MUX104
Synthèse d'image et réalité virtuelle

Fractales

Pierre Cubaud
cubaud @ cnam.fr

mars 2020

le cnam

1. Fractales, relief

Formes naturelles

- Le besoin : appréhender la complexité des formes naturelles

  eau / terre / feu / vie

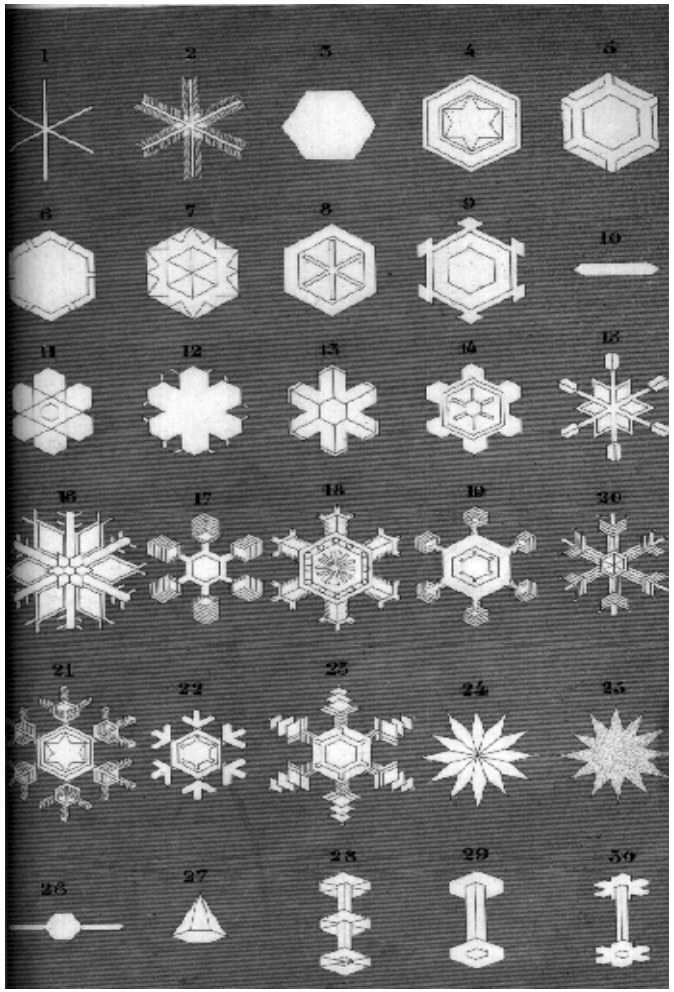
- Première approche : récurence et/ou hasard simulé

- Exemples de tous les jours :

  - Le chou-fleur

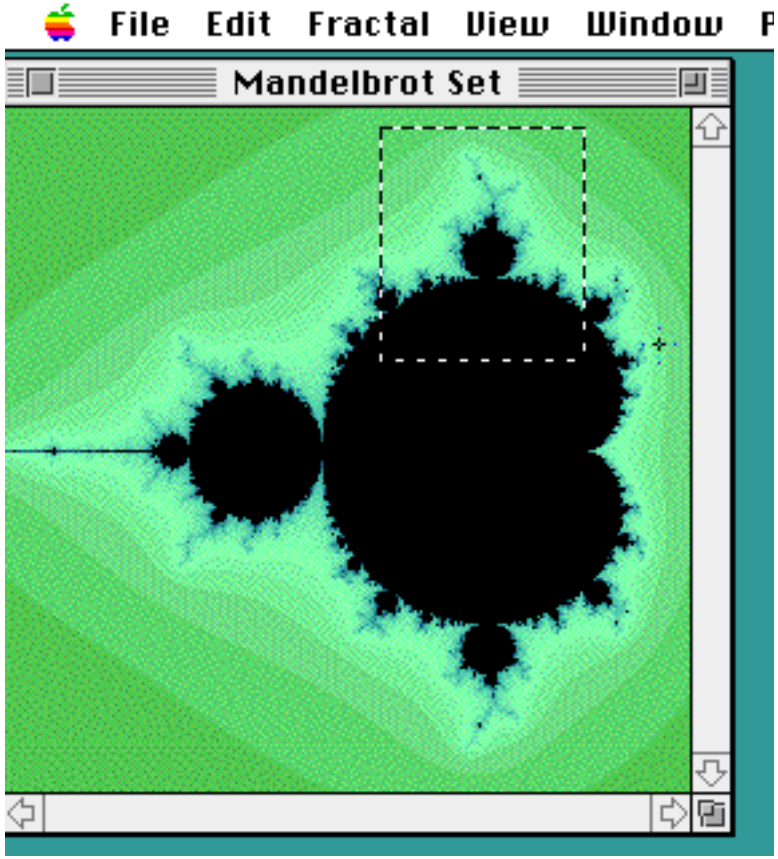
  - Cristal de sel que l'on casse (Hauÿ)

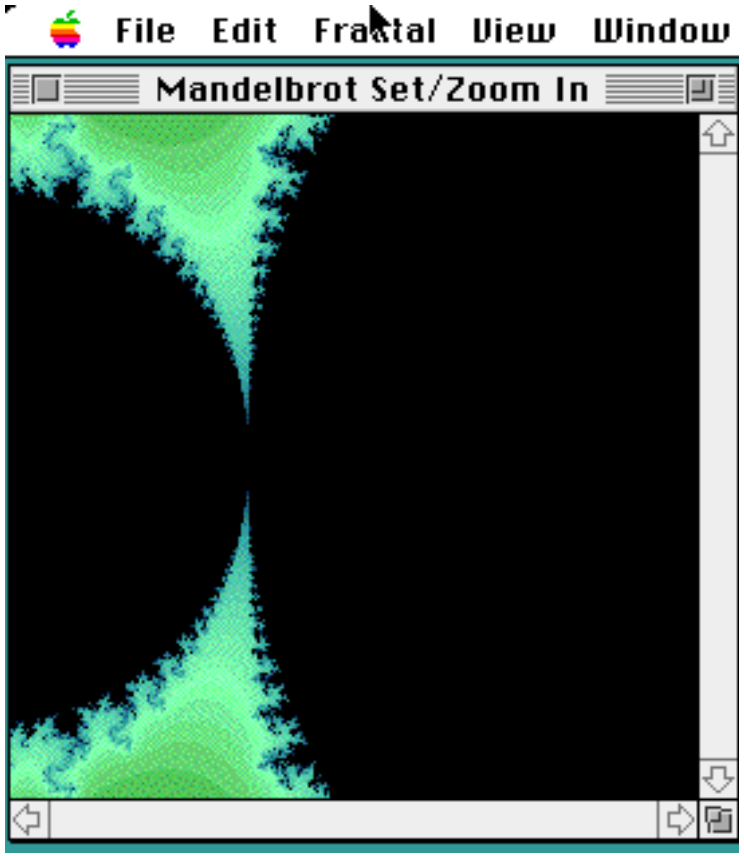
  - Cristaux de neige et de givre



Kaemtz - Traité de météorologie, trad. Martins 1843

L'ensemble de Mandelbot





MandelBrowser, Jess Jones, MacOS, 1995

zoom infini ! => détail à la demande

- On étudie la fonction itérative complexe :

$$z(n+1) = z(n)^2 + c$$

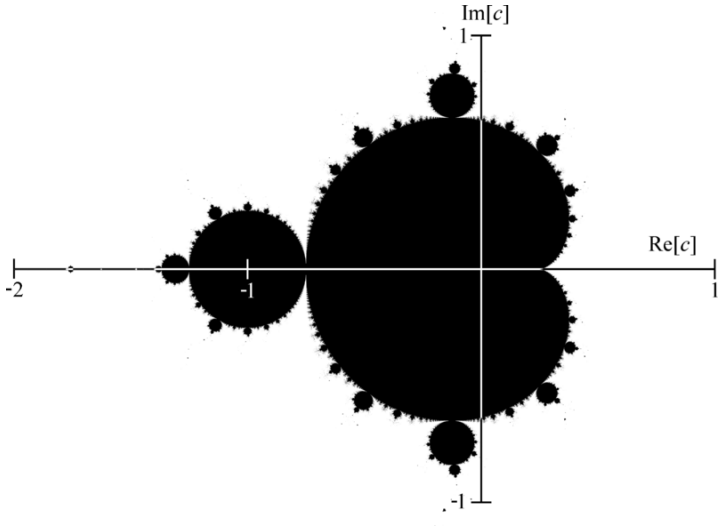
avec $z(0) = c$

Selon la valeur de c, la suite tend soit vers l'infini, soit vers une constante

- Exemples :

c = 1+i

itération 1	module de z = 3.162
itération 2	module de z = 9.899
itération 3	module de z = 97.01
itération 4	module de z = 9409

