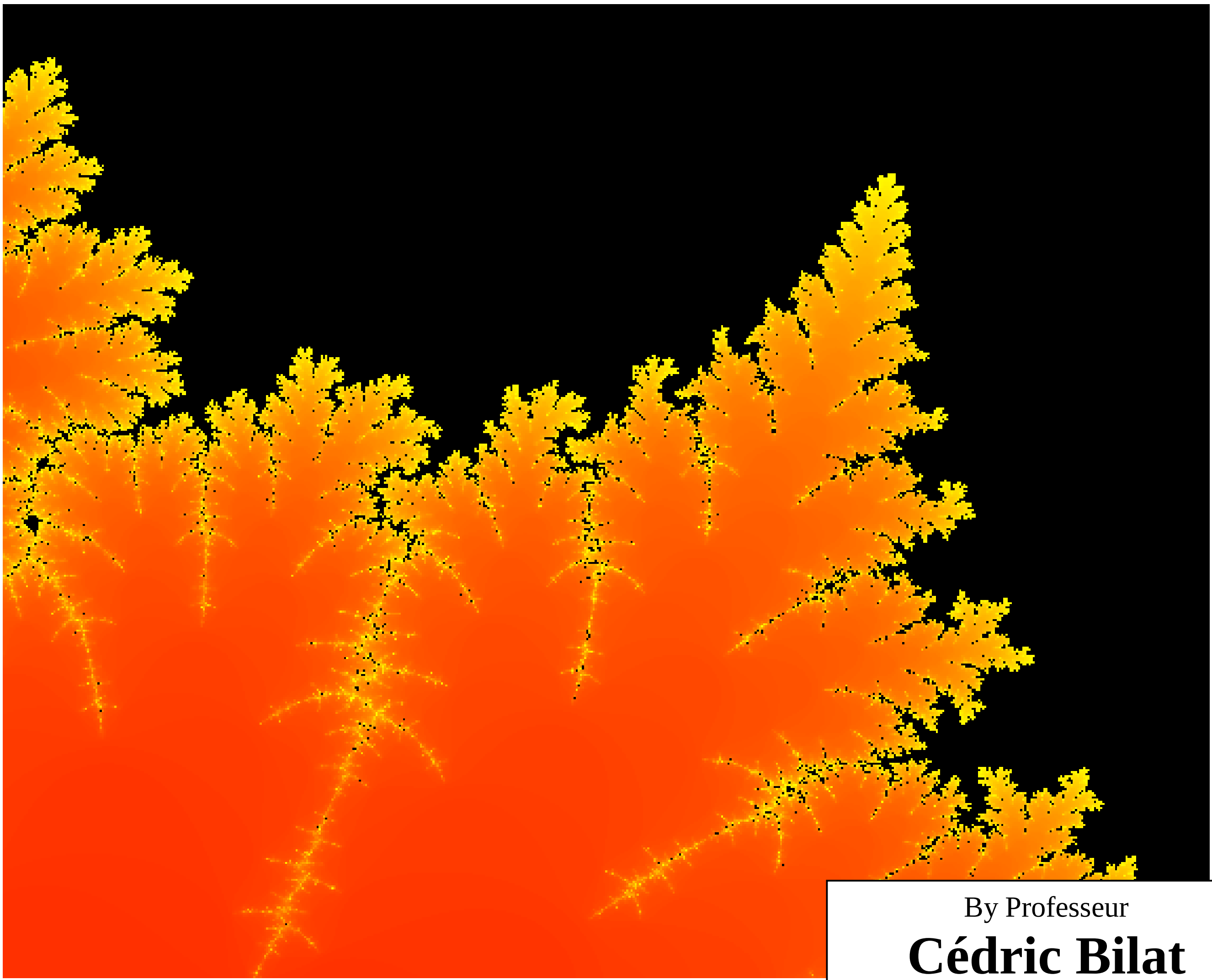
Fractale

MandelBrot & Julia



By Professeur
Cédric Bilat
Cedric.Bilat@he-arc.ch

Contexte

Principe

On s'intéresse ici à des fractales 2D que l'on représentera sous la forme d'une image bitmap. Chaque pixel (x,y) de l'image sera colorié à l'aide d'une couleur *HSB*. La définition du fractale définira la couleur du pixel (x,y).

MandelBrot et Julia :

Chacun de ces deux fractales associe à chaque pixel (x,y) une *suite* $(z_i\,(x,y))_i$. Cette *suite* dépend du pixel (x,y) de l'image à laquelle elle est rattachée. Comme toute *suite* de nombres, cette suite peut converger ou bien au contraire diverger.

La couleur du pixel (x,y) ne dépendra que de cette suite $(z_i\,(x,y))_i$ rattaché au pixel (x,y), et non de la couleur des pixels voisins.

Principe

Soit $N \in \mathbb{N}^*$ un nombre fixé, ie un paramètre. Si la *suite* $(z_i(x,y))_i$ converge vers un nombre fixe, le point (x,y) appartient au fractal et on le colore en noir. Dans la cas contraire, si l'on a pu montrer grâce à un z_s, avec $s \le N$ que la suite va diverger, on colorie le pixel en fonction de s. Par exemple s pourait induire la *hue* de la couleur au *prorata* de sa position dans $[0,N]$.

