Clément MUTEZ Groupe F

Problème scientifique informatique

Je suis tout seul à effectuer mon projet. Je l'ai effectué tout au long des séances ainsi que chez moi. Au début j'ai surtout travaillé pendant les heures de TD, et très peu chez moi, mais à la fin j'ai surtout travaillé chez moi car j'avais accumulé du retard. J'ai passé 12h à travailler en td et environ 15-20h à travailler chez moi.

Afin de tester les fonctions une à une, j'ai mis à disposition des **Exotests**, qui permettent de tester chaque fonction, il suffit juste d'enlever et de rajouter les doubles slashs dans le main. Ces fonctions **Exotest** m'ont permis tout au long de mon projet de pouvoir tester mes fonctions. La fonction qui permet de tout effectuer s'appelle Program. Elle permet de tester toutes les fonctions sur toutes les photos.

Je vais maintenant vous expliquez le déroulement de mon projet et je vais vous expliquer le fonctionnement de certaines de mes fonctions.

J'ai commencé par créer ma classe **MyImage**, c'est je pense ce qui m'a pris le plus de temps. En effet comprendre comment le bitmap fonctionne prend beaucoup de temps, mais une fois compris les fonctions qui suivent se font plus rapidement. Aussi les fonctions de conversions m'ont pris du temps surtout pour la fonction int vers endian. Aussi j'ai directement créé ma classe Pixel afin de pouvoir créer une matrice de pixel, cela permet de facilité la suite pour faire les traitements d'image.

Dans fonction **MyImage**, j'ai donc commencé par créer un constructeur qui, à partir d'un fichier bitmap, le traduit dans ma classe afin que je puisse la traiter par la suite. Dans ce constructeur, j'identifie les valeurs de chacun de mes attributs de mon champ et je créer ma matrice Image, constitué de pixel (de la classe **Pixel**). Donc pour ce constructeur, je vais pour chaque attribut traduire de **endian-to-int** la valeur de l'attribut (que j'ai créé par la suite), et ensuite j'ai créé une boucle qui parcours mon tableau et ma matrice image afin de pouvoir créer ma matrice image à partir de mon tableau de bitmap.

Pour mes propriétés, je ne les ai pas créées tout de suite, mais seulement quand j'en avais besoin. J'ai donc au final rendu accessible et modifiable les attributs suivants : la taille du fichier, la largeur, la hauteur et la taille de l'Offset, car pour certaines fonctions, ces attributs sont modifiés.

J'ai ensuite créé mais deux fonctions de conversion. Ma fonction de conversion de **endian-to-int** est simple et remplie juste un tableau de taille 4 (afin d'être réutilisable dans la fonction **from-image-to-file**) avec une boucle. Mon autre fonction de conversion est en revanche plus compliqué. Pour celle si j'ai séparé 4 cas différents : cas où la valeur à convertir est inférieur à 256^1, inférieur à 256^2, inférieur à 256^3 et supérieur à 256^3. Je vous laisse ensuite regarder mon programme pour le comprendre.
Je les ai tout de suite testées dans **Exotest2**. Une fois testée et validée, je les ai introduites dans mon constructeur.

J'ai ensuite créé ma fonction **from-image-to-file**. Dans cette fonction j'ai effectué exactement l'inverse du constructeur, à l'aide de la fonction de conversion.

J'ai ensuite testé si tout fonctionnait bien dans **Exotest3**.

Après, j'ai créé les fonctions **Noir-et-blanc** et **Nuance-de-gris** assez rapidement. Pour **Noir-et-blanc**, j'ai juste fait deux cas, si la valeur de la couleur est supérieure ou inférieure à 128. Pour **Nuance-de-gris**, j'ai juste fait la moyenne des couleurs rouge, vert et bleu pour chaque pixel. Je les ai tout de suite testées dans ma fonction **Exotest4.**