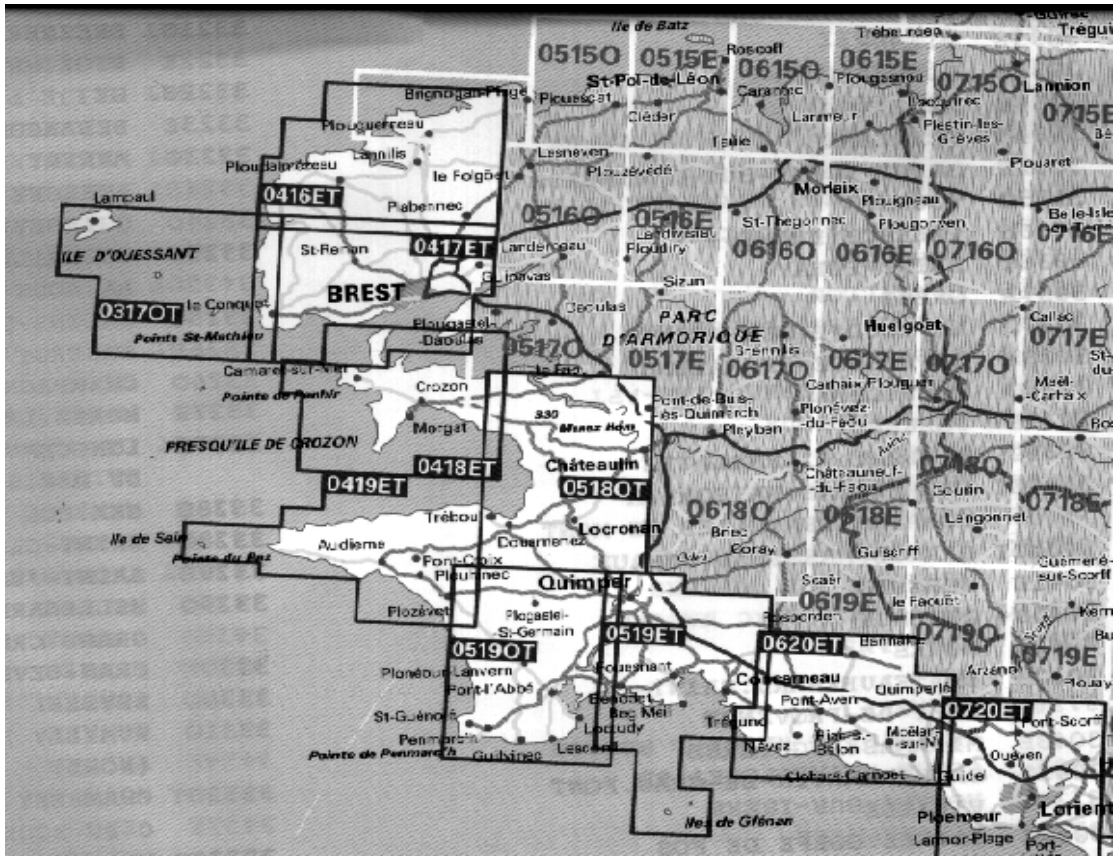


c = -1 + 0.25 i

itération 1	module de z = 0.5
itération 2	module de z = 0.559
itération 3	module de z = 0.731
itération 80	module de z = 0.493

- L'ensemble de Mandelbrot est constitué par tous les c tels que z^2+c reste fini quelque soit le nombre d'itération

Exemple des côtes de la bretagne



© IGN - Catalogue des cartes série bleue et Top 25

Loi de L.F. Richardson

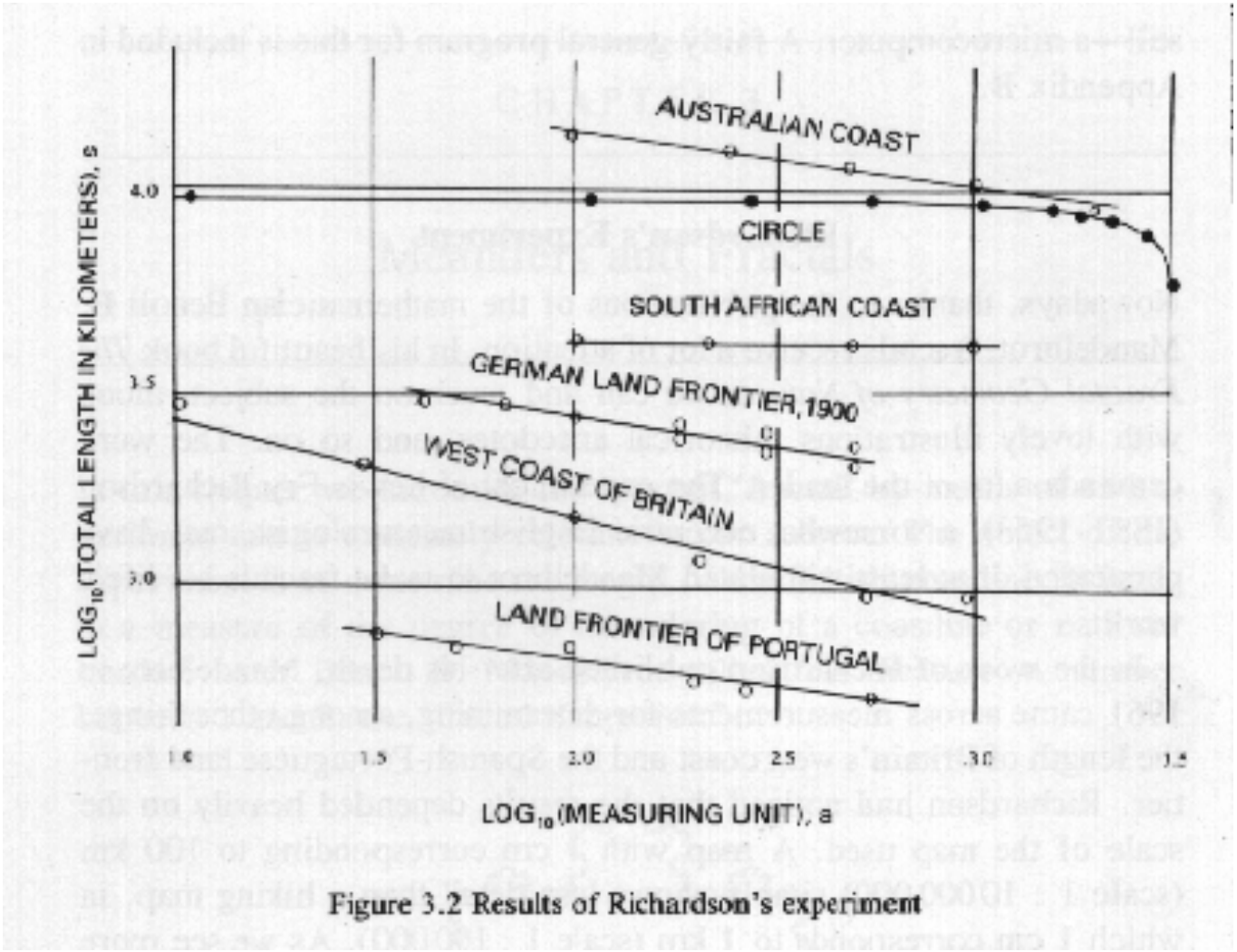
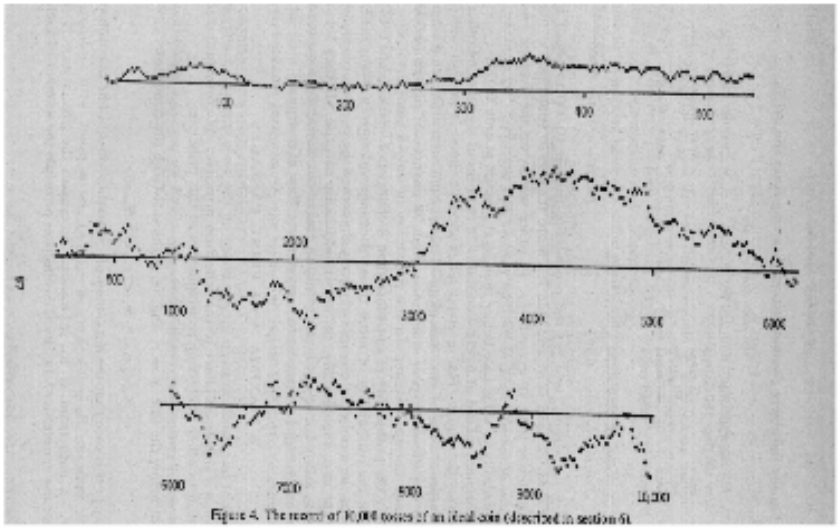


