ROUMEGOUS Adrien
SI Frédéric

« Quadtrees : Compression d’image»

Rapport Projet Programmation C Avancée.

Sommaire

Introduction.

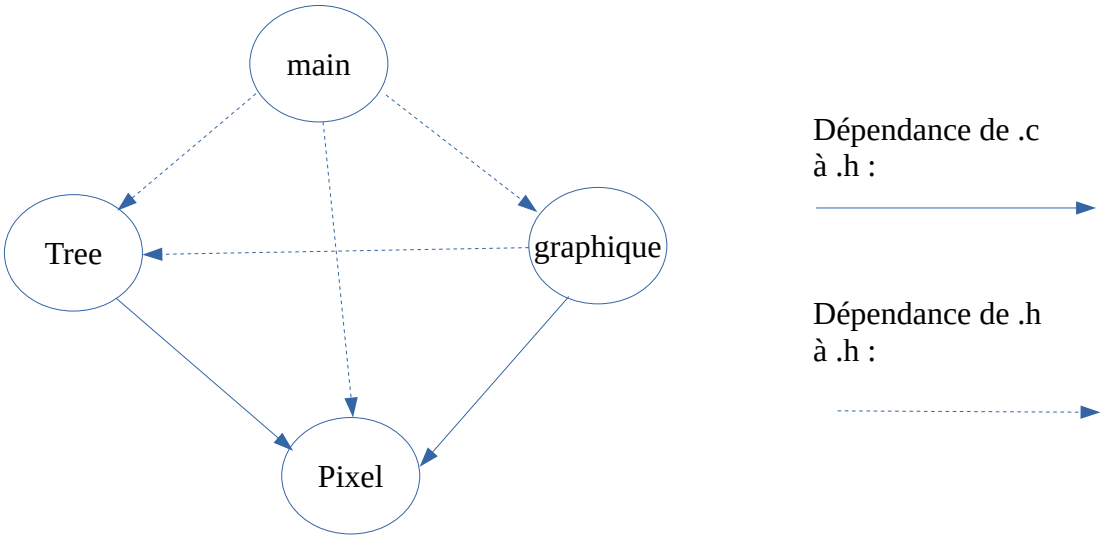
Pour ce projet, nous avons développé une application graphique pouvant encoder , décoder et compresser avec ou sans perte des images en utilisant des **Quadtrees**. Dans ce rapport, vous pourrez voir le fonctionnement général du projet ainsi qu’un guide dédié aux utilisateurs. Nous expliquons également comment le projet a été découpé en modules et expliquer l’implémentation des fonctions.

Architecture.

Voici l'ensemble des modules que nous avons cherché à implémenter :

- **Graphique.c** : Ce module va s'occuper de l'affichage de l'arbre et gérer les boutons du menu. Il va également s'occuper de la partie chargement d'image.
- **Pixel.c** : On peut considérer ce module comme une subdivision du module graphique : en effet, il s'occupe des calculs de moyenne d'image et des modifications de pixels dans l'image.
- **Tree.c** : Ce module se charge de la création du **quadtree** et de la gestion des nœuds. Il va aussi permettre de lire l'arbre pour pouvoir faire la sauvegarde binaire.
- **Main.c** : Pilote le programme.

Voici donc le découpage de notre projet :



Choix d'implémentation.

- Choix des structures :

Data : Le type **data** va nous permettre de connaître la taille de la partie de l'image dont on a besoin et sa position Nord Ouest.

Color : Le type **Color** va nous permettre de connaître les 4 couleurs *rgba.*

Node : Nous avons décidé qu'un nœud devait connaître la position de l'image qui lui est attribuée, donc le data(voir **struct data**) , la couleur moyenne, son parent pour la partie minimisation etses 4 fils et le **level** où se trouve le nœud.

-Couleur moyenne des pixels de l'image :

Le premier objectif qu'on s'est fixé fut d'afficher l'image avec la moyenne de tous les pixels de l'image, c'est à dire avoir une image monochrome. On utilise donc les fonctions de MLV pour récupérer la couleur de chaque pixel de l'image, puis on a fait un calcul pour avoir la moyenne de la zone (ici toute l'image) puis il ne restait plus qu'à appliquer la couleur à chaque pixels.

-Subdivision en quadtree :

Chaque nœud correspond à une zone d'image. On divise récursivement en 4 autres nœuds, puis on recalcule la moyenne des couleurs des quatre nœuds suivants, ainsi que les nouvelles coordonnées et la taille de l'image.

-Minimisation :
Pour la minimisation, on parcourt jusqu'à la dernière feuille, puis on remonte vers le nœud parent pour comparer la couleurs des 4 feuilles. On ne garde qu'une seule feuille par couleur.
Pendant la phase de subdivision , si la zone d'image a une seule couleur, on arrête la subdivision pour cette zone d'image.

Il reste une deuxième phase de minimisation, qui est de fusionner 2 sous-arbres s'ils sont identiques.

-Sauvegarde :

On convertit les 4 couleurs en binaire, puis on écrit en sortie dans un fichier.

Difficultés rencontrées :

-Fuite mémoire et plantage de machine :

Au début dans la structure du nœud , nous n'avons pas mit l'attribut permettant de gérer la couleur. Par conséquent, on a du calculer la couleur moyenne et la modification des pixels lors de la phase d'affichage de l'image. Pour éviter de modifier l'image d'origine, on a dupliqué l'image pour la modifier, tout en gardant l'image d'origine pour le calcul de la moyenne des couleurs.

Solution trouvée:
Nous avons décidé d'ajouter un attribut **color** de type **MLV_Color** dans la structure du nœud.

-Utilisation de la fonction **MLV_convert_color_to_rgba** :

Pour écrire les fichiers d'encodage en sortie, on voulait convertir la couleur de type **MLV_Color** codée sur 32 bits en 4 entiers de type Uint8. Mais à chaque appel de la fonction **MLV_convert_color_to_rgba** , on a eu une erreur de segmentation .

Solution trouvée :
On a créé une structure **Color** qui regroupe 4 attributs **int** pour chaque couleur **rgba** pour remplacer le type **MLV_Color** dans la structure du nœud. Il ne reste plus qu'à convertir les 4 couleurs en binaires dans un tableau de **int**.

La minimisation en fusionnant 2 arbres est ce qu'il nous reste à résoudre.