TP 4— JOLIS DESSINS (MANDELBROT)

http://www.gallium.inria.fr/~scherer/teaching/colles/2012/tp4.pdf

Les questions en rapport avec les TP, Caml ou la programmation en général sont les bienvenues et peuvent être envoyées à `gabriel.scherer@gmail.com`. N'hésitez pas !

Le but de ce TP est de faire de jolis dessins. On utilisera pour cela des mathématiques, et le module `graphics` de `Caml Light`.

1 PRÉLIMINAIRES

Une bonne méthode pour obtenir des dessins amusants vient de la dynamique complexe. En deux mots, on choisit un polynôme complexe P, et l'on étudie la suite $P^n(z)$, où P^n est la fonction polynomiale itérée ($P^{n+1} = P^n \circ P$) et z un point du plan complexe.

On peut se poser de nombreuses questions au sujet de ces suites $P^n(z)$ (la périodicité par exemple). On s'intéressera ici à un critère très simple : on se demande si elles sont bornées en norme.

On va "dessiner" la question, en affichant à l'écran une partie du plan complexe en mettant les points en noir et blanc (par exemple) selon que la suite des normes correspondant est bornée ou non.

Pour ne pas se prendre la tête, on va commencer par les polynômes intéressants les plus simples possibles, à savoir ceux de la forme $P_c = X^2 + c$, $c \in \mathbb{C}$. La suite $(P_c^n(z))_{n\in\mathbb{N}}$ dépend des deux paramètres c et z. Dans un dessin à deux dimensions, seul un paramètre complexe varie, il faut donc faire un choix. Si l'on décide de faire varier z, on obtient des fractales de Julia; si l'on décide de faire varier c, on obtient la fractale de Mandelbrot. La fractale de Mandelbrot est plus simple puisqu'on peut faire un choix facile pour z : on prend $z = 0$, et on obtient de bons résultats. Les fractales de Julia, au contraire, dépendent fortement du choix de c et il peut être délicat de trouver des valeurs produisant un dessin satisfaisant : clairement, $c = 0$ n'est pas très un choix très excitant.

1.1 OPÉRATIONS COMPLEXES

▷ **Question 1.** On représente les complexes par le type `float * float`. Écrire les fonctions `add`, `mul`, `norme2` d'addition, de multiplication et de norme au carré dans $\mathbb{C}$. ◁

Le critère de divergence est très simple : on regarde la suite jusqu'au terme N (pour N très grand), et si le carré de la norme de l'une des valeurs est supérieure à R (pour R assez grand), on considère que la suite n'est pas bornée. C'est très rigoureux. On prendra par la suite $N = 50$ et $R = 4$.

▷ **Question 2.** Définir des valeurs `limite_iterations = 50` et `limite_norme2 = 4`.. Définir ensuite une fonction `mandelbrot : (float * float) -> (int * (float * float))` qui, étant donné c en paramètre, calcule[1] les valeurs de $P_c^n(0)$. La fonction doit s'arrêter :

- si $n = N$
- si $n < N$ mais $|P_c^n(0)|^2 > R$

Dans les deux cas, on renvoie le couple $(n, P_c^n(0))$. ◁

2 AFFICHAGE

On calculera et affichera seulement des valeurs discrètes de la fonction `mandelbrot` : concrètement, on parcourira avec deux boucles les valeurs `(x,y)` avec `0 <= x < taille_x` et `0 <= y < taille_y`. Il faut donc établir une correspondance entre ces valeurs et les points dans la région du plan complexe qui nous intéresse.

Dans le cas de l'ensemble de mandelbrot, on utilisera la traduction suivante :

$$\begin{cases} x \to 2\frac{x}{taille_x} - 1.5 \\ y \to 2\frac{y}{taille_y} - 1. \end{cases}$$

▷ **Question 3.** Écrire une fonction `normalise : int -> int -> float` telle que $normalise(k, n) = \frac{k}{n}$.

