Visualisation rapide de formes fractales

Gilles Gouaty, Eric Tosan, Eric Guérin

Visualisation rapide de formes fractales

Gilles Gouaty, Éric Tosan, Éric Guérin

LIRIS – Université Claude Bernard Lyon 1
ggouaty,et,eguerin@bat710.univ-lyon1.fr

Résumé : *Notre objectif est d'obtenir le rendu 3D en temps interactif d'images fractales. Les figures que nous cherchons à visualiser reposent sur un modèle introduit et développé au LIRIS : le modèle d'IFS projeté [11]. Nous rappelons dans un premier temps les principaux concepts sur lesquels s'appuient le modèle d'IFS [6, 7] puis nous donnons la définition du modèle que nous utilisons. Le système de visualisation que nous proposons est inspiré de l'algorithme de* BARNSLEY *permettant la visualisation des IFS [2, 1]. Nous proposons un nouvel algorithme qui rend compte de la multi-résolution sous-jacente aux fractales. Nous présentons ensuite une optimisation qui permet de limiter les effets néfastes du zoom sur la complexité de l'algorithme. Pour conclure, les résultats de performance des différents algorithmes sont présentés sous forme d'un tableau comparatif.*

Mots-clés : Fractales, IFS, visualisation rapide, OpenGL, multi-résolution.

1 Modèle fractal utilisé

La théorie des IFS (Iterated Function System) est un outil puissant pour l'analyse et la synthèse d'images fractales. Outre son aspect mathématique rigoureux, elle offre des méthodes simples pour générer des formes fractales. BARNSLEY a exploité une des caractéristiques principales des figures fractales qui est leur structure autosimilaire. Son idée a été de coder une image fractale à l'aide d'un ensemble d'opérateurs contractants qui traduit cette propriété. Cet ensemble d'opérateurs ainsi constitué est appelé IFS et la figure codée par cet IFS peut être générée en itérant ces opérateurs.

1.1 Modèle IFS

L'espace dans lequel on se place est un espace métrique complet (E, d). On note $\mathcal{H}(E)$ l'ensemble des compacts non vides de (E, d), il est muni d'une métrique induite par d. Soient A et B deux compacts d'un espace métrique (E, d), la distance de HAUSDORFF de A à B est définie par :

$$d_H(A, B) = \max\{\max_{p\in A}\min_{q\in B} d(p, q), \min_{p\in A}\max_{q\in B} d(p, q)\}$$

Un *IFS* est un ensemble fini $\mathcal{I} = \{T_0, \ldots, T_{N-1}\}$ d'opérateurs contractants sur E.

L'opérateur de HUTCHINSON associé est défini par :

$$K \in \mathcal{H}(E) \mapsto \mathcal{I}K = \bigcup_{i=0}^{N-1} T_i K \in \mathcal{H}(E)$$

Si les N opérateurs $T_i : E \to E$, $i = 0, \ldots, N-1$ sont de constantes de contraction s_i, cet opérateur est contractant pour la distance d_H et de constante de contraction $s = \max\{s_0, \ldots, s_{N-1}\}$.

À tout IFS on peut associer un opérateur de HUTCHINSON. On peut donc appliquer le théorème du point fixe.

Théorème 1.1 *Pour chaque IFS $\mathcal{I}$, il existe un compact unique non vide $\mathcal{A}$ de $\mathcal{H}(E)$ tel que : $\mathcal{A} = \mathcal{I}\mathcal{A}$. $\mathcal{A}$ est appelé attracteur de $\mathcal{I}$ et sera noté $\mathcal{A}(\mathcal{I})$.*

Il est donc possible d'associer à tout IFS une figure, qui est point fixe de l'opérateur de HUTCHINSON associé. L'attracteur $\mathcal{A}$ possède la propriété d'autosimilarité au sens large : $\mathcal{A} = T_0\mathcal{A} \cup ... \cup T_{N-1}\mathcal{A}$.

Étant donné un alphabet $\Sigma = \{0, \ldots, N-1\}$, un IFS indicé est une famille finie $\mathcal{T} = (T_i)_{i\in\Sigma}$. On note Σ^n l'ensemble des mots de longueur n, Σ^* l'ensemble des mots finis , Σ^ω l'ensemble des mots infinis, et Σ^∞ l'ensemble des mots finis ou infinis.

Théorème 1.2 *On appelle* fonction d'adressage *d'un IFS indicé $\mathcal{T} = (T_i)_{i\in\Sigma}$ la fonction ϕ définie par :*

$$\begin{array}{rccl} \phi : & \Sigma^\omega & \longrightarrow & E \\ & \rho = \rho_1\rho_2\ldots & \longmapsto & \phi(\rho) = \lim_{d\to\infty} T_{\rho_1}\ldots T_{\rho_d} p \end{array}$$

La fonction d'adressage nous permettra par la suite de caractériser les algorithmes de visualisation.

