MUX104
Synthèse d'image et réalité virtuelle

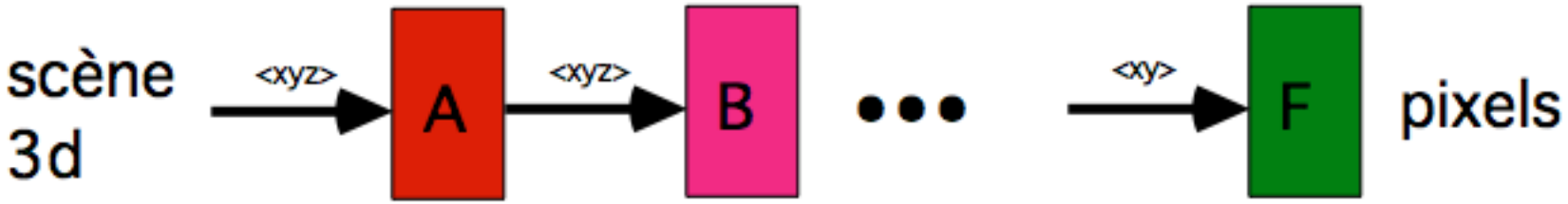
Synthèse procédurale
hasard simulé

Pierre Cubaud
cubaud @ cnam.fr

le cnam

mars 2020

graphisme procédural = ce qui ne peut pas être réalisé seulement par le pipeline standard de rendu



A- transformation des coordonnées locales en coordonnées de la scène (ou du"monde")

B- transformation en volume de vue canonique (cube 0-1)

C- élimination des objets hors volume de vue et découpe (clipping) des objets à la frontière

D- élimination des faces cachées

E- projection en 2D

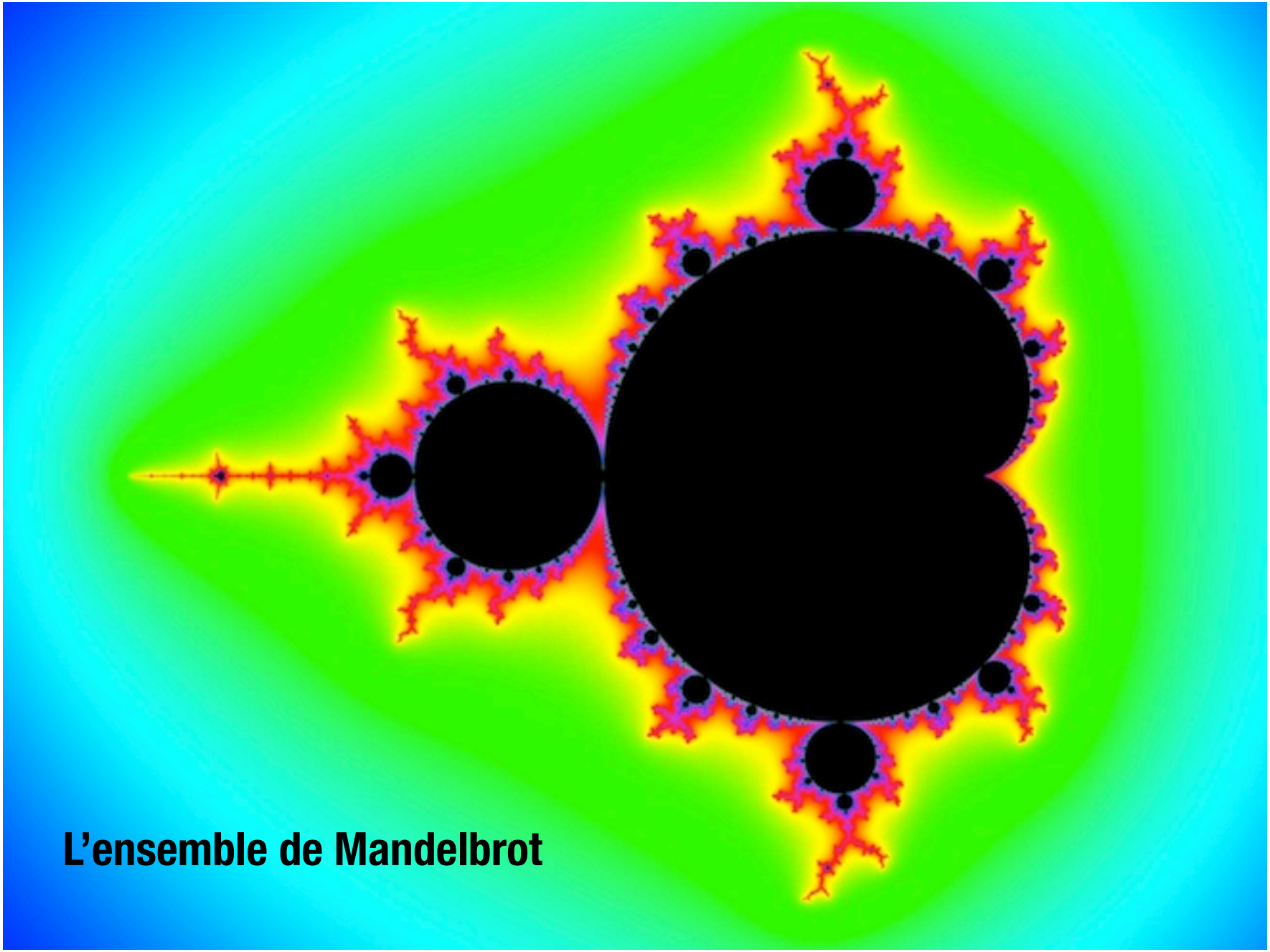
F- coloriage des faces

avec les GPU, ce n'est plus le cas => l'animation temps réel devient possible (et donc IHM)

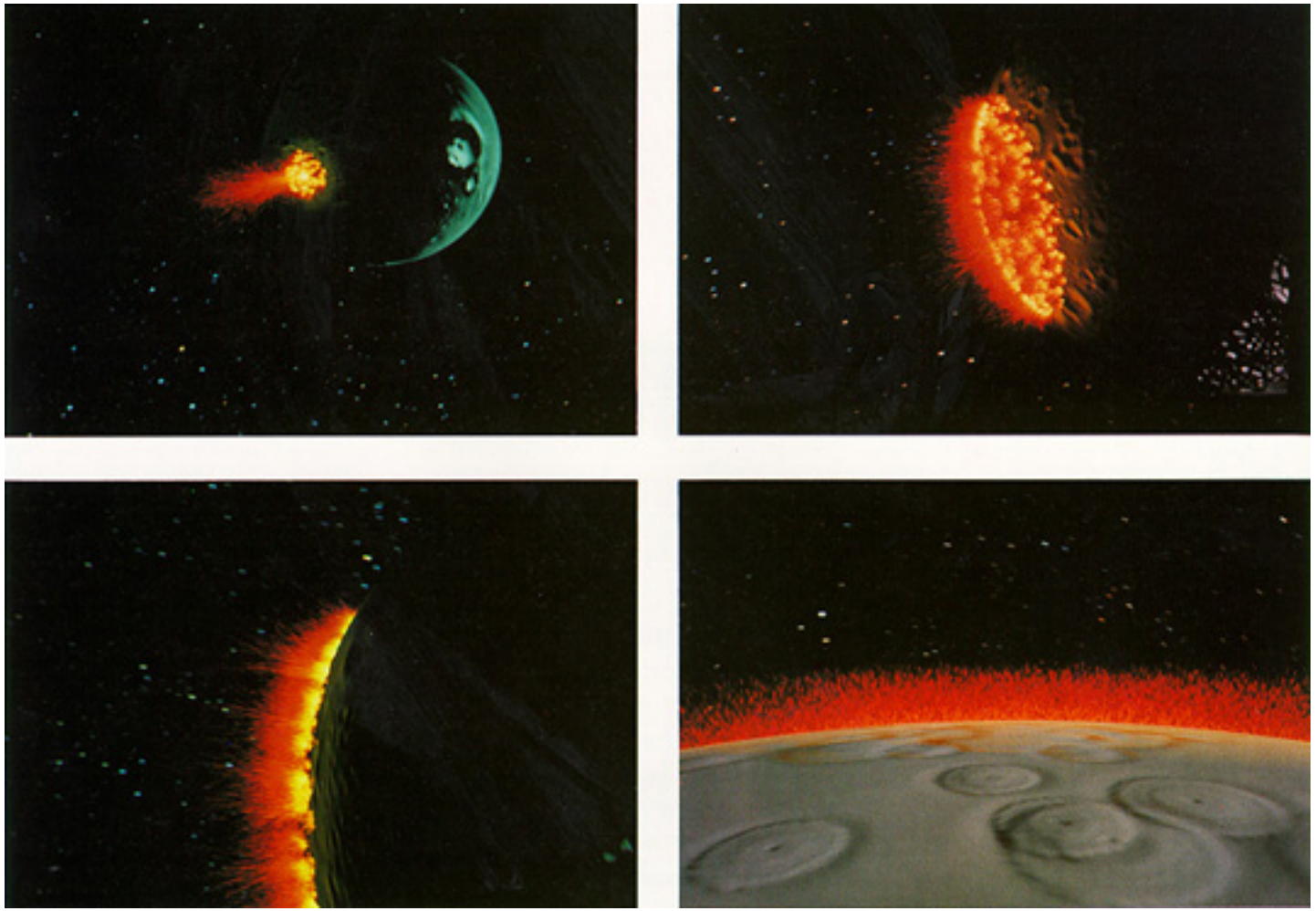
nombreuses techniques, basées sur le hasard

quelle différence entre "procédural" et "génératif" ??? connaissance a priori du résultat ?

