Arnaud Bodin
Université de Lille 1

Capes
2009-2010

Ensemble de Mandelbrot

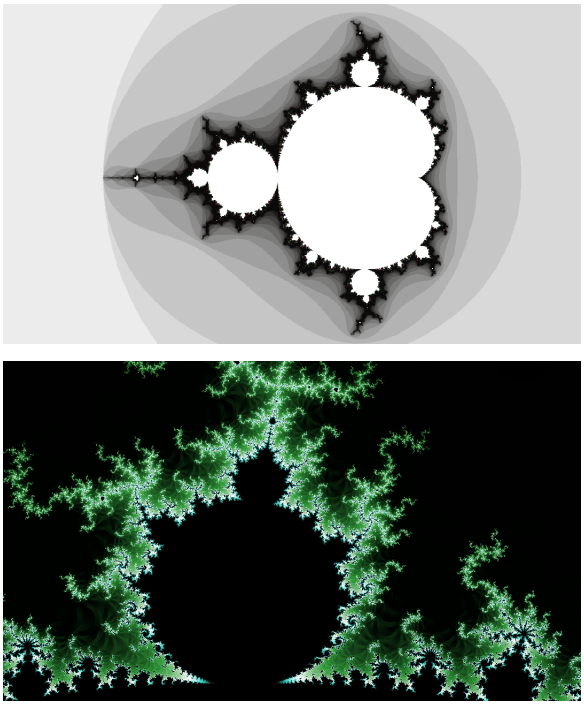
Il s'agit ici de donner un petit aperçu de l'ensemble de Mandelbrot qui est un objet fractal ; il est l'objet de recherches actives depuis une trentaine d'années.
L'objet est très simple à définir. Fixons une constante $c \in \mathbb{C}$. À partir de cette constante nous définissons une suite $(z_n(c))_{n\in\mathbb{N}}$ par récurrence :

$$\begin{cases} z_0(c) = 0 \\ z_{n+1}(c) = z_n(c)^2 + c \end{cases}$$

Il est plus facile de définir l'ensemble de Mandelbrot $\mathcal{M} \subset \mathbb{C}$ par son complémentaire :

$$\mathbb{C} \setminus \mathcal{M} := \left\{ c \in \mathbb{C} \mid \lim_{n\to+\infty} |z_n(c)| = +\infty \right\}$$

Donc l'ensemble des valeurs $c \in \mathbb{C}$ pour lesquels la suite des modules de $z_n(c)$ ne tend pas vers l'infini est l'*ensemble de Mandelbrot*.



En haut l'ensemble de Mandelbrot (en blanc), en bas un zoom d'une partie de l'ensemble (en noir). Les différentes couleurs sont là surtout pour faire joli et correspondent à des vitesses de divergence différentes.

Commentaires généraux :
- C'est un ensemble d'une immense complexité ! Il a une structure *fractale*, c'est-à-dire que l'image globale se retrouve en plus petit dans certaines parties de l'ensemble.
- Vous trouverez sur le web des centaines d'images toutes plus belles les unes que les autres, mais un grand plaisir est d'explorer soit même l'ensemble à l'aide du logiciel adéquat[1] et de découvrir des régions inexplorées[2].

Commentaires "spécial Capes" :
- Il s'agit ici de donné l'énoncé d'un lemme qui permet de détecter des points qui ne sont pas dans l'ensemble de Mandelbrot.
- Ce lemme est appliqué à l'écriture d'un algorithme pour dessiner l'ensemble de Mandelbrot. Je donne l'algorithme sous une forme universelle et pour la *TI Voyage 200*.
- On peut aussi dire que la cardioïde d'équation $\rho(\theta) = \frac{1}{2}(1-\cos\theta)$ apparaît sur la figure.

Lemme.

1. *Si* $|c| > 2$ *alors* $|z_n(c)| \to +\infty$ *(donc* $c \notin \mathcal{M}$*).*
2. *Fixons* $c \in \mathbb{C}$*, s'il existe* $n_0 \in \mathbb{N}$ *tel que* $|z_{n_0}(c)| > 2$ *alors* $|z_n(c)| \to +\infty$ *(donc* $c \notin \mathcal{M}$*).*

Bien sûr la réciproque de (2) est vraie par définition : si $c \notin \mathcal{M}$ alors il existe n_0 tel que $|z_{n_0}(c)| > 2$.

Démonstration. Commençons par la partie (1) : soit donc $|c| > 2$. Pour simplifier l'écriture nous notons pour tout $n \geq 0$, $z_n = z_n(c)$. Nous allons montrer que nous avons pour tout $n \geq 1$:

$$(\mathcal{H}_n) : \qquad |z_n| > |c|^n.$$

Comme $|c| > 2$ la suite divergera. Nous allons raisonner par récurrence.

Initialisation. Pour $n = 1$, $z_1 = c$ donc $\mathcal{H}_1$ est vraie.

[1] Xaos, Fractint,...

[2] Certaines personnes gardent même secrètes les coordonnées des images qu'elles obtiennent !

