Alexis Theas-Laban & Nader Khedr

L3 - IGM | 2020-2021 | Projet de programmation C

Compression d'image avec des quadtrees

Rapport

Table of contents

Introduction

Une application graphique pour compresser et visualiser des images en utilisant des quadtrees.

Installation

Le makefile fourni permet de générer l'exécutable avec la commande **make**.

Pour revenir à l'état d'origine utilisez :

```
make clean
```

Lancement

Le programme se lance avec la commande suivante :

```
./yaic [OPTION...]
```

Options

-o FILE

Ouvre une image quadtree nommé FILE. FILE doit avoir l'une des extensions suivantes : qtc, qtn, gmc, gmn.

--time-min FILE

Mesure et affiche le temps nécessaire pour minimiser FILE.

--test-load

Charge et affiche une image qtc depuis le dossier img.

--test-save

Prend une image et la sauvegarde dans le dossier img au format qtc.

Documentation

Pour générer la documentation technique utilisez :

```
doxygen doc/doxyfile
```

Architecture

Modules

area : représente une zone de l'image.

draw : permet d'afficher un quadtree.

minimize : permet la minimisation de quadtree.

quadtree : contient la structure d'un quadtree et toutes les fonctions de manipulation d'arbre associées (allocation, libération, hauteur, etc).

gui : affiche l'interface graphique.

tree_linked_list : une structure de données pour stocker des quadtrees. Ce module est exclusivement utilisé pour la minimisation de quadtree et la libération de graph.

encode : fonctions relatives au sauvegarde et chargement de quadtree/graph.

bit_buffer : une structure de données pour stocker des bit un à un. Utilisé par le module encode dans la sauvegarde de quadtree qtc, qtn.

bitmap : permet d'obtenir la couleur moyenne d'une image et son taux d'erreur.

color : représentation d'une couleur par un entier 32 bits.

main : point d'entrée du programme.

Implémentation

Minimisation

Une table de hachage permet de catégoriser les quadtrees en fonction de leurs couleurs moyennes. Chaque entrée de cette table contient une liste chaînée de quadtree dont la couleur moyenne est similaire.

Lorsqu'un sous-arbre est inspecté, on recherche un arbre avec une distance inférieure à une certaine valeur dans la liste chaînée correspondante. Si cet arbre existe, les deux sous-arbres sont rattachés.

Un facteur de regroupement (REGROUP_FACTOR dans minimize.c) permet de définir le degré d'inclusion d'un nœud dans une entrée de la table. Plus cette valeur est haute et plus les quadtrees comparé sont nombreux.

Il est donc possible de faire varier cette valeur afin d'obtenir de meilleurs taux de compression.

Encodage

Quadtree

L'encodage d'arbre au format qtn et qtc nécessite l'écriture de bit un à un dans un fichier texte. La structure **BitBuffer** permet de répondre à cette problématique en stockant les bits dans un tableau de caractère bit par bit. Le bit actuel est indiqué par un indice **bit_pos**.

DAG

L'encodage de graph dirigé et l'itération de cette structure en générale nécessite de ne pas visiter deux fois le même nœud. Pour cela on définit un champ d'un seul bit dans la structure de notre quadtree pour indiquer si un nœud a déjà été visité.

Par ailleurs l'encodage nécessite de numéroter les nœuds de notre graph/quadtree. Nous avons donc déclaré un second champ pour contenir l'identifiant du nœud courant.

Interface

Nous réutilisons la structure area définit précédemment afin de représenter une zone de collision. Une structure **Button** est composée d'une boîte de collision, d'une image et d'un message affiché lors du survol avec le curseur.

Conclusion

Difficultés rencontrées

La minimisation des quadtrees n'est pas assez efficace. Certaines branches avec une même couleur moyenne ne sont pas comparées. Par ailleurs, le parcours dans une entrée de la table de hachage a une complexité linéaire. Un arbre binaire de recherche aurait pu être utilisé pour répondre à ce problème. Mais nous ne l'avons pas implémenté par manque de temps.

Le test des fonctions de minimisation en générale a posé quelques problèmes. En effet, il fallait pouvoir inspecter la structure des nœuds dans notre quadtree. Nous avons dû créer un module spécifique pour la visualisation des nœuds dans notre arbre.