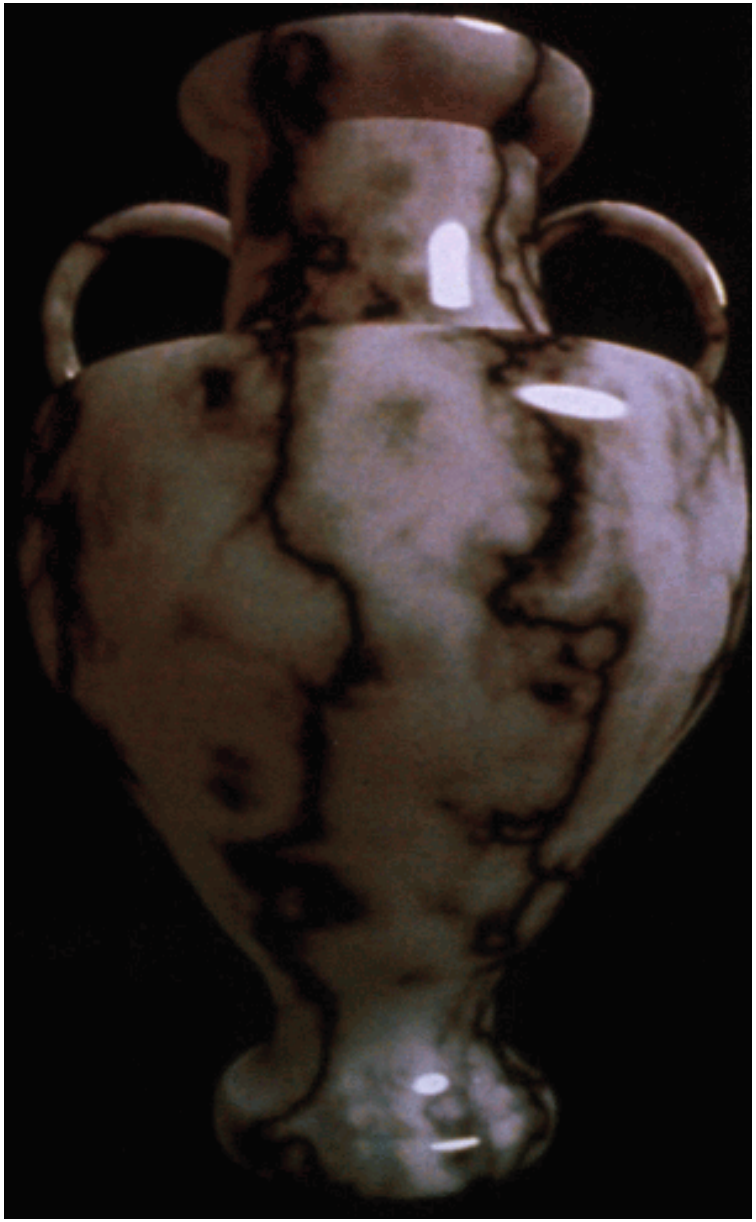
B. Mandelbrot
1982



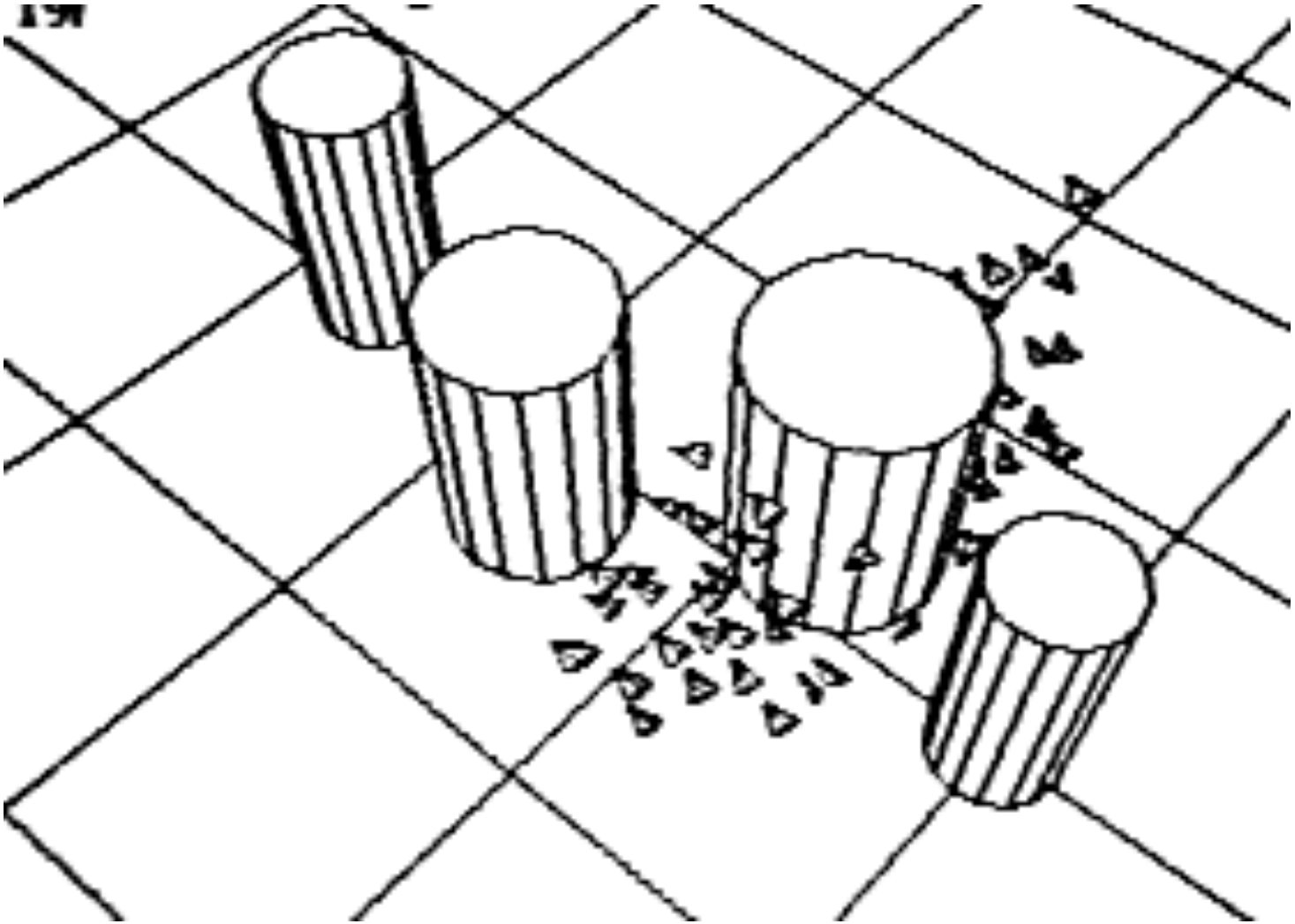
L’ensemble de Mandelbrot



Particules de Reeves (film Star Trek et SIGGRAPH 1983)



