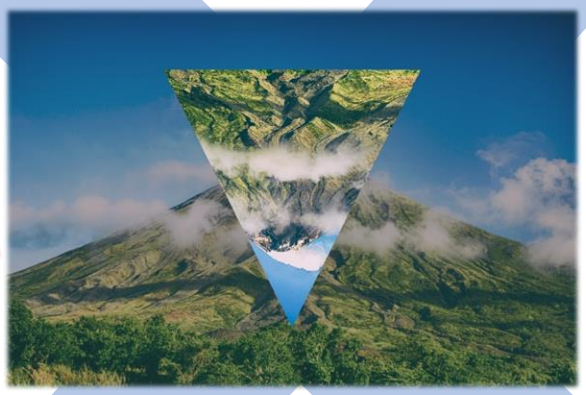
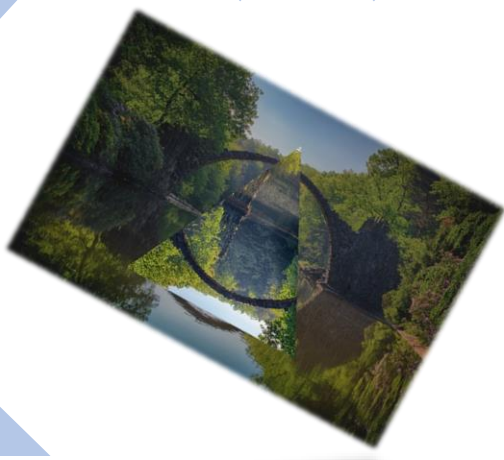
Zajac Alexandre TD A

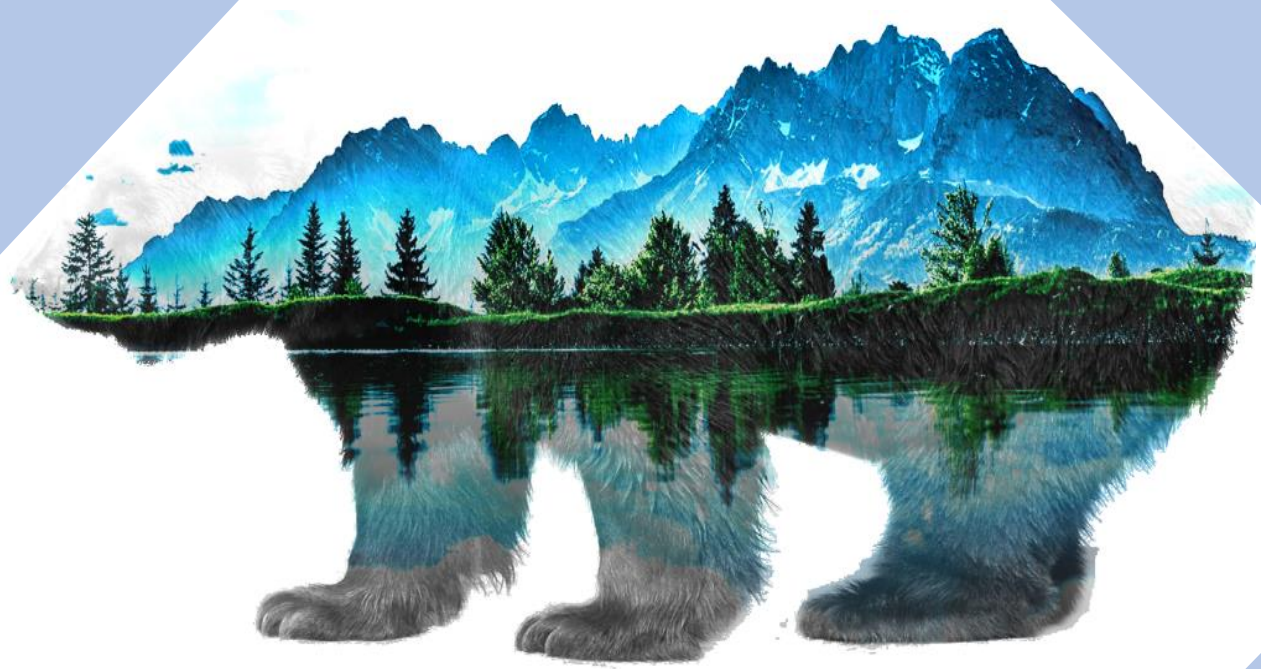
ESILV Semestre 4





# Problème scientifique informatique :

# Projet de traitement d'images en C#



# Structure des données

## Organisation du raisonnement :

Le code à été rédigé en programmation orientée objet en utilisant deux classes :

- La classe MyImage (imposée par le sujet) qui va nous permettre d'effectuer la majorité des traitements sur l'image.
- La classe Pixel qui nous aidera à différencier la nuance rouge, verte ou bleue (si l'on se place dans l'espace RGB) ou Hue (nuance), saturation et valeur (si l'on se place dans l'espace HSV).

