IFS (3) génération aléatoire des IFS

par Patrick Chalmeau et Vu Tha (TC), APTIC “Exploration Mathématique” du lycée Louise Michel de Bobigny (93)

enseignant : François Gaudel

rappel :

IFS signifie *Iterated Function Systems*, c’est-à-dire : systèmes de fonctions itérées. C’est une méthode pratique de réalisation d’images fractales, qui permet également la compression d’images quelconques. Dans la première partie (**Présentation générale**, page 83), nous en expliquions le principe et l’intérêt.

Le second exposé (**Distance de Hausdorff et application dans les IFS**, page 87) permettait de comprendre les propriétés et surtout l’existence des IFS grâce à l’introduction d’une distance appelée distance de Hausdorff, et du théorème du point fixe ; le théorème du collage nous donne alors une méthode pour fabriquer des images à partir de modèles.

Enfin, dans ce troisième exposé nous expliquerons comment nous avons utilisé la notion de dimension fractale pour améliorer la qualité de nos œuvres.

L’écran ayant une définition limitée, tous les points se trouvent sur l’image de la figure limite F_∞ au bout de très peu d’étapes. En admettant que la distance de Hausdorff qui sépare une figure de F_∞ soit divisée par 2 à chaque étape, au bout de 10 étapes, la distance initiale a été divisée par 1024, et par un million au bout de 20 étapes.

Si on part d’un point, qui est lui-même une figure F, la figure itérée $F_n=T^n(F)$ est constituée de points qui sont très rapidement indiscernables de ceux de F_∞. Or chacun de ces points est obtenu par composition successive de n des applications affines qui constituent l’opérateur de Hutchinson.

Supposons que la figure F_n soit en pratique indiscernable de F_∞. Appelons p le nombre d’applications affines qui constituent l’opérateur de Hutchinson T. Numérotons les applications affines qui constituent T :

$$f_1, f_2, \ldots, f_p.$$

Chaque point de F_n est repéré par les n numéros successifs des applications affines qui permettent de l’obtenir : c’est une suite de n nombres entiers compris entre 1 et p,

$$(d_1, d_2, \ldots, d_i, \ldots, d_n)$$

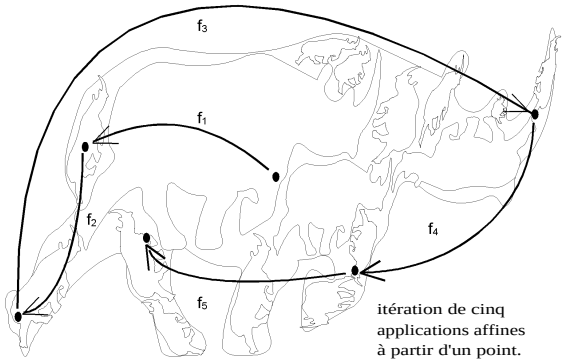
Par cette suite de n applications affines, tous les points de l’écran auront des images indiscernables, c’est-à-dire un même pixel de cet écran.

Partons d’un point quelconque de l’écran, et appliquons lui indéfiniment l’une des applications f_i, où l’indice i est tiré au sort à chaque étape parmi les valeurs 1, 2, … p.

A partir de la n-ième étape, nous affichons les points successifs obtenus. Ils sont tous indiscernables de ceux de F_∞.

Qui plus est, nous sommes assurés de tirer au sort à un moment ou un autre n’importe quelle suite $(d_1, d_2, \ldots, d_i, \ldots, d_n)$, et donc de voir s’afficher le point correspondant de F_n.

Nous avons donc là une autre méthode, très rapide, pour tracer des IFS : si $p = 10$ et $n = 20$, la méthode initiale nous oblige à calculer 10^{20} images du point de départ, alors que par cette nouvelle méthode, il n'y a qu'un point calculé à chaque itération, et, pour en obtenir un million, ce qui est amplement suffisant pour nos écrans 1 million et 20 calculs suffisent (cent mille milliards de fois moins).



il y a cependant un petit problème …

Si on tire les applications avec la même probabilité pour chacune, on se retrouve, après disons un million d'étapes, avec un nombre de points pratiquement identique pour chaque application. Résultat : la figure n'est pas homogène, les applications très contractantes, débouchant sur de petites figures, donneront des parties d'écran très denses, alors que les parties plus grandes du collage ne seront pas totalement remplies.

Pour nous, il s'agit d'obtenir des images homogènes et donc **la probabilité d'aller dans une zone doit être proportionnelle à sa taille**. Pour obtenir des dégradés ou des parties d'écran plus ou moins foncées, nous avons là aussi la base d'une méthode consistant seulement à moduler les p probabilités en fonction du résultat voulu.

Notre problème est maintenant de calculer les p probabilités pour avoir une figure homogène, ou, ce qui revient au même, la taille de chacune des composantes du collage dans F_∞.

Mais F_∞ est en général une **figure fractale.** Sa mesure, et celle de ses composantes, pour être calculées, exigent que l'on se place dans une **dimension non entière**.

Il est facile de calculer la mesure d'un objet auquel on a fait subir une contraction (ou une dilatation) de rapport k lorsque l'on travaille en dimension 1, 2 ou 3 : la mesure initiale est multipliée par k, k^2 , ou k^3. Mais les choses se compliquent un peu lorsqu'on a affaire à des fractales : le calcul est toujours le même dans son principe, la mesure initiale est multipliée par k^d. Le problème est de calculer la dimension d qui change avec la figure.

