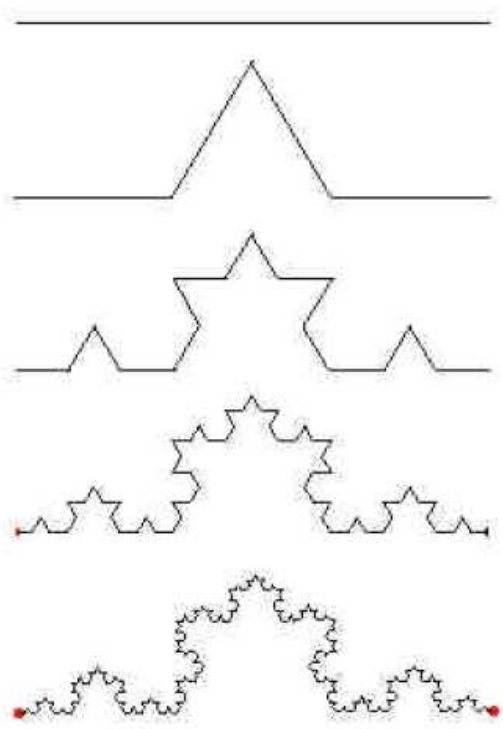
Une courbe fractale : courbe de Von Koch (1904)



On note n le nombre d'étapes. On démontre que le nombre de points à l'étape n est N = 4^n+1.
n sera limité à 9 (qui nécessite 262 145 points pour la construction de la courbe),
Pour une première simulation, on prendra n=6 (4 097 points)
La longueur l du segment sera initialisée à 1.

1) Ecrire une fonction [u, v] = etape(x, y)
Où x est un tableau a deux entrées contenant les abscisses x(1) et x(2) respectivement de l'extrémité gauche et de l'extrémité droite du segment. De même, y est un tableau contenant les ordonnées y(1) et y(2) de ces deux points.
u et v sont des tableaux à trois valeurs représentant les trois points intermédiaires (u pour les abscisses et v pour les ordonnées)

2) En raison de la symétrie, nous allons d'abord construire la première partie de la courbe pour ensuite en déduire la partie symétrique.
Ecrire une fonction [xcourbe, ycourbe] = courbe1(n) renvoyant la première partie de la courbe à l'étape n.
Où lequel xcourbe et ycourbe représenteront respectivement le vecteur global des abscisses et des ordonnées des points de la courbe.
3) Ecrire une fonction [xcourbe,ycourbe]=symétrie(xcourbe, ycourbe) qui détermine la partie symétrique de la courbe.

4) Ecrire un programme qui utilise les fonctions précédentes. Après avoir déterminé les coordonnées des points de la courbe, vous utiliserez la fonction xpoly(xcourbe,ycourbe) pour la construire.

LES FRACTALES EN ALGORITHMES

LE FLOCON DE VAN KOCH

```
function initialiser_graphique1
xbasc()
square(0,-1/(sqrt(3)*2),1,sqrt(3)/2)

function [tab]=decouper_segment1(a,b)
tab=zeros(5,1)
tab(1,1)=a
tab(2,1)=a+(b-a)/3
tab(3,1)=a+(b-a)/3*(1+exp(%i*%pi/3))
tab(4,1)=a+2*(b-a)/3
tab(5,1)=b

function dessiner_segment(tab)
xpoly(real(tab),imag(tab),"lines",0)

function [r]=r1(tab)
r=tab*exp(%i*%pi/3)

function [r]=r2(tab)
r=1+(tab-1)*exp(%i*(-%pi/3))

function [s]=sym(tab)
s=conj(tab)

function [tab]=une_etape1(t)
n=length(t)
tab=zeros(4*n-3,1)
for i=1:n-1
u=decouper_segment1(t(i,1),t(i+1,1))
for j=1:4 do tab(4*(i-1)+j,1)=u(j,1), end
end
tab(4*n-3,1)=t(n,1)

function dessiner1(tab)
initialiser_graphique1()
dessiner_segment(sym(tab))
dessiner_segment(r2(tab))
dessiner_segment(r1(tab))

function flocon_de_Van_Koch
n=input('A quelle étape voulez-vous obtenir le flocon de Van Koch ?')
while n<=0 |int(n)<>n
disp('IL FAUT SAISIR UN ENTIER POSITIF','ERREUR !')
n=input('A quelle étape voulez-vous obtenir le flocon de Van Koch ?')
end
tab=[0;1]
for i=2:n do t=une_etape1(tab),tab=t,end
if n<2 then, t=tab, end
initialiser_graphique1()
dessiner1(t)
T=input('Voulez-vous une étape de plus ? ("o" ou "n")')
while T<>'o' & T<>'n' do disp('ERREUR !!!'), T=input('Voulez-vous une étape de plus ? ("o" ou "n")'),
end
while T=='o' do tab=t, t=une_etape1(tab)
dessiner1(t)
T=input('Voulez-vous une étape de plus ? ("o" ou "n")')
```

```
while T<>'o' & T<>'n' do disp('ERREUR !!!'), T=input('Voulez-vous une étape de plus ? ("o" ou "n")'),
end
end
```

