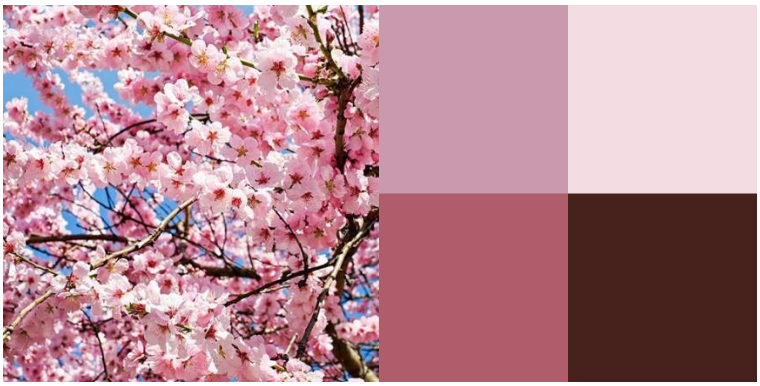
MAURIN Christina
ODOR Thibault

Harmonie des Couleurs

Compte Rendu 4

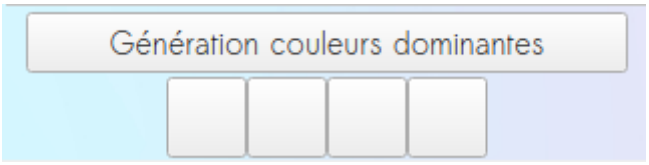
Développement de l'application avec Qt Creator (suite) :

Cette semaine nous continué le développement de l'application avec Qt Creator. La semaine précédente, nous avions une fonction *test_kmean.cpp* que nous avons renommée plus convenablement en *couleursDominantes.cpp* qui génère une palette de 4 couleurs. Ce sont les 4 couleurs dominantes de notre image. Exemple :



Nous voulons donc récupérer ces 4 valeurs pour notre application afin de pouvoir avoir en option le choix d'utiliser l'une d'entre elles. Il faut donc lier notre programme avec le code de notre application.

Le but est de cliquer sur un bouton "Afficher les couleurs dominantes" qui appelle donc notre fonction et qui met à jour 4 boutons dont chacun des fonds est l'une des couleurs.



Dans un premier temps, bien que l'on récupère avec QString le chemin du fichier, la première difficulté a été de récupérer le chemin pour notre fonction *lire_image_ppm()* car celle-ci utilise la fonction *fopen*() qui lit des char[]. Après plusieurs tests infructueux (ex: utilisation de .toStdString().c_str()), nous avons réussi à faire la transition avec *.toLocal8Bit().constData().*

```
char nomImage[cheminFichier.length()+1];
strcpy(nomImage, cheminFichier.toLocal8Bit().constData());
couleursDominantes(nomImage, couleur1, couleur2, couleur3, couleur4);
```

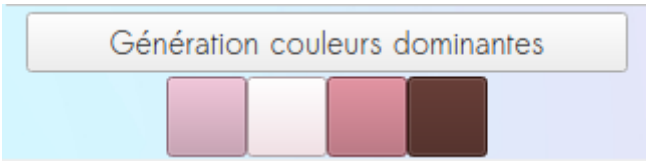
MAURIN Christina
ODOR Thibault

Nous avons ensuite modifié notre fonction couleursDominantes() afin de pouvoir récupérer facilement les nouvelles valeurs de nos quatre couleurs dans notre application en les mettant chacune en arguments.

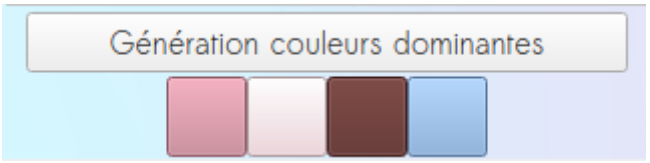
Couleur1, couleur2, couleur3 et couleur4 sont définis par une struct OCTET. La fonction *setStyleSheet(QString("background-color: rgb(%1, %2, %3)")* requiert que lorsque l'on ajoute des "*.args()*", ces derniers contiennent des entiers, nous devons donc faire un cast en insérant un *(int)*.

```
ui->d_color_1->setStyleSheet(QString("background-color: rgb(%1, %2, %3)")
                                .arg((int)couleur1.r)
                                    .arg((int)couleur1.g)
                                   .arg((int)couleur1.b));
```

Le code ci-dessus est le code permettant d'actualiser la couleur de notre bouton1 "d_color_1", on lui applique une feuille de style dont le background sera en rgb avec les 3 arguments qu'on lui ajoute. On fait de même pour nos autres boutons. Ainsi, on obtient le résultat suivant :



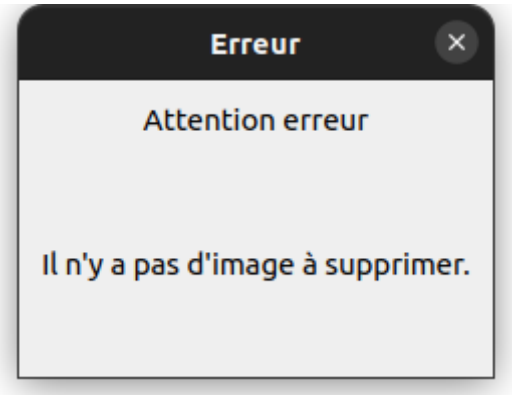
A noter que l'on peut cliquer plusieurs fois dessus et la génération modifie les couleurs.



