IF112 - Projet d'Informatique
2023 - 2024

Yannick Bornat - yannick.bornat@enseirb-matmeca.fr
Guillaume Bourmaud - guillaume.bourmaud@enseirb-matmeca.fr
Clémence Gillet - clemence.gillet@enseirb-matmeca.fr
Rémi Giraud - remi.giraud@enseirb-matmeca.fr

TP1 : Git - Gestion de versions
Lecture / Écriture dans un fichier

Les objectifs de cette séance sont :
- Savoir utiliser Git, un logiciel de gestion de version.
- Savoir lire, créer et écrire dans un fichier sur le disque à partir d'une représentation en mémoire. On appliquera à l'écriture d'une image, au format PPM.

1 Utilisation de Git

Quand on programme, potentiellement à plusieurs, sur des projets un peu plus conséquents on va très souvent utiliser un système de gestion de version décentralisé. Ce type de logiciel va vous permettre de travailler à plusieurs en même temps sur un projet ainsi que de conserver en ligne un historique des versions, vous permettant notamment de revenir en arrière en cas de problème.

Suivez entièrement le tutoriel qui explique le fonctionnement du logiciel de gestion de version Git, qui est un des plus utilisés au monde : `https://thor.enseirb-matmeca.fr/ruby/docs/repository/git`

Vous y trouverez une vidéo explicative ainsi que toutes les instructions pour créer votre dépôt et manipuler les révisions. Vous utiliserez ce dépôt pour vos rendus de code à chaque fin de séance. Vous trouverez votre dépôt dans l'onglet Repository du Projet IF112 - G#.

Un rappel des commandes est également disponible ici : `https://thor.enseirb-matmeca.fr/ruby/docs/repository/git-cheat-sheet`

2 Représentation d'une image

Dans l'ordinateur, un codage très répandu pour les images est de mémoriser, pour chaque point (ou pixel), les composantes rouge, verte et bleue qui composent sa couleur (codage RGB pour Red/Green/Blue). Chaque composante de couleur est codée par un octet (0 : aucune contribution, 255 : contribution maximale). Ainsi, un pixel noir sera codé par (0, 0, 0), un pixel rouge par (255, 0, 0), un pixel vert par (0, 255, 0), un pixel bleu par (0, 0, 255) et un pixel blanc par (255, 255, 255). Toutes les nuances intermédiaires sont autorisées pour permettre environ 16 millions de teintes différentes.

Par convention, les pixels sont organisés par lignes. Pour chaque ligne, les pixels sont mémorisés de gauche à droite. Les lignes sont rangées de haut en bas. Ainsi, pour mémoriser une image de 10 pixels de large par 10 pixels de haut, on utilisera un tableau de taille 300 (largeur 10 x hauteur 10 x 3 composantes). Chaque composante de couleur étant codée sur 8 bits, elles seront mémorisées dans des types char. La figure 1 illustre l'ordre de codage des valeurs.

Bien entendu, cette représentation est incomplète, puisqu'elle ne rend pas compte de la géométrie de l'image (ses dimension entre autres). Ces valeurs seront donc mémorisées à part.

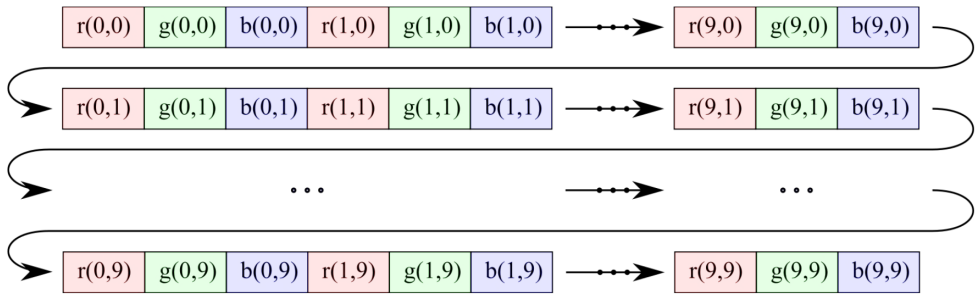

Figure 1: Ordre de codage des composantes de pixel

3 Le format PPM

Un fichier image est un fichier comme un autre, mais dont les données codent, selon un certain format, la représentation d'une photo, d'un dessin ou d'une figure quelconque. À partir des années 80, les systèmes de chez Microsoft ont utilisé une extension sur trois lettres à la fin du nom de fichier pour aider à en identifier le format. Cette habitude est restée, mais n'est pas nécessaire, seule l'organisation des données dans le fichier a de l'importance. Parmi les formats d'image les plus connus, figurent le format "bitmap" de chez Microsoft, dont l'extension usuelle est .BMP, ou le format ISO/CEI 10918-1, couramment appelé JPEG, dont l'extension est généralement .JPG. Pour chaque format, les premiers octets forment un en-tête dont le seul rôle est de permettre l'identification dudit format (indépendamment du nom de fichier, et donc de son extension). Nous allons nous intéresser au format Portable Pixmap, associé à l'extension .PPM, qui permet de programmer rapidement des fonctions de lecture et d'écriture. Un fichier de ce format contient toujours (dans l'ordre) :