Ken Perlin
SIGGRAPH 84-5



« Boids » C. Reynolds, SIGGRAPH’87

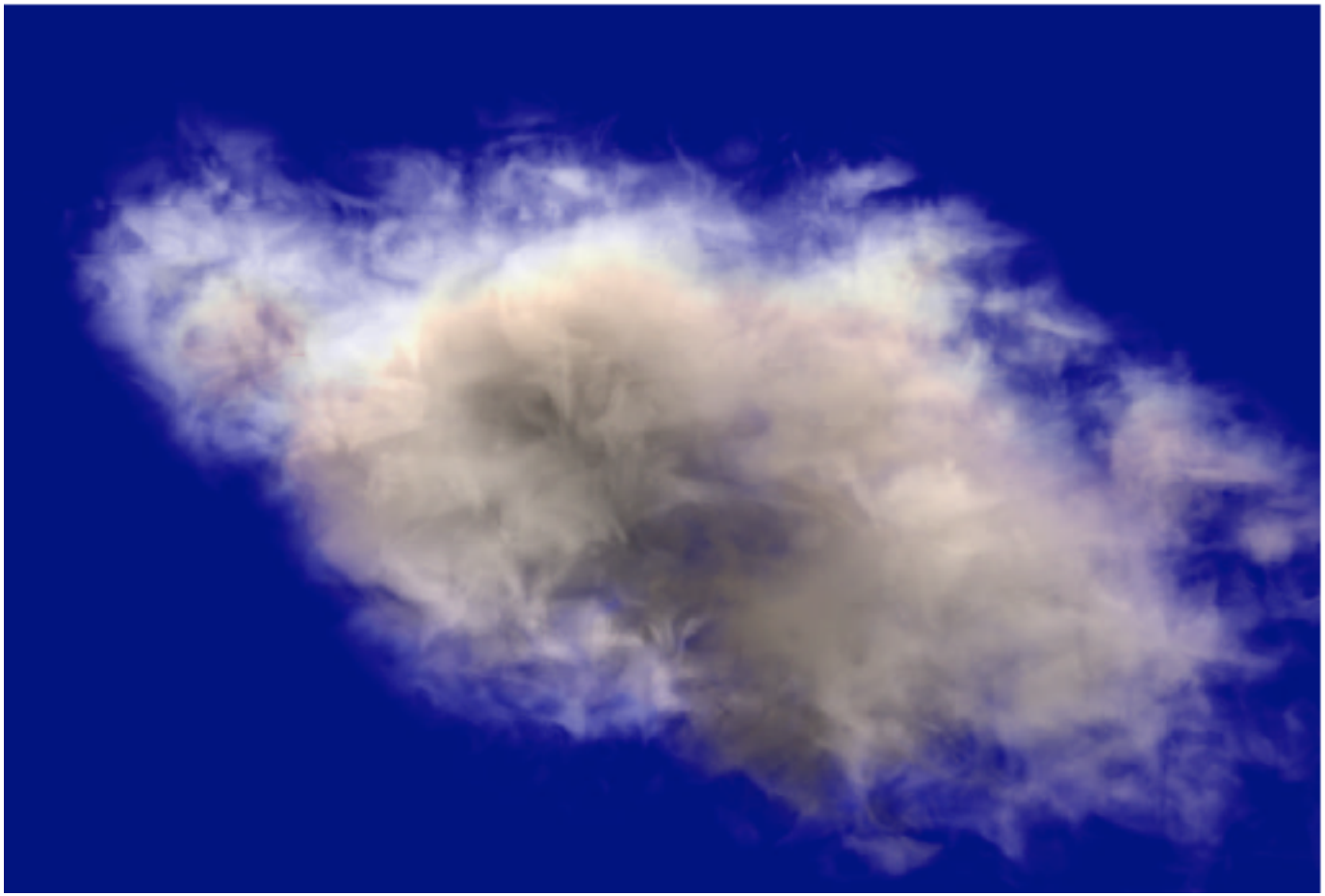
Karl Sims - Panspermia - Prix Ars Electronica, 1991

Musgrave, "Other state", 1992

Les algoristes (Siggraph, 1995)

```
if (creation && object of art && algorithm && one's own algorithm) {
include * an algorist *
}
elseif
(!creation || !object of art || !algorithm || !one's own algorithm)
{
exclude * not an algorist *
}
```

http://www.verostko.com/algorist.html



Ebert, 1995 (?)

P. Muller et al. SIGGRAPH 2001

Figure 18. Somewhere in a virtual Manhattan.

Procedural Modeling of Buildings

Pascal Müller* ETH Zürich

Peter Wonka† Arizona State University

Simon Haegler* ETH Zürich

Andreas Ulmer* ETH Zürich

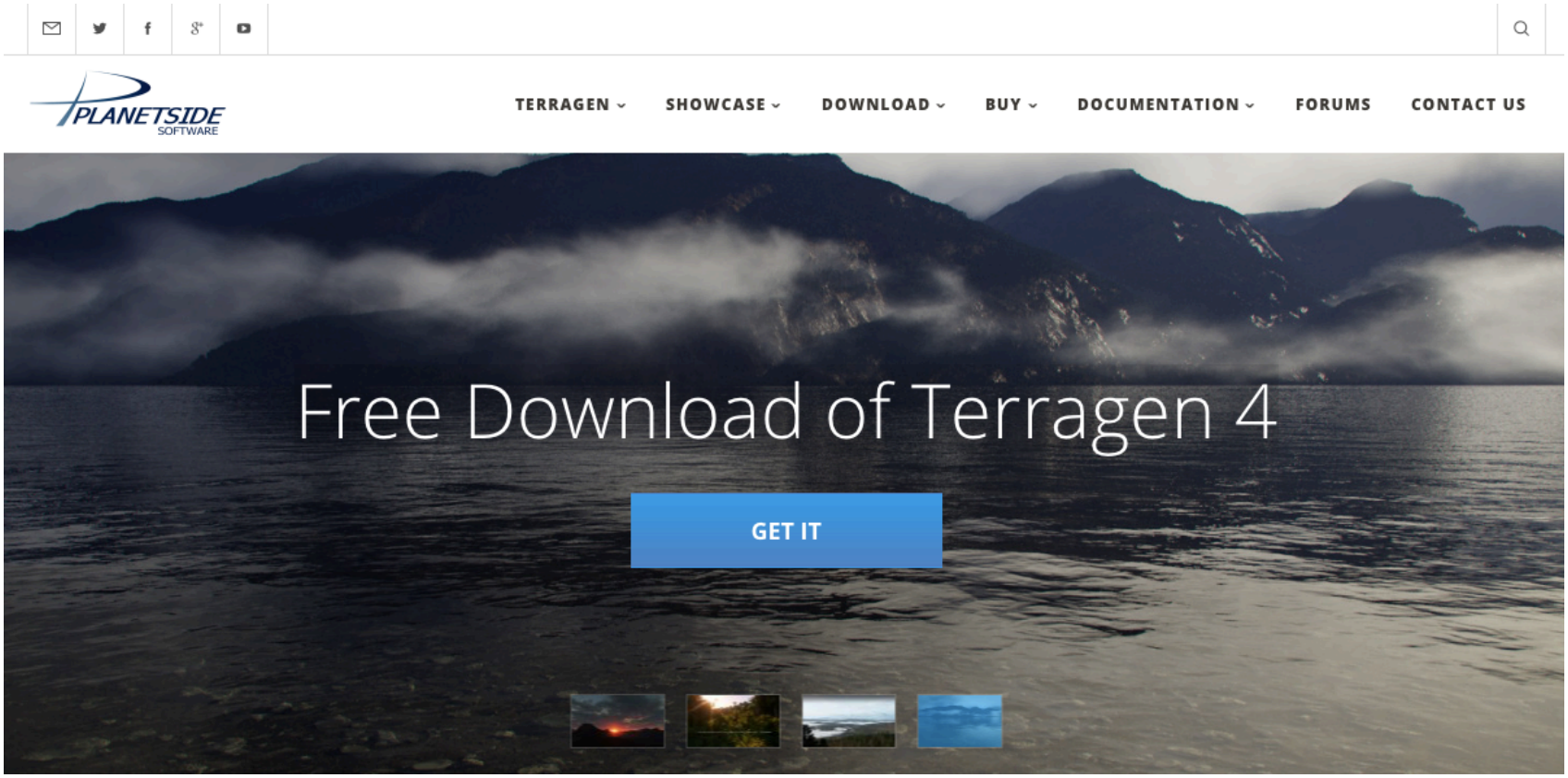
Luc Van Gool* ETH Zürich / K.U. Leuven

Figure 1: This figure shows the application of *CGA shape*, a novel shape grammar for the procedural modeling of computer graphics architecture. First, the grammar generates procedural variations of the building mass model using volumetric shapes and then proceeds to create façade detail consistent with the mass model. Context sensitive rules ensure that entities like windows or doors do not intersect with other walls, that doors give out on terraces or the street level, that terraces are bounded by railings, etc.