Nous avons également implémenté une condition qui affiche une boîte de dialogue qui indique une erreur si l'on n'a pas sélectionné d'image auparavant.

MAURIN Christina
ODOR Thibault

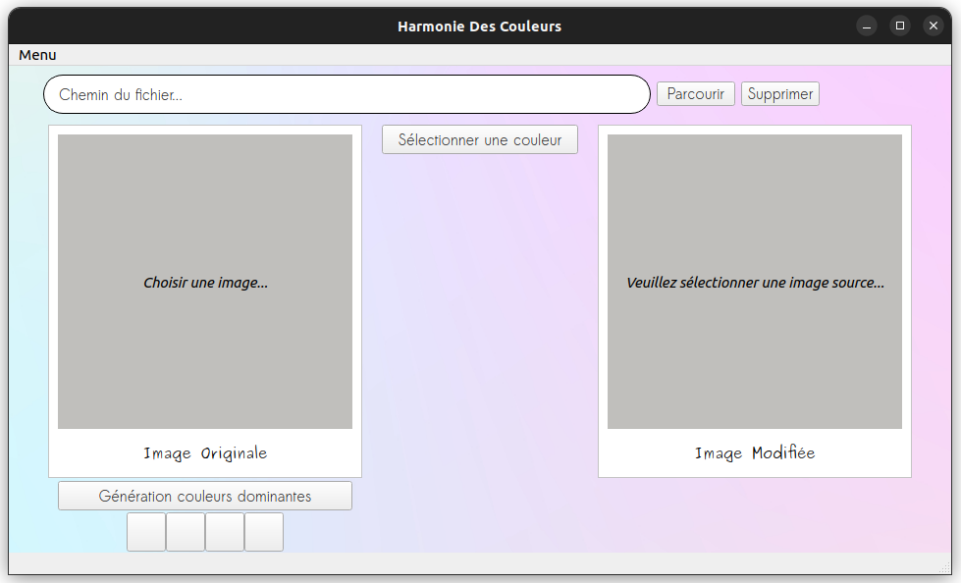
Parmi les fonctionnalités implémentées, nous avons également créé un bouton supprimer qui permet de supprimer l'image ainsi que son chemin.



Nous avons aussi ajouté sur notre interface graphique, les éléments qui permettront par la suite de générer notre image modifiée. Pour le moment il n'y a pas d'interaction mais cela nous permet visuellement de réfléchir à comment nous allons structurer le reste de notre application.

MAURIN Christina
ODOR Thibault

Voici le rendu actuel :



MAURIN Christina
ODOR Thibault

Algorithme :

La semaine dernière nous avions commencé à mettre en place deux types d'harmonisation : l'harmonisation Analogue et Complémentaire. Les résultats que l'on avait obtenus étaient plutôt décevants. Après de nombreuses expertises et débogages, nous avons fini par trouver que l'erreur venait de notre fonction de conversion HSL à RGB qui est appelée après avoir harmonisé notre image afin de pouvoir écrire l'image.

Ainsi nous avons perdu un peu de temps à résoudre ce problème mais nous proposons pour cette semaine trois types d'harmonisation différentes qui sont cette fois-ci beaucoup plus convaincantes.

Ci-dessous, voici nos résultats. Tout d'abord, nous avons : l'image de base et sa palette de couleur. Puis ligne par ligne : l'harmonisation de manière Analogue (utilisant donc les quatre couleurs de la palette), ensuite nous avons l'image harmonisée de manière Complémentaire puis l'image harmonisée de manière Triadique. Pour chacune des harmonisations, nous avons limité les couleurs voisines à un angle de 10°.

Nous avons également appliqué l'harmonisation sur des photographies prises par un APN et avons cette fois-ci rencontré des problèmes d'harmonisation spatiale.

Harmonisation triadique

Nous pouvons clairement voir un souci au niveau de la fenêtre située à droite sur la photo de droite, et, au niveau du mur sur la photo de gauche. Cela est dû à l'absence de la prise en compte de l'harmonisation spatiale de la composition de notre photo.

Pour la semaine prochaine :

Pour la semaine prochaine nous prévoyons de continuer à mettre en place les différentes harmonisations demandées et commencer à s'intéresser aux algorithmes permettant de réduire les artefacts présents sur les photos.

Également nous prévoyons d'avoir une application fonctionnelle nous permettant au minimum d'afficher une image harmonisée de manière Analogue et si le temps nous le permet Complémentaire et Triadique.

Sources :

https://www.techniques-de-peintre.fr/theorie-des-couleurs/harmonie-coloree/

https://en.wikipedia.org/wiki/HSL_and_HSV

https://doc.qt.io/