1.2 Modèle IFS projeté

Nous nous intéressons à la visualisation de formes fractales en 3D. On se place dans un espace de modélisation affine E^s représenté dans l'espace homogène $\mathbb{R}^{3+1} = \mathbb{R}^4$. L'objet modélisé est défini comme la projection de l'attracteur d'un IFS créé dans un espace d'itération affine E de dimension n. Les application affines sur E définissant les transformations de l'IFS sont des matrices $T_i \in \mathbb{R}^{(n+1)\times(n+1)}$. Une matrice de projection établit le passage de l'espace d'itération vers l'espace de modélisation : $P \in \mathbb{R}^{4\times(n+1)}$. Nous avons donc le modèle suivant [11] :

$$(P, \{T_0, \ldots, T_{N-1}\}), P \in \mathbb{R}^{4\times(n+1)}, T_i \in \mathbb{R}^{(n+1)\times(n+1)}$$

P est une transformation 3D qui permet d'avoir un modèle plus souple que la simple donnée de l'IFS :

1. Avec $n = 3$, on peut, en agissant seulement sur P, positionner l'attracteur dans la scène.
2. P permet également de modifier le type de l'espace d'itération. S'il est préférable d'exprimer les transformations de l'IFS dans un espace barycentrique, on peut utiliser un ensemble de matrices barycentriques pour les transformations et un ensemble de points de contrôle $(p_0, \ldots, p_n)$ comme colonnes de P.
3. On peut également utiliser P pour créer un effet de perspective.
4. Avec $n > 3$, on peut modéliser des objets qui n'auraient pas pu l'être avec le modèle IFS simple. En particulier, les IFS projetés peuvent être vus comme une généralisation des formes à pôles telles que les courbes ou surfaces de BÉZIER, ou bien les B-splines [11].

On peut combiner ces effets (projection, passage de coordonnées barycentriques en coordonnées homogènes, changement de repères, perspective, ...) en calculant le produit de plusieurs matrices.

2 Algorithme classique de visualisation

Notre système s'appuie sur la librairie graphique *OpenGL* qui, par l'intermédiaire de primitives, permet d'obtenir le rendu rapide d'images 3D en utilisant au mieux les capacités de la carte graphique.

OpenGL gère la conversion des coordonnées de scène en coordonnées de clipping, la division des perspectives et le Z-buffer. Il suffit de spécifier à OpenGL les matrices de visualisation et de perspective. Ces calculs sont effectués sur les cartes graphiques compatibles OpenGL. En bénéficiant de cette accélération matérielle, cette économie de calcul se traduit par un allègement de la charge sur le microprocesseur.

Nous avons choisi d'implémenter un système de visualisation spécifique qui gère lui-même les matrices de visualisation et de perspective. Nous fournissons à OpenGL des points directement exprimés en coordonnées de clipping, et nous laissons les tests de profondeur à la charge d'OpenGL. Dans la partie suivante, nous expliquons les raisons de ce choix, qui est lié aux caractéristiques particulières des IFS.

2.1 Approximation d'un attracteur

Notre méthode de visualisation est inspirée de l'*algorithme déterministe* de BARNSLEY [2]. Celui-ci consiste à calculer un ensemble de motifs dans l'espace de la scène jusqu'à un profondeur d fixée. En posant :

$$K_d = \mathcal{I}^d K = \bigcup_{\rho\in\Sigma^d} T_{\rho_1}\ldots T_{\rho_d} K$$

on a : $\lim_{d\to\infty} \mathcal{I}^d K = \mathcal{A}(\mathcal{I})$.

L'attracteur d'un IFS peut être approximé en calculant K_d pour d assez grand. Le choix de K est arbitraire. Dans la suite, K sera un polytope à k sommets S_i représenté par une matrice S à k colonnes.

On note E l'espace d'itération, E^s l'espace de scène ou espace de modélisation, E^c l'espace de clipping et K, K^s et K^c les ensembles de points correspondants dans E, E^s et E^c. L'attracteur projeté $P\mathcal{A}(\mathcal{I})$ est approximé par :

$$K_d^s = PK_d$$

En posant $P^c = WVP$, on prend en compte la caméra. L'attracteur projeté dans l'espace de clipping est approximé par :

$$K_d^c = WVK_d = P^cK_d$$

$V \in \mathbb{R}^{4\times4}$ étant la *matrice de visualisation* et $W \in \mathbb{R}^{4\times4}$ la *matrice de perspective*.

2.2 Complexité

L'évolution de l'algorithme de BARNSLEY s'exprime à l'aide d'un arbre (voir figure 1).

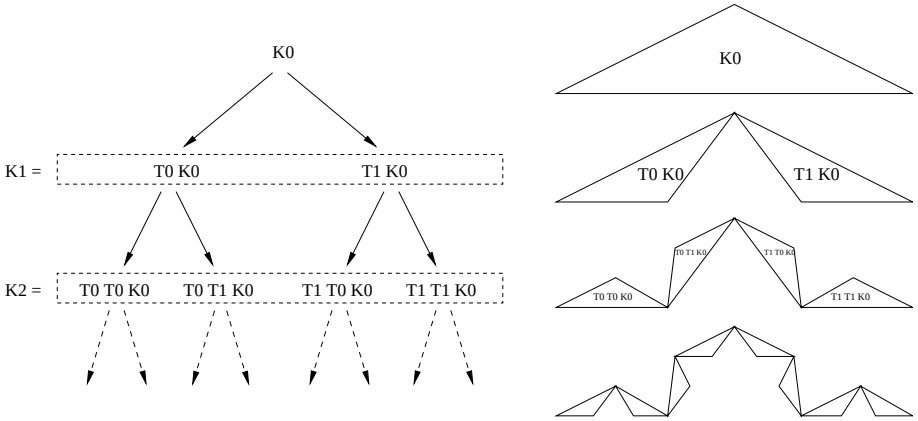


FIG. 1 – Arbre de construction d'un IFS selon l'algorithme de BARNSLEY, et approximants associés