Définition

Définition

Plus généralement, appelons E_N ***-Fractale*** un tel fractal. Il s'agit d'un ensemble de couples $(x,y)\in\mathbb{R}^2$**,** sous la forme:

$$\boxed{\left\{(x,y)\in\mathbb{R}^2 \,\middle|\, \textit{la suite} \quad z_i=\left\{z_i(x,y)\right\}_{i\in J\subset\mathbb{N}} \quad N-\textit{converge}\right\}\subset\mathbb{R}^2}$$

où N est un paramètre que l'on s'est fixé une fois pour toute.

Critère de N-divergence de la suite $z_i(x,y)$

Pour déterminer si la suite $\left(z_i\,(x,y)\right)_i$ est divergente ou convergente, on calcule un certain nombre d'élément de celle-ci.

$$z_1, z_2, z_3, \ldots$$

On applique ensuite un critère mathématique dans lequel on passe les $z_1, z_2, z_3, \ldots$ Ce critère

$$\boxed{isDivergent(z_i)}$$

nous permet de conclure à une *divergence* ou à une *convergence*. La finesse du E_N *-Fractale* dépend du nombre N d'éléments de la suite que l'on prend pour se donner les moyens de déterminer la convergence ou la divergence de la suite.

On ne peut pas calculer tous les éléments de la suite $\left(z_i\,(x,y)\right)_i$, il y a en a une infinité ! On en calcul seulement, et au maximum : N . On s'arrête des que le premier des $z_1, z_2, z_3, \ldots z_N$ répond *vrai* à la sollicitation

$$isDivergent(z_i)$$

Posons, s'il existe, s le premier z_i satisfaisant ce critère de divergence. Il se peut que ce critère ne soit jamais réalisé pour le N que l'on s'est fixé, dans ce cas, le pixel (x,y) est noir.

Couleur

Le pixel (x,y) sur lequel on a construit la suite $(z_i(x,y))_i$ est alors colorié selon la règle

$$color_N((x,y)) = \begin{cases} noir & \text{si} \quad s \text{ n'existe pas} \\ HSB(h(s),1,1) & \text{sinon} \end{cases}$$

où s est le premier $i \le N$ ou l'élément z_i satisfaisait au critère de divergence.

Note

Le fractale est l'ensemble des points (x,y) noir pour lesquels la suite $(z_i(x,y))_i$ à N -convergé!
La partie intéressante est pourtant la partie colorée où la suite $(z_i(x,y))_i$ à $N-$divergé.

Résumer

MandelBrot et *Julia* seront donc défini par

- une *suite* spécifique $(z_i(x,y))_i$ associé à chaque pixel (x,y)
- un critère de divergence, ie une fonction booléenne $isDivergent(z_i(x,y))$
- un paramètre N fournissant non pas un fractale, mais une famille de fractale.

Famille

Selon la valeur de N ,on n'obtient pas le même fractale! Il y a donc plusieurs fractales de *Julia* et plusieurs fractales de *Mandelbrot* selon la valeur de N choisie! Il est donc:

- important de citer cette valeur de N .
- intéressant de faire varier cette valeur de N .

On parle de la ***famille*** des fractales de *Julia* et de la famille des fractales de *Mandelbrot*.

Important:

Les E_N *-Fractaux* sont intéressants lorsque l'on effectue des zooms sur certaine partie de ceux-ci et que l'on réitère le zooming plusieurs fois. Si l'on représente sous forme graphique rectangulaire un E_N *-Fractale*, il est donc important de définir la zone dessinée:

$$[xMin,xMax]x[yMin,yMax] \subset \mathbb{R}^2$$

Mandelbrot

Introduction

Ce fractal découvert par *Benoît Mandelbrot* dans les années 80 présente de nombreuses particularités. Une des plus intéressantes est que l'infini complexité de la géométrie n'est le résultat que d'un algorithme excessivement simple!

Définition

Soit $N \in \mathbb{N}^*$, et $x \in \mathbb{R}, y \in \mathbb{R}$.

$(x, y) \in Mandelbrot_n$ ssi la suite $(z_k)_{k \in \mathbb{N}}$ *définit par*

$$\begin{cases} z_{k+1} = z_k^{\ 2} + (x + iy) & \forall k \geq 0, k \in \mathbb{N} \\ z_0 = 0 \end{cases}$$

est *N-convergente*. Le critère de divergence est le suivant:

Si $\|z_k\| > 2$ pour $k \in [0, N]$, alors la suite $(z_k)_{k \in \mathbb{N}}$ est divergente.

Ainsi, si $\|z_k\| < 2 \ \ \forall k \in [0, n]$, alors on conclut que la suite $(z_k)_{k \in \mathbb{N}}$ est *N-convergente*, et que $(x, y) \in Mandelbrot_n$. Le pixel (x, y) est alors noir. Les pixels colorés sont ceux pour lesquels la suite z_k est divergente, et la couleur du pixel (x, y) dépend alors du premier $k \leq N$ pour lequel on a $\|z_k\| > 2$

