Table des matières

Structures de Données

Classe MyImage

Attributs

Les attributs de la classe MyImage et leurs significations sont listées dans le tableau ci-dessous :

Attributs	Signification
string image_type	Type de l'image
int offset	Taille offset
int file_length	Taille de l'image
int width	Largeur de l'image
int height	Hauteur de l'image
int bits_color	Nombre de bits par couleur
Pixel[,] image_Pixel	Matrice de pixels
string name	Nom de l'image
string file_path	Nom du fichier où on le trouve

Constructeurs

public MyImage(string myfile)

Le constructeur permet la lecture d'un fichier bitmap et sa transformation en instance de classe MyImage. On convertit l'image en tableau de byte puis on récupère les attributs grâce à ce dernier.

public MyImage(MyImage picture)

Le constructeur permet de récupérer les attributs d'une autre variable MyImage. Cela permet entre autres de faire des copies d'une image.

public MyImage()

Avec ce constructeur, on initialise tous les attributs à 0 sauf certaines valeurs qui ne changent jamais par exemple le nombre de bits par couleur et la taille offset.

public MyImage(string name, string image_type, int offset, int file_length, int width, int height, int bits_color, Pixel[,] image_Pixel)

Avec ce constructeur, on peut crée une nouvelle variable MyImage avec des attributs modifiés ce qui permet de faire apparaitre des nouvelles images.

Méthodes

public void From_Image_To_File()

Prend une instance de MyImage et la transforme en fichier binaire respectant la structure du fichier bitmap. On récupère les attributs de la classe et on les rajoute au fur et à mesure dans une liste de byte qu'on convertira en tableau de byte puis en fichier bitmap.

public static int Convertir_Endian_To_Int(byte[] tab)

Méthode qui convertit une séquence d'octets au format Little Endian en entier. Le paramètre est un tableau de bytes contenant la valeur en Little Endian. On additionne chaque valeur du tableau mais avant l'addition, la première valeur est multipliée par 16^0, la deuxième par 16^2, la troisième par 16^3 et ainsi de suite…

public static byte[] Convertir_Int_To_Endian(int data)

Méthode qui convertit un entier en séquence d'octets au format Little Endian.

public void AfficherFichier()

Méthode permettant d'afficher la structure d'un fichier pour une image.

public string AfficherDonnees()

Méthode permettant d'afficher les attributs de la variable MyImage.

public Pixel[,] Nuances_gris()

Méthode permettant le passage d'une photo couleur à une photo en nuances de gris.

public Pixel[,] Noir_blanc()

Méthode permettant le passage d'une photo couleur à une photo en noir et blanc

public Pixel[,] Miroir_horizontal()

Méthode permettant d'appliquer un effet miroir horizontal à une image

public Pixel[,] Miroir_vertical()

Méthode permettant d'appliquer un effet miroir vertical à une image

public void Retrecir(double facteur)

Méthode permettant de rétrécir une image

public void Agrandir(double facteur)

Méthode permettant d'agrandir une image

public void filtre()

Choix des filtres de la matrice de convolution

private void Convolution(int[,] convolution, bool flou)

Création d'une matrice de convolution

public static double Radian(double degre)

Convertir un angle en degré en radian

public void Rotation(double angle)

Méthode permettant d'effectuer une rotation sur une image selon un angle réel quelconque compris entre 0° et 360°

public double[] Convert_To_Polaire(int x, int y)

Méthode permettant de passer d'un système de cordonnées cartésiennes à un système de coordonnées polaires

public int[] Convert_To_Cartesienne(double[] polaire, double angle, int new_width, int new_height)

Méthode permettant de passer d'un système de coordonées polaires à cartésiennes

public void Histogramme()

Méthode permettant la création d'un histogramme selon 3 bytes Rouge, vert et bleu pour une image

private void RemplirBlanc(Pixel[,] image)

Méthode permettant de remplir de blanc une image

public MyImage CoderImage(MyImage Image2)

Méthode permettant de cacher une image secrète dans une image conteneur selon la stéganographie

public static Pixel MergePixel(Pixel pixel1, Pixel pixel2)

Méthode permettant de fusionner deux pixels en un seul

public void DecoderImage()

Prend une image où une deuxième est contenue dans cette dernière et sépare et retourne ces deux images

public static Pixel[] DetachPixel(Pixel pixel)

Prend un pixel et le sépare en deux grâce au binaire

public static MyImage Fractale_Mandelbrot()