Cet algorithme construit un arbre de profondeur d dont chaque nœud possède N fils. Le nombre de feuilles de cet arbre est N^d. Cet algorithme effectue en chaque branche un produit matriciel $\mathbb{R}^{4\times(n+1)} \times \mathbb{R}^{(n+1)\times(n+1)}$ pour calculer $M_{\rho i} = M_\rho T_i$. Arrivé dans une feuille de l'arbre, lorsque d atteint 0 , on effectue un produit matriciel $\mathbb{R}^{4\times(n+1)} \times \mathbb{R}^{(n+1)\times k}$ pour calculer $S_\rho^s = M_\rho S$. Le nombre de branches vaut $\frac{N^{d+1}-1}{N-1} - 1$. La complexité de cet algorithme est :

$$\left(\frac{N^{d+1}-1}{N-1} - 1\right)\widetilde{\nu} + N^d\nu$$

en notant $\widetilde{\nu}$ le temps de calcul de $M_\rho T_i$ et ν le temps de calcul de $M_\rho S$.

La figure 2 montre quelques images obtenues avec cet algorithme sur un même modèle d'IFS et avec des positions de caméra de plus en plus proches de la fractale.

La puissance de la modélisation fractale réside dans le fait que l'on peut modéliser des objets présentant des détails à un niveau de précision aussi grand que l'on veut. Le principal défaut de cet algorithme est de ne pas exploiter cette caractéristique des IFS. En effet, vu de loin, l'objet présente une structure complexe paraissant très détaillée. Lorsque l'on se rapproche, on se rend compte que l'objet n'est en fait qu'un ensemble de polygones grossiers, ne présentant plus aucun détail. Pour remédier à ce problème, nous devons adapter la profondeur d'itération en fonction de la position de la caméra.

3 Adaptation de la profondeur d'itération

Les généralisations de l'algorithme de BARNSLEY s'expriment à l'aide d'arbres de construction des itérés. Ces arbres ont des propriétés de localisation des attracteurs utilisables par les algorithmes de visualisation.

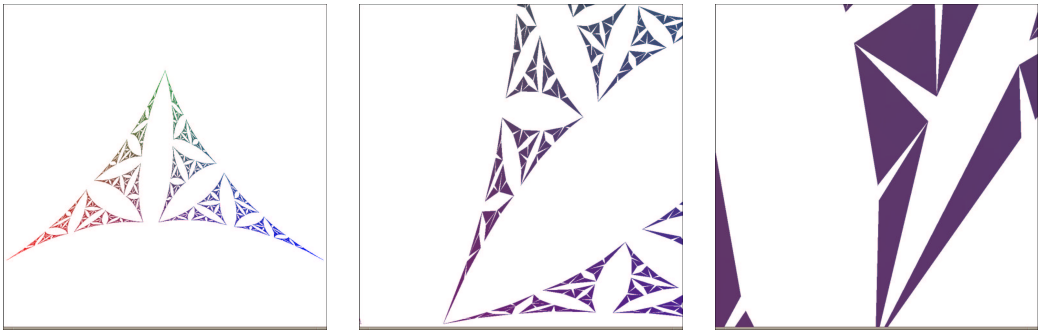


FIG. 2 – Images obtenues avec l'algorithme de BARNSLEY

3.1 Arbres de construction

L'algorithme de BARNSLEY consiste à générer un ensemble de K_ρ pour $\rho \in \Sigma^d$, soit l'ensemble des mots de longueur d. L'algorithme que nous proposons généralise l'algorithme de BARNSLEY : on calcule un ensemble de K_ρ où les ρ n'ont pas tous la même longueur. En notant L un ensemble de mots ρ correspondant à un calcul $T_{\rho_1} \dots T_{\rho_d} K$, on définit :

$$K_L = \bigcup_{\rho \in L} K_\rho$$

en posant $K_\rho = T_{\rho_1} \dots T_{\rho_d} K$ avec $K \in \mathcal{H}(E)$ et $\rho = \rho_1 \dots \rho_d \in \Sigma^*$.

L est un ensemble fini de mots finis sur Σ.

En notant $\preccurlyeq$ la relation préfixe définie par : $\forall \rho \in \Sigma^*, \forall \sigma \in \Sigma^\infty, \rho \preccurlyeq \sigma \Leftrightarrow \exists \tau \in \Sigma^\infty, \rho\tau = \sigma$, on dit que L est une coupure de Σ^ω lorsque tout élément de Σ^ω possède un préfixe unique dans L soit :

$$\forall \sigma \in \Sigma^\omega, \exists ! \rho \in L, \rho \preccurlyeq \sigma$$

On associe à toute coupure L l'ensemble $\widehat{L}$ constitué de ses préfixes. Muni de la relation préfixe, $\widehat{L}$ constitue un arbre dont la racine est le mot vide, l'ensemble des feuilles est L, et l'ensemble des nœuds N-aires est $\widetilde{L} = \widehat{L} \setminus L$, comme le montre la figure 3.