Résumé

Soit $N \in \mathbb{N}^*$, et $x \in \mathbb{R}, y \in \mathbb{R}$

$(x, y) \notin Mandelbrot_n$ ssi $\exists s \in [0, N]$ tel que $\|z_s\| > 2$,

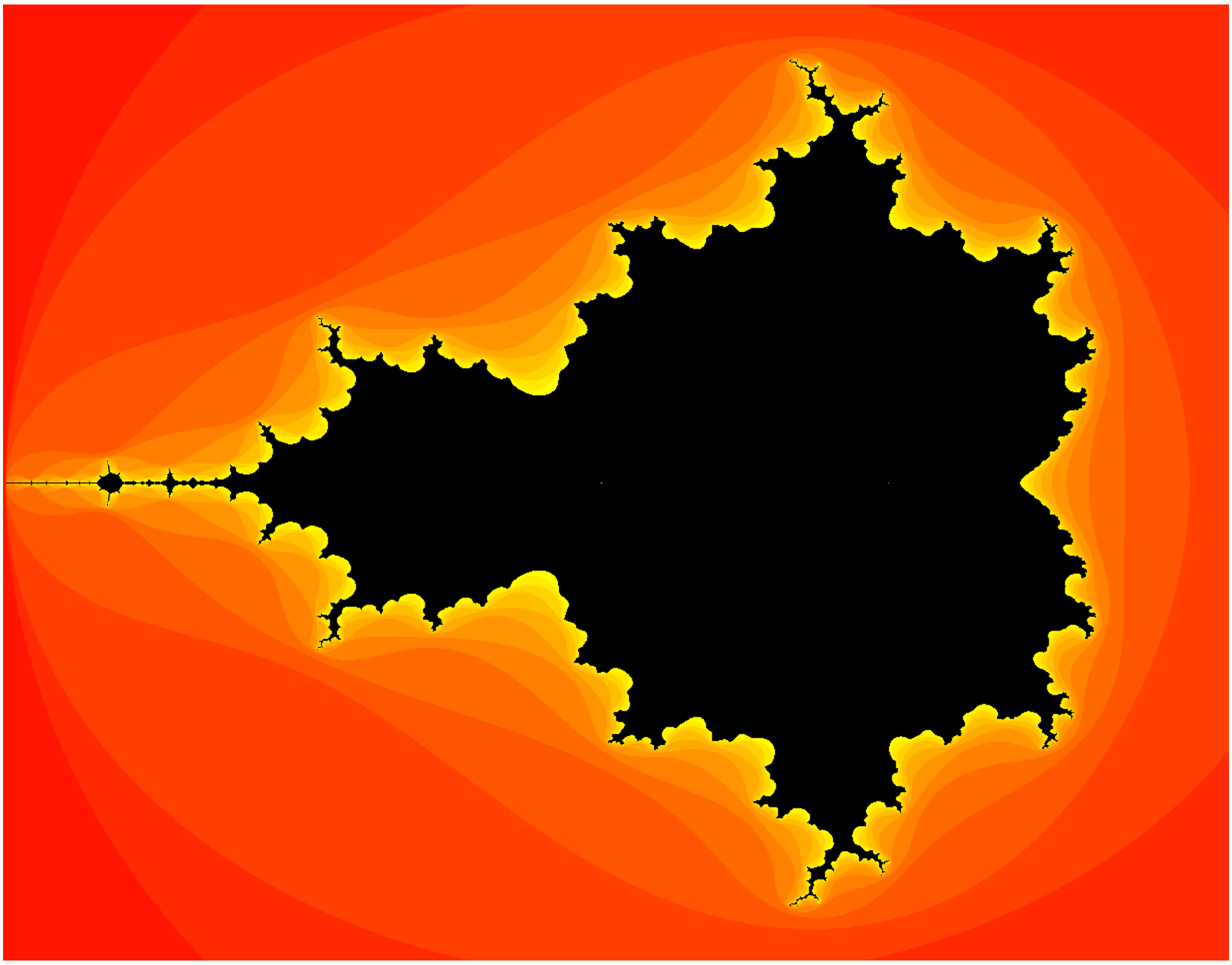
avec z_k nombre complexe défini par

$$\begin{cases} z_{k+1} = z_k^{\ 2} + (x + iy) & \forall k > 0, k \in \mathbb{N} \\ z_0 = 0 \end{cases}$$

