ESILV Année 2

Projet Scientifique Informatique

Auteurs :
M. Victor Lenoble
M. Guillaume Turcas

Encadrant :
François Boisson

Rapport Final du
Mercredi 17 Avril 2019

1 Le Menu

Le menu est à la fois intuitif et épuré. Il permet d'avoir accès à toutes les fonctionnalités du programme directement en navigant dedans, sans à avoir à toucher au code. Même lorsque l'on voudra changer d'image, il y'aura une option faite pour ça, et si on appuie sur un mauvais bouton, par exemple on va par erreur dans le sous menu Fonctions Simples alors que l'on désirait aller dans le sous menu Fonctions Avancés, nous avons toujours la possibilité, en appuyant sur 0, à retourner au menu précédent. Cela fait partie des quelques touches personnelles que nous avons fait sur ce programme.

2 NoirEtBlanc()

Cette fonction, prend l'image de départ, et prend pour chaque pixel, la moyenne entre les trois valeurs, rouge, vert et bleu. Elle change la valeur du Pixel de départ pour la remplacer par cette moyenne. Une fois cette opération effectuée sur chaque Pixel, l'image qui en sort est en noir et blanc. Elle renvoie une matrice de Pixel.

3 Negatif()

Cette fonction, prend l'image de départ, et prend pour chaque pixel, sa valeur, et elle soustrait cette même valeur à 255. Elle change la valeur du Pixel de départ pour la remplacer par cette nouvelle valeur calculée. Une fois cette opération faite sur chaque Pixel, l'image qui en sort est en négatif, donc chaque couleur est inversée dans l'image. Cette fonction fait partie des fonctions supplémentaires qui nous est demandé au TD5. Elle renvoie une matrice de Pixel.

4 Rotation(int demande)

En fonction de la demande initiale faite dans la class Program, on prend chaque pixel de l'image de départ, et on le place dans une nouvelle image de dimension personnalisée en fonction encore une fois de la demande, à un endroit bien précis, afin qu'au final, nous obtenions une image avec une rotation précise (90, 180, 270). Elle renvoie une matrice de Pixel.

5 Miroir(char demande)

En fonction de la demande initiale faite dans la class Program, on prend chaque pixel de l'image de départ, afin de le placer dans une nouvelle image à un endroit bien précis en fonction si vous demandez un miroir horizontal ou vertical. À la fin on obtient une image avec un "filtre" miroir affecté dessus. Elle renvoie une matrice de Pixel.

6 Redim(int redim)

En fonction de la demande initiale faite dans la class Program, on prend l'image de départ, et soit on supprime un pixel sur deux s'il faut la réduire afin de placer ceux qui restent dans une image deux fois plus petite, soit on fait que chaque pixel prenne deux fois plus de place pour les placer dans une image deux fois plus grande. Elle renvoie une matrice de Pixel.

7 FonctionAvance(int[,] matriceConv, int decal, int div)

La fonction FonctionAvance est celle qui permet l'application des matrices de convolution sur une image. Pour se faire, elle va prendre chaque pixel un à un, et va créer une nouvelle matrice. Ceci se faisant par le passage d'une autre fonction, la fonction PixelToConv qui est celle qui permet de faire en sorte qu'un pixel, devienne une matrice 3*3 de pixel en fonction de ceux qui l'entoure, puis cette matrice 3*3 est multipliée avec la matrice de convolution par le biais d'une autre fonction MultiMat3p3, et elle ne garde que le pixel du centre de cette matrice. Ensuite elle lui applique un diviseur et un décalage en fonction de ce qui a été établi dans la class Program. Appliqué à chaque pixel, et nous avons une nouvelle image qui a subi la matrice de convolution. En fonction de la valeur de cette matrice, plusieurs effets peuvent se produire (flou, repoussage, détection de contour...). Cette fonction retourne une matrice de Pixel.

8 Pixel[,] Fractal()

La fonction Fractal permet se dessiner la fractale de Mandelbrot. On commence par définir la taille de l'image pour connaître le nombre de points du plan complexe à étudier. On définit le nombre d'itérations qui décide si la suite (z =

z*z +c) créée par les nombres z et c converge en ce point. On utilise les calculs sur partie réelle et partie imaginaire pour ne pas avoir besoin de classe nombre imaginaire. Si la suite dépasse 2 on déduit qu'elle diverge. Au bout d'un nombre d'itérations choisi on considère que la suite converge. La fonction retourne la matrice de pixels (en noir et blanc).

9 Histogram()

Cette fonction, qui à partir de l'image de départ, permet de prendre chaque pixel un à un, récolte chaque valeur de ce pixel (la valeur rouge, la valeur verte et la valeur bleue), et le place dans une matrice qui fait 768 pixels de large, et un max de hauteur (le max représentant la valeur la plus de fois représentée dans la matrice de tableau d'entier (de Pixel) qui forme l'image). Elle compte donc la valeur de chaque sous pixel, et l'incorpore dans une image de taille variable. Pour que cette image reste en .bmp, et que le programme puisse lire le header, nous changeons la valeur max, pour en faire un multiple de quatre supérieur à cette valeur. Et c'est tout. L'image est alors créée de toute pièce par le programme, et renvoie une matrice de Pixel.

10 ImageImage(Pixel[,] image2)

La fonction ImageImage permet de cacher une image (de matrice de Pixel image2) dans l'image avec laquelle on travaille actuellement. Elle commence par réduire le nombre de couleurs en réduisant de 8 bits à 4 bits chaque pixel pour notre image initiale, une division par 16 que l'on applique aussi à la deuxième image sans remultiplier par 16 pour la deuxième image. On insère alors chaque pixel de la deuxième image comme résidu, sur les pixels de la première image.
(Note : la taille finale est fixée sur la première image, si la deuxième image est plus grande alors une partie des data est perdue.)

TrouverImageCachee()

La méthode TrouverImageCachee() permet de réaliser l'étape inverse, par des divisions euclidienne on retrouve pour chaque pixel les résidus potentiels d'une image cachée. En les multipliant par 16 on retrouve une image proche de celle qui a été cachée précédemment, parfois décalée ou tronquée. La méthode retourne une matrice de pixels

11 AfficherImage(Pixel[,] imageAffich, string fichier)

Cette fonction, qui prend au départ la matrice de Pixel que toutes les autres fonctions précédentes ont créé, et le fichier Image de départ, afin de créer une nouvelle image à partir de ça. Pour cela il créé un fichier de taille 54 (pour le Header) + la taille de la matrice de Pixel. Ensuite, il remplit le Header à partir du fichier de départ, puis modifie les différentes valeurs en lien avec la taille, à l'aide, encore une fois, de la matrice de Pixel. Ensuite il donne un par un, chaque valeur de Pixel au fichier, à l'aide notamment, de la fonction LitEndian. Une fois que c'est fini, le programme ouvre le ficher.