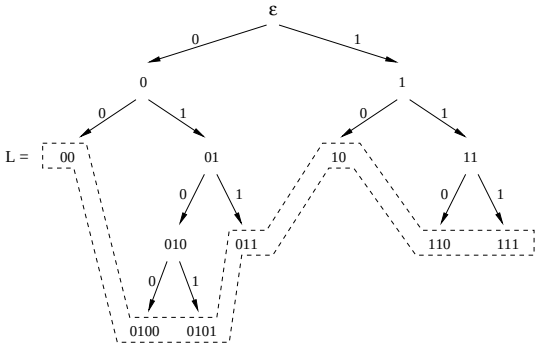


FIG. 3 – Exemple de coupure de Σ^ω sur l'aphabet $\Sigma = \{0, 1\}$, et son arbre de construction N-aire associé

En notant $|\widehat{L}|$ le nombre de nœuds de $\widehat{L}$, $|L|$ le nombre de feuilles, et $|\widetilde{L}|$ le nombre de nœuds N-aires, $|\widetilde{L}|$ peut s'exprimer en fonction de $|L|$ et N de manière indépendante de la forme de l'arbre : $|\widetilde{L}| = \frac{|L|-1}{N-1}$. Le nombre de branches est : $N|\widetilde{L}| = \frac{N}{N-1}(|L|-1)$. La complexité des algorithmes associés à ces arbres est :

$$\frac{N}{N-1}(|L|-1)\widetilde{\nu} + |L|\nu = |L|\left(\frac{N}{N-1}\widetilde{\nu} + \nu\right) - \frac{N}{N-1}\widetilde{\nu}$$

3.2 Englobants des attracteurs

À toute coupure est associé un englobant de l'attracteur.

Proposition 3.1 *Si $K \in \mathcal{H}(E)$ est un englobant de l'attracteur $\mathcal{A}(\mathcal{I})$, alors K_ρ contient tous les $\phi(\rho\sigma)$, autrement dit tous les points dont l'adressage est préfixé par ρ :*

$$\forall K \in \mathcal{H}(E), \mathcal{A}(\mathcal{I}) \subseteq K \Rightarrow \forall \rho \in \Sigma^*, \phi(\rho\Sigma^\omega) \subseteq K_\rho$$

Preuve 3.1

$$\phi(\rho\Sigma^\omega) = T_{\rho_1} \dots T_{\rho_d} \mathcal{A}(\mathcal{I}) \subseteq T_{\rho_1} \dots T_{\rho_d} K = K_\rho$$

Cette propriété permet d'obtenir des englobants d'attracteurs associés aux coupures, à savoir :

Proposition 3.2

$$\forall K \in \mathcal{H}(E),\ \mathcal{A}(\mathcal{I}) \subseteq K \Rightarrow \mathcal{A}(\mathcal{I}) \subseteq K_L$$

Preuve 3.2

$$\mathcal{A}(\mathcal{I}) = \phi(\Sigma^\omega) = \phi(L\Sigma^\omega) = \bigcup_{\rho \in L} \phi(\rho\Sigma^\omega) \subseteq \bigcup_{\rho \in L} K_\rho = K_L$$

Cette propriété des englobants s'étend aux projections d'attracteurs :

$$P^c \mathcal{A}(\mathcal{I}) \subseteq P^c K_L = K_L^c$$

Le calcul d'un englobant d'IFS est un problème difficile dans le cas général [4, 5, 3, 10]. Il est relativement simple de calculer une sphère englobante d'un IFS [8]. Notre approche nécessite un englobant qui soit un polytope. Une condition suffisante pour que $\mathcal{A}(\mathcal{I}) \subseteq K$ est que $\mathcal{I}K \subseteq K$. Autrement dit, si $\mathcal{I} = (T_i)_{i \in \Sigma}$, alors K est un englobant si $\forall i \in \Sigma, T_i K \subseteq K$. Lorsque les transformations T_i sont exprimées dans des matrices de MARKOV, il existe un englobant très simple à calculer : il s'agit du simplexe. Il est tel que $S = Id$. Nous considérons par la suite que les IFS sont définis par un ensemble de matrices de MARKOV.

3.3 Contrôle de la résolution

Lorsque K_ρ^c est suffisamment petit sur l'image, il n'est plus nécessaire de poursuivre les itérations car on sait que les itérés suivants seront tous inclus dans K_ρ^c. La taille de K_ρ^c telle qu'elle apparaît dans l'espace de clipping est donc un critère d'arrêt qui permet d'adapter la profondeur d'itération de manière pertinente en fonction de la position de la caméra.

Cet algorithme permet une visualisation fine des attracteurs. Les approximants K_L^c sont constitués d'un ensemble de polygones de petite taille. Lorsque l'on se rapproche de l'objet, ces polygones grossissent jusqu'à ce que leur taille dépasse le seuil ε. À ce moment là, chaque polygone se divise en un ensemble de polygones plus petits. Si ε est de l'ordre de quelques pixels, la transition s'effectue de manière imperceptible. La figure 4 montre quelques images obtenues avec cet algorithme lorsque la caméra se rapproche de la fractale.

3.4 Complexité

$|L|$ dépend de plusieurs paramètres. D'après la définition de l'algorithme, L est l'ensemble des itérés dont la taille apparente est inférieure à ε c'est à dire : $L = L(\varepsilon, P^c) = \{\rho / \mathrm{diam}(K_\rho^c) \leq \varepsilon\}$. Lorsque ε diminue, le nombre de calcul augmente selon une fonction dépendant de la dimension fractale de l'objet. Par définition, la dimension fractale d'un compact K est :

$$D_F(K) = \lim_{\varepsilon \to 0} \frac{\ln(N_B(\varepsilon, K))}{\ln(\frac{1}{\varepsilon})}$$

où $N_B(\varepsilon, K)$ est le nombre minimum de boules de diamètre inférieur à ε nécessaire pour recouvrir K.