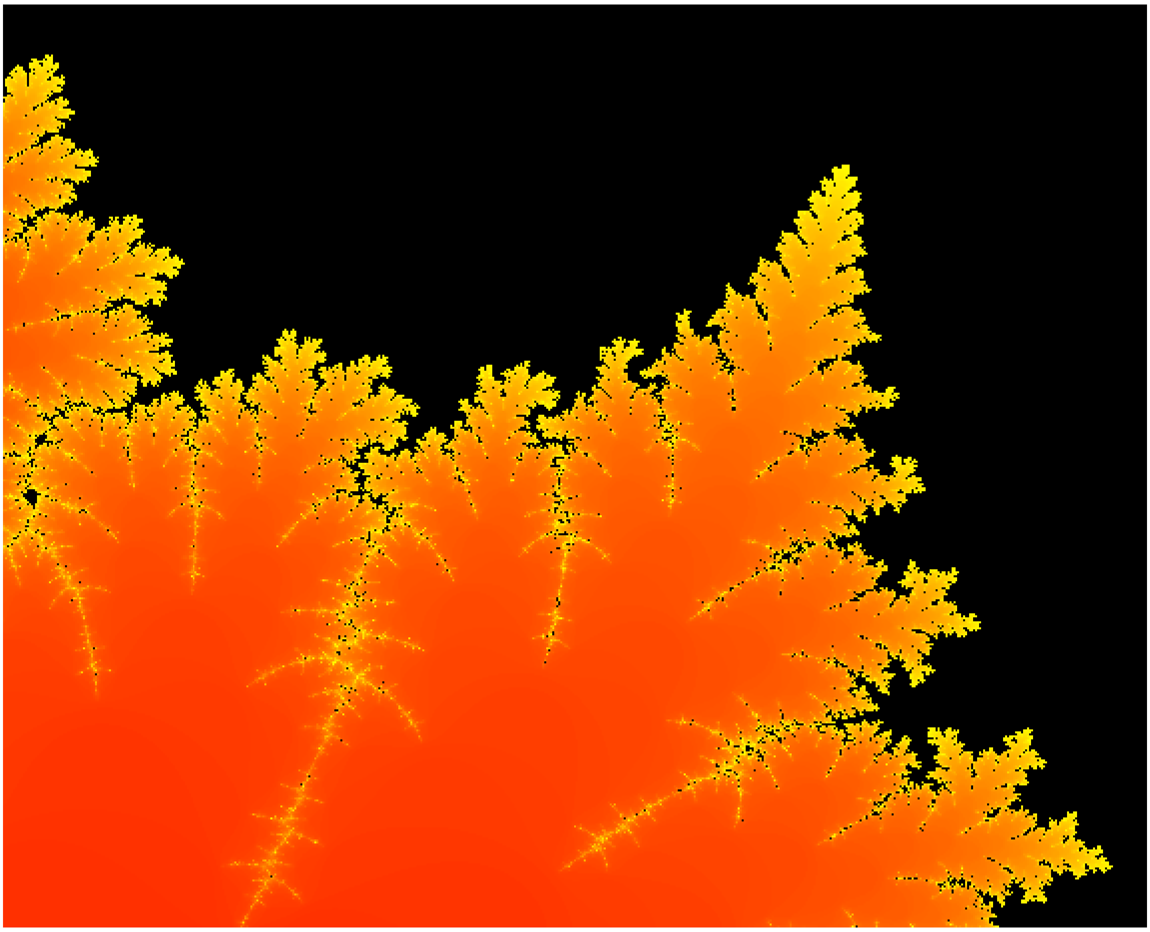
Si le (x, y) n'appartient pas au fractale, alors sa couleur dépend du plus petit s pour lequel $\|z_s\| > 2$, sinon le pixel (x, y) est noir !

Exemple 1

N=12 [x1,x2] x [y1,y2] = [-2.1 , 0.8] x [-1.3 , 1.3]



Exemple 2 N=102 [x1,x2] x [y1,y2] = [-1.3968,-1.3578] x [-0.03362 , 0.0013973]



Julia

Introduction

Ces fractaux ont été découverts par *Gaston Julia*. Ils sont très proches du fractal de *Mandelbrot*.

- Pour *Mandelbrot*, la suite est définit par
  - $z_0 = 0 + 0i$
  - $z_{k+1} = z_k^{\ 2} + c$ avec $c = x + iy$
- Pour *Julia*, la suite est définit par
  - $z_0 = x + yi$
  - $z_{k+1} = z_k^{\ 2} + C$ avec C un paramètre supplémentaire, dit **paramètre de Julia**.

Dans les deux cas (x, y) représente un pixel dont on se demande s'il appartient au fractal ou non.

Famille

Julia est donc une double famille de fractale. Cette famille est paramétrée par

- N (idem *Mandlebrot*)
- C (spécifique à *Julia*)

Définition Soit $N \in \mathbb{N}^*$, $C \in \mathbb{C}$ et $x \in \mathbb{R}, y \in \mathbb{R}$

$(x, y) \in Julia_{n,c}$ si et seulement si la suite $(z_k)_{k \in \mathbb{N}}$ *définit par*

$$\begin{cases} z_{k+1} = z_k^{\ 2} + C & \forall k > 0, k \in \mathbb{N} \\ z_0 = x + iy \end{cases}$$

est *N-convergente*. Le critère de divergence est le suivant:

Si $\|z_s\| > 2$ pour $s \in [0, N]$, alors la suite $(z_k)_{k \in \mathbb{N}}$ est divergente.

Ainsi, si $\|z_k\| < 2 \ \forall k \in [0, N]$, alors on conclut que la suite $(z_k)_{k \in \mathbb{N}}$ est *N-convergente*, et que $(x, y) \in Julia_{N,C}$.

Note

En faisant varier $C \in \mathbb{Q}$, on obtient plusieurs fractaux de *Julia* différent. *Julia* est donc une double famille de fractale, à cause des deux paramètres variables $N \in \mathbb{N}^*$ et $C \in \mathbb{C}$.

Résumé

Soit $N \in \mathbb{N}^*$, $C \in \mathbb{C}$ et $x \in \mathbb{R}, y \in \mathbb{R}$