Même équipe, SIGGRAPH 2006

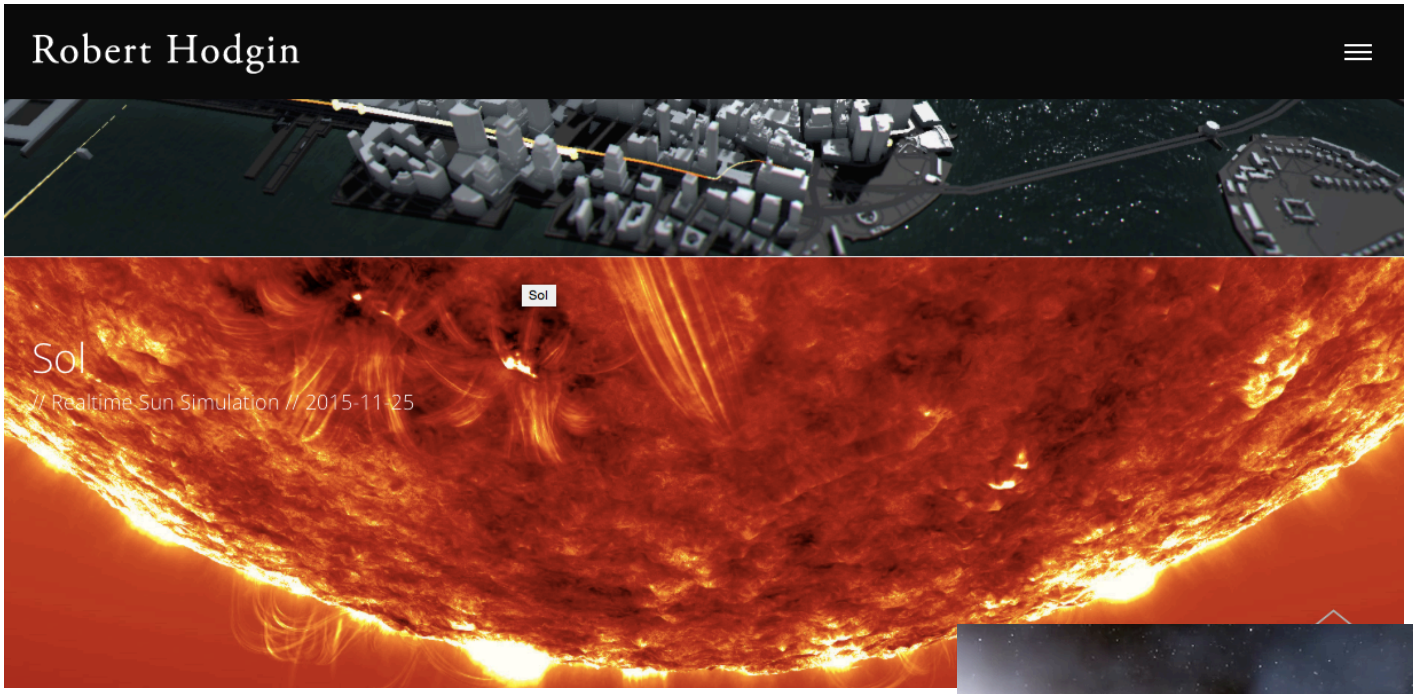
Based on real building footprints, the city was generated with 190 manually written *CGA shape* rules. Hence, the whole model is a rule-based composition of 36 terminal objects (plus 4 tree types and the environment)

Daniel White, 2009
http://www.skytopia.com/project/fractal/mandelbulb.html



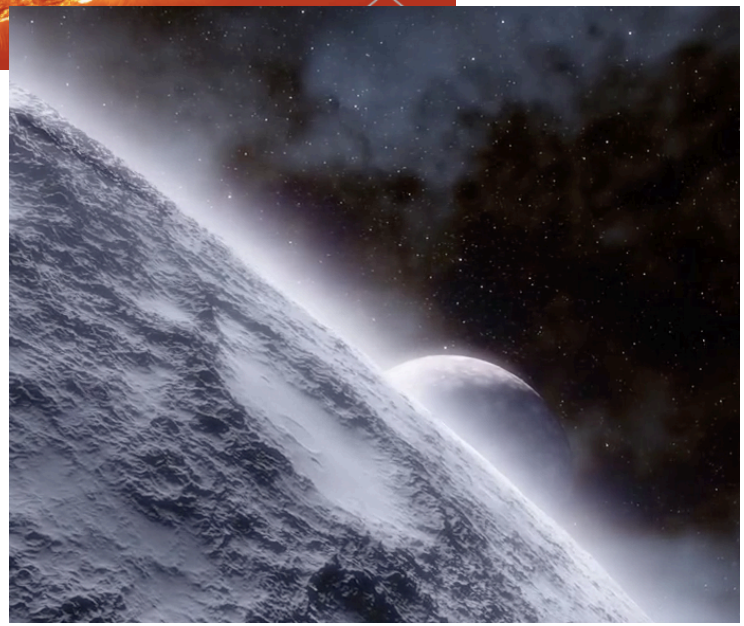
Terragen (planetside.co.uk)

http://www.planetside.co.uk/gallery/f/tg2/buzzzz-mt-copy-3_resize.jpg.html

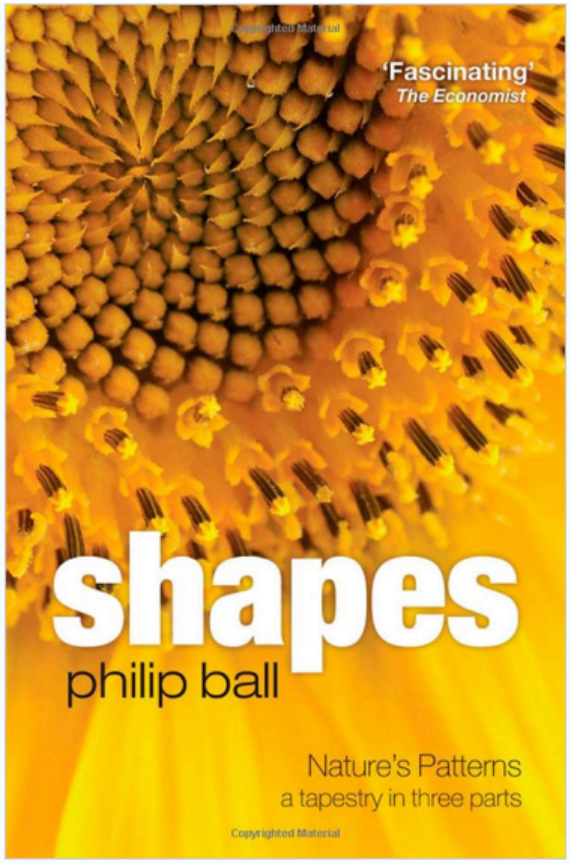


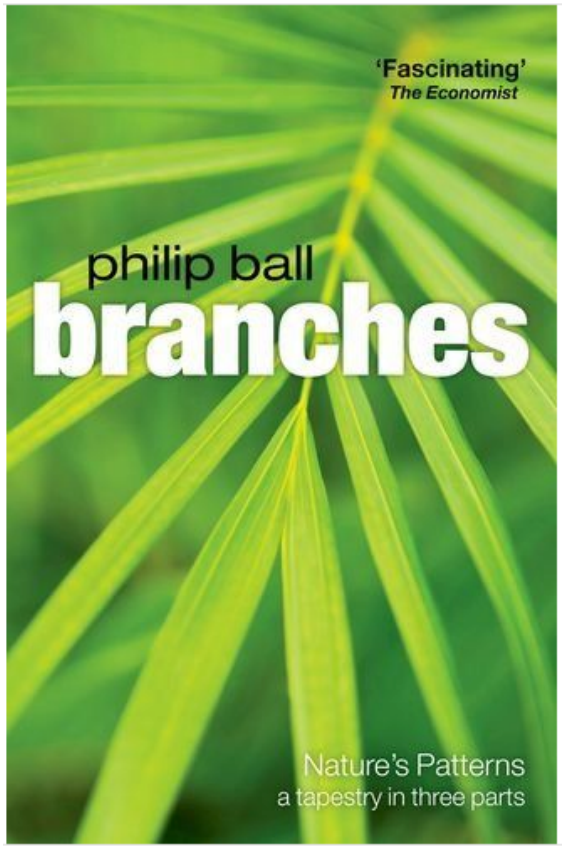
roberthodgin.com

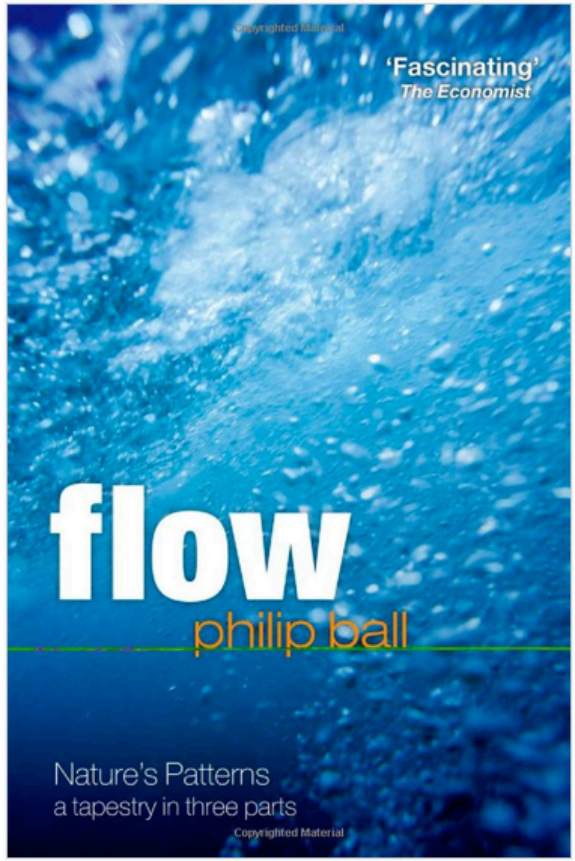
+ spaceengine.org



Bibliographie







THE NATURE OF CODE

DANIEL SHIFFMAN

How can we capture the unpredictable evolutionary and emergent properties of nature in software? How can understanding the mathematical principles behind our physical world help us to create digital worlds? This book focuses on the programming strategies and techniques behind computer simulations of natural systems using Processing.

