Université Clermont Auvergne

École Universitaire de Physique et d'Ingénierie

Diplôme : Master Imagerie et Technologie pour la Médecine (TechMed)



**Tableau de bord de rapport**

**Présenté par :** LEHARA Lyes

**Thème :** Bibliothèque de traitement d'images en C+

**Dirigé par :** Mme. PERY Emile

**Année universitaire :** 2024/2025.

|  |  |
| --- | --- |
| Date | 31 mars 2024 |
| Taches réaliser | 1. Rédaction de cahier de charge. 2. Comprendre le travail à réaliser 3. Planification des taches de la première semaine. 4. Suivre les étapes de gestion du projet  * **Conception** * Compréhension de cahier de charge * Compréhension de la problématique * Conception de projet * Entrainement sur les outils à utiliser (Programmation en C++). * **Développement** * Implémentation des bibliothèque namespace V1.1, V1.1 et V2.0 * **Test et validation** * Réalisation des tests **unitaires** * **Documentation et déploiement** * Rédaction de la documentation finale et le rapport |
| Difficulté rencontré |  |
| Solution apporté |  |

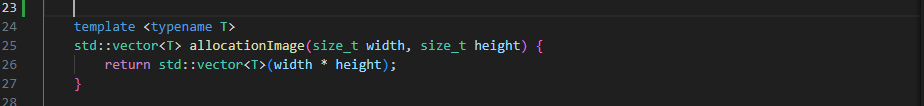
|  |  |
| --- | --- |
| Date de début - date de fin | 01 Avril – 06 Avril 2025 |
| Taches réaliser | 1. Lecture des cours de C++ 2. Faire de recherche sur visuel studio code 3. Configuration de l’éditeur pour exécuter le C++ 4. Réaliser les exercices demandés en C++ 5. Préparation du rapport 6. Planification des taches de la deuxième semaine |
| Difficulté rencontré | 1. Problème d’exécution de C++ sous Windows 2. Difficulté trouver sur les pointeurs en C++ |
| Solution apporté | * Installation d’un outil MinGW qui permet de compiler et d’exécuter des ficher C/C++. * Pratiquer les exercices sur les pointeurs. * Comprendre les tableaux * Une bonne pratique sur les fonctions Template et des types génériques |

|  |  |
| --- | --- |
| Date de début - date de fin | 07 Avril – 13 Avril 2025  13 Avril – 18 avril 2025 |
| Taches à réaliser | 1. Comprendre le codage des images 2. Comprendre l’allocation de mémoire pour une image avec C++ 3. Création des images en C++ (image banche, image damier et sinusoïdale) 4. Sauvegarder des fichiers. raw en .pgm ou .ppm pour pouvoir les visualiser de les manipuler 5. Application des filtre LUT pour les images |
| Difficulté rencontré | 1. Lecteur des fichiers. raw |
| Solution apporté | * Utilisation d’un logiciel Photoshop ou photopea pour le lecteur des fichiers .raw * Installation d’une extension **PPM/PBM/PGM** **Viewer** pour vs code |

* **Quelques captures pour le namespace1**

Voici quelques captures de réponse pour le namespace 1 et plus de details dans le rapport de TER

1. **Allocation d’images :**

****

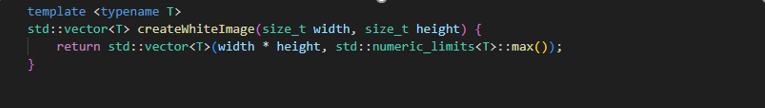
Cette fonction sert à allouer dynamiquement une image sous forme de vecteur 1 dimension de taille width × height (longeur et hauteur).

* **Template <typename T>**

C’est une fonction template de type T (un type générique), elle peut fonctionner avec n’importe quel type (int, float, unsigned char, etc.).

* **size\_t** est un entier non signé (typiquement utilisé pour les tailles).
* Même si une image est naturellement en 2D [i][j], on la stocke souvent en 1D pour simplifier la gestion de mémoire

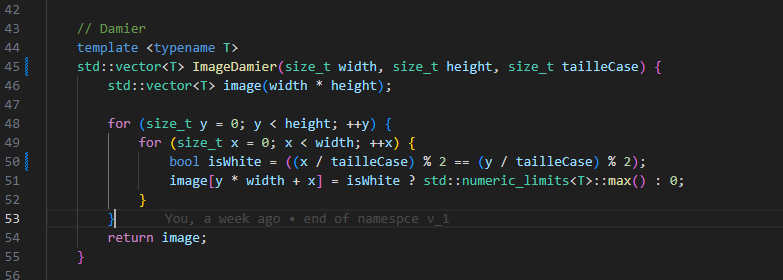
1. **Images Blanche, damier et sinusoadal :**
   * **Images Blanche**

****

* Cette fonction permet de créer une image blanche de taille hauteur (height) multiplié par la longueur (width) d’une dimension, est stockée dans un vecteur dynamique de type générique T grâce à la fonction template.
* ***std::vector<T>(width \* height, std::numeric\_limits<T>::max()) :*** cette ligne permet d’initialisé chaque pixels avec la valeur maximale possible pour le type T et cette valeur correspond à la couleur blanche en niveaux de gris.

