Université Paris Diderot
L3 Informatique

Conduite de Projet
Année 2018-2019

Cimp [1] — Un éditeur d'images

version 1.0

> *Raster images are stored in a computer in the form of a grid of picture elements, or pixels. These pixels contain the image's color and brightness information. Image editors can change the pixels to enhance the image in many ways. The pixels can be changed as a group, or individually, by the sophisticated algorithms within the image editors.*
>
> — *https://en.wikipedia.org/wiki/Image_editing*

1 Introduction

Le but de ce projet est d'implémenter en langage C un logiciel de manipulation d'images matricielles ("raster graphics editor"). Les éléments de spécification qui suivent se divisent en un *projet minimal* (dont on s'attend à ce qu'il soit implémenté en intégralité) et des suggestions d'extensions, dont l'implémentation pourra consolider votre note.

Le projet est à faire en trinômes. La progression régulière de votre travail au cours du semestre, de même que le bon usage des outils et des pratiques de conduite de projet présentés dans ce cours, seront au moins aussi importantes pour votre évaluation que le résultat final. Les informations pratiques utiles vous seront fournies sur la page Moodle du cours. [2]

2 Projet minimal

Visualisation d'images. La visualisation des images en cours de traitement se fera à l'aide de la bibliothèque libre SDL 2.0. [3] Cette librairie, minimaliste mais suffisante pour les besoins de ce projet, permet notamment l'ouverture à l'écran d'une ou plusieurs fenêtres, le transfert d'images stockées en mémoire vers ces fenêtres, la capture d'événements souris et clavier se produisant dans celles-ci, ainsi que la capture des événements sur celles-ci. [4] La documentation détaillée de SDL se trouve sur son wiki officiel [2]. Vous pourriez également consulter un tutoriel qui introduit quelques bonnes pratiques par rapport à l'utilisation de la bibliothèque [3].

Interface utilisateur. L'accès aux fonctionnalités du logiciel se fera essentiellement à l'aide d'une *interface en ligne de commandes* (CLI) – autrement dit en mode texte depuis un terminal. La capture d'événements dans les fenêtres ouvertes par SDL pourra permettre de compléter à la souris certaines demandes d'opérations sur les images affichées (*e.g.* demande de sélection d'une région par commande, suivi d'une sélection effective à la souris). Votre interface doit être suffisamment bien conçue pour autoriser ce type d'entrées de façon fluide et non ambiguë

1. Cimp – Console Image Manipulation Program
2. http://moodlesupd.script.univ-paris-diderot.fr/course/view.php?id=6660
3. https://www.libsdl.org
4. Entrée, sortie, perte ou gain de focus, redimensionnement, fermeture...

(opération en cours clairement explicitée, possibilité d'annuler celle-ci lorsqu'elle n'est pas encore complétée, [5] etc).

Le choix du langage de commandes est libre, mais il devra être suffisamment riche pour permettre l'invocation des fonctionnalités décrites ci-dessous.

Chargement et sauvegarde d'images. On doit pouvoir à tout moment charger en mémoire une ou plusieurs images, transférer une image vers une fenêtre déjà ouverte ou vers une nouvelle fenêtre. Les formats d'images supportés doivent être raisonnablement variés : `jpg`, `bmp`, `png`, etc. [6] Inversement, une image transformée doit pouvoir à tout moment être sauvegardée, sous son format d'origine ou sous un autre format, sous son nom d'origine ou un autre.

Sélection de régions. Toute modification uniforme des pixels d'une image (*e.g.* remplissage) doit pouvoir s'appliquer soit à l'image toute entière, soit à une région de l'image (rectangulaire, délimitée par tracé libre à la souris, regroupant tous les pixels d'une couleur donnée avec ou sans marge de tolérance, etc.)

Le programme doit permettre la sélection de telles régions – et mieux encore, l'ajout ou la suppression d'une région à une sélection déjà existante. La région en cours de sélection doit être clairement visible à l'écran, et désélectionnable à tout moment.

