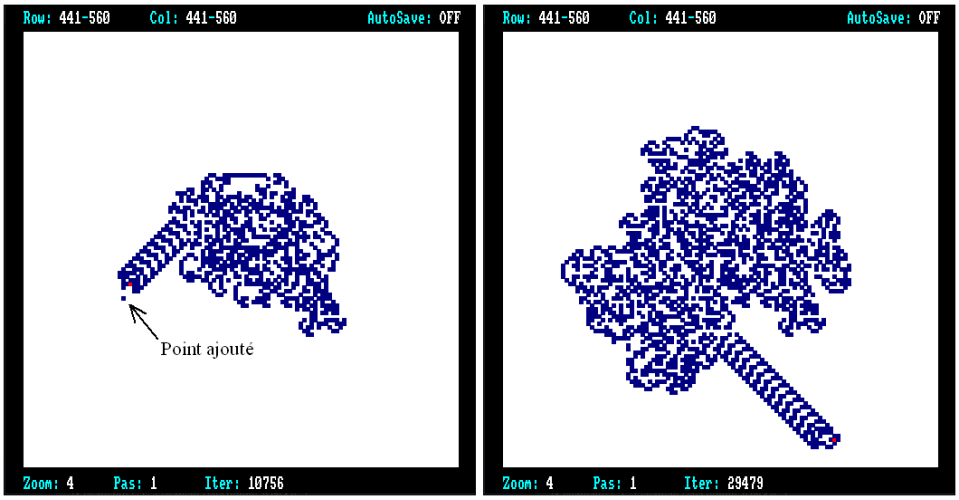
5. Comment aider (ou perturber) la fourmi ?

La fourmi met beaucoup de temps avant d'entamer la construction d'une route. Nous pouvons l'aider en plaçant quelques cases bleues au voisinage de la fourmi : il y a alors des chances pour que la construction de la route commence beaucoup plus tôt.

Sur l'exemple suivant, la présence d'un cadre autour de la fourmi a permis de réduire le temps de démarrage de la route à environ 4500 itérations.

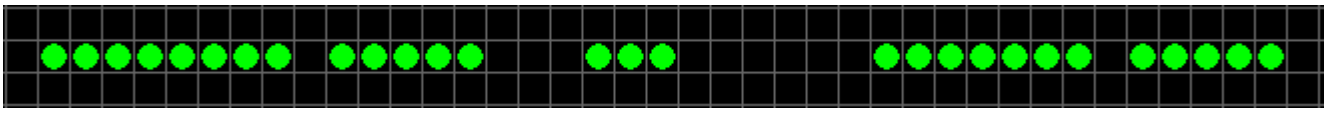


A l'inverse, si on veut perturber la fourmi il suffit de placer un obstacle sur sa route ; sur l'exemple suivant l'ajout d'un seul point a suffi à ramener la fourmi dans sa phase chaotique et à lui faire effacer la route qu'elle avait tracée ! Elle parvient toutefois à en tracer une autre … au bout de presque 30000 itérations !



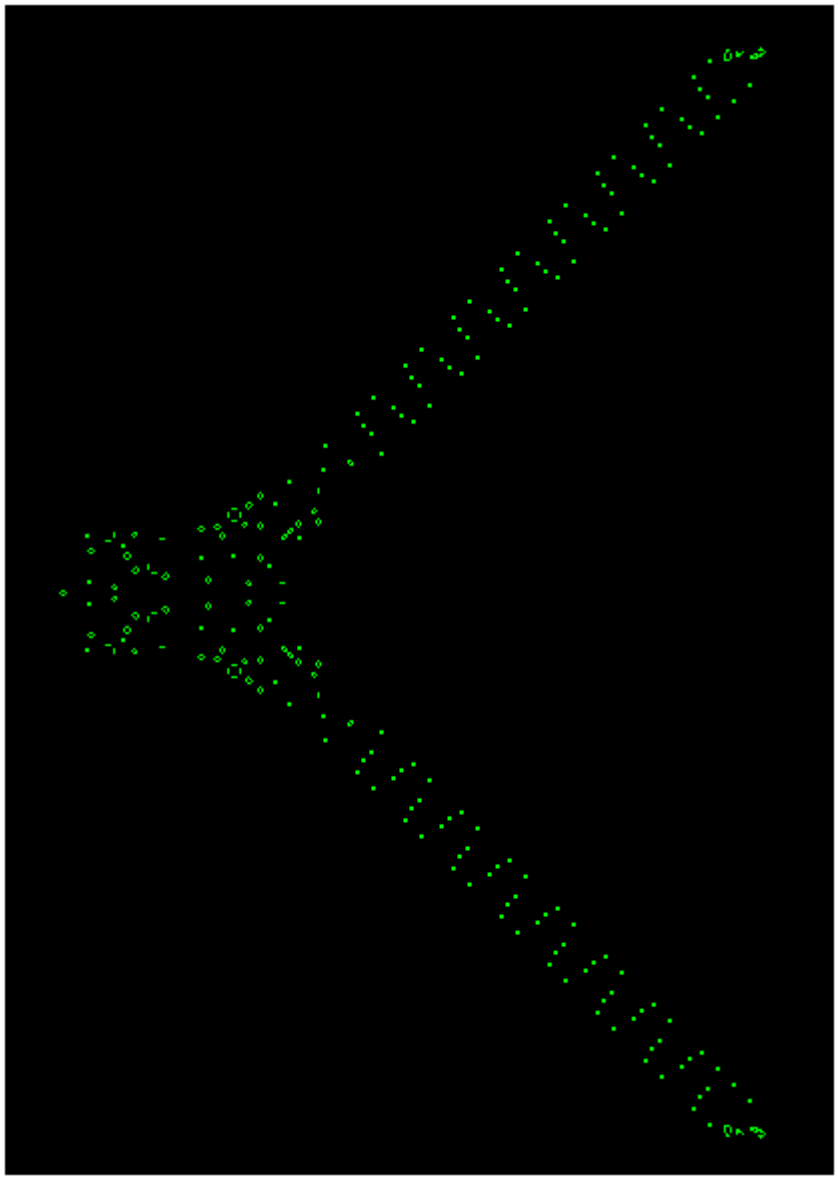
6. D'autres constructeurs de routes

Le jeu de la vie peut aussi générer des motifs évoquant des routes. Pour le montrer nous partirons du motif suivant, d'apparence assez anodine :



En suivant l'évolution de ce motif avec le programme **`vie.bas`** décrit dans le précédent article, nous

constatons qu'après environ 3000 générations, le motif a évolué pour former deux « routes » situées à ± 45° de l'horizontale :



Un agrandissement montre que chaque « route » est formée de « blocs » (ensembles de 4 cellules disposées en carré et constituant des motifs stables) répartis de manière sinusoïdale : des routes pleines de virages, plutôt difficiles à parcourir !