Hérédité. Soit $n \geq 1$. Supposons que $\mathcal{H}_n$ soit vraie. Alors

$$\begin{aligned}
|z_{n+1}| &= \left|z_n^2 + c\right| \\
&\geq \left||z_n^2| - |c|\right| \qquad \text{c'est la seconde inégalité triangulaire} \\
&\geq \left|(|c|^n)^2 - |c|\right| \qquad \text{par l'hypothèse de récurrence } \mathcal{H}_n \\
&= |c|^{n+1}\left||c|^{n-1} - \frac{1}{|c|^n}\right| \\
&\geq |c|^{n+1}(|c|^{n-1} - 1) \qquad \text{car } |c| > 2 \\
&\geq |c|^{n+1} \qquad \text{car } |c| > 2
\end{aligned}$$

Donc $\mathcal{H}_{n+1}$ est vraie.

Conclusion. Par le principe de récurrence, pour tout $n \geq 1$, $\mathcal{H}_n$ est vraie.

Pour la partie (2) on procède de la même façon et on montre par récurrence que

$$|z_n(c)| \ \geq 2^{n-n_0+1}, \quad \text{pour } n \geq n_0.$$

□

Appliquons ce lemme pour dessiner l'ensemble de Mandelbrot à l'aide d'un ordinateur.
On note $x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$ les coordonnées de la fenêtre que l'on désire afficher :

$$\{c \in \mathbb{C} \mid x_{\min} \leq \mathrm{Re}(c) \leq x_{\max} \text{ et } y_{\min} \leq \mathrm{Im}(c) \leq y_{\max}\}.$$

On recouvre cette fenêtre par $N \times N$ pixels : Le pixel (i, j) correspond donc à la valeur : $c \in \mathbb{C}$ tel que $\mathrm{Re}(c) = x_{\min} + i\frac{x_{\max}-x_{\min}}{N}$ et $\mathrm{Im}(c) = y_{\min} + j\frac{y_{\max}-y_{\min}}{N}$.
On fait le calcul de la suite $(z_n(c))$ pour chaque c correspondant à un pixel, mais on itère la suite au maximum $k_{\max}$ fois. Si à un moment $|z_n(c)| > 2$ alors par lemme précédent on sait que $c \in \mathcal{M}$ (donc on stoppe le calcul et on affiche le pixel correspondant). On a décidé si le point c est dans l'ensemble de Mandelbrot ou pas après $k_{\max}$ itération (au lieu d'une infinité) : on ne dessine donc qu'un ensemble approché de l'ensemble de Mandelbrot, mais il faut bien arrêter les itérations au bout d'un temps fini.

```
Pour i := 0 jusqu'à N
  Pour j := 0 jusqu'à N
    c := xmin + i(xmax-xmin)/N + I(ymin + i(ymax-ymin)/N)
```

```
    Tant que (k < kmax et abs(z) <= 2) faire
      z := z^2+c;
      k := k+1;
    Fin du "tant que"
    Si (abs(z) <= 2) alors afficher le pixel (i,j)
  Fin du "pour j"
Fin du "pour i"
```

Vous pouvez adapter le principe en écrivant un système de deux équations pour la partie réelle et la partie imaginaire de z_n.
Implémentation : voici le code pour la calculatrice *TI Voyage 200*. Le principe est bien sûr le même que l'algorithme ci-dessus mais il ne faut pas négliger l'aspect pratique d'une écriture de code. Voici ce que cela donne :

```
:mandel()
:Prgm
:ClrDraw:ClrGraph
:-1.5->xmin:1.2->xmax:0->ymin:1.5->ymax
:
:Local xpix,ypix
:232/5 -> xpix:102/5 -> ymax
:
:Local i,j,k,z,c
:
:For i,0,xpix
:For j,0,ypix
:  xmin+i/xpix*(xmax-xmin)+I*(ymin+j/ypix*(ymax-ymin)) -> c
:  0 -> z : 0 -> k
:  While k<8 and abs(z)<=2
:    z^2+c -> z
:    k+1 -> k
:  EndWhile
: If abs(z)>2 Then
:   PxlOn j,i
: EndIf
:EndFor
:EndFor
:EndPrgm
```

Quelques commentaires :

– Ici on affiche les pixels qui ne sont pas dans l'ensemble de Mandelbrot ! Attention cela prend plusieurs minutes pour faire les calculs ! C'est pour cela qu'ici on affiche que la moitié de l'ensemble (avec les valeurs avec partie positive de $\mathrm{Im}(c)$).
– L'écran de la *TI Voyage 200* fait 232×102 pixels, en raison de la longueur des calculs on affiche seulement une zone de $1/5$ des dimensions de l'écran. En plus le pixel de coordonnées (i, j) est sur la ligne i et la colonne j, d'où le `PxlOn j,i` et pas `i,j` (mais en plus on affiche le dessin la tête en bas). Toujours pour des raisons de limites de la calculatrice le nombre d'itérations est (trop) faible : ici $k < 8$, ($k < 100$ serait mieux, mais trop long).
– "`->`" désigne la touche de stockage d'une valeur "sto" et "`I`" désigne le nombre complexe i (en bleu au-dessus de la lettre "i").
– Pour stopper un calcul trop long (ou qui a planté) faite "shift+on" ("shift" est la flèche blanche juste au-dessus de "on").
– Il faut donc voir ce programme plus comme un moyen de démontrer vos capacité de programmation, quand vous serez prof vous pourrez le faire à vos élève sur des ordinateurs. L'algorithme théorique présenté avant est déjà pas mal pour le jour du Capes...