Buy **The Nature of Code** eBook Bundle

Download the entire book directly from the author.

- DRM-free
- includes PDF, ePub, and Kindle formats
- Delivered by email, no login necessary
- Receive lifetime book updates for free

Name Your Price:

$10.00

Donate a percentage to the Processing Foundation.

10%

Amount to Processing: $1.00
Amount to Author: $9.00

Buy Now

Get the print edition!

CreateSpace | Amazon

Problem with the quality of your print?

Read the Entire Book Online for Free

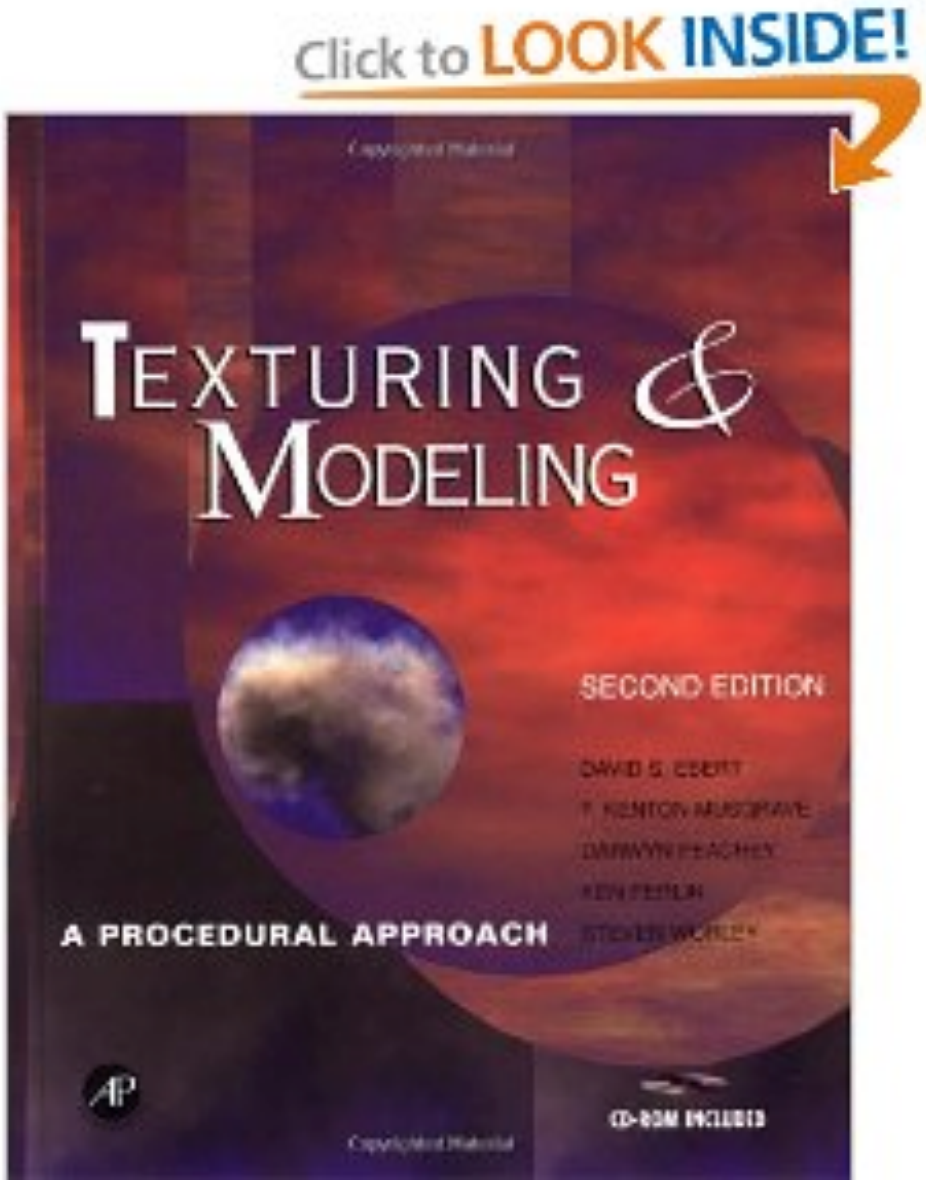
The complete book is available as HTML with interactive Processing.js examples.

Read Now

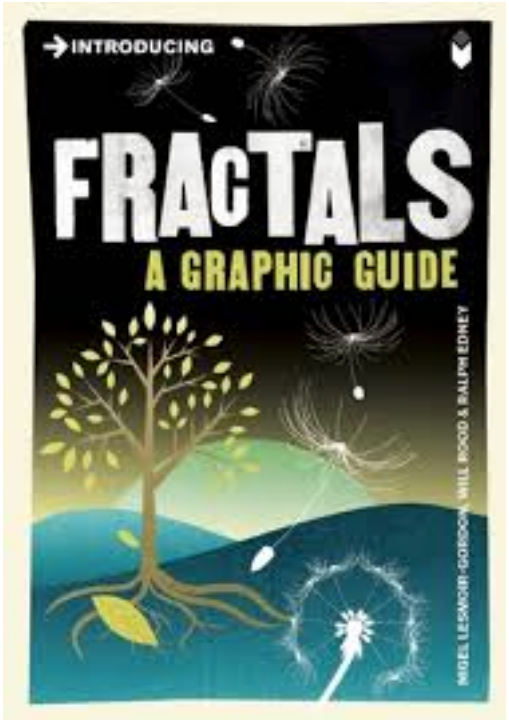
Download the book's code

All of the source files for building the book and the Processing code examples are available on github.

Download Code

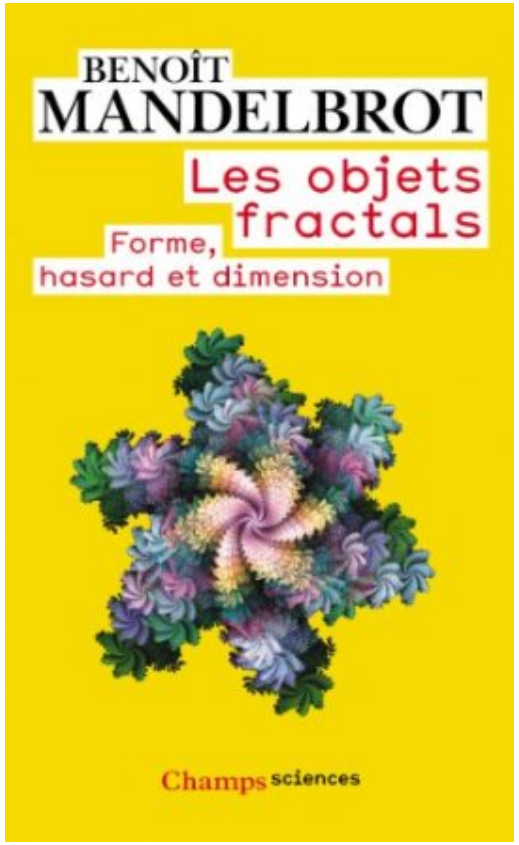


du code, mais pas toujours clair



très sympa
$10

et bien sûr


BENOÎT
MANDELBROT
Les objets fractals
Forme, hasard et dimension
Champs sciences


BENOÎT
MANDELBROT
Fractales, hasard et finance
Champs sciences

2. Hasard simulé

☛ Delahaye *Logique, informatique et paradoxes*
Belin 1995 (chap. 4)

Qu'est-ce qu'un nombre aléatoire ?

On veut produire une suite de "0" et de "1" de manière imprévisible, c'est à dire en éliminant toute répétitivité "suspecte" (ex: la séquence 010101…)

On veut également que les proportions de "0" et de "1" soient égales à long terme (loi des grands nombres) et que leur variation suive la loi du logarithme itéré.

Critères insuffisant : ex. de π

Complexité de Kolmogorov & Chaitin

On defini la complexité K(n) d'une suite de chiffres de longueur n comme la longueur du programme nécessaire pour le calculer (l'afficher).