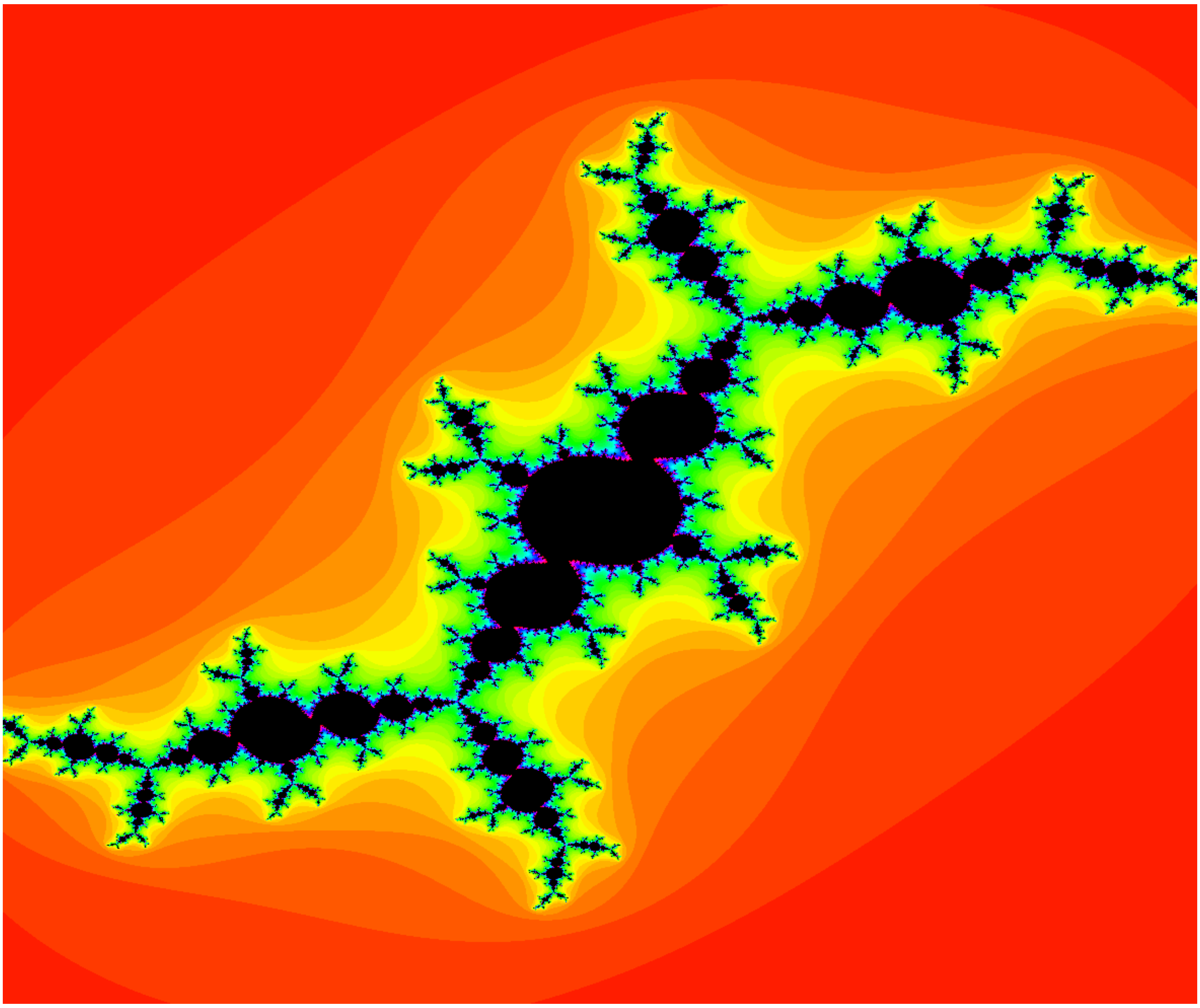
$(x,y) \notin Julia_{n,c}$ ssi $\exists s \in [0,N]$ tel que $\|z_s\| > 2$,

avec z_k nombre complexe défini par

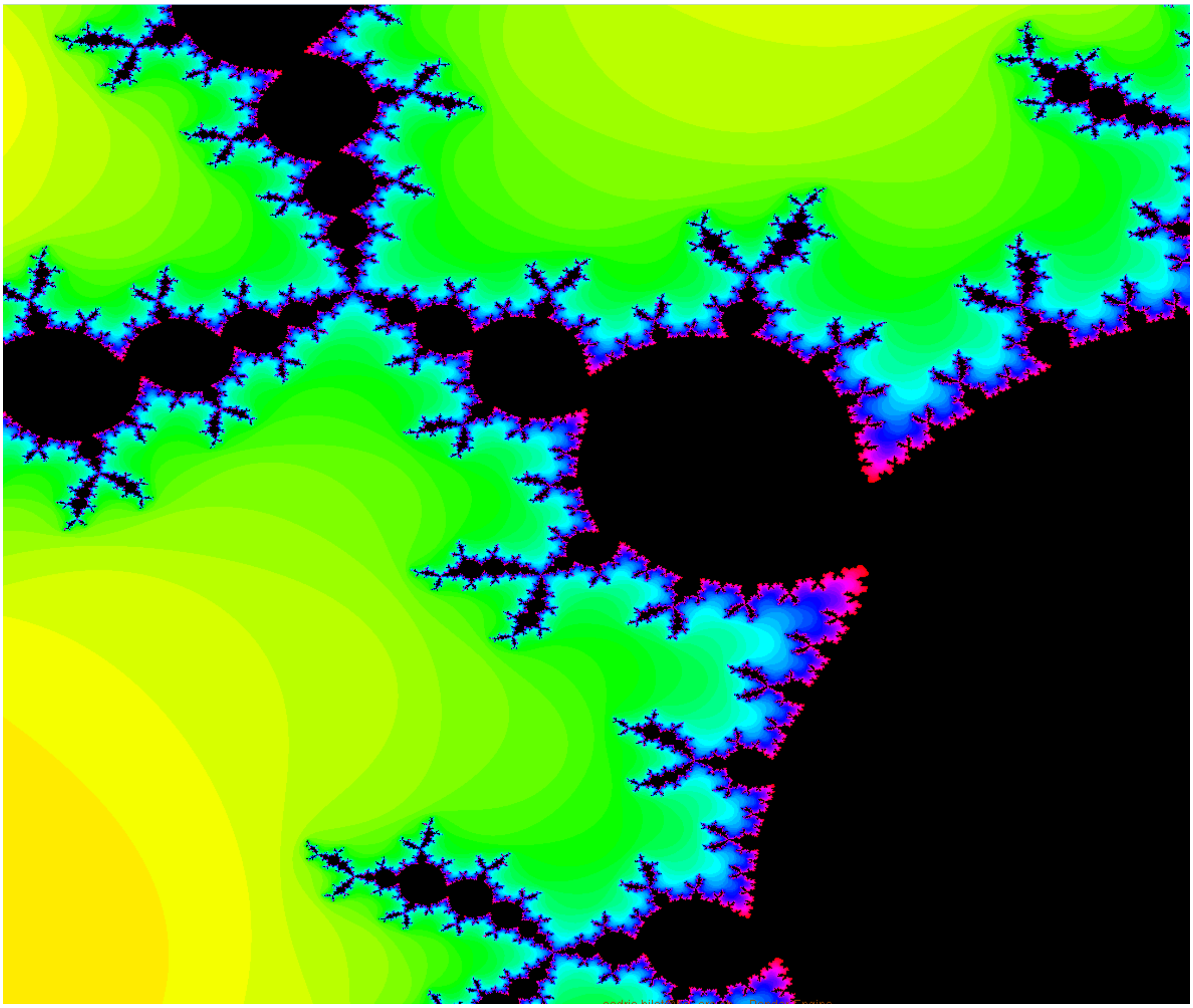
$$\begin{cases} z_{k+1} = z_k^{\ 2} + C & \forall k \geq 0, k \in \mathbb{N} \\ z_0 = x + iy \end{cases}$$