Fonction qui crée une fractale de Mandelbrot

Innovations

Notre projet comporte plusieurs innovations. Voici les fonctions qui la compose :

public void Innovations()

Méthode Innovation. Permet à l'utilisateur de choisir entre les 4 différentes innovations crées.

public Pixel[,] Sepia()

Applique un filtre sepia à l'image

public void Seuillage()

Applique un filtre seuillage à l'image, saturation des couleurs selon R,G,B

public void Negatif()

Fonction retournant une image au négatif, pour chaque pixel par passage au complémentaire au blanc

public void Pixélisation()

Fonction permettant de pixéliser une image en conservant une netteté suffisante

public void Primairisation()

Fonction permettant d'exagérer la saturation des couleurs de chaque pixel les faisant tendre vers le rouge, le vert ou le bleu

Classe Pixel

Attributs

Les attributs de la classe Pixel et leurs significations sont listées dans le tableau ci-dessous :

Attributs	Signification
Byte red	Octet rouge constituant un pixel
Byte green	Octet vert constituant un pixel
Byte bleu	Octet bleu constituant un pixel

Constructeurs

public Pixel(byte B, byte G, byte R)

Constructeur prenant en paramètres la valeur de chaque byte bleu, vert et rouge

Méthodes

public static Pixel En_gris (Pixel pixel1)

Méthode permettant de transformer un pixel d'une image en nuances de gris.

public static Pixel Noir_et_Blanc(Pixel pixel1)

Méthode permettant de transformer un pixel d'une image en noir ou blanc.

Compression d'image

Aujourd'hui, la compression d'image est fondamentale. En effet, tous les smartphones et les caméras utilisent un format JPEG, format compressé d'image. De plus, 90% des images que nous trouvons sur internet sont en JPEG.

L'algorithme de compression d'image JPEG analyse chaque section de l'image, trouve et supprime chaque élément que l'œil humain ne peut pas percevoir.

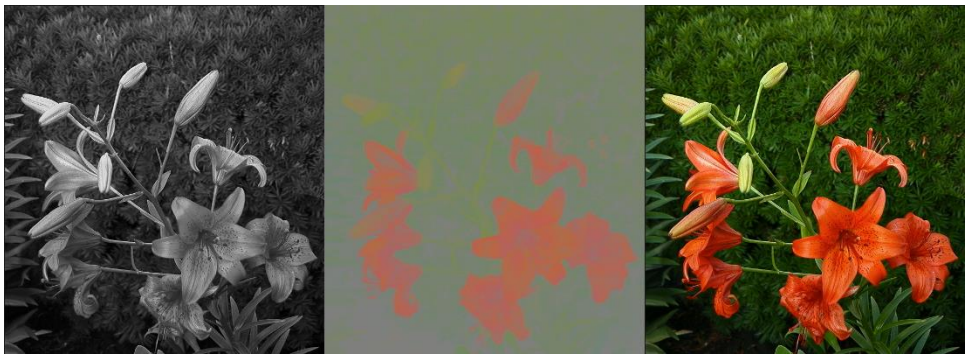
Normalement lorsqu'on cherche à réduire le poids d'image en Mo, une diminution du poids entraine une perte de "détails" donc de qualité pour notre image.

Cependant la résolution de l'image est inchangée, c'est-à-dire qu'on conserve le même nombre de pixels) on obtient en même temps une perte de qualité.

Principe de fonctionnement de la conversion en JPEG :

L'œil humain est constitué de plus de 100 millions de batônnets peu sensibles à la lumière et 6 à 7 millions de cônes sensibles aux couleurs. Trois types de cônes existent, un rouge, un vert et un bleu, comme pour les bytes d'un pixel !

Les batônnets sont beaucoup plus nombreux que les cônes donc notre œil est plus sensible à la luminosité d'une image plutôt qu'à a ses couleurs.



Chrominance

La première étape de l'algorithme de conversion JPEG d'une image est l

L'image originelle est composée de pixels et chaque pixel est composé d'un byte rouge, vert et bleu qui prennent chacun une valeur de l'image a une valeur de 0 à 255

Les bytes rouges sont convertis en byte Y représentant la "Luminance"

Les bytes verts sont convertis en byte Cb représentant la "Chrominance bleu"

Les bytes bleu sont convertis en byte Cr représentant la "Chrominance rouge"

Ces opérations sont réversibles et on ne perd pas de donnée. Voir l'image ci-dessous.