**PSI : Projet image**

**Présentation Projet**

L’idée est de créer un outil qui lit une image dans un format donné (bitmap ), traite cette image (agrandir, rétrécir…) et sauvegarde l’image dans un fichier de sortie différent de celui donné en entrée (toujours format Bitmap).

**Structure du code**

Pour cela, il faut d’abord lire tous les bits du fichier et les stocker dans un autre tableau de bit qu’on utilisera pour récupérer les infos concernant l’image un par un. Les pixels sont récupérés par trois (bleue, vert et rouge) et sont associés à un emplacement précis dans une matrice de pixels.

Ainsi, nous avons créé 4 classes en plus du Main :

Classe - Pixel

Un pixel est constitué de 3 composants qui sont les 3 couleurs primaires et qui correspondent donc à mes attributs : des entiers compris entre 0 et 255 pour le bleue, le vert et le rouge qu’on obtient à travers le constructeur en entrant les valeurs, ou bien en copiant un autre pixel.

Ces attributs sont accessibles en lecture (pour récupérer les valeurs et les traiter selon les besoins), et en écriture (effectuer les modifications nécessaires).

Les méthodes permettent de modifier la couleur envoyée par le pixel en gris (selon la moyenne de la somme des 3 couleurs primaires) ou bien en noir/blanc en modifiant les valeurs des 3 attributs à l’identique.

Classe - MyImage

Cette classe est assez imposante car c’est là que la récupération de l’image va se faire et le transfert dans une matrice de la classe Pixel, mais aussi les traitements d’images et la restitution dans un fichier de sortie différent de celui donné en entrée.

Les attributs sont donc les informations propres à l’image tel que le nom, type d’image, taille du fichier, dimensions… qu’on affichera mais aussi sur lesquels on apportera des modifications ce qui implique de donner accès en lecture mais pas en écriture car on ne les traites pas en dehors de la classe.

On retrouve aussi un tableau contenant tous les bits et un autre contenant seulement le header, mais aussi une matrice de la classe Pixel qui contiendra les valeurs de RVB.

Un constructeur récupère donc toutes ses informations et les organises, un autre est vide car il est utile dans le cas de la création d’image (rien à récupérer).

Les méthodes de traitement d’image sont publiques car elles seront lancées depuis le Main.

Complex: classe servant au calcul des coefficients pour afficher la fractale.

QR code : Classe incomplète, nous n’avons pas réussi à faire le Qr code. Ainsi, il y une conversion du messagee en bit mais pas d’affichage de notre QR code dans cette classe.

**Bilan de projet**

Tout au long de ce projet, plusieurs connaissances acquises lors de notre formation universitaire ont été mises à contribution. Les notions des cours de POO nous a donné une base pour pouvoir concevoir notre code.

Malgré le fait que nous manquions de connaissances sur les méthodes de traitement d’image, cela nous a permis d’en apprendre davantage à ce sujet en l’utilisant et en cherchant les ressources pour répondre au mieux à nos nombreux problèmes de modélisation.

**Travail d’équipe :** Pour ce qui est du travail en équipe, cette partie s’est bien déroulée malgré les horaires chargés des deux membres de l’équipe. La communication entre les parties était adéquate, notamment via Messenger et les rencontres prévues hebdomadairement entre les membres ont été respectées. Nous avons manqué de ressources à quelques reprises ce qui nous a freinés sur le plan d’avancement du projet. Cependant, nous avons fait de notre mieux pour effectuer ce projet au mieux de nos connaissances.

**Respect de l’échéance :** Nous avons respecté les délais de rendus. Cependant, le nombre d’heures prévues pour ce projet a été grandement dépassé dû aux imprévus rencontrés lors de la réalisation de celui-ci.

**Compression d’image**

La compression d'image est une application de la [compression de données](https://fr.wikipedia.org/wiki/Compression_de_donn%C3%A9es) sur des [images numériques](https://fr.wikipedia.org/wiki/Image_num%C3%A9rique). Cette compression a pour utilité de réduire la redondance des données d'une image afin de pouvoir l'emmagasiner sans occuper beaucoup d'espace ou la transmettre rapidement.

La compression d'image peut être effectuée avec perte de données ou sans perte.

La compression sans perte est souvent préférée là où la netteté des traits est primordiale : schémas, dessins techniques, icônes, bandes dessinées.

La compression avec perte, plus radicale, est utile pour les transmissions à bas débit, mais dégrade la qualité de l'image restituée.

Les méthodes de compression sans perte sont également préférées là où la précision est vitale : balayages médicaux ou [numérisations](https://fr.wikipedia.org/wiki/Num%C3%A9risation) d'images pour [archivage](https://fr.wikipedia.org/wiki/Archivage_%C3%A9lectronique). Les méthodes avec perte restent acceptables pour des photos dans les applications où une perte mineure de fidélité (parfois imperceptible) est tolérée pour réduire les coûts de stockage ou d'envoi.

**Pourquoi peut-on compresser ?**

Parce qu'une image où chaque point serait parfaitement indépendant des autres n'aurait pour nous pas d'intérêt : une image ne nous est utile que si elle contient des [corrélations](https://fr.wikipedia.org/wiki/Corr%C3%A9lation), qui dès lors qu'elles existent peuvent permettre les compressions en question.

**Liens :**

<https://fr.wikipedia.org/wiki/Compression_d%27image>

<http://igm.univ-mlv.fr/~dr/XPOSE2013/La_compression_de_donnees/jpeg.html>

<https://images.math.cnrs.fr/Compression-d-image.html>

<https://interstices.info/de-fourier-a-la-compression-dimages-et-de-videos/>

<https://cagnazzo.wp.imt.fr/files/2013/05/compression_05_image.pdf>

<http://serge.wacker.free.fr/technoprimaire/c2i/revisions/formats_image.pdf>