Programmation fonctionnelle – TD

Fractales

Décembre 2018

L'objectif de ce TD est de permettre la visualisation d'ensembles fractals célèbres : l'ensemble de Mandelbrot [1] et des ensembles de Julia [2].

Les notes de bas de page donnent des indices, qu'on n'est pas obligé de lire.

On pourra se référer à Hoogle [3] pour la documentation sur l'API standard.

Pour faciliter la mise au point, on pourra enfin utiliser la fonction `trace` pour afficher certaines informations lors de l'évaluation des fonctions.

1 Rappels sur les ensembles de Mandelbrot et Julia

Pour $c_0, c \in \mathbb{C}$, on définit la suite de nombres complexes $(z_{c_0,c,n})_{n\in\mathbb{N}}$ par :

$$\begin{cases} z_{c_0,c,0} = c_0 \\ z_{c_0,c,n+1} = z_{c_0,c,n}^2 + c \end{cases}$$

L'ensemble de *Mandelbrot* $\mathcal{M}$ est alors l'ensemble des points c tels que $(z_{0,c,n})_{n\in\mathbb{N}}$ est bornée.

L'ensemble de *Julia* $\mathcal{J}_c$ (relatif à $\mathcal{M}$) pour le paramètre c est l'ensemble des points c_0 tels que $(z_{c_0,c,n})_{n\in\mathbb{N}}$ est bornée.

On peut montrer que ces ensembles sont inclus dans le disque du plan complexe de centre 0 et de rayon 2. Étant donné un nombre maximum d'itérations m, on fera l'approximation suivante : la suite est bornée si et seulement si tous ses éléments d'indices inférieurs à m ont un module inférieur ou égal à 2.

On utilisera le type `Complex` [4] de Haskell pour les calculs [5].

2 Calcul de la suite et divergence

Q1. Proposer une fonction `fractale` qui étant donné le paramètre `c`, et un terme `z` de la suite définie ci-dessus, calcule le terme suivant ;

1. `https://fr.wikipedia.org/wiki/Ensemble_de_Mandelbrot`
2. `https://fr.wikipedia.org/wiki/Ensemble_de_Julia`
3. `https://www.haskell.org/hoogle/`.
4. Instancié avec des réels double précision.
5. Indice : comme le type résultant est un peu long à écrire on pourra utiliser `type` pour définir un *alias*.

Q2. Proposer une fonction `divergence` qui étant donné un nombre maximum d'itérations `m`, une fonction `f` qui calcule l'itération suivante de la suite comme dans la question précédente, et une valeur initiale complexe `z`, donne le plus petit $n \leq$ `m` tel que $\texttt{f}^n(\texttt{z})$ a un module supérieur à 2 [6] [7] [8]. S'il n'existe pas de tel n, la fonction donne la valeur 0.

3 Création des images

On se donne un rectangle du plan complexe défini par deux points opposés `c1`, en haut à gauche, et `c2`, en bas à droite. On se donne également une largeur d'image `l`, en nombre de points. La hauteur `h` associée sera calculée pour préserver le ratio défini par `c1` et `c2`.

On veut associer un point du rectangle à chaque point de l'image ainsi définie.

Q3. Proposer une fonction `hauteur` qui calcule la hauteur `h`, en nombre de points, d'une image de largeur `l`, avec le ratio largeur/hauteur défini par `c1` et `c2` [9] ;

Q4. Proposer une fonction `points` qui à partir de `l`, `h`, `c1` et `c2` donne une matrice de `l`×`h` nombres complexes, discrétisant de manière régulière le rectangle défini par `c1` et `c2` [10] ;

Q5. Proposer une fonction `mandlebrot` qui étant donnés un nombre maximal d'itérations `m` et une matrice `css` de nombres complexe (par exemple obtenue avec la Q4), calcule une matrice d'entiers, dans laquelle l'élément d'indice `i`,`j` représente le rang de divergence (comme défini à la Q2) de la suite $(z_{c_0,c,n})_{n\in\mathbb{N}}$, pour $c =$`css!!i!!j` et $c_0 = 0$ [11] ;

Q6. Proposer une fonction `julia` qui étant donnés un nombre complexe `c`, un nombre maximal d'itérations `m` et une matrice `css` de nombres complexe (par exemple obtenue avec la Q4), calcule une matrice d'entiers, dans laquelle l'élément d'indice `i`,`j` représente le rang de divergence (comme défini à la Q2) de la suite $(z_{c_0,c,n})_{n\in\mathbb{N}}$, pour $c =$`c` et c_0=`css!!i!!j` ;

4 Création des fichiers images

Afin de s'affranchir des problèmes liés à l'affichage graphique, on va engendrer des images au format PGM [12].

On utilise la variante « Plain PGM » dans laquelle l'image est un fichier texte (et non pas un fichier binaire) décrivant chaque point par une unique valeur entière répresentant une nuance de gris : de 0 qui correspond au noir, à la valeur maximale (voir ci-après) qui correspond au blanc.

Le fichier commence par une ligne contenant la chaîne de caractères `P2`, puis une ligne contenant la hauteur et la largeur de l'image séparés par un espace, puis une ligne contenant la valeur maximale d'un point de l'image.

Viennent ensuite les points, rangés par ligne, avec une valeur par point. Chacune de ces valeurs doit avoir au moins un espace, ou un saut de ligne, avant et après elle. Normalement,

6. Indice : on pourra utiliser les fonctions `take`, `takeWhile`, et `iterate`.
7. Indice : la fonction `zip` peut vous permettre de « numéroter » les élements d'une liste avec leur indice.
8. Indice : pour accélérer un peu les calculs, on pourra écrire et utiliser une fonction qui calcule le carré de la norme d'un complexe.
9. Indice : pour transformer un entier en réel, on pourra utiliser la fonction polymorphique `fromIntegral`.
10. Indice : le résultat de cette fonction pourra être écrite sous la forme d'une liste en compréhension.
11. Indice : on pourra utiliser (deux fois) `map` pour appliquer une fonction à tous les éléments de la matrice.
12. `http://netpbm.sourceforge.net/doc/pgm.html`

les lignes du fichier ne doivent pas faire plus de 70 caractères, ce qui implique qu'on devra en général écrire une ligne de l'image sur plusieurs lignes du fichier, mais nous ignorerons cette contrainte.

Q7. Proposer une fonction `show'` qui transforme un entier en une chaîne de caractères, en ajoutant un espace à la fin ;

Q8. Proposer une fonction `fichier` qui transforme une matrice d'entiers telle qu'obtenue aux Q5 ou 6, en une chaîne de caractères représentant le contenu d'un fichier PGM ;

Q9. Écrire la fonction principale, appelant `mandelbrot` ou `julia` et écrivant l'image correspondante dans un fichier [13] ;

Q10. On pourra regarder en particulier les images produites avec la largeur `l`=1024, le nombre d'itérations `m`$= 256$, les valeurs `c1`$= -2 + 1.2i$ et `c2`$=1 - 1.2i$ pour `mandelbrot` et `l`=1024, `m`$= 512$, `c1`$= -2 + 1.2i$, `c2`$=2 - 1.2i$ et `c`$= -0.8 + 0.156i$ pour `julia`.

13. Indice : on pourra utiliser la fonction `writeFile`.