- deux octets d'en-tête au format ASCii, il s'agit de la chaîne "P6"
- un octet de séparation (nous prendrons le caractère espace : ' ')
- la largeur de l'image en pixels, codée au format ASCii
- un octet de séparation (encore le caractère espace : ' ')
- la hauteur de l'image en pixels, codée au format ASCii
- un caractère de séparation (toujours le caractère espace)
- l'intensité maximale des composantes couleur, au format ASCii. On prendra la valeur "255"
- un dernier caractère de séparation (un caractère espace)
- les valeurs des composantes de pixels ($r_{(0,0)}$, $g_{(0,0)}$, $b_{(0,0)}$, $r_{(1,0)}$, $g_{(1,0)}$, ...) au format **binaire**, mises les unes derrière les autres.

NB : Le format binaire consiste à écrire les données de façon brute sans chercher à les représenter de façon lisible par l'utilisateur. Ce format est plus efficace en termes de taille mémoire et de vitesse de traitement. Pour écrire les données au format binaire, les fonctions les plus appropriées sont `fputc` ou `fwrite`. Avec ce format, une valeur comprise entre 0 et 255 n'occupe qu'un octet dans le fichier.

4 Création d'images simples

Écrivez un programme qui crée un fichier image de 10 par 10 pixels contenant une couleur unie de votre choix. Si le format est correct, la taille du fichier doit être de 313 octets[1].

Vous pouvez alors ouvrir votre fichier avec la commande `eog` sous Linux.

[1]Il peut arriver que sur les systèmes Windows, ce fichier fasse 316 octets. Cela vient d'un codage des sauts de ligne spécifique. Si tel est votre cas, remplacer les sauts de lignes par des espaces devrait corriger le problème. En tout état de cause, il est conseillé d'utiliser les machines de l'école sous Linux.

Dans le cas d'un TP à distance, il est conseillé de d'abord récupérer l'image sur votre ordinateur, puis de l'ouvrir.

- Sous Linux, selon votre distribution, vous pouvez utiliser l'un des programmes suivants : `eog`, `gnome-open`, `kde5-open`, `feh`, `gwenview`.
- Sous Windows, Si le format n'est pas connu, vous pouvez installer *irfanview* (`https://www.irfanview.com/`) qui est gratuit en usage personnel (lors de l'installation, vous pouvez lui interdire de se référencer comme programme par défaut pour visionner les images, de cette façon, il ne perturbera pas vos réglages précédents).
- Le support d'Apple pour ce format est inconnu, une mise à jour de ce document sera proposée dès que possible.

Si votre programme de visualisation affiche votre couleur unie en tant qu'image, félicitations.

Écrivez maintenant un programme capable de créer une image de taille un peu plus conséquente (600 pixels x 400 pixels), représentant un disque d'un diamètre 200 pixels en son centre. Si votre image s'affiche correctement, vous savez créer une image.

Usage général : Un outil très pratique est *ImageMagick*, qui permet la conversion simple de fichiers image. Ainsi, la commande `convert image.ppm image.jpg` vous permettra de convertir votre image au format .jpg pour réduire sa taille sur le disque et y accéder plus facilement. Vous pouvez même appeler cette commande directement depuis votre programme C grâce à la fonction `system` (`man 3 system` pour plus de détails).

5 Création d'images complexes (ensemble de Mandelbrot)

Pour illustrer la création d'images complexes, nous allons utiliser la figure de l'ensemble de Mandelbrot. Qui est une figure fractale. L'objectif sera donc de calculer cette figure, puis de créer un fichier permettant de la stocker sur le disque. On considère la suite complexe définie par l'équation suivante :

$$U(0) = Z; \quad U(n+1) = U(n)^2 + Z \tag{1}$$

L'ensemble des Z tels que $U(n)$ converge s'appelle l'ensemble de Mandelbrot. Dès qu'un terme a un module supérieur ou égal à 2, la suite diverge. En pratique, on calculera jusqu'au 85e terme, si le dernier terme calculé a un module inférieur à 2, on considérera que la suite converge.