Figure 3.2 Results of Richardson's experiment

[LAUWERIER] p. 30

Modèles de reliefs

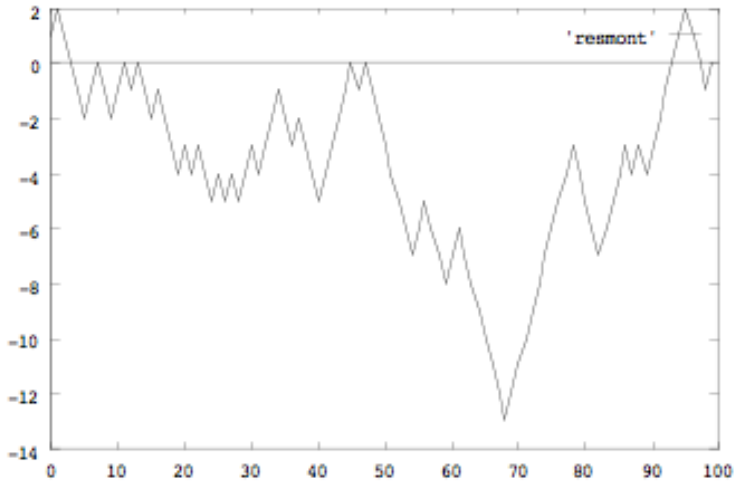
- Exemple de marche aléatoire : le gain au jeu de pile ou face :



Feller - An introduction to probability theory… - vol 1. - Wiley, 1950. p. 87

```
main()
{
int s,i;

 s1=354675; /* racines du generateur aleatoire */
 s2=78364;

 s=0;
 for(i=1;i<=100;i++)
 {
  if (Uniform()<0.5){s++;} else {s--;}
  printf("%d \n",s);
 }
}
```

Visualisation avec GNUplot :

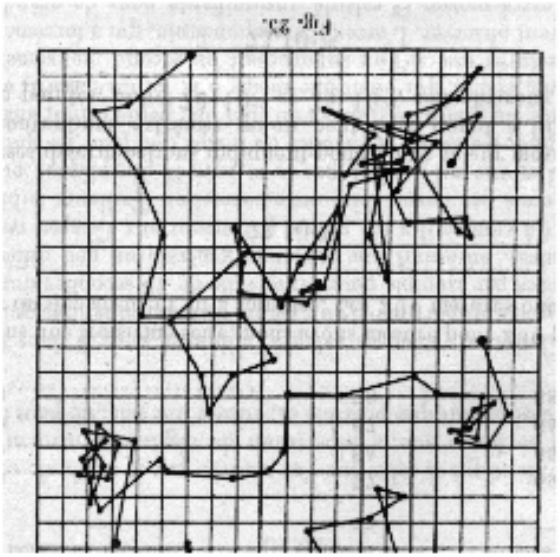


2. Bruit brownien fractionnaire (Fbm)

Mouvement brownien

‘ La figure ci-jointe reproduit trois dessins obtenus en traçant les segments qui joignent les positions consécutives d'un même grain de mastic, à 30 secondes d'intervalle. C'est le demi-carré moyen de tels segments qui vérifie la loi d'Einstein. L'un de ces dessins contient 50 positions consécutives d'un même grain. Ils ne donnent qu'une idée très affaiblie du prodigieux enchevêtrement de la trajectoire réelle. Si, en effet, on faisait des pointés de secondes en secondes, chacun de ces segments rectilignes se trouverait remplacé par un contour polygonal de 30 côtés relativement aussi compliqué que le dessin reproduit ici, et ainsi de suite. ”

J. Perrin Ann. Chimie et Physique, 8ème série, 18, 1909. reprod. in Œuvres scientifiques de J. Perrin, CNRS, 1950 p. 217-8

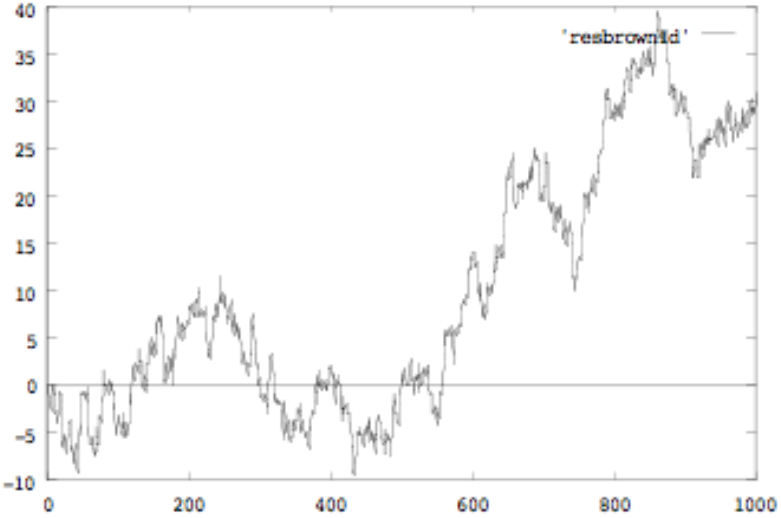


1 carreau = 5mm pour 1.7μm

voir aussi : “Les atomes”

```
main(){
int t;double y;

/* racines du generateur aleatoire */
 s1=354675;s2=78364;
t=0.0;y=0.0;
 printf("%.2f %.2f\n",t,y);
 for(t=0;t<TMAX;t++) {
  y += Normal(0.0,D);
  printf("%d %f \n",t,y);
 }
}
```

Visualisation avec GNUplot :



L'écart-type D ne modifie que l'amplitude du mouvement (en ordonnée ici), mais pas la “rugosité” du signal. Celle-ci dépend finalement de N

- Subdivion récursive du mouvement brownien

  On choisit (xd,yd) la position de départ (t=0)

  On choisit (xf,yf) la position finale à date t=T

  Celle-ci suit une loi normale de moyenne (xd,yd) et de variance 2DT

  La position intermédiaire à date T/2 connaissant le départ et l'arrivée suit elle-aussi une loi normale de variance 2D(T/2)=DT

  On montre que sa position moyenne est la moyenne des positions initiales et finales

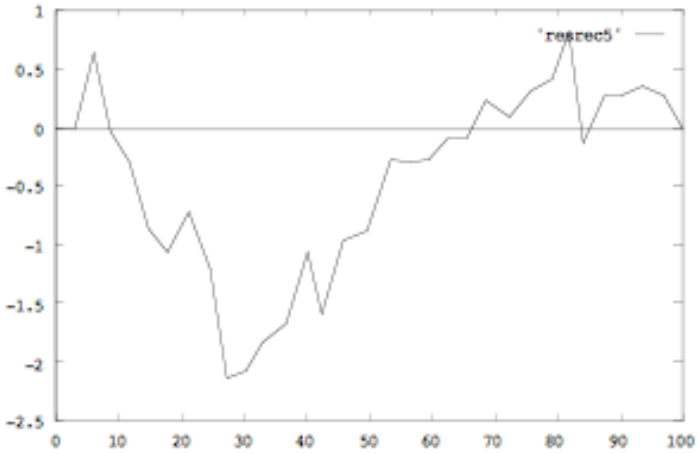
  (S. Karlin *Initiation aux processus aléatoires* Dunod 1969 p. 301)

  IMPORTANT : Ceci parce que la loi normale est stable pour l'opération d'addition (" infiniment divisible "). Ce n'est pas la seule . Point de départ des travaux de P. Lévy, puis de B. Mandelbrot

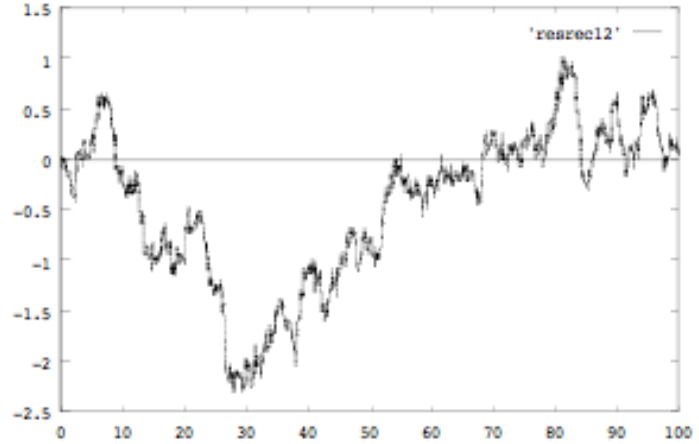
Algorithme largeur d'abord !!

```
#define D 1.0
#define N 12

main()
{
s1=354675;s2=78364;
/* racines du generateur aleatoire */

  fin=Nouveau(100.0,0.0,NULL);
  deb=Nouveau(0.0,0.0,fin);
  s=sqrt(D);
  for(i=1;i<=N;i++){
   p=deb;
   while (p!=fin){
    xm=Normal((p->x+p->suiv->x)/2.0,s);
    ym=Normal((p->y+p->suiv->y)/2.0,s);
    q=Nouveau(xm,ym,p->suiv);
    p->suiv=q;
    p=q->suiv;
   }
   s /= sqrt(2.0);
  }

 p=deb;
 while (p!=NULL) {
  printf("%f %f\n",p->x,p->y);
  p=p->suiv;
 }

}
```

Avec N=5 : 33 points



Avec N=12 : 4097 points



Mouvement brownien fractionnaire

*Mandelbrot. *Une classe de processus stochastiques homothétiques à soi ; application à la loi climatologique de H.E. Hurst.* CRAS t. 260, pp. 3274-7 (mars 65)

rep. in B. Mandelbrot. *Fractales, hasard et finance.* Flammarion, 1997 pp. 26-9)

On introduit le paramètre 0<H<1 (facteur de Hurst) dans la variance du processus brownien :

$$var(X(t)) = D.t^{(2H)}$$

Le processus reste self-similaire, mais apparaît un phénomène de dépendance à long terme.