Exemple : dimension du triangle de Sierpinski : lorsqu'on lui fait subir une homothétie de rapport 2, sa mesure est multipliée par 3 (voir la figure, page 84).

On doit donc avoir $2^d = 3$, d'où, en prenant le logarithme népérien de chaque membre :

$$d \times \ln(2) = \ln(3),$$

et finalement une dimension d égale à :

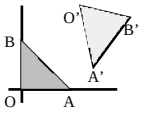
$$d = \frac{\ln 3}{\ln 2}$$

Calcul des probabilités des différentes applications affines

Une application affine est définie par un triangle de base non aplati, et un triangle image. Le rapport des aires de ces deux triangles nous donne bien entendu une information sur la façon dont l'aire d'une figure quelconque est transformée par cette application affine. Partant de cette idée, nous sommes partis d'un repère orthonormé $(O, \vec{i}, \vec{j})$, correspondant au triangle rectangle isocèle OAB, et de l'expression analytique d'une application affine, qui est :

$$\begin{cases} x' = a\,x + b\,y + e \\ y' = c\,x + d\,y + f \end{cases}$$

où x et y sont les coordonnées d'un point M et x' et y' sont les coordonnées de son image M'.



Le triangle *OAB* a pour image le triangle *O'A'B'* pour lequel :

$$\overrightarrow{O'A'} : (a, c) \; ; \; \overrightarrow{O'B'} : (b, d)$$

L'aire du triangle *OAB* était 1/2 ; celle du triangle *O'A'B'* est égale à la moitié de la valeur absolue du déterminant $a\,d - b\,c$. Nous définissons alors de façon naturelle (par abus de langage) le coefficient de contraction k de notre application affine qui est :

$$k = \sqrt{\left| \begin{matrix} a & b \\ c & d \end{matrix} \right|} = \sqrt{|a\,d - b\,c|}$$

L'image finale doit avoir la même mesure que la réunion de ses images par les applications affines composant l'opérateur de Hutchinson, puisqu'elle est invariante par cet opérateur. Appelons k_1 le coefficient de contraction de l'application f_1 et ainsi de suite jusqu'à k_p, coefficient de contraction de f_p. Si la mesure de F_∞ est posée égale à 1, et si sa dimension fractale est d, nous avons supposé que la mesure de $f_1(F_\infty)$ est $k_1{}^d$, et ainsi de suite jusqu'à f_p.

On obtient alors, à condition que les diverses composantes ne se recouvrent pas, la relation:

$$\boxed{k_1^d + k_2^d + \ldots + k_n^d = 1}$$

Il suffit de faire varier d de manière à ce que cette relation soit vérifiée pour trouver la dimension fractale de la figure et du même coup les probabilités qui sont égales à $k_i{}^d$.

Le programme que nous avons réalisé en Turbo-Pascal fait plusieurs choses :

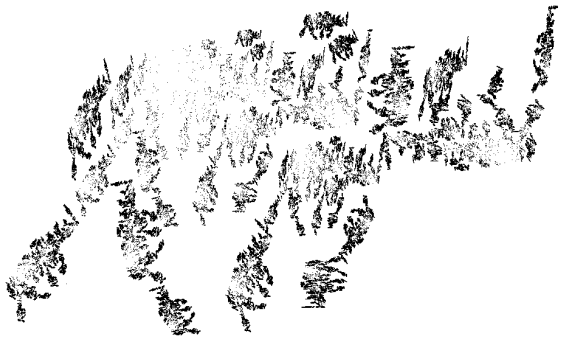
— Tout d'abord, il permet de calculer les expressions des applications affines à partir des coordonnées des images du triangle de base : pour calculer a, b, c, d, e, f, six inconnues, nous disposons de six équations :

$$\begin{cases} x_{A'} = a\,x_A + b\,y_A + e \\ y_{A'} = c\,x_A + d\,y_A + f \\ x_{B'} = a\,x_B + b\,y_B + e \\ y_{B'} = c\,x_B + d\,y_B + f \\ x_{C'} = a\,x_C + b\,y_C + e \\ y_{C'} = c\,x_C + d\,y_C + f \end{cases}$$

que nous résolvons par la méthode du pivot de Gauss. Nous pouvons ainsi travailler avec plus que les 32 triangles qu'autorise FDESIGN, et avec plus de précision.

— Ensuite, le programme calcule les probabilités de chacune des applications affines. Le résultat peut alors être utilisé directement avec un programme très puissant appelé FRACTINT, qui permet par exemple de régler les couleurs, d'avoir une meilleure définition, etc …

Nos probabilités se sont révélées excellentes : les images sont bien homogènes. (Comparer les différentes versions du rhinocéros).



[image obtenue en prenant toutes les probabilités égales]

— Enfin, un module, achevé après le Congrès, permet de préparer des animations à partir de deux fichiers triangles : il suffit de calculer les fichiers triangles intermédiaires pour calculer une série d'images utilisables ensuite pour transformer, par exemple, une poule en lapin … [NDLC : ou une vessie en lanterne ?]