5.1 Calcul pour un point donné

Écrivez une fonction `convergence`, dont les arguments sont deux flottants x et y (Z = x + iy) et retournant 0 si la suite converge, le nombre d'itérations sinon. Pour aider à la validation, voici quelques valeurs que doit renvoyer convergence :

X	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
Y	2.0	1.5	1.0	0.5	0.4	0.35	0.3	0.29	0.28
convergence	0	1	2	4	6	9	34	58	0

5.2 Codage des couleurs et écriture de l'image

Pour coder les couleurs, vous choisirez une relation simple comprenant deux composantes constantes, la troisième étant le triple de la valeur obtenue par la fonction convergence. Par exemple, si est une valeur de convergence, la couleur associée sera : (r,g,b) = (0, 3*c, 255)

Écrivez le programme qui balaye chaque pixel de l'image. Pour chaque pixel, votre programme calculera les coordonnées du point du plan correspondant, la valeur de convergence pour les coordonnées ainsi calculées, et finalement la couleur du pixel considéré. Cette couleur sera écrite dans le fichier.

Pour calculer les équations de projection, vous pourrez vous aider de la Figure 2. De manière générale, prenez l'habitude de toujours travailler avec papier/stylo pour faire des schémas et vous aider à représenter les structures.

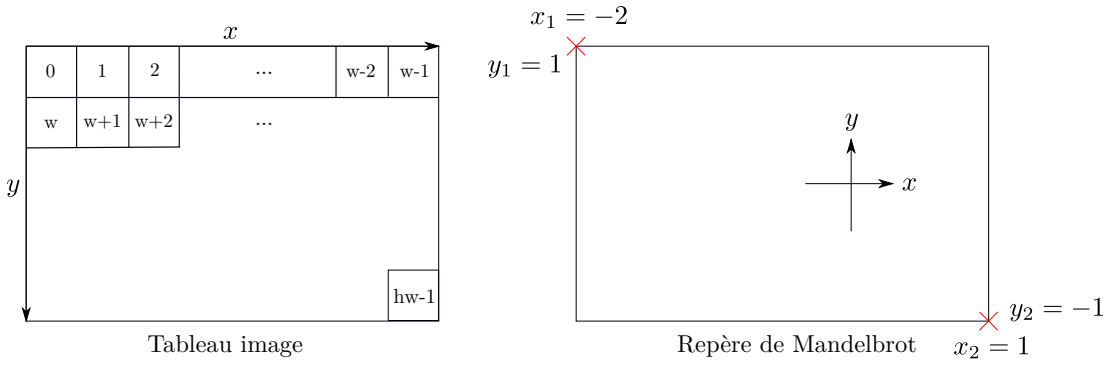

Figure 2: Visualisation du repère de Mandelbrot dans le cas $x_1 = -2$, $y_1 = 1$, et $x_2 = 1$, $y_2 = -1$.

Paramètres : Pour vous assurer d'une image finale correcte, vous prendrez des dimensions de 900 pixels (largeur) par 600 (hauteur). Le pixel en haut à gauche représentera les coordonnées ($x_1 = -2$, $y_1 = 1$), et le pixel en bas à droite représentera les coordonnées ($x_2 = 1$, $y_2 = -1$). Pour aider les futures modifications de votre programme, définissez ces valeurs comme des constantes à l'aide de `#define`.

6 Bonus : Création de séries d'images

6.1 Images indépendantes

Modifiez le programme de la question 4) pour que, dans une boucle de 10 itérations, les points (x_1, y_1) et (x_2, y_2) se rapprochent lentement du point $(-0.99, 0.3)$. Chaque itération créera un fichier image (im0.ppm à im9.ppm). Faites attention à respecter les proportions. Après exécution, en observant successivement les images, un effet de zoom apparaît.

6.2 Assemblage en vidéo

Les ordinateur de l'école sont équipés de l'outil `ffmpeg` qui permet de rassembler plusieurs images en une vidéo. La syntaxe minimale en ligne de commande est : `ffmpeg -i im*.ppm video.mpg`

Assemblez vos images en une vidéo finale. Si vous utilisez plus de 10 images, la syntaxe devient la suivante (pour un numéro sur trois caractères) : `ffmpeg -i im%03d.ppm video.mpg`

Pour plus d'informations, consultez la page de manuel : `man ffmpeg`.

Certaines options sont intéressantes en plus d'être parfois nécessaires selon les codecs installés sur votre machine : `-framerate <r>`, `-qscale 0`, `-vcodec libx264`.