LE TRIANGLE DE SIERPINSKI

```
function initialiser_graphique3
xbasc()
square(0,0,1,1)
xpoly([0;1;0.5],[0;0;sqrt(3)/2],"lines",1)

function [tab]=decouper_segment2(a,b)
tab=zeros(6,1)
tab(1,1)=a
tab(2,1)=(a+b)/2
tab(3,1)=a+(b-a)/2*exp(%i*%pi/3)
tab(4,1)=b+(a-b)/2*exp(%i*(-%pi/3))
tab(5,1)=(a+b)/2
tab(6,1)=b

function dessiner3(tab)
xpoly(real(tab),imag(tab),"lines",0)
t=[tab(2,1);tab(3,1);tab(4,1)]
xfpoly(real(t),imag(t),1)

function [t]=une_etape2(tab)
n=length(tab)/2
t=zeros(6*n,1)
for i=1:n
u=decouper_segment2(tab(2*i-1),tab(2*i))
dessiner3(u)
for k=1:6 do t(6*(i-1)+k,1)=u(k,1), end
end
xpoly([0;0.5;1],[0;sqrt(3)/2;0],"lines",0)

function triangle_de_Sierpinski
n=input('A quelle étape voulez-vous obtenir le triangle_de_Sierpinski ?')
while n<=0 |int(n)<>n
disp('IL FAUT SAISIR UN ENTIER POSITIF','ERREUR !')
n=input('A quelle étape voulez-vous obtenir le triangle_de_Sierpinski ?')
end
tab=[0;1]
initialiser_graphique3()
for i=2:n do t=une_etape2(tab),tab=t,end
T=input('Voulez-vous une étape de plus ? ("o" ou "n")')
while T<>'o' & T<>'n' do disp('ERREUR !!!'), T=input('Voulez-vous une étape de plus ? ("o" ou "n")'),
end
while T=='o' do t=une_etape2(tab), tab=t
T=input('Voulez-vous une étape de plus ? ("o" ou "n")')
while T<>'o' & T<>'n' do disp('ERREUR !!!'), T=input('Voulez-vous une étape de plus ? ("o" ou "n")'),
end
end
```