***Le traitement d'une image suit le plan suivant*** : déclaration d'une nouvelle instance de la classe MyImage, on remplit la matrice de pixels avec les pixels RGB de manière inverse par rapport au ligne (à cause du format bitmap 24 bits) et ensuite on traite l'image par les méthodes du TD 2, 3, 4 et 5. On écrit ensuite dans un tableau de bytes de sortie nos pixels modifié depuis la matrice de pixels et l'associe au fichier « Resultat.bmp ». Le constructeur fichier va littéralement décomposer le tableau de bytes en entrée du fichier en attributs de la classe, en fonction de la structure d'un fichier bitmap 4 bits.

***Pour la classe MyImage***, en plus des attributs imposés par le sujet, on note l'utilisation de 4 autre attributs qui nous serviront pour la suite du projet : IsASquare, SwitchDimensions, IsAllGrey et BytesToAdd. Le premier nous servira dans le cas de la rotation car la méthode de rotation est différente si l'image est carrée ou rectangulaire. Le deuxième est également utilisé lors de la rotation car s'il faut retourner un rectangle de 90 ou 270 degrés, alors il faut changer les dimensions dans le header, d'où le nom du champ. IsAllGrey détermine si l'image est monochrome en comparant la valeur des pixels lors de la construction de la matrice de pixels correspondante à l'image. Le dernier permet à l'algorithme de traiter des images de n'importe quelle largeur (alors que les bitmap dib 24 bits sont gérées par multiples de 4 bits).

***Pour le TD 1***, nous avons deux méthodes de conversion de format little endian en entier, ce qui va nous faciliter la tâche pour certains traitements. La méthode FromImageToFile(string file) utilise la méthode GetOutputTab() afin d'obtenir à la fin du traitement les pixels modifiés dans la matrice de pixels et de pouvoir écrire notre tableau de bytes résultant dans un fichier de sortie. Une méthode ChangeHeightWidth() permet d'inverser la hauteur et largeur de notre image si nécessaire.

***Pour le TD 2***, l'agrandissement d'une image est proposé pour des multiples de 20, 50 ou 80 %. En effet, on va effectuer l'agrandissement d'une part de la hauteur, puis de la largeur de l'image (d'où deux fonctions différentes). Pour chaque pourcentage, l'algorithme va calculer la plus petite fraction de nombres consécutifs correspondantes (ex : 50% = 1,5 fois l'image = 3/2).

Puis on va effectuer un parcours de notre nouvelle image plus grande, et dès que nous serons à la position du dénominateur de la fraction alors on place un pixel qui a pour valeur la moyenne des deux pixels adjacents. Sinon, on place un pixel déjà existant sur l'image d'origine en prenant en compte le décalage car l'image de résultat sera plus grande. C'est donc une répartition constante des pixels qui va s'opérer. Pour faire en sorte que l'utilisateur puisses agrandir de l'ordre de tous les multiples de 20, 50 ou 80 % il faut donc faire intervenir cet algorithme plusieurs fois (car 40% = 1,4 fois l'image qui n'as pas de fraction de deux nombres consécutifs donc on agrandit de 20 puis de 20). On itère ce raisonnement tant que l'agrandissement demandé par l'utilisateur n'est pas atteint pour les deux dimensions de l'image : il est ici intéressant de voir qu'il peut agrandir sans tenir compte du ratio de pixels. Pour le rétrécissement, c'est le même principe pour 20 ou 50 % uniquement, en raison de la perte de qualité rapide lors du rétrécissement. La méthode pour la rotation prend en paramètre un angle, et est différente selon si l'image est un carré ou un rectangle. Les formules déterminées pour chaque en fonction de la forme de l'image ont été déterminé à la main par récurrence. Le passage en nuances de gris d'une image s'effectue par une moyenne sur chaque pixel de la somme de ses nuances tandis que le passage en noir et blanc s'effectue par le remplacement des pixels ayant une valeur > 128 en moyenne par des pixels blancs et inversement par des pixels noirs. La dernière méthode du TD superpose deux images en faisant la moyenne des valeurs de chaque nuance pour chaque pixel où il y a superposition. On fait attention de superposer la plus petite sur la plus grande dans le cas de différence de taille.

***Pour le TD 3***, il s'agit d'une seule et unique méthode que l'on applique à l'image et qui a pour paramètre le noyau de convolution. On traite les bords en créant une image ayant une taille plus grande que notre objet courant d'une valeur proportionnelle à la taille du noyau. C'est sur cette image que l'on effectue la convolution : Pour chaque pixel du parcours, on effectue la somme de la multiplication des coefficients du noyau avec les valeurs des pixels de chaque couleur, et on la normalise pour ensuite l'affecter au pixel considéré.