Si le (x,y) n'appartient pas au fractale, alors sa couleur
dépend du plus petit $s \leq N$ pour lequel $\|z_s\| > 2$, sinon le pixel (x,y) est noir !

Exemple 1 N=52 C= -0.12+0.85i [x1,x2] x [y1,y2] = [-1.3 , 1.3] x [-1.4 , 1.4]

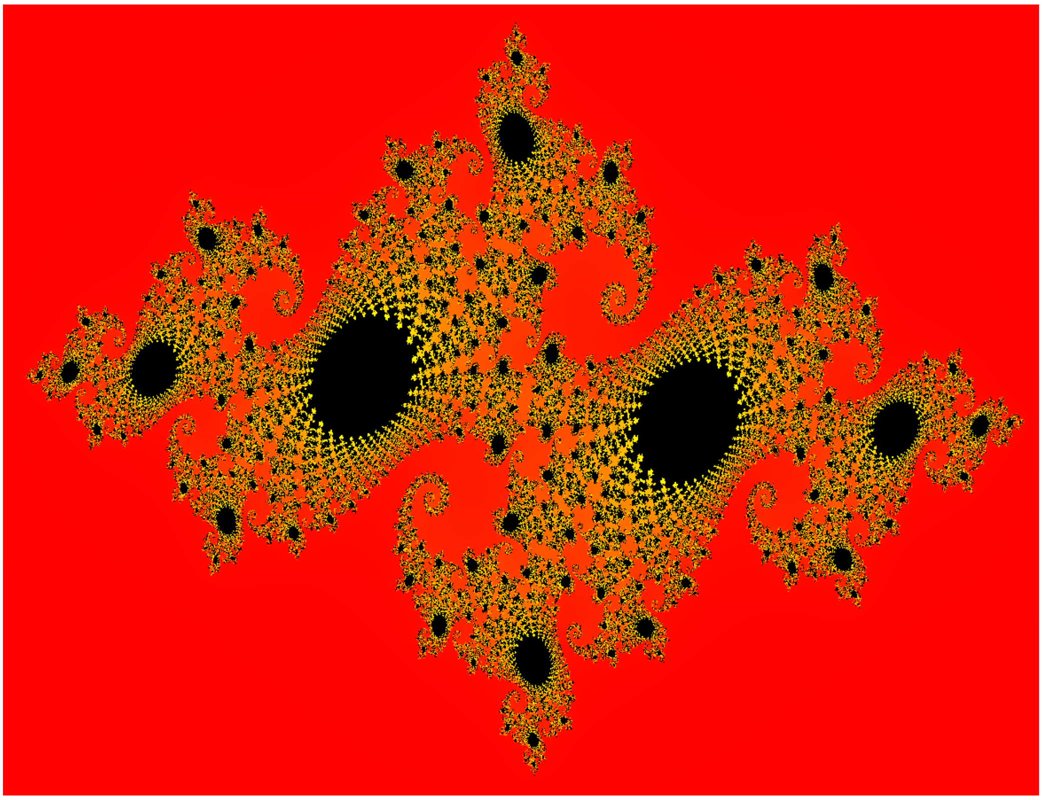


Exemple 2 N=52 C= -0.12+0.85i [-0.327167, -0.086667] x [-0.2156, 0.0434]