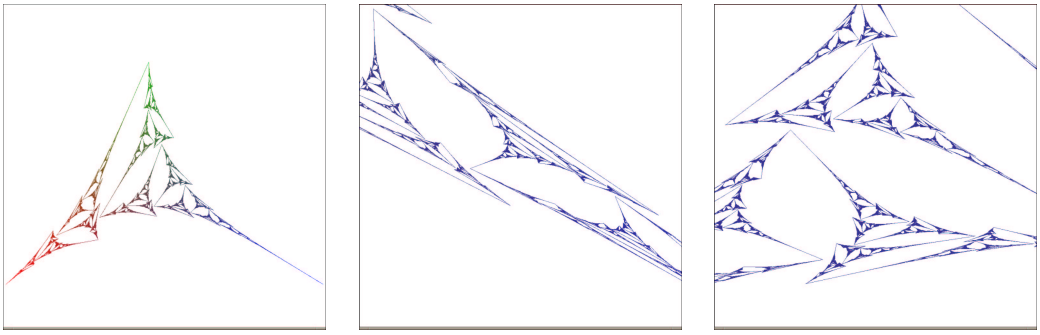


FIG. 4 – Images obtenues avec l'algorithme à contrôle de résolution

On a intuitivement $|L| \approx N_B(\varepsilon, P\mathcal{A}(\mathcal{I})) = O((\frac{1}{\varepsilon})^{D_F}) = O(\varepsilon^{-D_F})$ lorsque ε tend vers 0. Notons que $D_F \leq 3$ car l'espace de modélisation est de dimension 3.

Effectuer un zoom sur la figure fractale revient à changer le seuil ε. En introduisant $H(\alpha)$ une homothétie de rapport α on a :

$$L(\varepsilon, H(\alpha)^{-1}P^c) = L(\alpha\varepsilon, P^c)$$

car : $\text{diam}(H(\alpha)^{-1}K_\rho) \leq \varepsilon \Leftrightarrow \text{diam}(K_\rho) \leq \alpha\varepsilon$.

L'évolution de $|L|$ en fonction du facteur de zoom α est : $O(|L|) = \alpha^{-D_F}\varepsilon_0^{-D_F}$ en supposant la résolution d'image ε_0 fixée.

Nous disposons d'un algorithme permettant de visualiser un attracteur d'IFS qui tient compte de la multi-résolution. Cependant, la complexité de cet algorithme est un inconvénient majeur. Au fur et à mesure que la caméra se rapproche de la fractale, le temps de calcul augmente de manière importante. De plus, lorsque la fractale traverse le plan $\{z = 0\}$ dans l'espace de la caméra, l'algorithme boucle. En effet, quelle que soit la profondeur de subdivision, les polygones qui traversent ce plan ont une taille apparente infinie.

4 Élagage de l'arbre

Le nombre de composants calculés de K_L^c augmente en α^{-D_F}, mais peu d'entre eux sont visibles. Nous proposons d'améliorer notre algorithme en tenant compte de la propriété d'appartenance des itérés à la fenêtre de visualisation. Cette fenêtre de visualisation correspond au champ de vision de la caméra dans la scène.

4.1 Algorithme

On note C le champ de vision de la caméra dans la scène et C^c le cube de clipping $C^c = WVC = [-1, 1]^3$.

Nous savons que lorsque K est un englobant de l'attracteur, alors n'importe quel itéré dans l'arbre englobe toute sa descendance. Lorsqu'un itéré K_ρ est extérieur à la fenêtre de visualisation ($K_\rho \cap C = \emptyset$), toute sa descendance l'est également. Il en est de même pour les itérés situés à l'intérieur de la fenêtre de visualisation ($K_\rho \subseteq C$). Certains itérés peuvent chevaucher la fenêtre ($K_\rho \cap C \neq \emptyset \wedge K_\rho \cap C \neq K_\rho$). Dans ce cas on ne peut rien dire sur leur descendance.

Le test d'inclusion que nous proposons est basé sur les 12 demi-espaces qui définissent le cube de clipping $C^c = [-1, 1]^3 = \bigcap_{k=1}^{6} D_k$ avec $D_k, k \in [1, \ldots, 6]$ correspondant aux faces de C^c. On définit également les 6 demi-espaces complémentaires fermés $\overline{D}_k$, *demi-espaces extérieurs*.

Pour la coordonnée x, on a :
– $D_1 = \{(x, y, z), x \geq -1\}, \overline{D}_1 = \{(x, y, z), x \leq -1\}$
– $D_2 = \{(x, y, z), x \leq 1\}, \overline{D}_2 = \{(x, y, z), x \geq 1\}$
Les quatre autres demi-espaces correspondent aux coordonnées y et z.

Le test sur un itéré K_ρ prend l'une des 3 valeurs définies ainsi :
– inclus : $\forall k \in [1,\ldots,6], K_\rho^c \subseteq D_k$
– exclu : $\exists k \in [1,\ldots,6], K_\rho^c \subseteq \overline{D}_k$
– indéterminé : dans les autres cas.
En utilisant ce test, il est possible d'élaguer l'arbre de construction de manière importante. Lorsqu'un itéré est exclu, on peut l'éliminer. Lorsqu'il est intérieur, on poursuit l'algorithme, sans refaire de test d'inclusion, jusqu'au cas d'arrêt relatif à la taille apparente des itérés sur l'image. Lorsqu'il est indéterminé, on effectue le test sur la taille. Si cet itéré n'est pas assez petit, on poursuit l'algorithme en conservant le test d'inclusion sur les itérés suivants.

