BUT-Info.3 2023-2024
Module "3 en 1"
Projet n°2

Format de compression d'image numérique
Codage hiérarchique par *QuadTree*
— Objectif visés —

Développement "Full Stack" d'un `CoDec` *de compression/décompression d'image*
Création d'une bibliothèque partagée `shared object`.

---

l'objectif est de créer un format d'échange (`Codeur` ⇌ `Fichier` ⇌ `Décodeur`) d'image compressée basé sur une décomposition hierarchique (`QuadTree`). Ce `CoDec` prendra la forme d'une bibliothèque `libqtc.so`.

Les *fichiers d'échange* devant être compatibles entre `CoDec`, la structure du format sera strictement imposée pour chaque étape. Ces structures seront précisées dans les documents annexes afférents à chaque étape.

Fonctionalités

Schéma de codage/décodage

Ce format (nommé `QTC`) devra prévoir :

ⓐ un schéma avec ou sans perte (*lossy/lossless*) - format `'Q1'`

ⓑ un schéma *lossy/lossless* en codage différentiel & code à longueur variable (*VLC*[(1)]) - format `'Q2'`

mode graphique

Une interface graphique dédiée devra permettre de gérer l'encodage et le décodage de ce format :

- Encodeur
  - ✓ charger / afficher une image (format d'entrée quelconque)
  - ✓ afficher l'image restructurée en mode hiérachique (grille `QuadTree`)
  - ✓ ajuster les paramètres de "perte"
  - ✓ prévisualiser et évaluer la qualité et le taux de compression
  - ✓ activer les mode différentiel/*VLC*
  - ✓ activer les options de sauvegarde au format `QTC`.
- Décodeur
  - ✓ charger / décompresser / afficher une image au format `QTC`.
  - ✓ activer les options de sauvegarde dans un autre format standard

mode 'terminal'

Le `CoDec` devra également pouvoir être utilisé sans interface graphique, directement en ligne de commandes, avec une gestion des options "façon `Unix`" (`$> codec -i input -l 0.9 -d -o output`).

Une liste d'options vous sera précisée en document annexe.

[(1)] Variable Length Coding

Développement "agile" et "modulaire"

Livrables intermédiaires

Il y a plusieurs étapes successives, chacune dépendant des précédentes.

☞ les chiffres `[x|y|z]` donnent une indication de l'importance de chaque phase pour chacun des 3 "Modules" en jeux : (`Prog.|Math.|Multimédia`) et le nombre de piments (🌶) correspond au niveau de difficulté et/ou au temps nécessaire.

ⓐ `CoDec` sans perte (*lossless*) :
- restructuration de l'image en `QuadTree` `back-[5|3|2]` (🌶🌶🌶)
- outils de visualisation / dialogue `front-[4|2|4]` (🌶🌶)
- création du fichier au format d'échange (mode `'Q1'`) `full-[5|2|3]` (🌶🌶🌶)

ⓑ `CoDec` avec perte (*lossy*) :
- outils de filtrage du `QuadTree` `back-[5|3|2]` (🌶🌶🌶)
- outils de contrôle / visualisation / dialogue `front-[3|3|4]` (🌶🌶🌶)
- adaptation des fonctionalité de lecture/écriture (mode `'Q1'`) `full-[5|2|3]` (🌶)

ⓒ `CoDec` *lossy/lossless* différentiel avec `VLC` :
- adaptation du `QuadTree` `back-[4|4|2]` (🌶🌶🌶)
- adaptation des outils de visualisation / dialogue `front-[2|4|4]` (🌶)
- adaptation du format d'échange (modes `'Q2'`) `full-[4|3|3]` (🌶🌶🌶)

Chaque étape sera développée sur 2 séances de `TP (4h)` et donnera lieu à un livrable sous la forme d'une archive complète (sources, `Makefile`).

Compatibilités ascendante(↑)|descendante(↓)|croisée(⇌)

le livrable à une étape devra inclure les fonctionalités de toutes les étapes précédentes

↑ un fichier `'Q1'` créé à l'étape ⓐ devra pouvoir être lu par le décodeur des étapes ⓑ et ⓒ.

↓ un fichier `'Q1'` créé à l'étape ⓒ devra pouvoir être lu par le décodeur des étapes ⓐ et ⓑ

⇌ un fichier `'Q*'` créé à n'importe quelle étape devra pouvoir être lu par le décodeur de quelqu'un d'autre.

Chaque étape devra donc être structurée sous forme modulaire, chaque module gérant une phase du `CoDec` :

① Conversions `Image` ⇌ `QuadTree`.
② Ecriture/Lecture `QuadTree` ⇌ `fichier.qtc`
③ `CoDec` avec pertes (étape ⓑ)
④ `CoDec` Diff+*VLC*(étape ⓒ)
⑤ mode graphique / mode 'terminal'
⑥ module principal

Bibliothèque partagée

Chaque livrable devra se compiler sous la forme d'une blibliothèque partagée `libqtc.so` regroupant les modules fonctionnels ① à ④, de deux modules d'interfaçage (graphique et 'terminal').

Les fichiers partagés `libqtc.so` devront être interchangeables d'un groupe à l'autre. De ce fait les *interfaces* (types et fonctions publiques) devront respecter des noms et formats imposés.

Vos architecture et fichiers de configuration `Makefile` devront permettre de créer et changer facilement la version de `libqtc.so` (code de version, identifiant de l'auteur...).

Ces éléments du cahier des charges seront à affiner en cours de développement mais doivent rester, à tout instant, bien présents à l'esprit des développeurs.

Travail en binôme

pour chaque étape il y a beaucoup de code à écrire et à assembler, mais les fonctionalités de codage et décodage sont en grande partie symétriques. Vous travaillerez donc en binôme : un personne s'occupant en priorité de la partie `Codeur`, l'autre en priorité de la partie `Décodeur`.

Les deux personnes contribuent à part égale à la mise en place des types, fonctions et modules, à l'ensemble des interfaces utilisateur et à la mise en place de l'architecture globale (`Makefile, libqtc.so`)