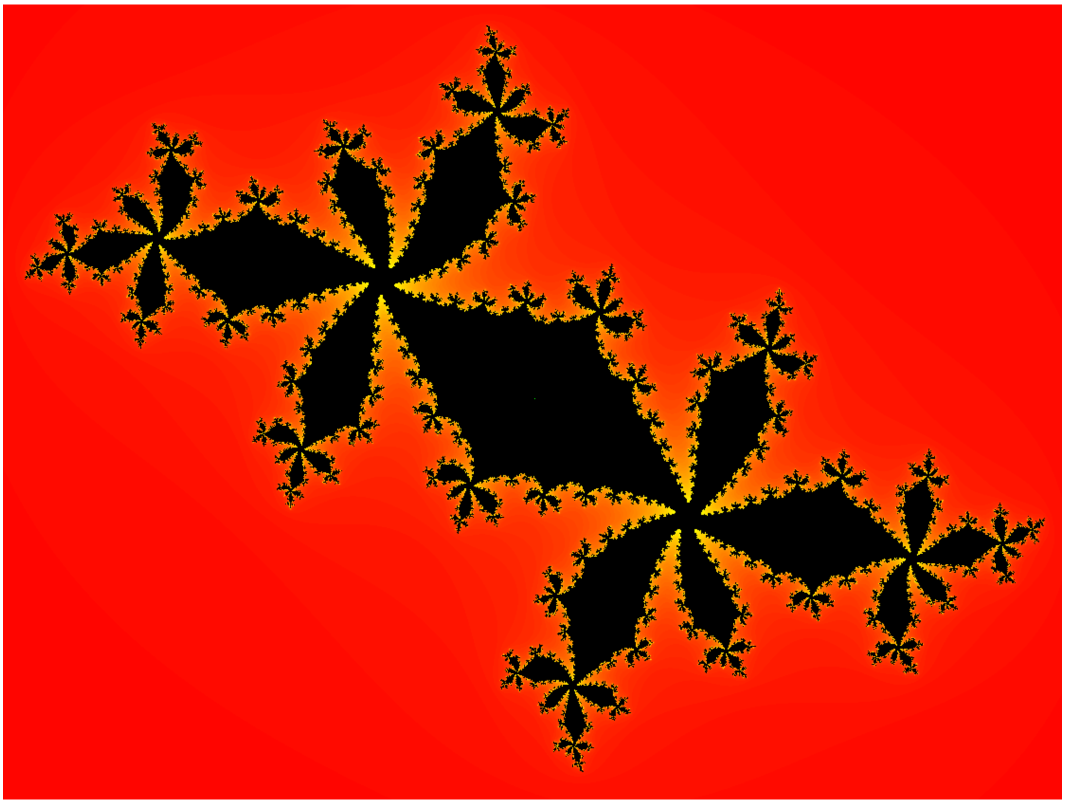


(Zoom exemple précédent)

Exemple 3 N=300 C=-0.745+0.1i [x1,x2] x [y1,y2] = [-1.7 , 1.7] x [-1 , 1]



Exemple 4 N=50 C= -0.52 +0.57i [x1,x2] x [y1,y2] = [-1.7 , 1.7] x [-1.2 , 1.2]



Implémentation

Conseils

(C1) Utiliser la version *fonctionnelle* de l'API Image fournie.

(C2) Utiliser une hiérarchie de classe

(C3) N'utilisez pas de classe *Complexe*, mais travailler composante par composante

Indication

Contexte

Soit $(i, j) \in \mathbb{N}^2$ un pixel de l'image.

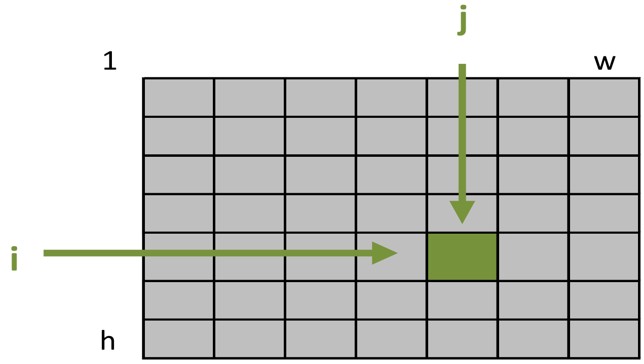
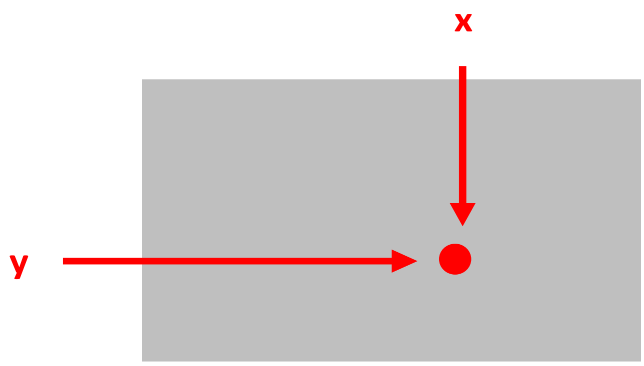


Image (wxh)

On convertit $(i, j) \in \mathbb{N}^2$ en $(x, y) \in \mathbb{R}^2$ un point du domaine mathématique auquel est associé le fractal. Utilisez par exemple la fonction

$$\boxed{toXY(i, j, x, y)}$$

de l'API Image, ou faite le travail de conversion vous-même.



Domaine Math

On associe ensuite une suite de nombre complexe $(z)_k$ à ce pixel. Selon la rapidité de convergence, ou la divergence, on associe une couleur à ce pixel. La couleur d'un pixel ne dépend donc pas de la couleur des pixels voisins, mais uniquement de cette suite $(z)_k$.