En pratique, H=3/4 se rencontre souvent dans la nature (séries climatologiques, reliefs…)

L'algorithme de subdivision peut facilement être modifié :

```
s /= sqrt(2.0);
```

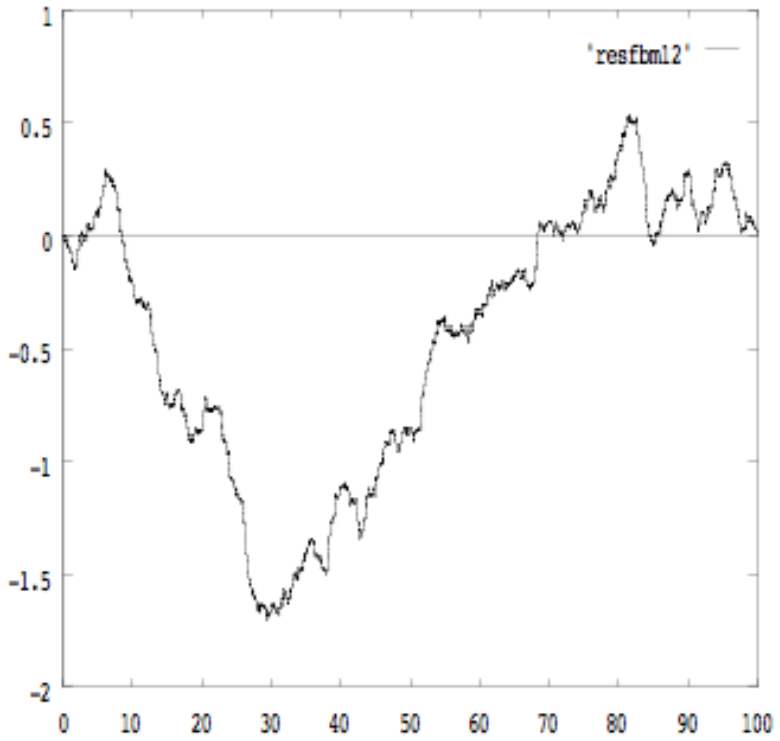
devient :

```
s /= pow(2.0,H);
```

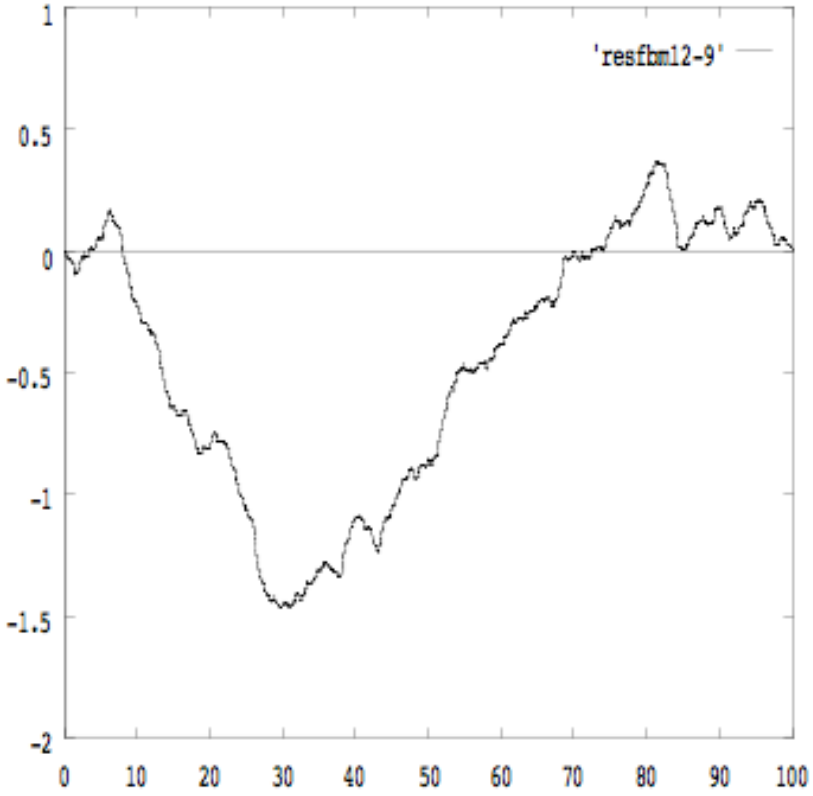
mais le processus produit n'est qu'une approximation du MBf

Exemples :

avec H =3/4= 0.75



avec H=0.90



comparer avec le mvt. brownien ordinaire (H=0.5)

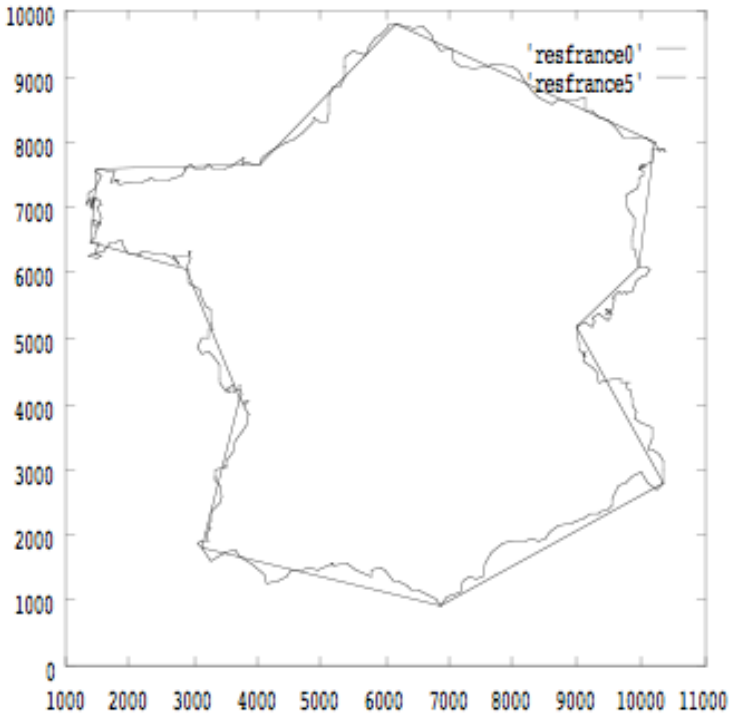
demo

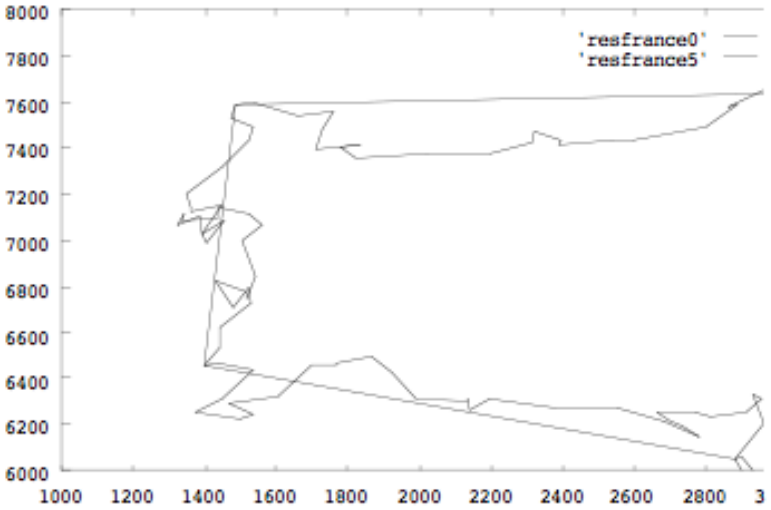
Modèles de terrains (bis)