Exemples :

- Pour la suite 1111…, le programme est

`boucle afficher("1") finboucle` => K petit

- Pour π et e, le programme existe et K est un peu plus grand

- Pour une séquence aléatoire, il n'existe pas de programme plus court que celui qui prend les chiffres un à un et les affiche.

$$\exists c \mid \forall n \;\; K(n) > n - c$$

Une séquence aléatoire est incomprésible

Calculer du pseudo hasard

Méthodes congruentielles (Lehmer, 1951)

$$U_k = (aU_{k-1}) \text{ modulo } m$$

Trois paramètres :

U_0 : le germe ("seed")

m : le diviseur (terme de congruence)

a : le multiplicateur

Exemple :

$$U_k = 3U_{k-1} \bmod 31$$

période = 31 car 31=2^5-1

U0 = 15 (premier avec 31)
U1 = 3*15 mod 31 = 45 mod 31 = 14
U2 = 3*14 mod 31 = 11
U3 = 3*11 mod 31 = 2
U4 = 3*2 mod 31 = 6
…

U30 = 15
U31 = 14 = U1 la période est atteinte

Autre germe :

U0 = 10 (premier avec 31)
U1 = 3*10 mod 31 = 30
U2 = 3*30 mod 31 = 28
etc…

Un bon générateur (Park & Miller, 1988) :

On choisi a = 7^5 = 16807 et m = 2^31-1

Si entiers sur 32 bits : overflow

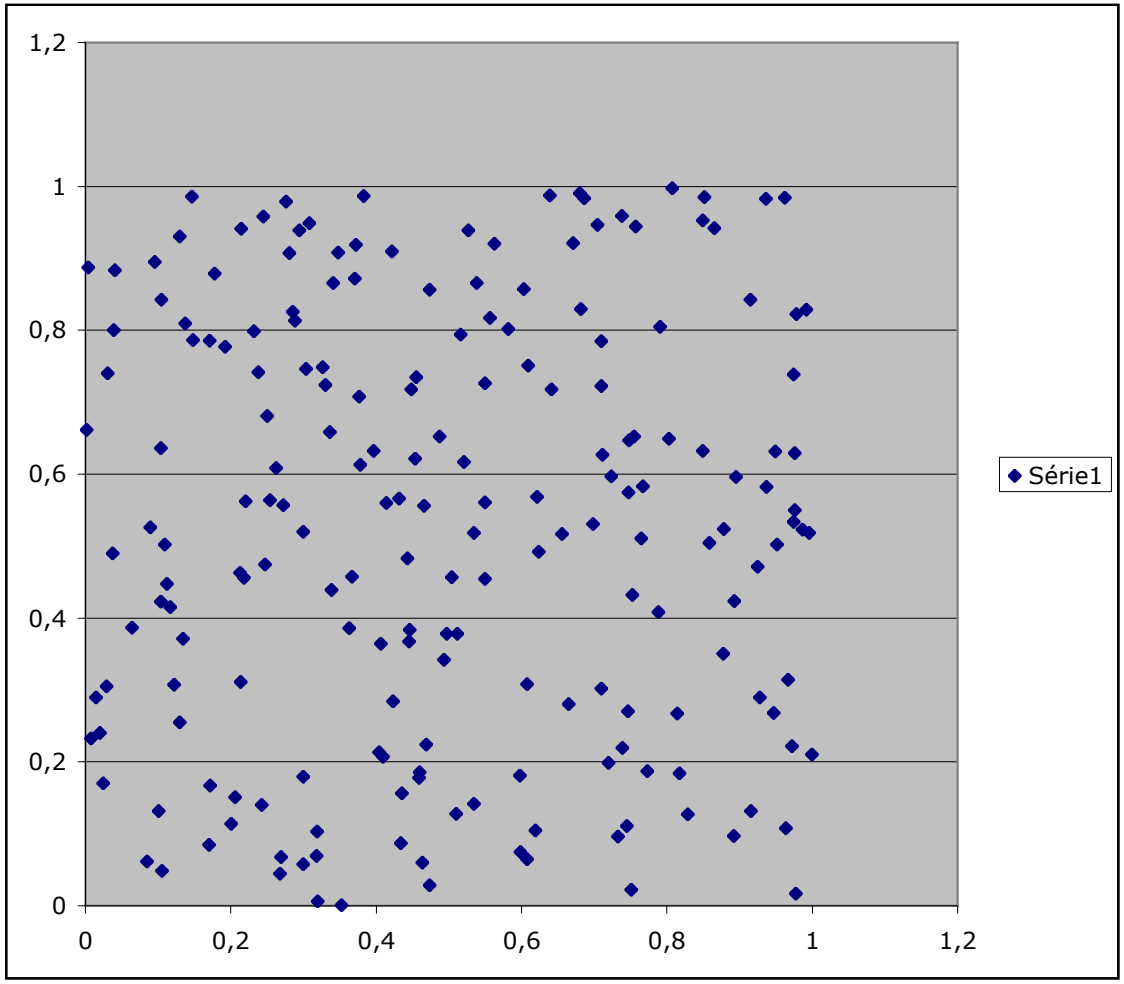
```
function random(var u:integer):real
const
      a=16807;
      m=2147483647;
      q=127773; (* m div a *)
      r=2836;   (* m mod a *)
var
      nu:integer;
begin
      nu:= a*(u mod q) - r*(u div q);
      if nu>0 then u:=nu else u:=nu+m;
      random:=u/m;
end;
```

☛ P. L'Ecuyer, Comm. ACM, vol 31, n°6, 1988

```
#include <stdlib.h>
#include <math.h>
int s1=354675;  /* germes */
int s2=78364;   /*du generateur aleatoire */

double Uniform() {
int Z;
div_t k; /* type predefini dans math.h */

  k= div(s1,53668);
  s1= 40014*(s1-k.quot*53668)-k.quot*12211;
  if (s1<0) {s1+=2147483563;}

  k= div(s2,52774);
  s2= 40692*(s2-k.quot*52774)-k.quot*3791;
  if (s2<0) {s2+=2147483399;}

  Z=s1-s2;
  if (Z<1) {Z+=2147483562;}
  return Z*4.656613E-10;
}
```

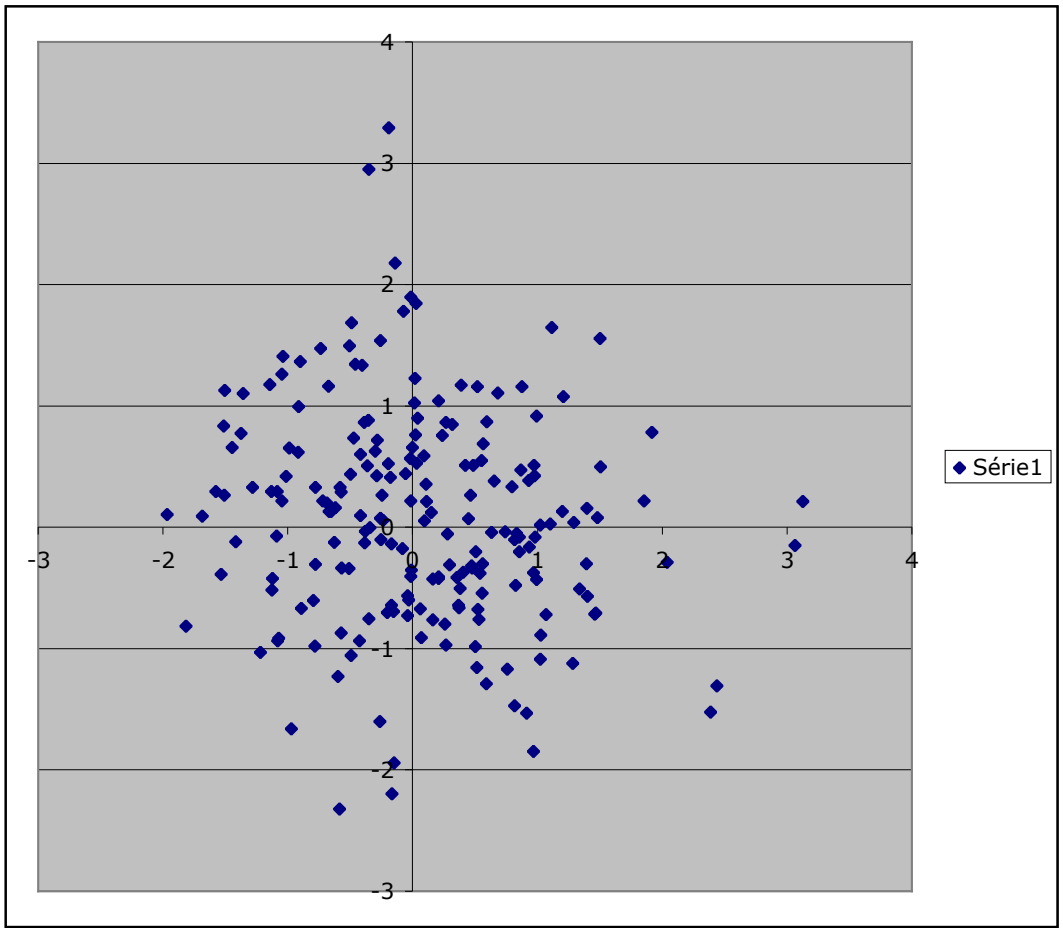
Période = 2.3*10^18

Répartition uniforme des points U[0,1]