Mandelbrot

Soit la suite

$$\begin{cases} z_0 = 0 \in \mathbb{C} \\ z_{k+1} = z_k^{\ 2} + (x + iy) \in \mathbb{C} \end{cases}$$

Posons

$$z_k = a_k + ib_k \in \mathbb{C}$$

Dès lors

$$\begin{aligned} z_k^{\ 2} &= (a_k + ib_k)^2 \\ &= a_k^{\ 2} + 2a_k b_k i - b_k^{\ 2} \\ &= \underbrace{(a_k^{\ 2} - b_k^{\ 2})}_{re} + \underbrace{2(a_k b_k)}_{im} i \end{aligned}$$

On a ainsi

$$\|z_k\|^2 = a_k^{\ 2} + b_k^{\ 2}$$

On s'arrête de calculer des éléments de la suite que si le nombre d'itération dépasse un seuil limite N que l'on s'est fixé à l'avance, où lorsque

$$\|z_k\|^2 = a_k^{\ 2} + b_k^{\ 2} > 4$$

Pour l'informaticien :

<u>Initialisation</u>

$$\boxed{\begin{aligned} a_0 &= 0 \in \mathbb{R} \\ b_0 &= 0 \in \mathbb{R} \end{aligned}}$$

<u>Itération</u>

$$\boxed{\begin{aligned} &aCopy = a \\ &a = (a^2 - b^2) + x \\ &b = 2 * aCopy * b + y \end{aligned}}$$

Arrêt

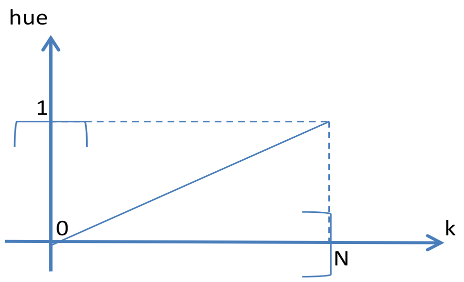
$$\boxed{stop \quad ssi \quad \begin{cases} a_k^{\ 2} + b_k^{\ 2} > 4 \\ k > N \end{cases}}$$

où $(x, y) \in \mathbb{R}^2$ est le point du domaine de math associé au pixel $(i, j) \in \mathbb{N}^2$ de l'image, et N un entier que l'on s'est fixé à l'avance. Plus ce paramètre N est grand, plus l'image sera raffinée.

Color

$$\boxed{color(i, j) = color(x, y) = \begin{cases} noir & \text{si} \quad k > N \\ HSB \quad avec \quad h = h(k) & \text{sinon} \end{cases}}$$

où la $hue \in [0,1]$ est fabriqué de manière linéaire selon la valeur de $k \in [0, N]$



$$\boxed{h(k) = \frac{1}{N}k \qquad \in [0,1]}$$

Vous pouvez aussi utilisez la classe ***Calibration*** fournie.

Julia

Le principe est le même, mais ici l'initialisation n'est plus donné par $(0,0) \in \mathbb{R}$, mais par $(x, y) \in \mathbb{R}^2$, et l'itération dépend elle du paramètre de Julia $(c_1, c_2) \in \mathbb{R}$ que l'on s'est fixé à l'avance. Pour l'informaticien :

Initialisation

$$\boxed{\begin{aligned} a_0 &= x \in \mathbb{R} \\ b_0 &= y \in \mathbb{R} \end{aligned}}$$

Itération

$$\boxed{\begin{aligned} &aCopy = a \\ &a = \left(a^2 - b^2\right) + c_1 \\ &b = 2 * aCopy * b + c_2 \end{aligned}}$$

où $(x, y) \in \mathbb{R}^2$ est le point du domaine de math associé au pixel $(i, j) \in \mathbb{N}^2$ de l'image, et $(c_1, c_2) \in \mathbb{R}$ la constante de *Julia*.

Domaine et Piège

Dans ce document le domaine de mat est donné selon le formalisme **mathématique**

[x1,x2] x [y1,y2] = [-2.1 , 0.8] x [-1.3 , 1.3]

alors que dans l'API image fournie le domaine est spécifié sous le format commun au **informaticien**, par « coins » :

x1,y1, x2,y2 = -2.1, -1.3, 0.8, 1.3

Hiérarchie

Gagner du temps

Si vous êtes malin, utiliser une hiérarchie de classe pour coder à la fois Julia et Mandelbrot. Utiliser une classe abstraite :

Fractale

pour coder les éléments communs à *Julia* et *Mandelbrot*. Les seuls éléments différents entre *Julia* et *Mandelbrot* sont :

- Le calcul de la suite
- La constante *C* d'initialisation de Julia

On résout le premier problème avec une méthode virtuelle pure dans la classe abstraite *Fractale*, par exemple

int indiceArret() ;

qui retourne le premier indice ou le critère d'arrêt est atteint, ou *N* sinon. Le second problème est résolu avec le constructeur de la classe *Julia*.

Animation

Pour *Julia* comme pour *Mandelbrot*, une animation sympathique est obtenue en faisant varier le paramètre *N* entre *nMin* et *nMax*. Par exemple prendre

$$n \in [30,100]$$

Vous pouvez utiliser la classe ***VariateurI*** fournie qui fait varier l'entier *n* entre $[n_{\min}, n_{\max}]$ de manière linéaire, d'abord sous forme croissante de $n_{\min} \to n_{\max}$ puis de manière décroissante de $n_{\min} \leftarrow n_{\max}$

Validation

Effectuer les variations suivantes :

- Taille de l'image (rectangulaire, pas carrée !)
- dg et db

Pour taille de l'image prenez par exemple

```
int dw = 16 * 80;
int dh = 16 * 60;
```

Pour les contraintes à satisfaire sur *dg* et *db*, utilisez

```
Devices ::printAll() ;
```

Speedup

Mesurer les coefficients de *speedup* des différentes implémentations.

Pour canevas, utiliser le document

speedup_simple.xls.

Au besoin adapter ce canevas.

End