On a utilisé un polygône de 13 côtés H = 0.7 N = 12

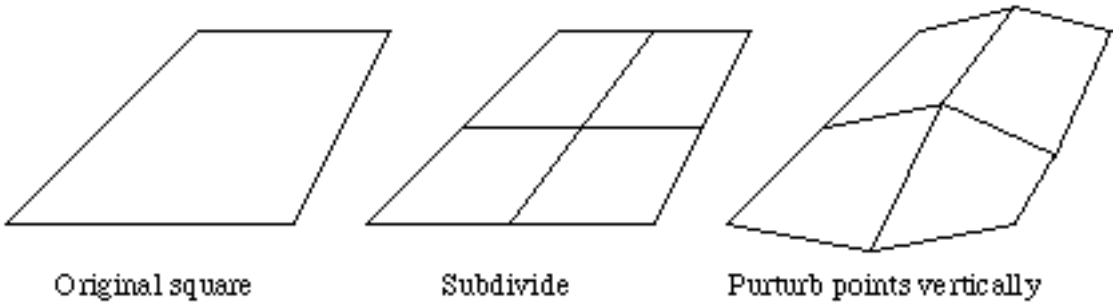
D = 5e4 pour le repère choisi :

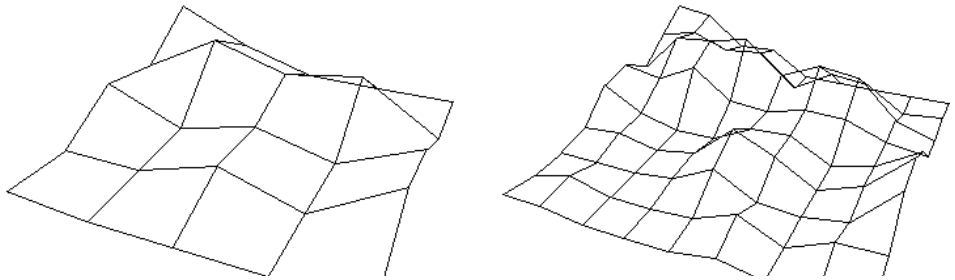
0<x<11000 et 0<y<11000



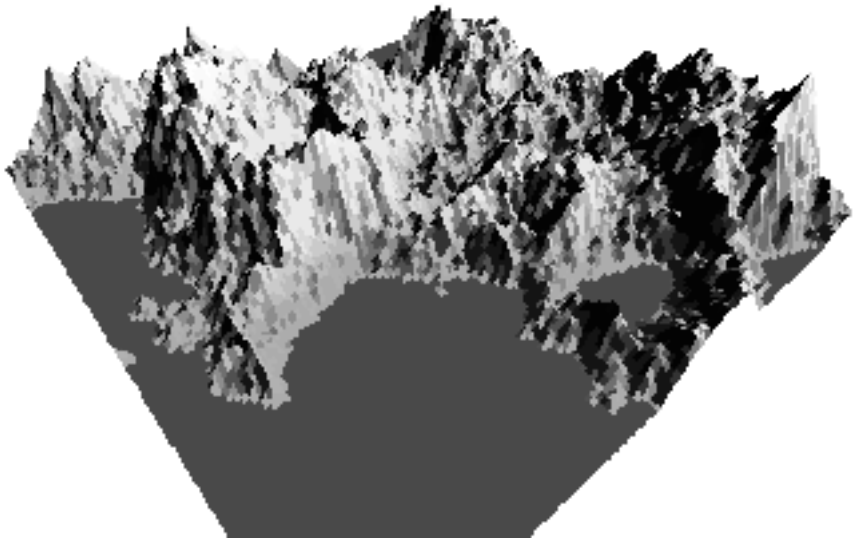


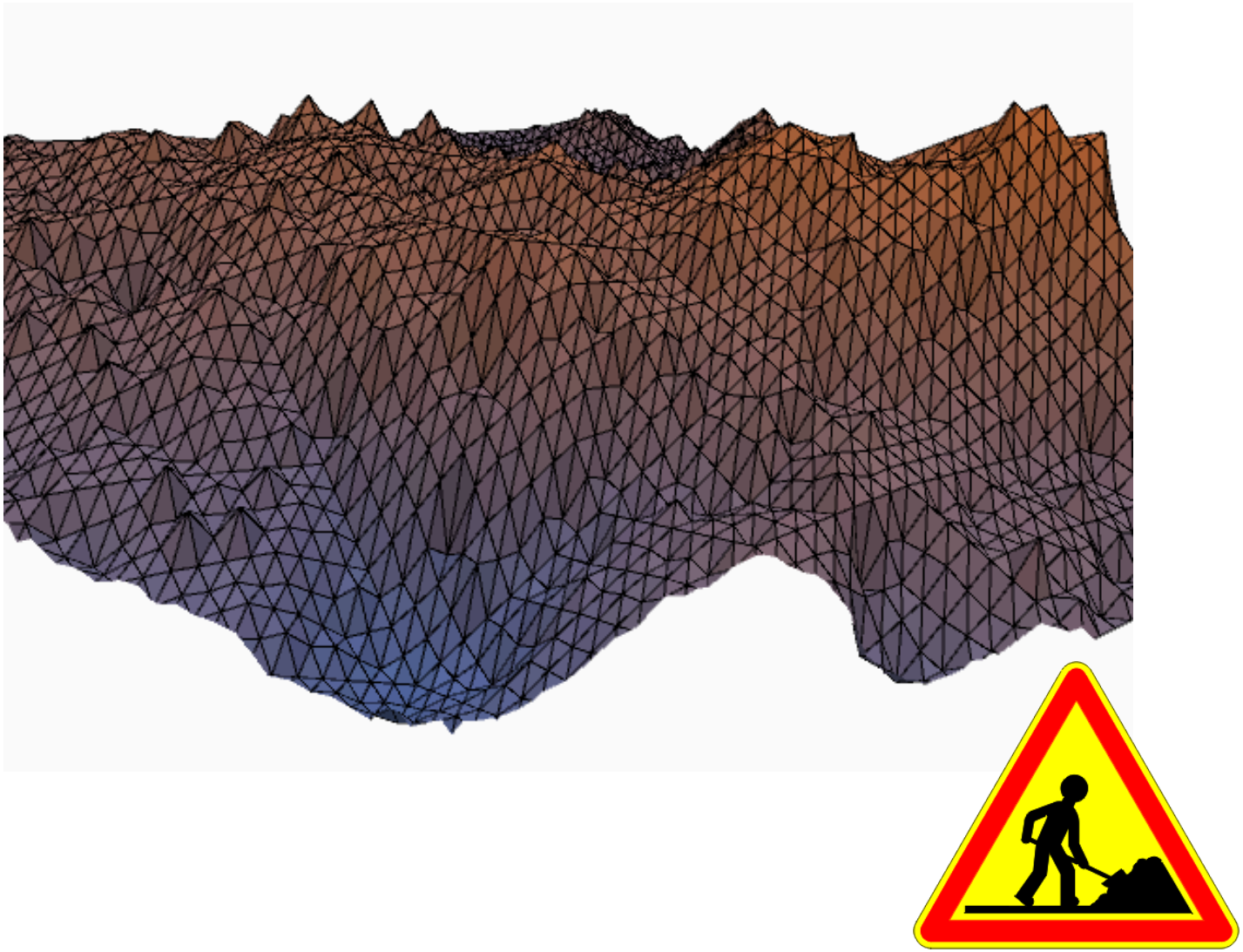
Passage en 3D



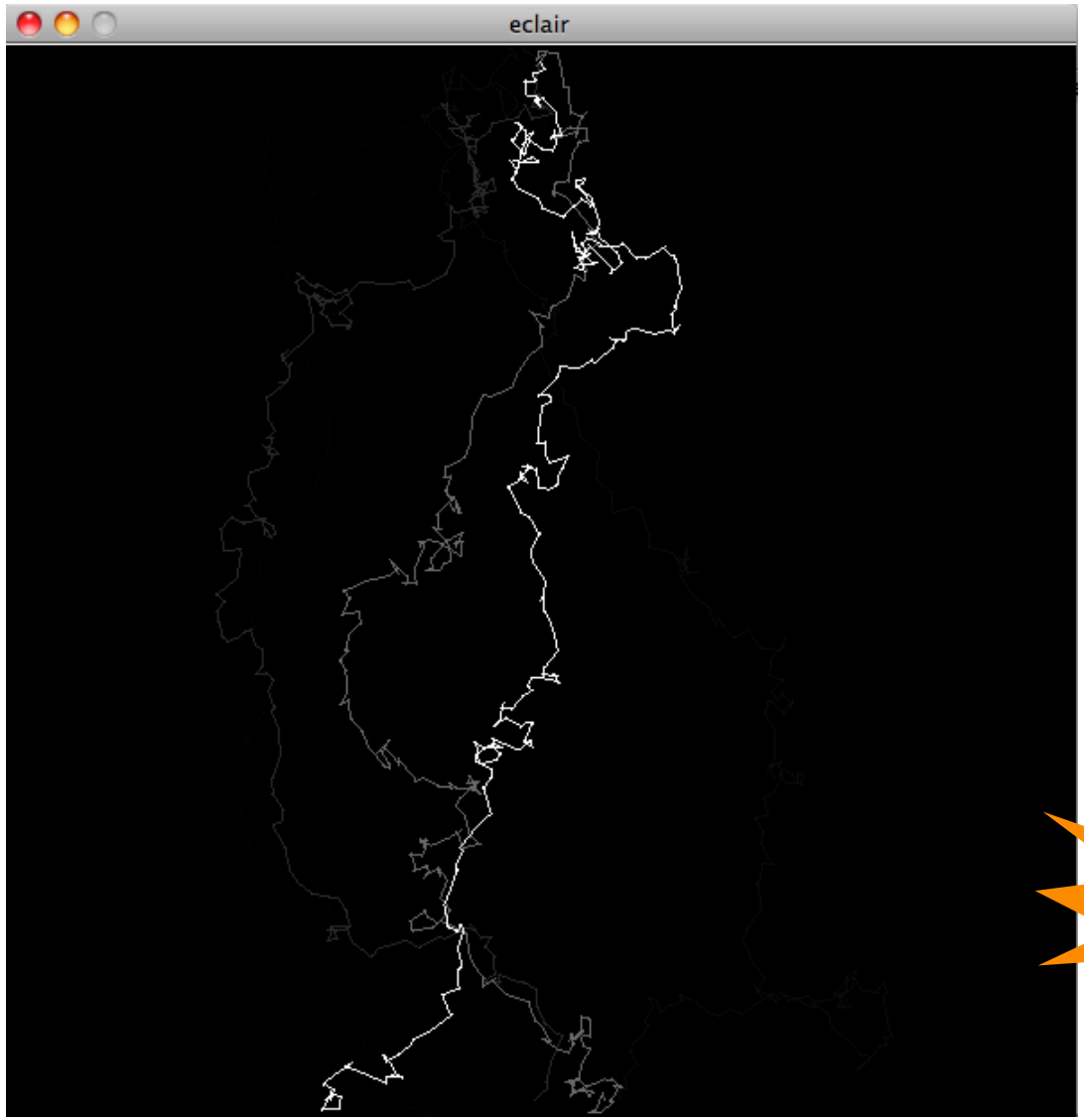


4x4





Nombreuses autres applications du Fbm



demo

3. Bruit de Perlin

```
main(k){float i,j,r,x,y=-16;while(puts(""),y++<15)for(x
=0;x++<84;putchar(" .:-;!/>)l&IH%*#"[k&15]))for(i=k=r=0;
j=r*r-i*i-2+x/25,i=2*r*i+y/10,j*j+i*i<11&&k++<111;r=j);}
```

http://www.mrl.nyu.edu/~perlin/doc/oscar.html

©A.M.P.A.S.® *(click here for legal notice)*

Noise and Turbulence

In 1997 I received a Technical Achievement Award from the Academy of Motion Picture Arts and Sciences for work I had done on procedural texture. For example, the NYU Torch on the right is made entirely from procedural textures (except for the text along the bottom). The flame, background, and metal and marble handle are not actually 3D models - they are all entirely faked with textures. A hi-res image of a marble vase I made using this technique can be found here. A bunch of other texture images I created can be found here.