Le test sur la taille pour les itérés chevauchant évite que l'algorithme boucle en subdivisant à l'infini les itérés contenant un point sur la frontière du cube de clipping.

Le résultat de ce test ($inclus$, $exclu$, ou $indéterminé$) n'est pas équivalent à la classification d'un itéré ($intérieur$, $extérieur$, ou $chevauchant$). En effet, un itéré peut être de type $indéterminé$ alors qu'il est $extérieur$ mais aux étapes suivantes, cette indétermination est levée (voir figure 5).

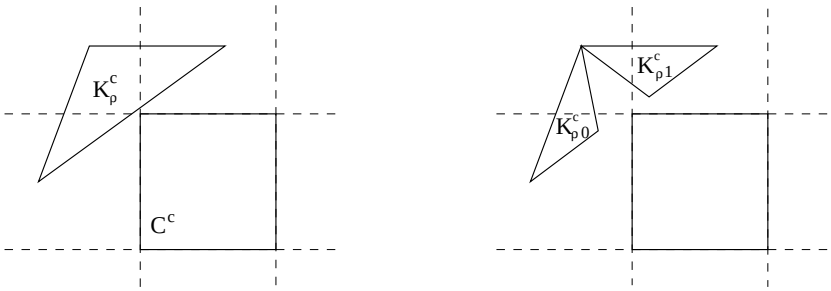


FIG. 5 – Exemple de calcul supplémentaire dû au test d'inclusion

La figure 6 montre quelques images obtenues avec cet algorithme. Les images du haut représentent ce que l'on voit. Les images du bas représentent la scène avec la fenêtre de visualisation et les itérés exclus.

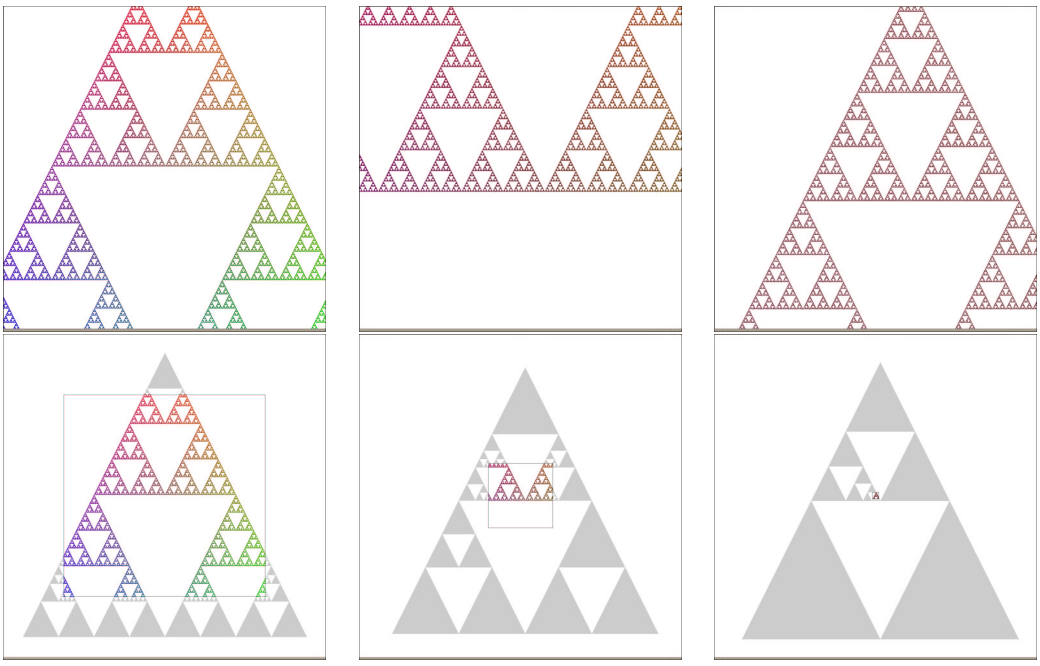


FIG. 6 – Visualisation des itérés exclus pour une caméra de plus en plus proche de la figure

4.2 Calcul d'inclusion

Pour afficher une figure représentée dans un espace homogène, il faut calculer l'intersection du cône positif engendré par K avec le plan $\{w = 1\}$:

$$K^E = \mathcal{C}^+(K) \cap \{w = 1\}.$$

La projection d'un itéré sur E^c est donnée par $\mathcal{C}^+(K_\rho) \cap \{w = 1\}$. Lorsque tous les w sont positifs, cette projection est un compact égal à l'enveloppe convexe de ces sommets. En posant :

$$x_{min} = \min_{(x,y,z)\in K_\rho} x \ , \quad x_{max} = \max_{(x,y,z)\in K_\rho} x \ ,$$