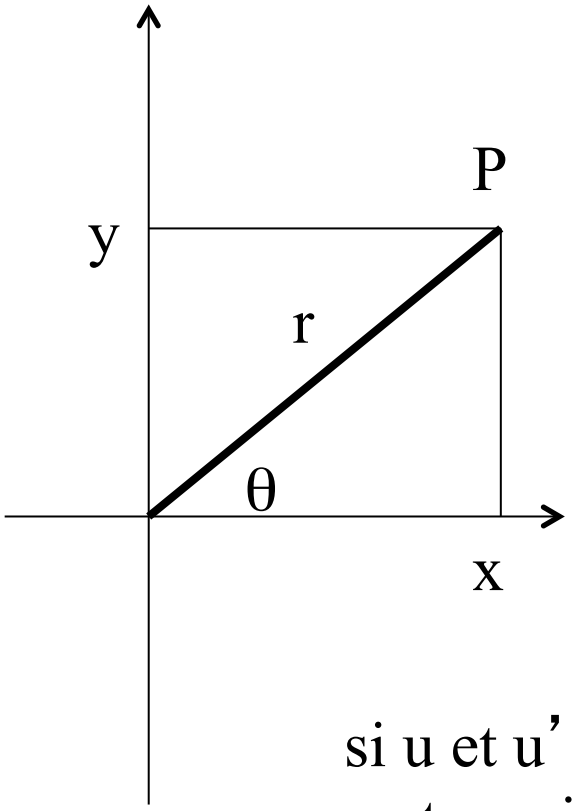
C'est la fonction `random()` de base
moyenne = 0.5 écart-type = 0.29

Répartition normale des points N(0,1)



moyenne = 0, écart-type = 1
on a N(m,s) = m + s*N(0,1)

Simulation d'une loi Normale N(0,1) : ex. de la méthode pôlaire



$$\theta = 2\pi * u$$

$$r = \sqrt{-2\log(u')}$$

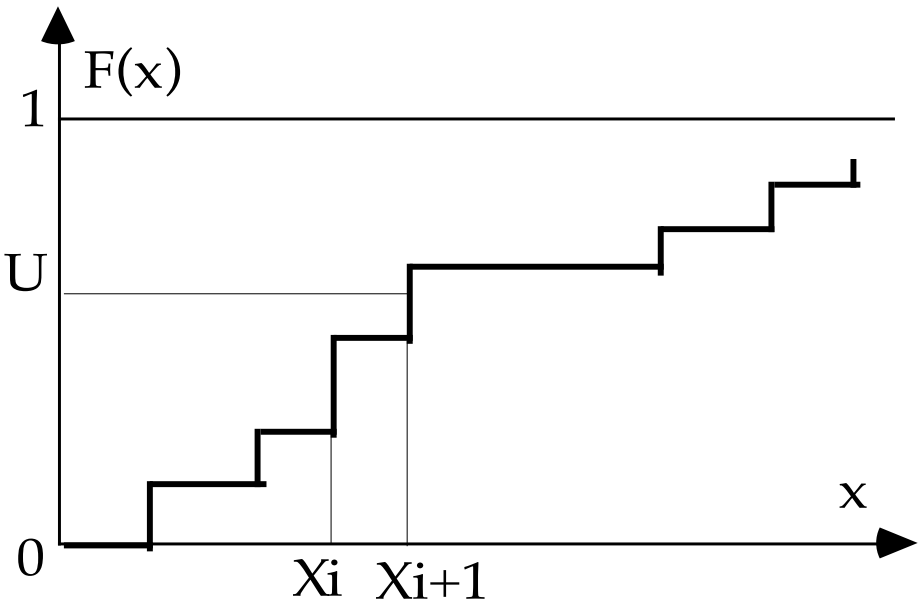
$$x = r\cos\theta$$

$$y = r\sin\theta$$

si u et u' suivent une loi uniforme U[0,1]
x et y suivent une loi normale N(0,1)

Génération par la loi inverse

On génère un nombre U, uniformément distribué entre 0 et 1.
On calcule ensuite $X = F^{-1}(U)$



cas d'une loi discretisée

cas loi uniforme :

$$F(x) = \frac{x-a}{b-a} = U$$

$$x = F^{-1}(U) = a + U(b-a)$$

cas loi exponentielle

$$F(x) = 1 - e^{-\lambda x} = U$$

$$X = -\frac{1}{\lambda}\log U$$

La loi de Poisson

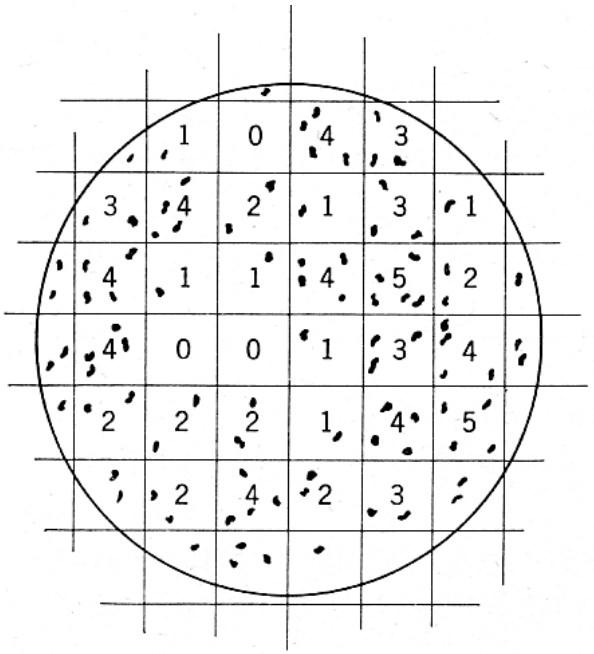


Figure 1. Bacteria on a Petri plate.

$$P[X = k] = p(k,\lambda) = e^{-\lambda}\frac{\lambda^k}{k!}$$

$$EX = \lambda \quad \sigma_X^2 = \lambda$$

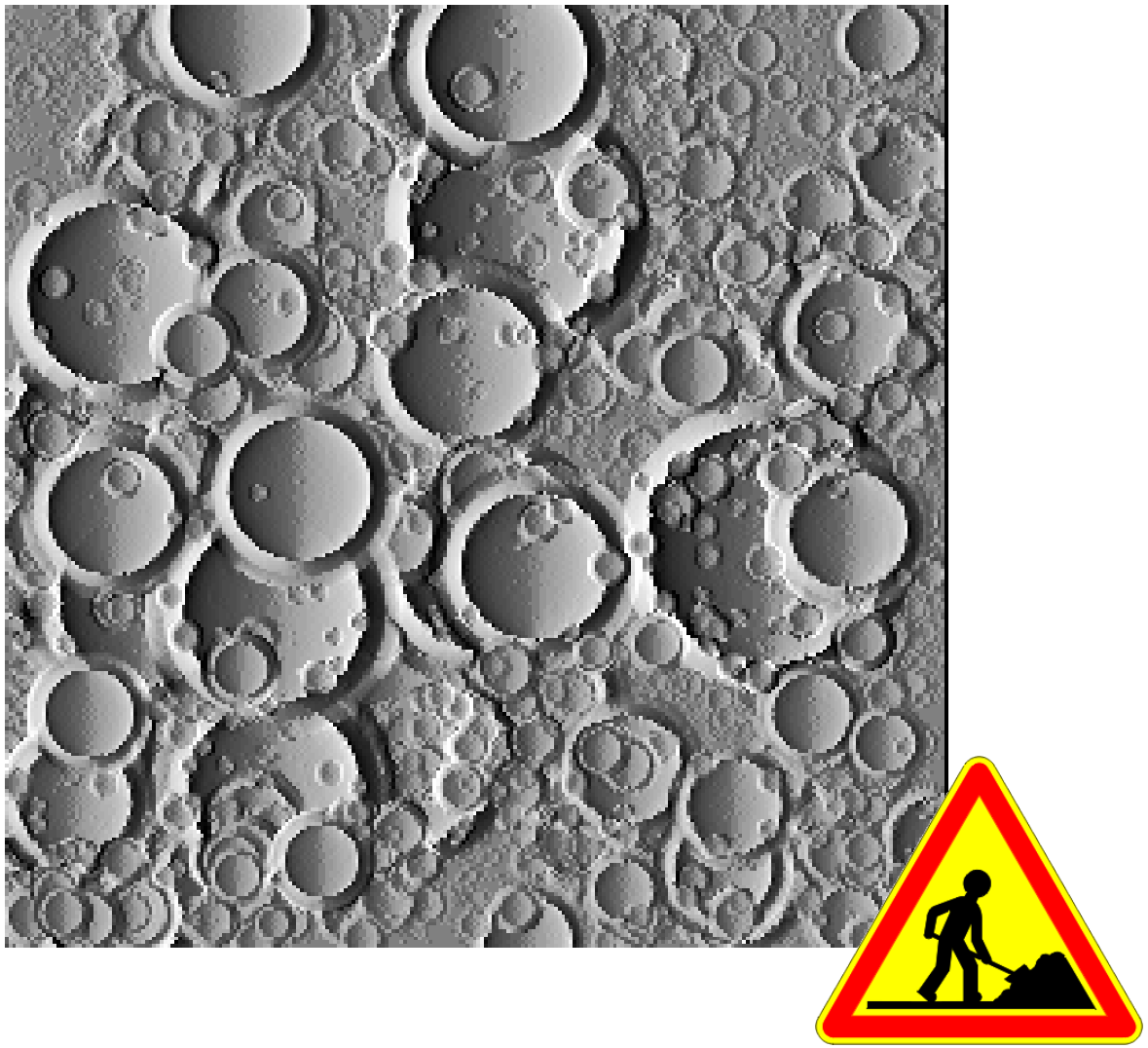
```
w:=1;k:=0
repeter
  u:=random(); w:=w*u; k++;
jusqu'a w<=exp(-lambda)
retourner k
```

