Procédé d'implémentation d'un fractal

L'implémentation d'un fractal se passe par l'implémentation de la tortue et d'un L-Systeme.
Un L-Systeme, est en quelque sorte le « langage » permettant la création graphique du fractal voulu. Le système de Lindenmayer est comparable à la modification à chaque itérations d'une liste. Chaque élément de cette liste subit alors une certaine application sur lui même. Cette application sur cet élément crée alors plusieurs nouveaux éléments pour la nouvelle liste. Puis on refait de même en prenant la nouvelle liste comme liste initiale. Un système est composé d'un **axiom** représentant un mot, une **rules** qui représente l'application agissant sur chaque élément de l'axiome, et d'i**nterp** représentant graphiquement l'**axiom**.
Un mot lui, est soit un **Symb** qui représente une lettre de l'alphabet du système, soit une **Seq** qui représente une liste de mots, ou bien soit une **branch** qui représente un mot. L'élément le plus petit d'un mot est **Symb**.
Pour l'implémentation d'un fractal, il est donc impératif de définir celui ci dans un **system**.
Rules est donc la fonction de substitution d'un **symbol**, pour chaque **symbol** on y applique cette fonction, qui peut être la transformation d'un **symb**, en une **Seq**, ou bien qui à un **symb** renvoie ce **symb**, etc.
Interp est l'interprétation graphique d'un **word**, et donc a une action « géométrique pour chaque lettre de l'alphabet du systèmes.
Viens ici le rôle de la Tortue, cf **turtle.mli**.

La tortue a une position exacte à chaque instant ainsi que l'angle qu'elle a par rapport à un axe horizontale. Cette position est en deux dimensions. On prend l'axe des abscisses et des ordonnées.
Il y a également un type **command** pour la tortue lui permettant de faire des actions géométriques dans le but de représenter graphiquement le chemin de la tortue en quelque sorte, donc pour représenter le fractal. Ce type peut tracer un trait en faisant avancer la tortue d'une certaine distance : **Line**, ou bien faire bouger la tortue sans tracer de traits : **Move**, ou bien modifier l'angle de la tortue par rapport à l'axe des abscisses : **Turn**, ou bien elle peut enregistrer une position courante, pour les '[' et **Restore** pour les ']' qui restaure la dernière position enregistrée non-encore restaurée.

On peut donc se faire la réflexion dans un premier temps à ne se préoccuper uniquement du système et de la tortue sans réellement se préoccuper de l'aspect graphique. Alors pour représenter le fractal, il faut remplir le système qui lui correspond. Après on remplit la partie graphique.

A partir de cela, pour le fractal, on va appliquer la fonction de substitution sur l’axiom à chaque itération. Il y a donc besoin d’analyser un mot, renvoyer sa nouvelle liste après substitution, puis afficher graphiquement la liste. Recommencer jusqu’à ce qu’on demande l’arrêt de l’itération, mais tout cela peut se faire à l’infini, ce qui est d’ailleurs l’aspect fascinant des fractales.

Voyons dans le document « vision de l’architecture du projet » comment nous voyons l’implémentation d’un fractal, de quels fonctions aurons nous besoin, comment gérer les problèmes, points d’attentions, points d’interrogations.