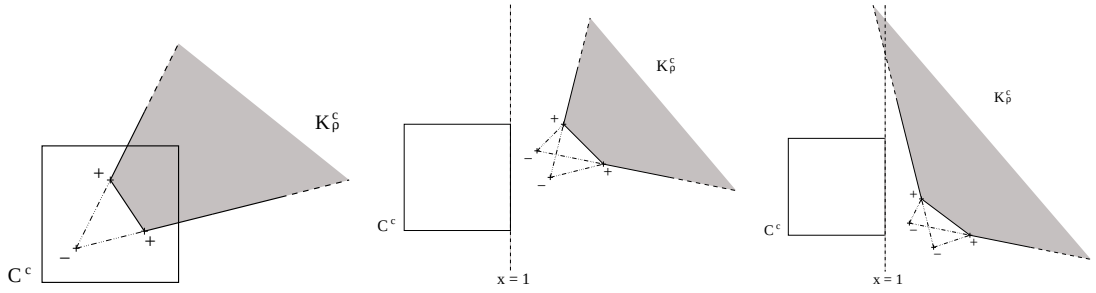
les tests sur les x se ramènent à :
– Condition d'inclusion : $x_{min} \geq -1 \wedge x_{max} \leq 1$
– Condition d'exclusion : $x_{max} \leq -1 \vee x_{min} \geq 1$
Lorsque l'on a certains w négatifs, $\mathcal{C}^+(K_\rho) \cap \{w = 1\}$ n'est pas un compact. La condition d'inclusion n'est jamais vérifiée, comme le montre la figure 7a.

En ce qui concerne la condition d'exclusion, il faut d'une part que les sommets dont w est positif vérifient les conditions précédemment définies, et d'autre part que les arêtes infinies engendrées par les points de fuite se trouvent dans le même demi-espace extérieur que les autres sommets.

On note :

$$x^+_{min} = \min_{\substack{(w,x_h,y_h,z_h)\in K_\rho \\ w>0}} \frac{x_h}{w} \ , \quad x^-_{max} = \max_{\substack{(w,x_h,y_h,z_h)\in K_\rho \\ w<0}} \frac{x_h}{w}$$

Considérons seulement le demi-espace $\{x \geq 1\}$. Pour vérifier la condition d'exclusion, il faut d'une part que tous les x qui ont un $w > 0$ soient supérieurs à 1 ; c'est à dire qu'il faut avoir $x^+_{min} \geq 1$. D'autre part, il faut que les arêtes infinies ne traversent pas le plan $\{x = 1\}$; il faut pour cela $x^+_{min} \geq x^-_{max}$, comme le montrent les figures 7b et 7c.



(a) Condition d'inclusion non respectée à cause de $w < 0$

(b) Condition d'exclusion vérifiée

(c) Condition d'exclusion non vérifiée car $x^+_{min} < x^-_{max}$

FIG. 7 – Cas particuliers de conditions d'inclusion et d'exclusion

La condition d'exclusion sur les x est : $(x^+_{max} \leq -1 \wedge x^+_{max} \leq x^-_{min}) \vee (x^+_{min} \geq 1 \wedge x^+_{min} \geq x^-_{max})$.

4.3 Complexité

Nous évaluons la complexité de cet algorithme en comptant le nombre de feuilles $|L|$ dans l'arbre de génération. Ces feuilles ce divisent en deux catégories : d'une part celles dont le polytope est suffisamment petit pour être affiché (inclus ou chevauchant), et d'autre part, celles dont le polytope est exclu :$|L| = |L_{image}| + |L_{exclu}|$.

La fenêtre de visualisation extrait des composantes réduites de la fractale (voir figure 6) :

$$L_{image}(\varepsilon_0, H(\alpha)^{-1}P^c) = \bigcup_{\gamma\in L_c} \gamma L(\varepsilon, H(\alpha)^{-1}P^c T_{\gamma_1} \dots T_{\gamma_n})$$

En supposant que l'effet du zoom $H(\alpha)^{-1}$ est « compensé » par les contractions $T_{\gamma_1} \dots T_{\gamma_n}$, on a :

$$|L_{image}(\varepsilon_0, H(\alpha)^{-1} P^c)| \approx |L(\varepsilon, P^c)|$$

Nous évaluons $|L_{exclu}|$ en raisonnant sur un exemple, celui du *triangle de* SIERPINSKI.

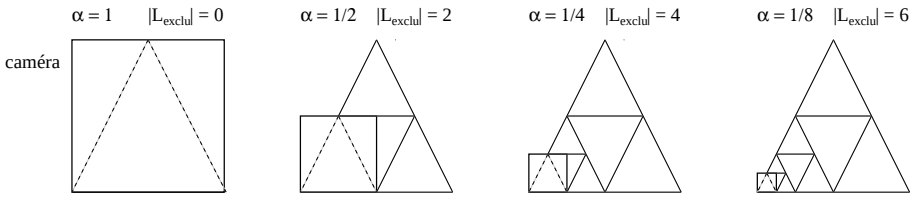


FIG. 8 – Évolution du nombre d'itérés exclus en fonction du zoom sur la figure fractale

Dans la figure 8, on constate que pour $\alpha = \frac{1}{2^n}$ on a $|L_{exclu}| = 2n$. On en déduit l'évolution de $|L_{exclu}|$ en fonction de α : $|L_{exclu}| = 2\frac{\ln(1/\alpha)}{\ln 2} = O(\ln(1/\alpha))$.