Feller - Proba. vol.1

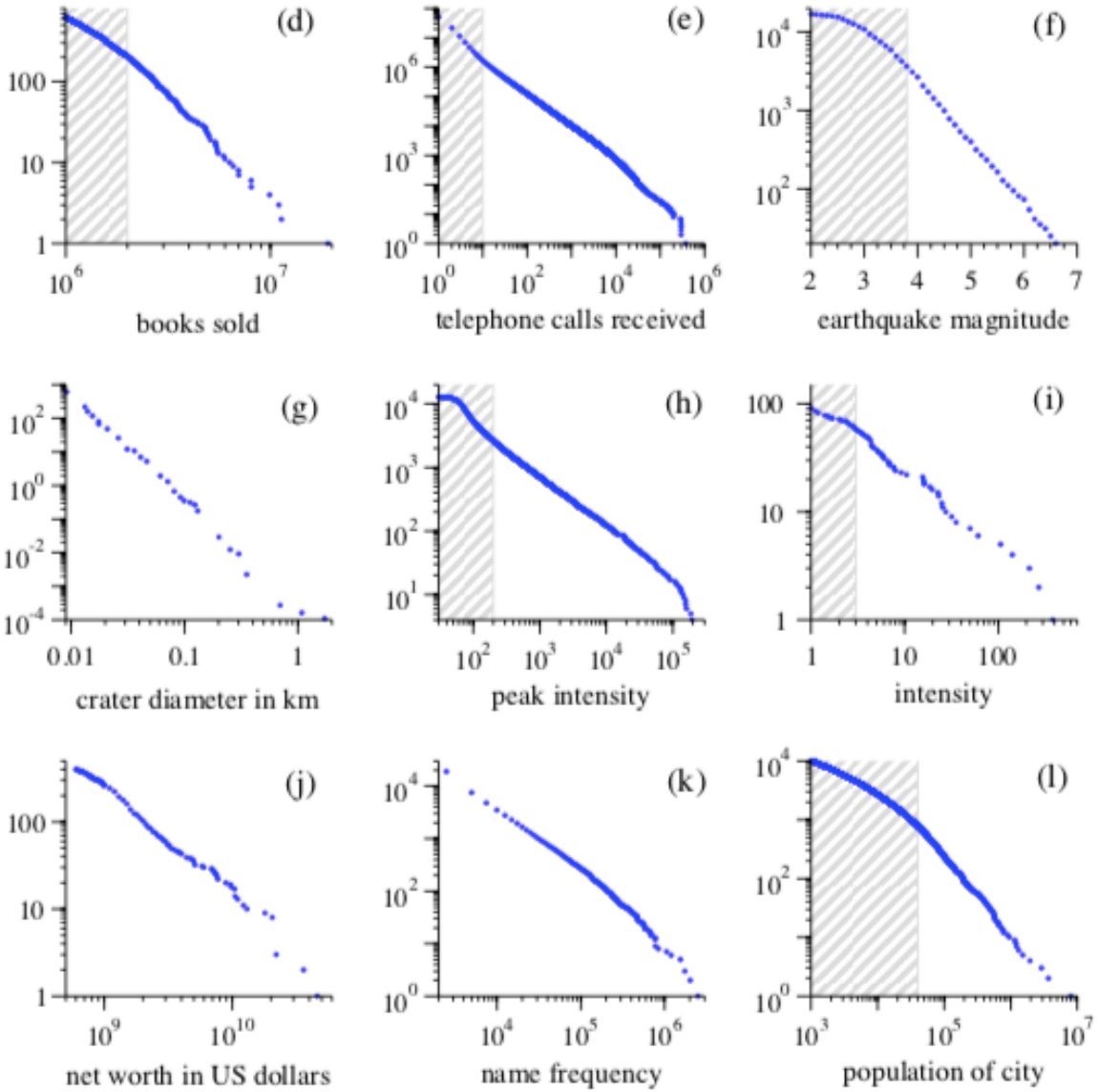
TABLE 4
EXAMPLE (*b*): FLYING-BOMB HITS ON LONDON

k	0	1	2	3	4	5 and over
N_k	229	211	93	35	7	1
$Np(k; 0.9323)$	226.74	211.39	98.54	30.62	7.14	1.57

Cas des cratères (pgmcrater dans netpbm)



La loi Puissance



http://www.piketty.pse.ens.fr/files/powerlaws80-20rule.pdf

génération de la loi Puissance(x0,x1,n) :

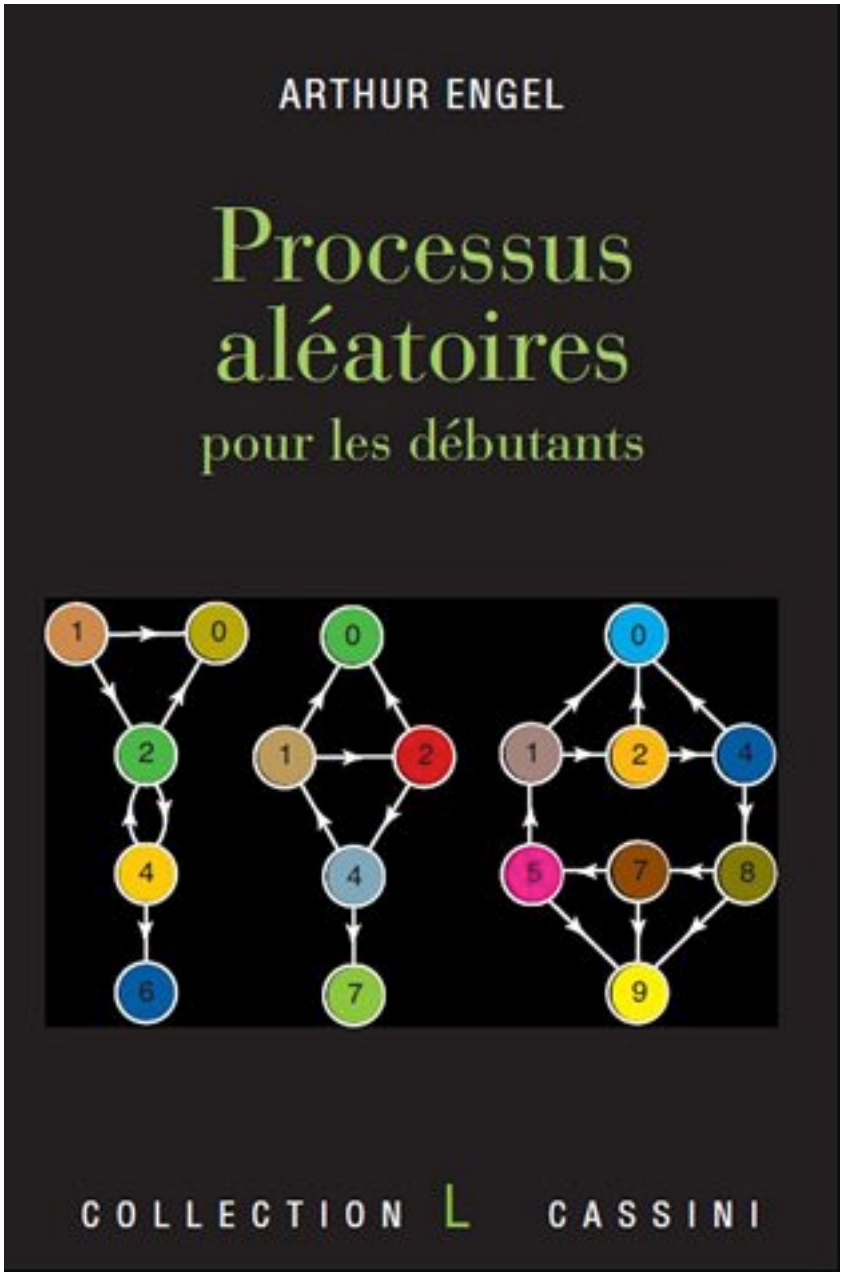
$$P(x) = C\,x^n \text{ for } x \in [x_0, x_1]$$

avec $$C = \frac{n+1}{x_1^{n+1} - x_0^{n+1}}.$$

On tire y = random(0,1) et renvoi de X loi Puissance avec :

$$X = \left(\frac{n+1}{C}\, y + x_0^{n+1} \right)^{1/(n+1)}$$
$$= \left[\left(x_1^{n+1} - x_0^{n+1} \right) y + x_0^{n+1} \right]^{1/(n+1)}$$

https://mathworld.wolfram.com/RandomNumber.html



A LIRE

15 euros
super clair

- chaines de Markov
- pas de simulation