Ensuite, pour la fonction **Miroir**, **Agrandir**, **Rétrécir** et pour la fonction **Rotation**, j'ai créé une matrice temporaire afin de pouvoir stocker la matrice modifiée dedans, et à la fin de la modification j'égalise les deux matrices. Pour faire ses fonctions j'ai d'abord fait des schémas au brouillon, ce qui m'a beaucoup aidé pour faire mes boucles et mes compteurs.
Pour la fonction **Miroir**, j'ai juste fait une boucle qui permet de parcourir la matrice image, et à l'aide d'un compteur et des bornes de la boucle, stocker la matrice « avec l'effet de miroir » dans la matrice temporaire.
Pour la fonction **Agrandir**, j'ai d'abord créé une matrice temporaire qui faisait deux fois la taille de la matrice image. Ensuite à l'aide d'une boucle je parcours la matrice image, et pour une case parcourue dans la matrice image, j'en remplis 4 dans la matrice temporaire (à l'aide de compteur).
Pour la fonction **Rétrécir**, je créer d'abord une matrice temporaire de taille 2 fois plus petite que la matrice image. Ensuite à l'aide d'une boucle je parcours la matrice temporaire, et je remplis une case de matrice temporaire avec « une case parmi 4 » de la matrice image, à l'aide du compteur qui lui avance de 2 en 2. J'ai dû créer avec cette fonction, la fonction : **Fonction-pour-retrecir** qui permet de vérifier si, une fois l'image rétrécie, cette image possède bien une largeur multiple de 4. Elle rajoute des colonnes de pixel noir pour compléter l'image pour qu'elle possède une largeur multiple de 4.

Je me suis ensuite attaqué aux filtres. Pour créer mes filtres j'ai utilisé deux fonctions : **MultiplicationMatriceConv** et **CréerLaMatriceContour**.
La fonction **CréerlaMatriceContour** permet de créer la matrice qui entoure le pixel de coordonné (i,j), en prenant en compte les 9 cas différents possibles. Dans les cas limites, la matrice contour possède des pixels de valeurs (0,0,0) pour ne pas influencer le résultat final. Au final la fonction retourne une matrice de pixel.
La fonction **MultpiplicationMatriceConv** permet d'effectuer la multiplication de la matrice contour par la matrice du filtre (selon la multiplication de la convolution). Je demande en paramètre le filtre, car il y a une différence si le filtre est Flou. Cette fonction retourne le pixel modifié.
Ensuite pour chacun de mes filtres (**DetectionDeContour, RenforcementDesBords, Repoussage** et **Flou)**, je créer la matrice du filtre, je créer une matrice temporaire pour stocker la matrice « filtré », et dans ma boucle qui parcours toute la matrice, je créer une matrice contour, je la multiplie à la matrice filtre et le rentre le pixel modifié dans la matrice temporaire pour chaque (i,j).

Pour créer ma **fractale**, j'ai créé une classe **Blanche** qui permet de créer une image de taille et de couleur voulue. Pour ce faire j'ai environ reproduit ma fonction **From-image-to-file**, mais avec seule donné la hauteur et la largeur de l'image.
Avec la classe **Blanche** je créer une image de taille 270*240 (d'après ce que je me suis renseigné sur internet pour faire une fractale) et donc de couleur blanche.

Ensuite une fois l'image créer, je la traite dans ma classe **MyImage**. Donc dans cette classe, je créer ma fractale dans la fonction **Fractale**. Dans celle-ci, je reproduis l'algorithme de Mandelbrot mais traduit en C#. Pour pallier le problème que C# ne traite pas les nombres imaginaires, j'ai fait la multiplication « à la main ».

Ensuite pour ma fonction **Histogramme**, je créer d'abord une matrice temporaire. Ensuite je parcours ma matrice image de colonne en colonne.

- Pour chaque colonne, je somme la valeur de rouge, vert et bleu de chaque pixel.
- Ensuite je fais la somme totale de chaque somme (rouge, vert et bleu), afin ensuite de pouvoir faire le pourcentage de « répartition » entre chaque somme de pixel.
- Ensuite une fois que j'ai ce pourcentage, je regarde combien il représente en termes de pixel par rapport à la taille de la colonne.
- Et enfin je remplis les colonnes de pixel rouge, vert ou bleu, en fonction du nombre de pixel calculé précédemment.

J'ai testé toutes mes fonctions durant l'avancement de mon projet. Je les ai ensuite insérées dans mon programme principal.

J'ai voulu à un moment créer une fonction Complexe pour faire ma fractale, qui après réflexion n'était pas utile, malheureusement je n'ai pas réussi à supprimer cette classe sans que cela ne gêne ma compilation. Je l'ai donc laissé, mais elle inutile pour mon projet.

Enfin je n'ai malheureusement pas eu le temps de traiter le TD5.

Clément MUTEZ Groupe F.