Écrire la fonction `point : int * int -> int * int -> float * float`, qui, étant donnés `(taille_x, taille_y)` et `(x,y)`, effectue la traduction des coordonnées dans le plan complexe. ◁

[1] récursivement, bien sûr

2.1 Affichage moche

Pour afficher une fenêtre avec des couleurs, on utilisera le module *graphics*. Mais pour commencer, on va essayer d'approximer la forme de l'ensemble en se content d'afficher du texte (c'est plus simple).

▷ **Question 4.** Écrire une fonction `mandelbrot_texte : int -> int -> unit` qui, étant donné les paramètres `taille_x` et `taille_y`, affiche `taille_y` lignes de `taille_x` lettres de large, représentant l'ensemble de mandelbrot : on affichera (`print_char`) le caractère `'X'` quand le point appartient à l'ensemble (cas d'arrêt $n = N$), `'-'` sinon. ◁

Vous pouvez tester votre fonction avec des valeurs raisonnables (il faut que cela tienne dans votre écran), par exemple 20x20.

2.2 Affichage en couleurs

On utilisera maintenant le module *graphics*. N'hésitez pas à consulter sa documentation dans le manuel.

Pour pouvoir utiliser les fonctions du module sans le préfixe `graphics` (`graphics__set_color`, etc.), on utilise la directive `#open "graphics" ;;`.

Les couleurs sont représentées (comme souvent en informatique) par un triplet (r, g, b), où r représente le taux de rouge, g de vert et b de bleu. La fonction rgb du module produit prend ces trois paramètres (entiers entre 0 et 255 inclus), et produit une couleur utilisable par les fonctions de dessin.

Pour produire une couleur à partir du résultat de `mandelbrot`, il est plus simple de travailler sur des triplets `(r,g,b)` normalisés (de `float` dans $[0, 1]$). On a donc besoin d'une fonction de conversion, d'un triplet normalisé vers une couleur `Caml`, qui réutilise `rgb`.

▷ **Question 5.** Écrire `caml_couleur : float * float * float -> color`, qui produit une couleur `Caml`à partir d'un triplet normalisé $(r, g, b) \in [0, 1]^3$. ◁

Il faut choisir une représentation graphique du résultat de `mandelbrot`. On commencera par une méthode simple, utilisant uniquement des niveaux de gris : $r = g = b = \frac{n}{N}$, où n est l'itération à laquelle s'est arrêtée la fonction `mandelbrot`.

▷ **Question 6.** Écrire `couleur : int * (float * float) -> color`, qui au résultat de `mandelbrot` associe une couleur `Caml`. ◁

```
let mandelbrot_graphique couleur taille_x taille_y =
  open_graph (printf__sprintf " %dx%d" taille_x taille_y);
  (* ...  *)
  let _ = read_key () in
  close_graph ();;
```

▷ **Question 7.** En utilisant le squelette ci-contre, écrire la fonction `mandelbrot_graphique` prenant une fonction `couleur` en paramètre, et affichant l'ensemble de mandelbrot dans un rectangle de dimensions `taille_x, taille_y`.

Pour afficher un point, on utilisera la fonction `plot : int -> int -> ()` prenant les coordonnées du point à afficher. Pour régler la couleur d'affichage (à changer pour chaque point à l'aide de la fonction `couleur`), on utilise au préalable `set_color : color -> unit`.

L'utilisation de la fonction `read_key` permet d'attendre l'appui sur une touche par l'utilisateur avant de fermer l'image. ◁

▷ **Question 8.** Tester la fonction `mandelbrot_graphique` avec d'autres fonctions `couleur`. On pourra essayer par exemple la traduction suivante :

$$\begin{cases} (n, z) \mapsto (0, 0, 0) & \text{si } n < N \\ (n, z) \mapsto (\frac{Re(z)^2}{|z|^2}, 0, \frac{Im(z)^2}{|z|^2}) & \text{quand } n = N \end{cases}$$

◁