I've written up a fun history and how-to site called **MAKING NOISE**. It includes a tutorial, examples, animations, and interactive applet diagrams, plus a description of a hardware implementation.

I then improved it, and wrote a paper about that. You can play with interactive demos of the improved version here.

You can play with designing noise-based textures yourself with a really nice interactive Java Applet created by Justin Legakis, and Hugo Elias has a nice web page about it, Also, the interactive fractal planet demo on my home page is made using these techniques.

It seems that my techniques found their way into the various software packages, such as Autodesk *Maya, Softlmage, 3D Studio Max, Dynamation, RenderMan,* etc., that folks use to make the effects for feature films, which is way cool. Movies look better now, and I guess that makes me a good American.

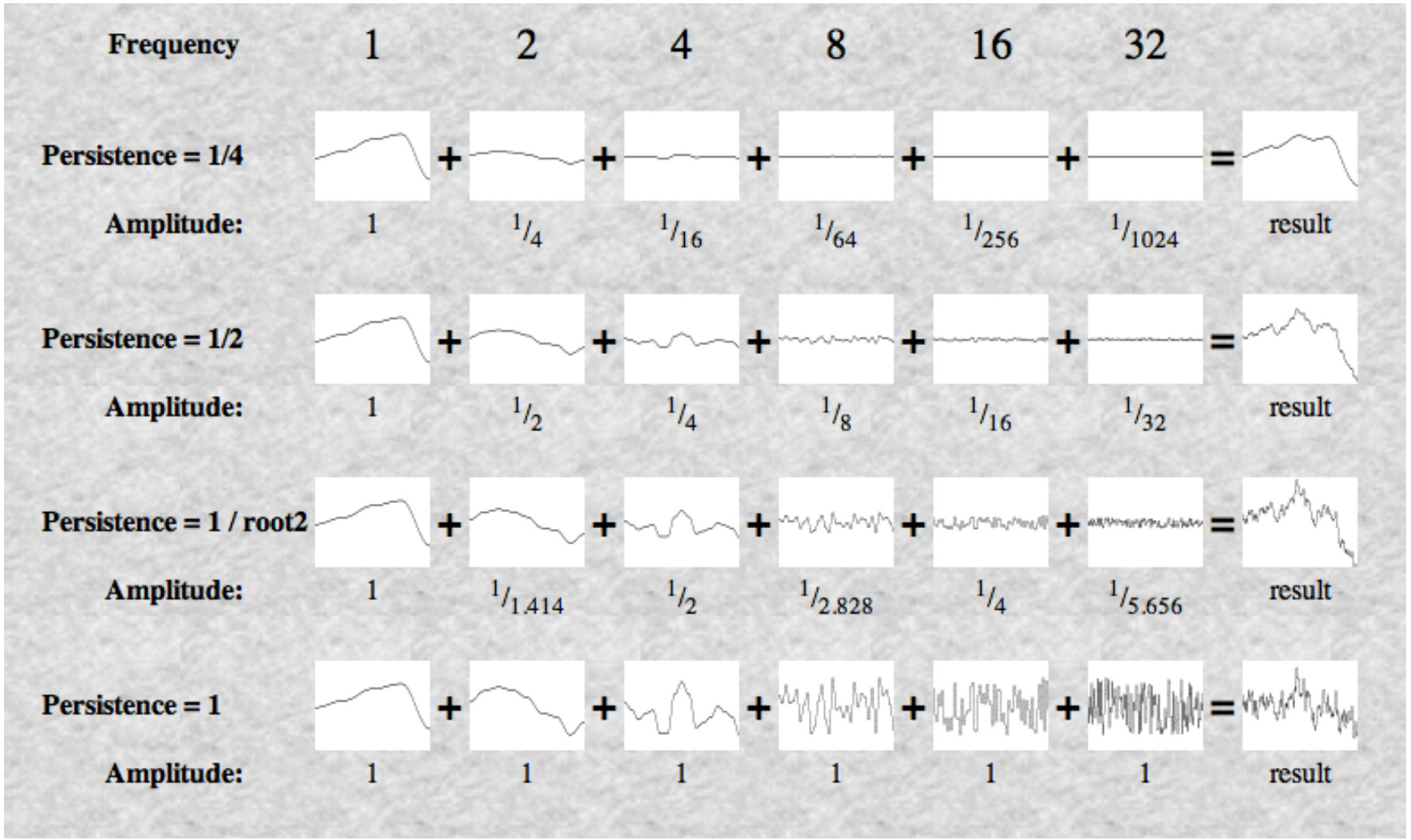
Here's what the award actually says:

> ***To Ken Perlin for the development of Perlin Noise, a technique used to produce natural appearing textures on computer generated surfaces for motion picture visual effects.***
>
> *The development of Perlin Noise has allowed computer graphics artists to better represent the complexity of natural phenomena in visual effects for the motion picture industry.*