**exemple :**

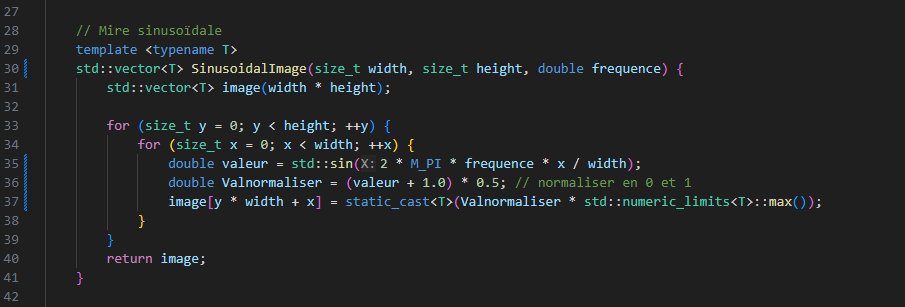
* + Si T = **unsigned char**, alors max = 255
  + Si T = **unsigned short**, alors max = 65535
  + **Damier**

****

* **ImageDamier** est une fonction générique (template de type T) permet de créer une image de taille hauteur multipliée par langueur de l’image. Elle prend en paramètre la hauteur et la langueur de l’image ainsi que **tailleCase** la taille d’un carré de damier (en pixel).
* Deux boucles pour parcourir tous les pixels de l’image ; la variable y pour parcourir les pixels de la hauteur et la variable x pour parcourir les pixels de la langueur.
* **bool isWhite = ((x / tailleCase) % 2 == (y / tailleCase) % 2)**

**image [y \* width + x] = isWhite ? std::numeric\_limits<T>::max() : 0**

À chaque itération de la boucle, on devise les indice x et y par la taille d’un carrées pour obtenir le numéro de la case dans laquelle on se trouve. Ensuite on applique le modulo 2 pour savoir si la case est paire ou impaire. Si les deux résultats sont identiques, cela veut dire qu’ont est dans la case blanche, sinon c’est une case noire. On utilise cette condition pour remplir l’image en mettant la valeur maximale possible si la case est blanche, ou 0 si elle est noire et on retourne l’image généré.

* + **Sinusoïdal**
* La fonction générique **SinusoïdaleImage** prend en paramètre la **langueur** et la **hauteur** d’une image ainsi que la **frequence** qui est le nombre de période de la fonction sinusoïdale. D’abord on crée une image vide de taille hauteur multiple par sa langueur et sera stocké dans un vecteur de type T.
* Deux boucles imbriquées pour parcourir chaque pixel de l’image.
* On calcule la valeur de période sinusoïdale en fonction de la position x et on obtient un résultat entre -1 et 1 puis on normalise la valeur obtenue en 0 et 1 car les images utilisent des valeur positive (0 à 255) :

**double valeur = std::sin(2 \* M\_PI \* frequency \* x / width);**

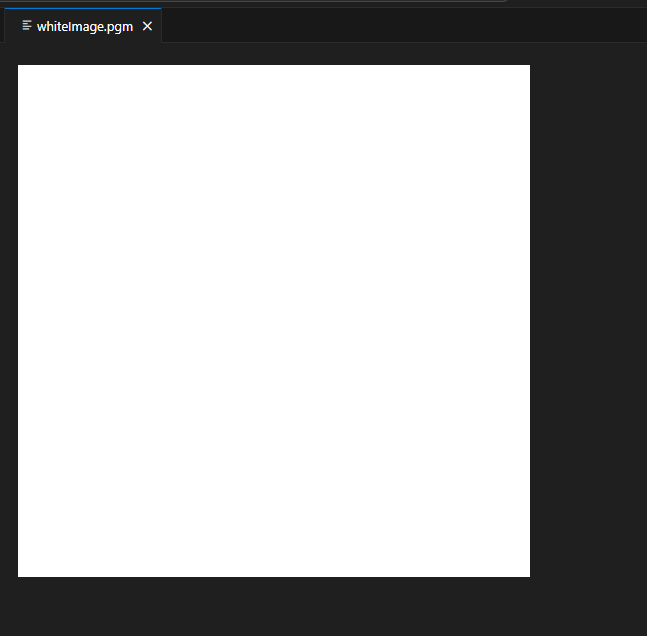
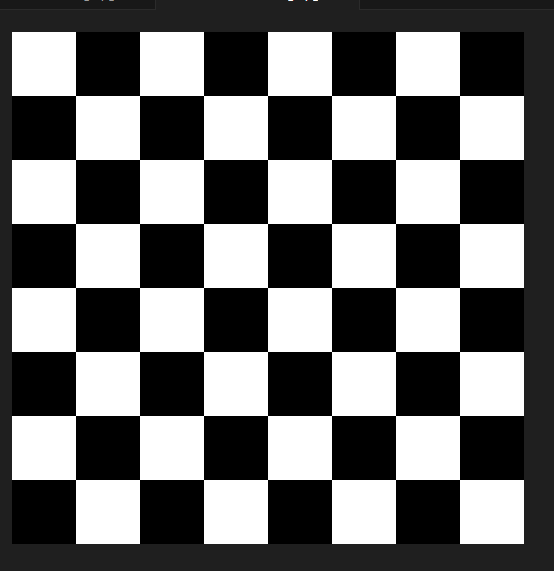
**double valeur = (val + 1.0) \* 0.5;**

* On convertit la valeur normalisée [0, 1] dans l’intervalle du type T :

Par exemple, si T = unsigned char, son max = 255, donc on obtient une intensité entre 0 et 255.

* Enfin, on retourne l’image obtenue.
* **Resultat obtenu**

Les figures montre le resultat obtenue apres l’appel aux fonctions



1. **Lecteur et écriture de fichiers images au format brut (.raw) :**
   * Lecteur de fichier image brut :