LE TAPIS DE SIERPINSKI

```
function initialiser_graphique2
xbasc()
square(0,0,1,1)
xpoly([0;1;1;0],[0;0;1;1],"lines",1)

function [t]=generer_centres(centre,cote)
```

```
t=zeros(8,1)
t(1,1)=centre+cote*(-1-%i)
t(3,1)=centre+cote*(-1+%i)
t(5,1)=centre+cote*(1+%i)
t(7,1)=centre+cote*(1-%i)
t(2,1)=(t(1,1)+t(3,1))/2
t(4,1)=(t(3,1)+t(5,1))/2
t(6,1)=(t(5,1)+t(7,1))/2
t(8,1)=(t(1,1)+t(7,1))/2

function [t]=decouper_carre(tab)
n=length(tab)
t=zeros(8*n,1)
k=log(n)/log(8)+1
for i=1:n
u=generer_centres(tab(i,1),(1/3)^k)
for j=1:8 do t(8*(i-1)+j,1)=u(j,1), end
end

function dessiner_carre(centre,cote)
a=centre+cote*(-1-%i)/2
b=centre+cote*(-1+%i)/2
c=centre+cote*(1+%i)/2
d=centre+cote*(1-%i)/2
t=[a;b;c;d]
xfpoly(real(t),imag(t),1)

function dessiner2(tab, cote)
for i=1:length(tab)/8
dessiner_carre(tab(i,1),cote)
dessiner_carre(tab(i+length(tab)/8,1),cote)
dessiner_carre(tab(i+length(tab)/4,1),cote)
dessiner_carre(tab(i+3*length(tab)/8,1),cote)
dessiner_carre(tab(i+length(tab)/2,1),cote)
dessiner_carre(tab(i+5*length(tab)/8,1),cote)
dessiner_carre(tab(i+3*length(tab)/4,1),cote)
dessiner_carre(tab(i+7*length(tab)/8,1),cote)
end

function tapis_de_Sierpinski
tab=[(1+%i)/2]
n=input('A quelle étape voulez-vous obtenir le napperon de Sierpinski ?')
while n<=0 |int(n)<>n
disp('IL FAUT SAISIR UN ENTIER POSITIF','ERREUR !')
n=input('A quelle étape voulez-vous obtenir le napperon de Sierpinski ?')
end
initialiser_graphique2()
dessiner_carre(tab,1/3)
if n>1 then, for i=2:n do t=decouper_carre(tab), tab=t, dessiner2(tab,1/3^i), end, end
T=input('Voulez-vous une étape de plus ? ("o" ou "n")')
while T<>'o' & T<>'n' do disp('ERREUR !!!'), T=input('Voulez-vous une étape de plus ? ("o" ou "n")'),
end
while T=='o' do t=decouper_carre(tab),tab=t,n=n+1
dessiner2(tab,1/3^n)
T=input('Voulez-vous une étape de plus ? ("o" ou "n")')
while T<>'o' & T<>'n' do disp('ERREUR !!!'), T=input('Voulez-vous une étape de plus ? ("o" ou "n")'),
end
end
```

Correction

```
function [u, v] = etape(x, y)

  xmilieu0 = (x(2)+x(1))/2;
  ymilieu0 = (y(2)+y(1))/2;

  xu= x(2)-x(1);   yu = y(2)-y(1);     // coordonnées d'un vecteur directeur du segment
  xn = y(1)-y(2);    yn = x(2)-x(1);   // vecteur normal au segment

  u(1) = x(1)+xu/3; v(1) = y(1)+yu/3;
  u(3) = x(2)-xu/3; v(3) = y(2)-yu/3;
  u(2) = xmilieu0 + xn*(sqrt(3)/2)/3;    v(2) = ymilieu0 + yn*(sqrt(3)/2)/3;
// sqrt(3)/2 est le rapport entre la hauteur et le cote d'un triangle équilatéral

endfunction

function [xcourbe, ycourbe] = courbe1(n)
xcourbe = zeros(1,N);  ycourbe = zeros(1,N);
xcourbe1 = zeros(1,N);  ycourbe1 = zeros(1,N);  xcourbe(2) = l;

for i=1:n
  jmax = 4^(i-1); // nombre de segments au début de l'étape i

  for j=1:jmax/2+1 // on travaille sur deux points j et j+1 (segment j)

    decalage = (j-1)*4; // un point est décalé en raison de l'apparition de nouveaux points
    x_init = xcourbe(j:j+1);
    y_init = ycourbe(j:j+1); // segment j
   [x_trans, y_trans] = etape(x_init,y_init); // segment transformé
    xcourbe1(decalage+1) = x_init(1); xcourbe1(decalage+5) = x_init(2);
    ycourbe1(decalage+1) = y_init(1); ycourbe1(decalage+5) = y_init(2);
    for k=1:3
      xcourbe1(k+decalage+1) = x_trans(k);
      ycourbe1(k+decalage+1) = y_trans(k); // mise dans le vecteur global
    end
  end
  xcourbe = xcourbe1; ycourbe = ycourbe1;
end

endfunction

function [xcourbe, ycourbe] = symetrie(xcourbe, ycourbe)
for i=1:4^n
  ycourbe(N-i+1) = ycourbe(i);
  xcourbe(N-i+1) = l-xcourbe(i); // seconde partie de courbe
end
endfunction

//  programmme principal

clf;    n = 6;   N = 4^n+1;    l = 1;
[xcourbe,ycourbe]=courbe1(n) ;
[xcourbe, ycourbe] = symetrie(xcourbe, ycourbe) ;
xpoly(xcourbe,ycourbe)
```