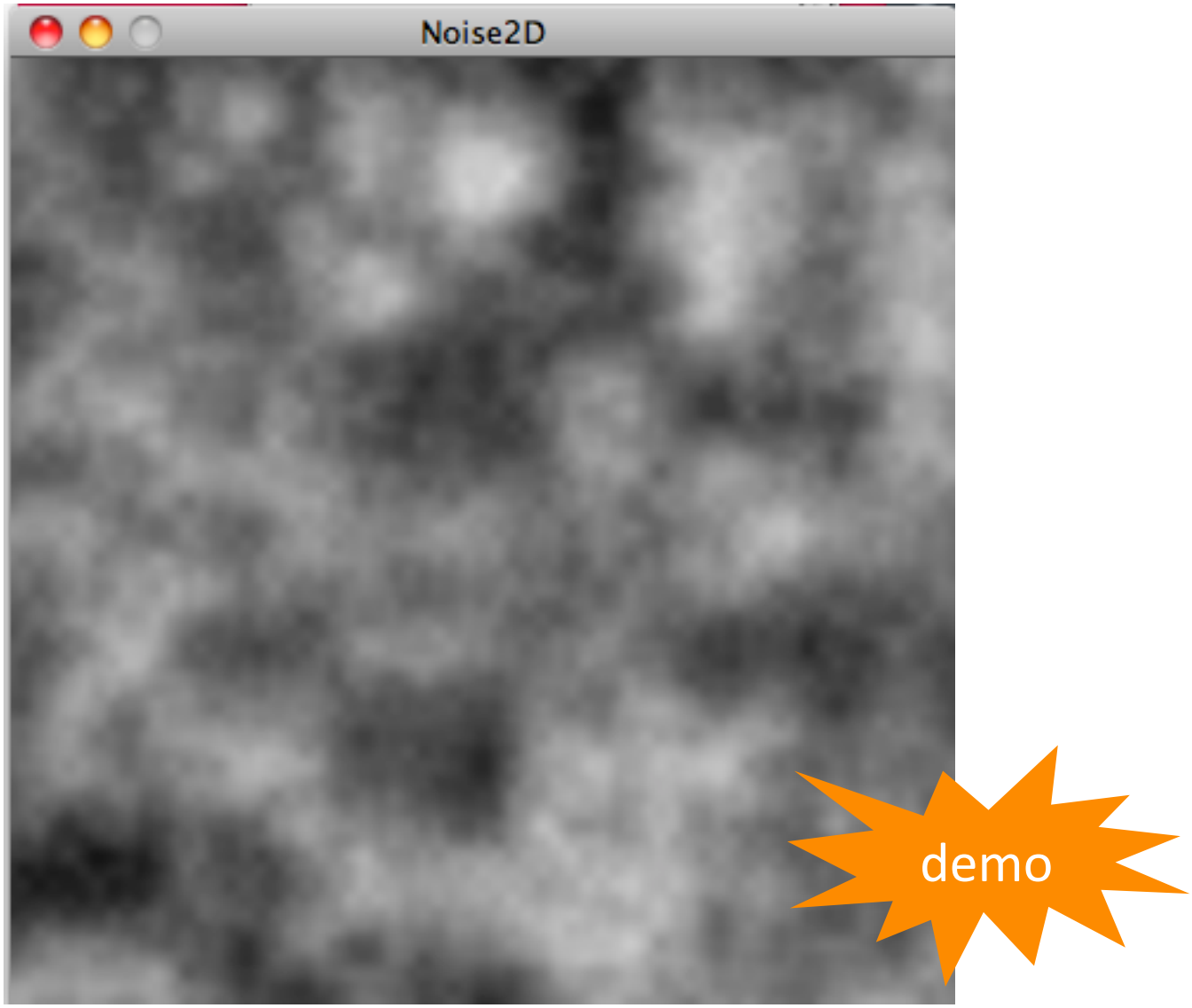
Needless to say, my Mom was very happy.

Go Here to see the source code for my original C implementation of *Noise*.

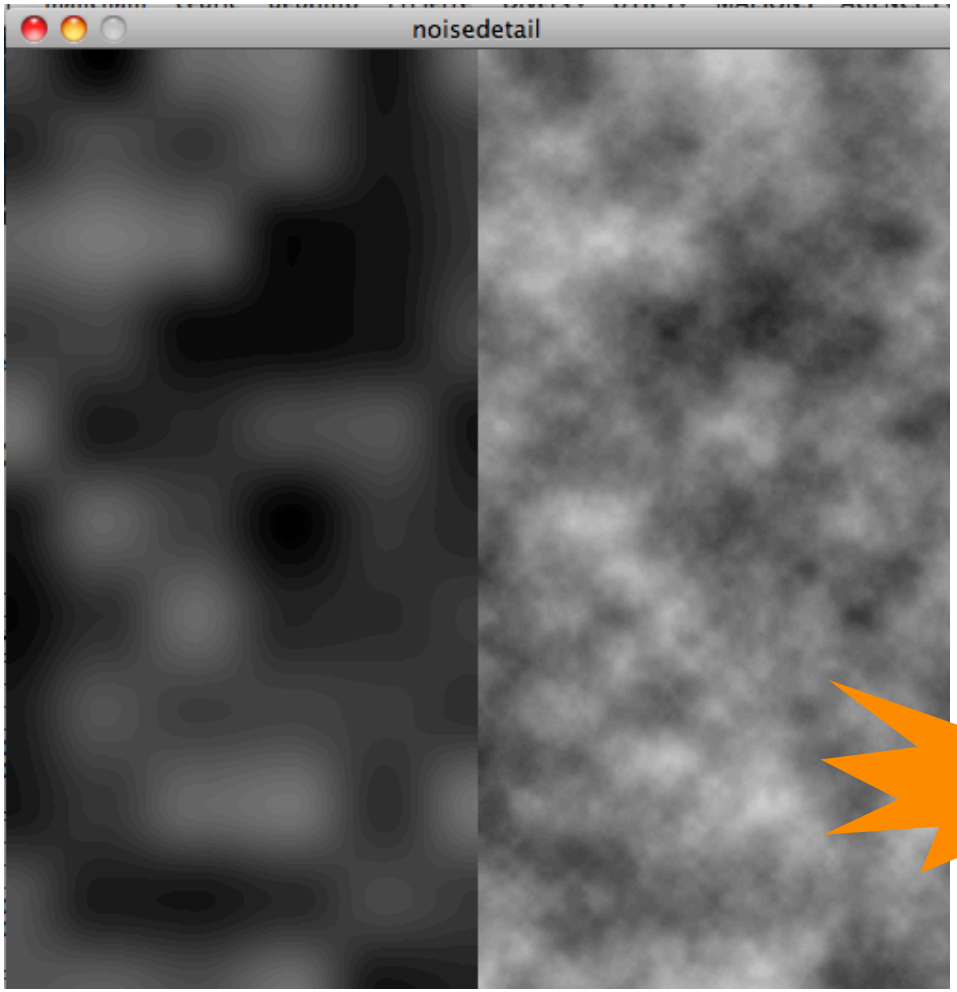
Principe : additionner des séquences RND en "frequences" doublées



