PROJET SCIENTIFIQUE INFORMATIQUE : RAPPORT

Pour ce second semestre d’année 2 à l’ESILV, nous devions réaliser un projet de traitement d’image afin d’accroître nos connaissances et nos pouvoirs de de réflexion en langage en C#.

# Explication des structures de données

## Classe Pixel

Pour simplifier le traitement ou la création d’image, nous avons voulu travailler en pixel. Cette classe sera construite à partir d’un ensemble de 3 bytes rouge, vert et bleu.

Dans cette classe, nous avons inclut une méthode MoyennePixel qui retourne la moyenne des valeurs des trois pixels afin de simplifier les calculs dans d’autres classes.

## Classe MyImage

Nous avons créé cette classe afin d’appliquer différents traitements à une image déjà existante, renseignée par l’utilisateur. Le constructeur va récupérer les données de l’images (notamment sa largeur, sa hauteur, la place qu’occupe ces données) qu’il va mettre dans un tableau d’octet. A partir de ce tableau, nous avons décidé de traiter l’image via une matrice de pixels.

Voici une liste des différents traitements basiques possibles :

* Copie de l’image
* Nuances de gris
* Noir et blanc
* Rétrécir
* Agrandir
* Rotation selon n’importe quel angle
* Miroir par rapport à l’un des côtés de l’image, au haut ou au bas de l’image et symétrie par rapport à l’un de ses angles.

Nous avons aussi donné la possibilité à l’utilisateur d’appliquer un filtre ou deux à une image. Il pourra notamment rendre son image floue, améliorer sa netteté, détecter ses contours, atténuer les bruits de l’image et faire apparaître ses dimensions.

L’utilisateur aura également la possibilité de cacher une image dans une autre, ou bien de dissocier deux images.

Nous avons également ajouté quelques innovations que l’on vous présentera plus bas.

## Classe Histogramme

Cette classe permet à l’utilisateur de mieux connaître les statistiques de son image via les histogrammes de la luminosité, du rouge, du vert et du bleu. Ces données ressortiront sous la forme d’un histogramme dont la hauteur sera définie par l’utilisateur mais dont la largeur sera imposée par le programme. Dans cette classe nous n’avons pas eu besoin des informations du header de l’image.

## Classe Fractale

Cette classe va nous permettre de construire deux types de fractales : la fractale de Mandelbrot et celle de Julia. Nous laisserons à l’utilisateur la responsabilité de choisir la hauteur et la largeur de la fractale. En revanche nous avons dû créer une image dans sa totalité : l’image en elle-même comme ses headers.

## Classe QrCode

Pour répondre au besoin de l’utilisateur, nous avons dû lui permettre de créer un Qr Code de dimension 21\*21 ou 25\*25 modules ou d’en lire un de ces mêmes dimensions. Ainsi tous les messages de moins de 45 caractères pourront être retransformés en Qr Code ou décodés. Pour pouvoir la mettre en œuvre, nous avons dû exploiter les classes de ReedSolomon.

## Classe Program

Dans cette classe nous avons créer l’interface utilisateur afin de simplifier l’expérimentation du programme au maximum. En lançant le programme, l’utilisateur aura le choix entre 6 types de créations ou de traitements d’images. En sélectionnant dans le menu principal une de ces options, il aura accès à un choix plus ciblé.

Nous avons tenté de limiter au maximum les sources d’erreurs pouvant venir de l’utilisateur qui ferait « planter » le programme. Par exemple nous avons normalisé le nom des fichiers appelés ou en sortie. Nous avons également sécurisé la sélection d’un menu en redemandant le numéro de l’option choisi en cas d’erreur. Une fois l’option chois, le programme fera appel à la fonction ou à la classe adaptée.

## Les Méthodes de Bases

Pour pouvoir mettre en application les classes précédentes, nous avons dû développer des méthodes basiques. En voici une liste non exhaustive :

* Convertir une variable de type int en une variable de type little endian (et inversement)
* Convertir une variable de type int en binaire (et inversement)
* Convertir les données d’une matrice de pixel en un tableau d’octet afin de les enregistrer au format bitmap
* Trier par insertion
* Déterminer le nombre de pixel correspondant à une nuance de gris ou de couleur dans une image.
* Convertir une variable de type string ou de type char en caractère de type alphanumérique (et inversement)
* Vérifier que le nom du fichier soit bien au format x.bmp

# Explication de l’innovation

Compte tenu des dates rapprochées de nos dernières séances, nous avons jugé que nous n’aurions pas le temps d’arriver à bout de la compression jpg. Nous avons donc décider d’apporter d’autres types de traitements d’image. En voici la liste :

* Négatif d’une image
* Faire ressortir une unique couleur sur une image
* Colorée l’image selon une seule nuance
* Faire une mosaïque avec la même image copié 4 fois mais de couleurs différentes

# Problèmes rencontrés

Nous avons dû adapter la répartition du travail entre nous deux en fonction du contenu demandé à chaque séance. Par exemple, il nous paraissait compliqué, voire impossible, de travailler à deux sur la même méthode de Qr Code. Il nous est donc arrivé de travailler en décalé, c’est-à-dire pendant que l’une travaillait sur le contenu de la séance, l’autre traitait le contenu de la séance suivante.

Ensuite, un réel souci rencontré a été de partager nos morceaux de codes. Nous avons tenté de nous familiariser avec GitHub, sans grand succès. Le premier obstacle rencontré a été l’installation de GitHub en elle-même. En effet, il nous a semblé que le tutoriel qui nous permettait de prendre en main le logiciel avait été fait pour une version antérieure de Visual Studio. Par conséquent, nous pensons qu’il n’était plus adapté à la version que nous utilisons. Ensuite, lorsque nous avons réussi à exploiter ce logiciel, le partage des données a été un autre problème : il est arrivé qu’une partie du code soit effacée définitivement. Pour cette raison, nous avons décidé de nous passer de ce logiciel par la suite. Nous devions donc nous envoyer les fichier .cs à la fin de chaque modifications.

Un autre frein a été que nous étions parties sur un traitement d’image par tableau et non par matrice de pixels. Ainsi, nous avons été contraintes de réadapter nos méthodes sue nous avions déjà codées.

Enfin, nous avons eu besoin de plus de temps que d’habitude pour programmer la création ou la lecture d’un Qr Code alors que nos TDs étaient rapprochés de la deadline finale. Pour plus d’efficacité et pour réussir à finir le projet dans les temps, nous avons dû nous répartir le travail de manière plus large.

# Auto-critique

Finalement nous avons réussi à aboutir ce projet dans les temps, malgré la complexité du sujet. Sur toute l’étendue du semestre, nous avons fourni un travail régulier, et nous estimons avoir toute les deux consacré la même implication.

D’un TD à l’autre, nous réussissions à terminer le contenu demandé lors de la séance précédente, ce qui nous a permis de ne pas prendre de retard et de poser des questions si besoin est. Notons que la méthode de rotation nous a demandé un peu plus de temps et de recherche, c’est donc la seule que nous n’avions pas réussi à terminer pour le TD d’après.

Il nous est arrivé ponctuellement d’avoir une baisse de motivation en ne réussissant pas à faire fonctionner une méthode sur laquelle nous avions déjà passé beaucoup de temps (par exemple la rotation ou le Qr Code). Néanmoins, nous faisions appel à notre coéquipière pour avoir un regard neuf et finalement réussir.

Même si nous avons dû faire face à un changement de professeur et au confinement, nous avons réussi à travailler ensemble à distance : nous nous appelions dés que cela était nécessaire.

En fin de compte, nous serions tout à fait prêtes à recommencer un projet ensemble.