***Pour le TD 4***, on a établi un nouveau constructeur de la classe MyImage qui construit une image blanche à partir d'une hauteur et d'une largeur. Pourquoi ? Car à partir de ces deux dimensions on peut aisément déduire toutes les informations nécessaires à la création d'un fichier bitmap. On a ensuite la possibilité de crée un carré, un triangle ou un triangle inversé au milieu de cette image. La dernière méthode permet de crée des histogrammes de n'importe quelle image. Le principe est simple, on crée 3 tableaux de taille 256 (nuances d'un pixel) ou chacune des positions représente la valeur associée à chaque nuance. Ainsi en parcourant toute l'image on récupère l'ensemble des intensités pour chaque nuance et on crée trois nouvelles images dont le maximum en couleur (rouge, verte ou bleue) est la nuance qui la plus grande intensité dans l'image, le reste des intensités est déterminé de manière relative à ce maximum. Si l'image est en noir et blanc, on crée un seul histogramme car les valeurs pour chaque pixel sont identiques.

# Partie Innovation

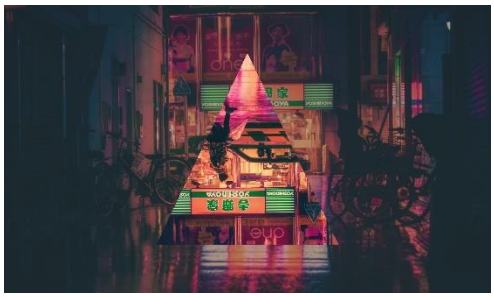
## Organisation des innovations :

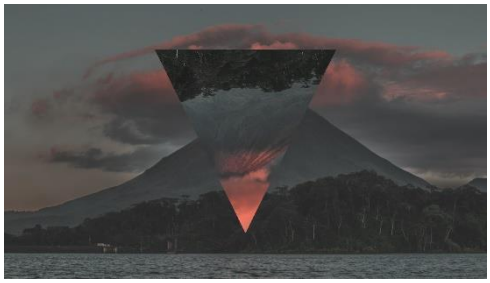
***Choix des innovations*** : Le projet présente trois sous-projets d'innovation. Le premier est la réalisation d'une fusion d'exposition (effet HDR). La deuxième est la réalisation de réflexion géométrique sur n'importe quelle image. Enfin, la dernière innovation est la double exposition avec deux images différentes.

***Pour l'effet HDR***, le traitement est un peu complexe. Il est impossible d'avoir un HDR parfait car il faudrait convertir notre image en 32 bits, faire de la fusion d'exposition et effectuer du tone-mapping pour la reconvertir en 24 bits. L'effet crée ici est donc une fusion d'exposition donnant l'apparence HDR. Cette innovation est proposée sur 3 suites de 4 images (Paris, rose et lake) qui possèdent toutes des expositions différentes. Pour une suite considérée le processus est le suivant : on crée une image caractéristique (Weight map) de l'exposition de chaque pixel pour chaque image de la suite, puis on normalise ces images en fonction de leurs coefficients. Enfin, on applique une transformation linéaire entre l'image caractéristique et son image correspondante dans la suite afin d'obtenir une exposition optimale pour chaque pixel. Voici les résultats (l'image du milieu est le résultat) :

***Pour la réflexion géométrique***, il s'agit d'une seule et même méthode qui permet de réaliser l'effet. Le principe est assez simple : on va créer une instance de la classe MyImage pour n'importe quelle image choisie par l'utilisateur, puis on va lui demander quelle forme il souhaite obtenir (carré, triangle ou triangle inversé). Nous allons dessiner la forme au centre de l'image qui contiendra l'ancienne image retournée de 180 degrés, et pour accentuer l'effet on a choisi d'accentuer le contraste et la luminosité dans la forme, et de les réduire en dehors. Ainsi à l'aide d'un système de masque il est possible de réaliser de jolies réflexions géométriques pour n'importe quelle image.

Voici quelques exemples :





***Pour la double exposition***, le traitement s'appuie sur le même principe que la réflexion géométrique. En effet, ce traitement prend en paramètre deux images, l'une a le label (DE) et l'autre sur fond blanc, le label (W) (il est impératif de suivre ce procédé). L'algorithme va créer un masque noir selon les contours de l'image sur fond blanc, puis va récupérer des informations caractéristiques telles que la ligne et colonne de début du masque sur l'image et la ligne et colonne de fin. Puis, on applique un dégradé linéaire à l'endroit ou les pixels sont noirs entre le masque et l'image de remplissage. Ce remplissage prend en paramètre une direction, et la possibilité de mettre l'image sur fond blanc en noir et blanc pour accentuer l'effet. Le programme nous montre alors le remplissage pour toutes les directions possibles, et les deux premières images conservent leur couleur tandis que les deux dernières sont en noir et blanc.

Voici quelques exemples :