http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

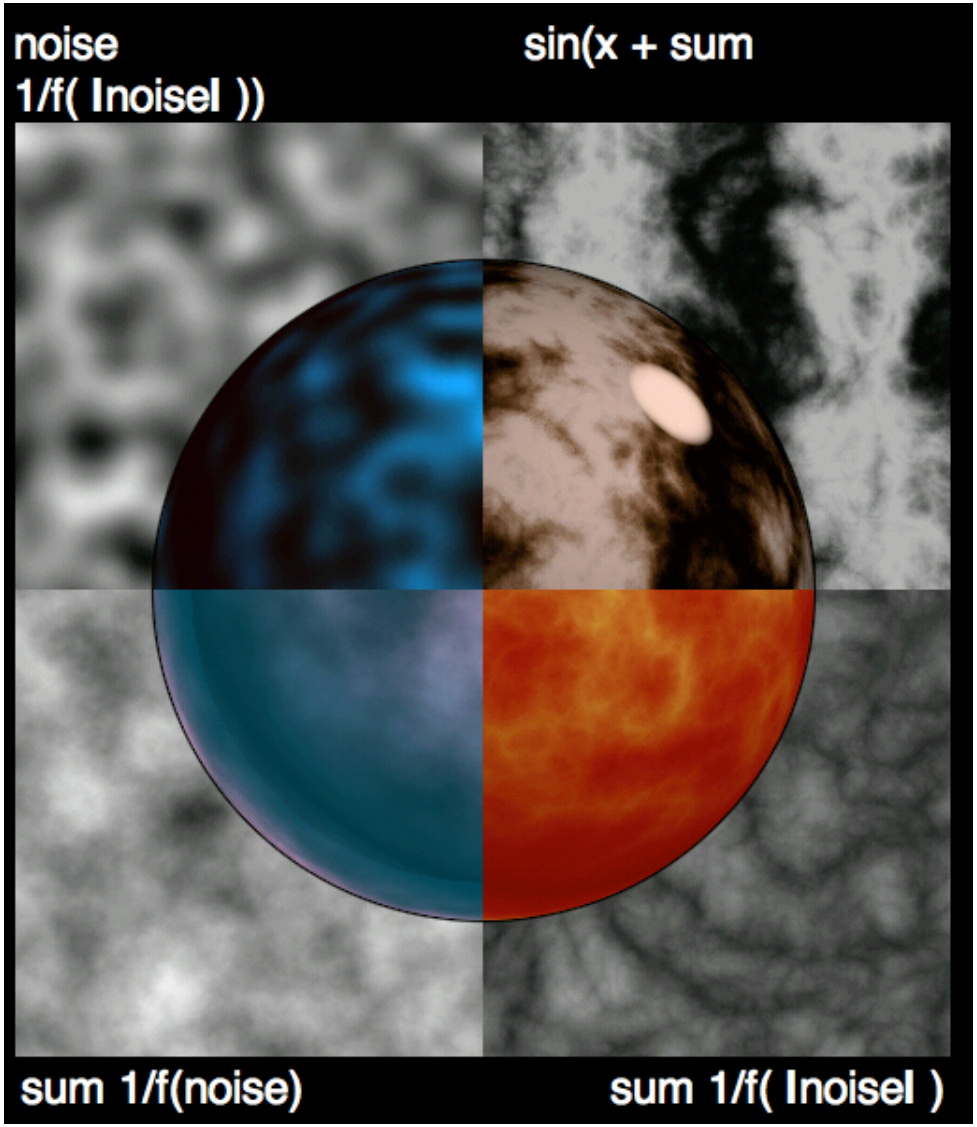

Noise2D
demo

Influence du niveau de détails ("octaves")



demo

detail = 1 detail=8 (fallof standard = 50%)



www.noisemachine.com (de Perlin aussi)

http://mrl.nyu.edu/~perlin/noise/ perlin noise.c
MOOC pcsCHI routeur mailcnam cedric deptinfo PrPierre Divers UTILS

Improved Noise reference implementation

Ken Perlin

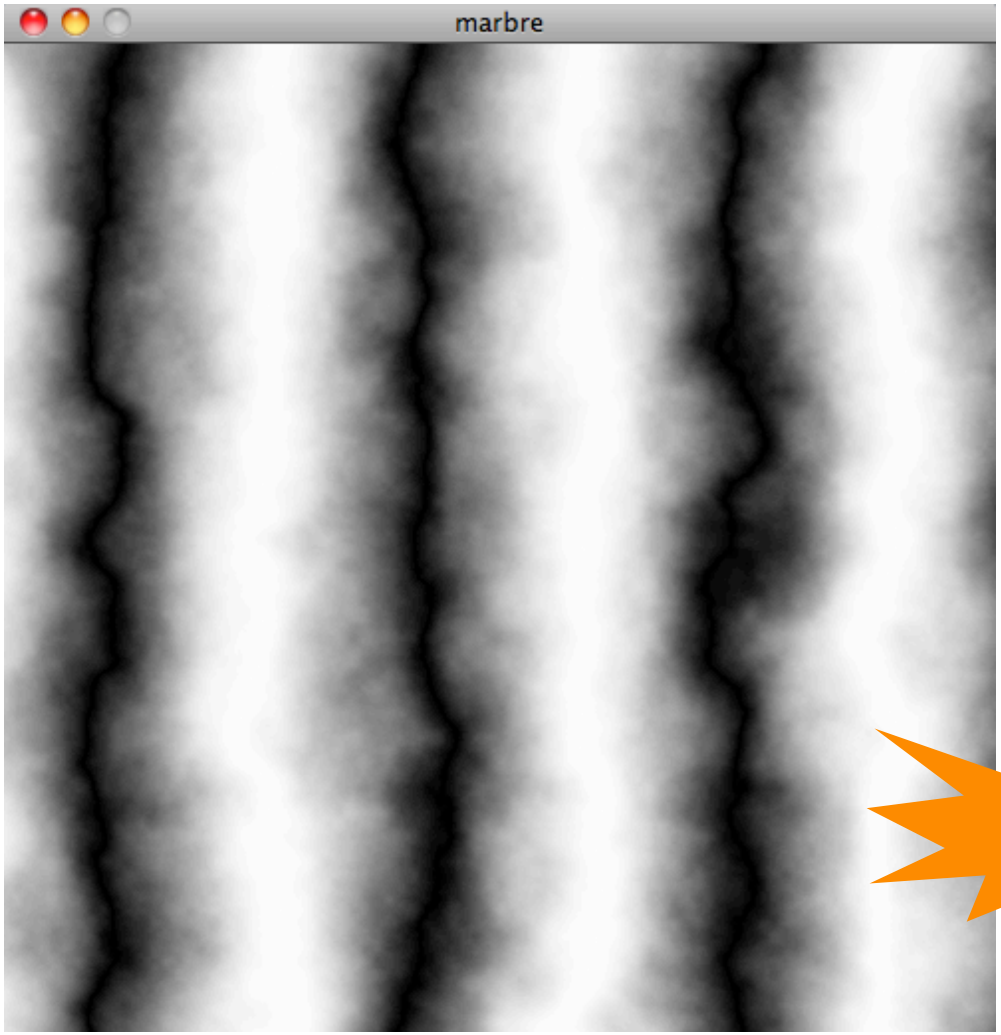
Smooth ball demo

Bumpy ball demo

This code implements the algorithm I describe in a corresponding SIGGRAPH 2002 paper.

```
// JAVA REFERENCE IMPLEMENTATION OF IMPROVED NOISE - COPYRIGHT 2002 KEN PERLIN.

public final class ImprovedNoise {
   static public double noise(double x, double y, double z) {
      int X = (int)Math.floor(x) & 255,                  // FIND UNIT CUBE THAT
          Y = (int)Math.floor(y) & 255,                  // CONTAINS POINT.
          Z = (int)Math.floor(z) & 255;
      x -= Math.floor(x);                                // FIND RELATIVE X,Y,Z
      y -= Math.floor(y);                                // OF POINT IN CUBE.
      z -= Math.floor(z);
      double u = fade(x),                                // COMPUTE FADE CURVES
```

Perturbation d'une structure périodique : exemple du "marbre"



demo

Pleins d'autres applications : voir livre/site de Schiffman

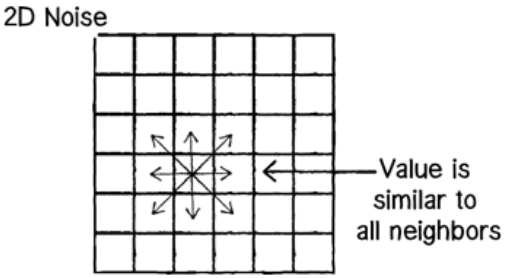
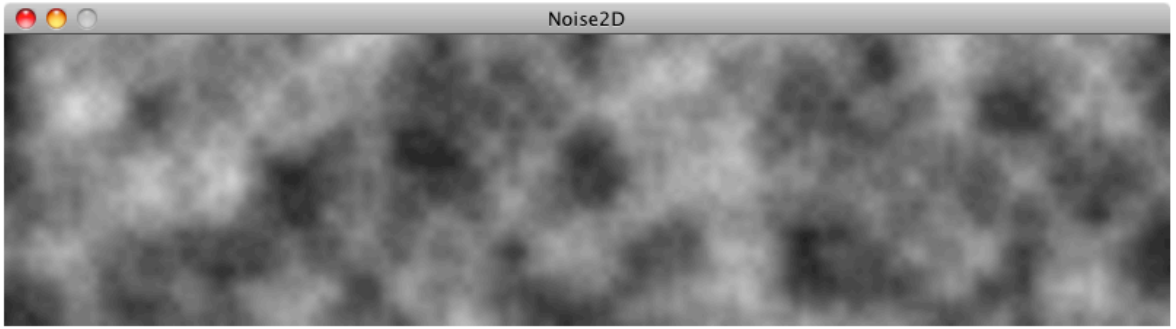


Figure I.11: 2D Noise

If you were to visualize this graph paper with each value mapped to the brightness of a color, you would get something that looks like clouds. White sits next to light gray, which sits next to gray, which sits next to dark gray, which sits next to black, which sits next to dark gray, etc.



This is why noise was originally invented. You tweak the parameters a bit or play with color to make the resulting image look more like marble or wood or any other organic texture.