Corrigé du TP n°3
Le flocon de Von Koch

Guillaume VERNADE - guillaume.vernade@ens.fr

2 décembre 2008

Affixe

On part d'un couple de coordonnées et il faut calculer l'affixe correspondant dans le plan complexe. Il suffit donc d'associer $y + i * y$ à (x, y). Pour cela on récupère la première coordonnée de p avec `p[1]` et la seconde avec `p[2]`.

```
affixe := p -> p[1]+I*p[2] ;
```

Je rappelle qu'on peut définir une fonction en l'écrivant sous la forme `NomDeLaFonction := Variables -> Résultat ;`

Pointaffixe

Il faut maintenant faire l'opération inverse, cad. associer le couple (x, y) au complexe $x + i * y$. Pour cela on récupère la partie réelle du complexe c avec `Re(c)` et sa partie imaginaire avec `Im(c)`.

```
pointaffixe_inverse := x -> [Re(x),Im(x)] ;
```

Il va maintenant falloir appliquer cette transformation à une liste de complexes. Pour cela on utilise `map` qui permet justement d'appliquer une fonction à tous les éléments d'une liste.

```
pointaffixe := L -> map(pointaffixe_inverse, L) ;
```

On peut aussi utiliser un `seq` pour créer une nouvelle liste.

```
pointaffixe := L -> [seq(pointaffixe_inverse(L[i]),i=1..nops(L))] ;
```

Rotation

Pour ce qui est de la rotation votre cours de maths vous donne la formule $a + (b - a) * e^{i*\theta}$ comme résultat de la rotation d'angle θ de b autour de a. Attention à ne pas oublier les parrenthèses autours des arguments.

```
rotation := (a, b, theta) -> a+(b-a)*exp(I*theta) ;
```

Transfo

Là l'affaire se corse, il faut, à partir de deux complexes a et e renvoyer les affixes des 5 points caractérisants les 4 segments post-transformation. Pour deux d'entre eux c'est facile, il s'agit de a et e. Les deux suivant (par ordre de simplicité) sont aussi faciles à calculer puisqu'il s'agit des points à 1/3 et 2/3 du segment $[a, e]$, cad. `b=a/3+2*e/3` et `d=2*a/3+e/3`. Enfin le dernier forme avec les deux précédents un triangle équilatéral, son abscisse s'obtient donc facilement en effectuant la rotation de b, de centre d et d'angle $\pi/3$, cad. `c=rotation(b,d,Pi/3)`.

```
transfo := proc(a,e)
  local b,c,d; % Dans le programme on crée les variables b,c et d qui ne
               % font pas partie de celles données lors de l'exécution, en
               % utilisant 'local' on précise qu'elles ne servirons qu'à
               % l'intérieur du programme et qu'il faut les effacer à la
               % fin. Si par hasard des variables b,c,d existaient avant
               % l'exécution du programme leur valeur n'est pas altérée.
  b := 2*a/3 + e/3 ;
  d := a/3 + 2*e/3 ;
  c := rotation(b,d,Pi/3) ;
  a,b,c,d,e; % la suite des 5 points est calculée en dernier et est donc
             % renvoyée par le programme.
end;
```

Un autre façon de créer des programmes/fonctions/procédures est de les écrire sous la forme :

```
NomDuProgramme := proc (Variables) %Pas de ;
Calculs ;
end ;
```

Flocon

Voila maintenant le programme principal. Il faut, à partir d'un couple d'affixe de points a et b calculer le résultat de la transformation appliquée n fois. C'est à dire qu'à partir d'un segment, la transformation de Koch en crée 4, puis à partir de ces 4 segments elle en crée 16, et ainsi de suite.

Si T est la transformation on doit calculer $T^n(a,b) = T(T^{n-1}(a,b)) = T^{n-1}(T(a,b))$, il y a donc deux méthodes :

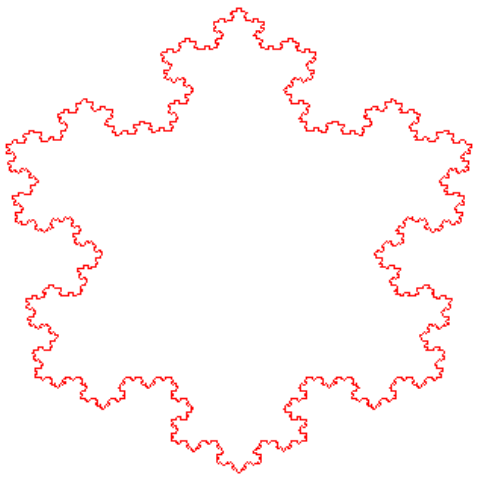
- calculer $T^{n-1}(a,b)$ puis appliquer T à chaque segment obtenu, le problème étant qu'il y en a plein et qu'on n'a pas forcément envie de faire une récurrence pour en calculer le nombre exact (qu'il faudrait donner pour la boucle)
- calculer $T(a,b)$ et appliquer T^{n-1} à chaqu'un des 4 segments obtenus. C'est ce que l'on va faire.

```
flocon:=proc(a,b,n)
  local L ;
  if n=0
    then
      a,b ; % Si n est nul, on renvoie le couple correspondant au segment
            % [a,b], c'est le cas d'arrêt qui fait que l'on ne va pas
            % appeler flocon un nombre infini de fois.
    else
      L := transfo(a,b) ; % L est la suite des 5 points qui définissent les
                          % 4 segments obtenus avec transfo.
      flocon(L[1],L[2],n-1), flocon(L[2],L[3],n-1), flocon(L[3],L[4],n-1), flocon(L[4],L[5],n-1) ;
      % On calcule T^(n-1) de chacun des 4 segments et on met les résultats
      % les un à la suite des autres.
    end;
end;
```

On remarquera qu'un certain nombre de points serons présents plusieurs fois (le dernier d'une liste est le même que le premier de la liste suivante) mais ce n'est pas grave. Cela l'aurait été plus si on avait voulu refaire des calculs dessus, par exemple si on avait choisi le première méthode.

Finalement

On a plus qu'à afficher les points avec un n assez grand pour qu'on ait l'impression qu'il y a bien des segments. n=5 suffit et on obtient :



Pour avoir le flocon entier il faut appliquer la transformation à un triangle équilatéral, ce que je fais.