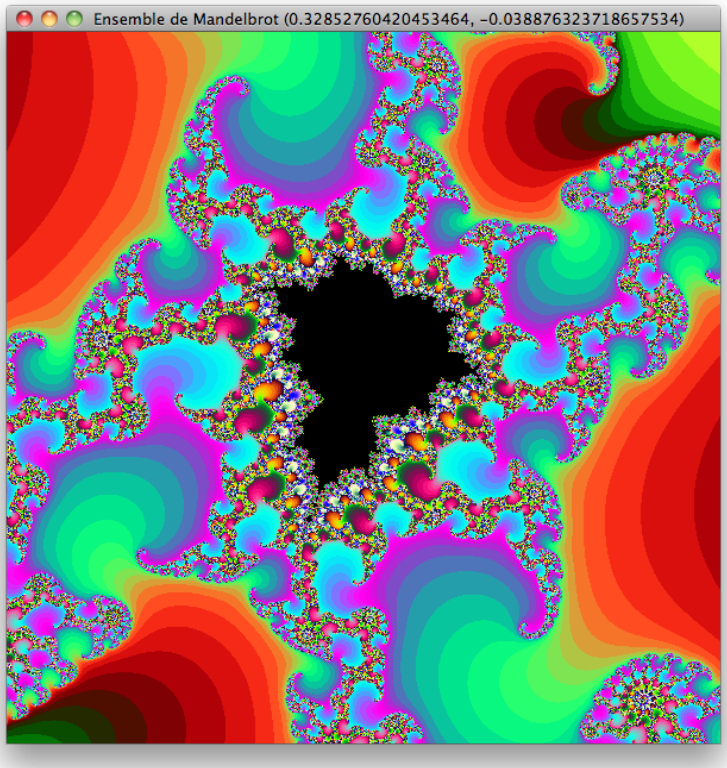
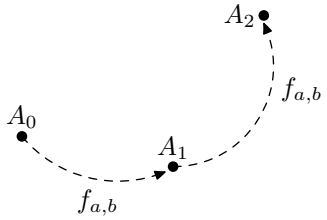
TP - Ensemble de Mandelbrot

Le but du TP est d'écrire quelques méthodes permettant de visualiser l'ensemble de Mandelbrot à l'écran.



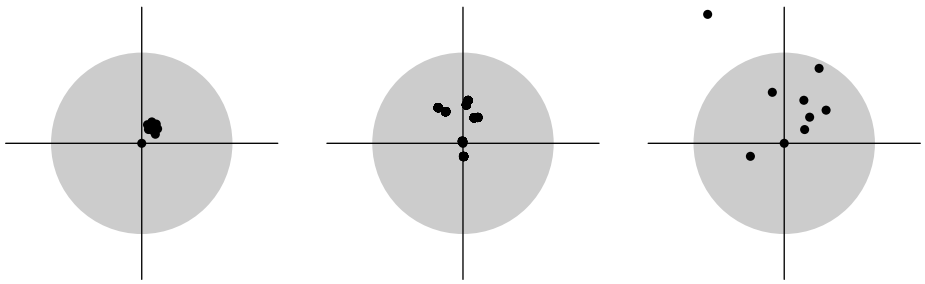
Qu'est-ce que l'ensemble de Mandelbrot ? Si (a, b) est un couple nombres réels fixé, on définit la fonction $f_{a,b}$ qui déplace chaque point du plan vers un autre point. Si A est un point de coordonnées (x, y) alors $f_{a,b}(A)$ est le point de coordonnées $(x^2 - y^2 + a, 2xy + b)$.



On peut imaginer un point A comme étant l'état d'un système à un instant t. Son image $f_{a,b}(A)$ est l'état du système à l'instant $t+1$.

Imaginons qu'à l'instant initial $t = 0$, le système soit situé à l'origine $A_0 = (0, 0)$. Observons l'évolution de ce point au cours du temps : si A_t est la position du point à l'instant t, alors $A_{t+1} = f_{a,b}(A_t)$ est sa position à l'instant $t+1$. Ici deux comportements sont possibles : ou bien le point reste éternellement proche (à distance au plus 2) de l'origine, ou bien il finit par s'éloigner (à distance supérieure à 2) et part à l'infini.

Le comportement observé dépend du système, c'est-à-dire des valeurs de a et b.



(a) $(a, b) = (0.3, 0.2)$ (b) $(a, b) = (0.33, 0.57)$ (c) $(a, b) = (0.45, 0.3)$

FIGURE 1 – En (a) et (b), le point reste proche. En (c) le point s'éloigne à l'infini.

Nous allons écrire un programme permettant de colorer chaque point, de coordonnées (a, b), en fonction du comportement : noir si l'évolution reste proche de l'origine, d'une autre couleur si le point s'éloigne (la couleur dépendra alors de l'instant auquel cela arrive). C'est cette image que l'on désigne par ensemble de Mandelbrot.

Exercice 1 : La fonction $f_{a,b}$.

Ecrire la méthode `double fx(double x, double y, double a, double b)` qui renvoie l'abscisse de l'image par $f_{a,b}$ du point de coordonnées (x, y).

Ecrire la méthode `double fy(double x, double y, double a, double b)` qui renvoie l'ordonnée de l'image par $f_{a,b}$ du point de coordonnées (x, y).

Exercice 2 : Couleur du point (a, b).

Ecrire la méthode `int calculeCouleur(double a, double b)` qui renvoie la couleur qui doit être attribuée au point (a, b).

Il s'agira de partir du point $(0, 0)$, de calculer la suite de point en appliquant $f_{a,b}$ successivement. Si les points restent toujours proches, on renvoie la couleur noire, sinon on renvoie une couleur dépendant du nombre d'application de $f_{a,b}$ au bout duquel le point s'est éloigné.

On vous fournit la méthode `int conversionCouleur(int nbIter)` qui renvoie cette dernière couleur.

Evidemment pour savoir si le point reste éternellement proche de l'origine, il faudrait calculer son évolution sur un temps infini, ce qui n'est pas envisageable. On fixe alors une borne `nbIterMax`, et l'on observe la trajectoire jusqu'à cet instant seulement. Au départ, `nbIterMax` vaut `200`, mais cette valeur pourra être modifiée lors de l'exécution.

Exercice 3 : Remplissage de l'image.

Ecrire la méthode `void remplitImage(BufferedImage image)` qui colore chaque pixel de l'image. Une image (objet de classe `BufferedImage`) de dimensions `largeur` et `hauteur` est donnée en paramètre.

On vous fournit les méthodes

— `double xCoord(double i)`

— `double yCoord(double j)`

qui convertissent les coordonnées entières d'un pixel (abscisse entre `0` et `largeur-1`, ordonnées entre `0` et `hauteur-1`) en coordonnées réelles.

Il faut parcourir tous les pixels de l'images. Pour chaque pixel, il faut convertir ses coordonnées pour obtenir un point (a, b). Il faut ensuite appeler la méthode `int calculeCouleur(double a, double b)` pour calculer la couleur du pixel. Enfin il faut colorer le pixel. Pour cela on utilisera la méthode `void setRGB(int i, int j, int c)` de la classe `BufferedImage` qui colore le pixel de coordonnées (i, j) avec la couleur c.

Exécution.

Au cours du TP, compiler le fichier `Mandelbrot.java`. Lorsque vous avez terminé, compiler et exécuter le fichier `MandelbrotFrame.java`. Vous pouvez alors parcourir l'image avec les flèches, zoomer avec les touches + et -, jouer sur la précision avec p et m.