En faisant l'hypothèse que cette évaluation peut se généraliser, on a :

$$|L| = |L_{image}| + |L_{exclu}| \approx |L(\varepsilon_0, P^c)| + O(\ln(1/\alpha))$$

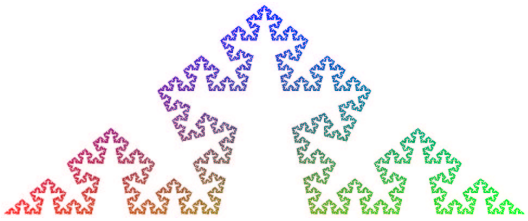


FIG. 9 – Courbe de VON KOCH

5 Résultats

Nous avons effectué des mesures de performances sur un exemple simple de fractale, la courbe de VON KOCH représentée sur la figure 9.

Nous présentons les performances obtenues avec plusieurs algorithmes de visualisation. La complexité de l'algorithme de BARNSLEY dépend de d, la profondeur d'itération. Le tableau 1 présente $|L|$ le nombre de feuilles calculées et f le nombre d'images par seconde en fonction de différentes valeurs de d.

	d	8	9	10	11	12	13
Algorithme de	$\|L\|$	256	512	1024	2048	4096	8192
BARNSLEY	f	118	59	30	15	7.5	3.8

TAB. 1 – Performances de l'algorithme de BARNSLEY

Le tableau 2 présente comparativement les performances obtenues avec l'algorithme multi-résolution et l'algorithme avec élagage. La résolution ε a été fixé de manière à ce que le résultat graphique soit de bonne qualité. Ce tableau présente les performances de ces algorithmes selon différentes positions de la caméra, où δ est la distance de la caméra au point de la fractale le plus proche.

	δ	1.2	0.6	0.3	0.15	0.1	10^{-4}	10^{-9}	10^{-14}
Algorithme multi-résolution	$\|L\|$	1024	3096	8192	32768	2097152			
	f	29	10.2	3.8	1	0.015			
Algorithme avec élagage	$\|L\|$	1024	2085	2009	2710	2867	1939	2696	2303
	$\|L_{image}\|$	1024	2046	1943	2661	2797	1855	2597	2174
	f	29	14.7	15.7	11.4	10.8	15.7	11.4	13.4

TAB. 2 – Performances de l'algorithme multi-résolution avec élagage

Nous remarquons que l'algorithme multi-résolution est très peu efficace lorsqu'il y a un zoom sur la fractale. L'algorithme avec élagage est beaucoup moins sensible à ce paramètre. Les performances de cet algorithme sont très liées à $|L_{image}|$, qui est représentatif de l'occupation de la fractale à l'écran. Les variations de $|L_{image}|$ sont dûes au fait que l'on peu zoomer sur des parties plus ou moins denses de la fractale. On peut remarquer que $|L|$ et $|L_{image}|$ varient très peu avec la distance, ainsi que f, ce qui valide l'efficacité de l'algorithme avec élagage.

6 Conclusion

Les caractéristiques particulières des IFS rendent possibles certaines optimisations qui permettent leur visualisation en temps interactif. Les résultats, tant d'un point de vue théorique (complexité) que pratique (simulations) prouvent l'efficacité de cette méthode.

Dans le cadre de futurs travaux, la question se pose d'appliquer cette méthode de visualisation à des modèles plus généraux. Ceci permettrait de simuler des objets plus variés, tels que ceux que l'on peut rencontrer dans le monde réel. On pourrait encore améliorer l'algorithme de visualisation. Dans l'élagage de l'arbre, il pourrait être intéressant d'estimer la visibilité de certaines parties de l'objet par rapport à d'autres, comme cela a été fait dans les travaux de modélisation multi-échelle [9]. Ceci aurait des avantages pour la visualisation d'objets dont la dimension fractale est très élevée.

Références

[1] M.F. BARNSLEY. *Fractals everywhere*. Academic press, 1988.

[2] M.F BARNSLEY and S. DEMKO. Iterated function systems and the global construction of fractals. *Proceeding of the Royal Society of London Ser A399*, pages 243–275, 1985.

[3] D. CANRIGHT. Estimating the spatial extent of attractors of Iterated Function Systems. *Computer & Graphics*, 18(2) :231–238, 1994.

[4] Serge DUBUC and Abdelkader ELQORTOBI. Approximations of fractal sets. *Journal of Computation and Applied Mathematics*, (29), 1990.

[5] Serge DUBUC and Abdelkader ELQORTOBI. The support function of an attractor. *Numer. Funct. Anal. and Optimiz.*, 14(3 & 4), 1993.

[6] C. GENTIL. Les fractales en synthèse d'image, le modèle IFS. Master's thesis, Université Claude Bernard Lyon 1, mars 1992.

[7] Eric GUERIN. *Approximation fractale de courbes et de surfaces*. Thèse de doctorat, Université Claude Bernard Lyon 1, 18 décembre 2002.

[8] Tomek MARTYN. Efficient ray tracing affine IFS attractors. *Computer & Graphics*, (25) :665–670, 1994.

[9] Franck PERBET. *Modélisation multi-échelle procédurale de scènes animées*. Thèse de doctorat, École doctorale, Sciences et technologies de l'information, Informatique, Février 2004.

[10] Jonathan RICE. Spatial bounding of self-affine iterated function system attractor sets. In *Graphics Interface'96*, pages 107–115, May 1996.

[11] Chems Eddine ZAIR. *Formes fractales à pôles basées sur une généralisation des IFS*. Thèse de doctorat, Université Claude Bernard Lyon 1, 11 juin 1998.