Copier, couper, coller. Le programme devra permettre de copier ou de couper une région d'image – en remplaçant dans ce dernier cas la couleur des pixels coupés par une couleur de fond par défaut ou choisie par l'utilisateur [7]. Les régions copiées ou coupées doivent pouvoir être collées à des coordonnées quelconques, dans l'image elle-même ou dans une autre image.

Transformations. Le programme doit permettre d'effectuer sur une image les transformations élémentaires suivantes :

- symétries verticale et horizontale
- rotation de l'image par un multiple de 90°
- recadrage de l'image :
  - par découpage rectangulaire (perte des pixels en dehors du rectangle de recadrage)
  - par agrandissement de la zone de travail (ajout d'une marge autour des pixels d'origine, colorée en une couleur de fond)
- modification de la taille de l'image [8]

Les transformations suivantes s'appliqueront à une image ou à la sélection courante :

- remplissage par une couleur donnée
- remplacement de couleur, avec ou sans marge de tolérance
- mise en négatif
- mise en niveaux de gris
- mise en noir et blanc

5. L'entrée des commandes peut se faire dans le shell via `getchar`/`scanf`, mais il est aussi envisageable d'entrer et de mémoriser la suite des caractères d'une commande par capture d'événements via SDL, la console affichant ces caractères un à un *comme si* elle avait repris la main. Le procédé complique légèrement l'ecriture de la boucle principale d'interaction, et ne fonctionne que si une fenêtre au moins ouverte par SDL a le focus.

6. Il faudra donc vraisemblablement intégrer à votre projet une ou plusieurs librairies externes supplémentaires – libres de droits

7. Cette couleur de fond pourra aussi être transparente si l'on choisit d'implémenter la gestion du canal alpha (*cf.* extensions possibles).

8. Cette transformation est non triviale par calcul direct, mais SDL vous donne la possibilité de l'implémenter à l'aide de fonctions prédéfinies.

— ajustement de la luminosité et du contraste

3 Extensions possibles

Afin de consolider votre note, voici une liste d'extensions possibles pour votre éditeur. Cette liste n'est pas exhaustive, et vous pouvez nous soumettre vos propres idées d'améliorations ou de nouvelles fonctionnalités. Les éléments que vous aurez traités devront être mentionnés dans votre rapport.

Traitements par lots. Les traitements précédents pourront être applicables automatiquement (en *batch*) à plusieurs fichiers d'images externes dont les noms seront soit explicités, soit spécifiés à l'aide des motifs suffisamment expressifs (*e.g.* `*.jpg`, `vacances*.png`, etc.)

Scripts de transformations. Des suites de commandes, éventuellement à paramètres, pourront être écrites dans des fichiers externes. Ces commandes pourront être exécutées séquentiellement par le programme en affectant une image chargée en mémoire, ou par traitement par lots.

Édition de scripts. Le programme pourra aussi permettre de nommer et rédiger les scripts précédents, de les nommer, de les exécuter en leur fournissant un ou plusieurs paramètres, de les modifier, de les sauvegarder et les recharger.

Undo/redo. Le programme doit permettre de revenir à un état antérieur de l'image, ou de revenir à l'un des états postérieurs à cet état antérieur encore présents en mémoire – le tout de manière suffisamment optimisée pour ne pas saturer la mémoire (*e.g.* en mémorisant le script de commandes induit par la suite de commandes effectuées depuis le chargement de l'image).

Gestion de la transparence (canal alpha). Certains formats d'image (comme PNG) permettent de gérer la transparence : en plus des valeurs de rouge, vert et bleu, on associe alors à chaque pixel une valeur d'opacité (c'est le *canal alpha*). Cette transparence peut être nulle (la valeur alpha est alors maximale), totale ou partielle. La transparence entre par exemple en jeu quand on colle une image sur une autre (cas d'utilisation réelle : une icône sur une page web ou un fond de bureau) : les pixels totalement transparents de l'image collée sont alors remplacés par ceux de l'image de fond ; quand à ceux qui sont partiellement transparents, ils sont mélangés avec les pixels de fond par la technique de l'*alpha blending*[9].

Transformations avancées. Outre les transformations décrites à la section précédente, le programme pourra permettre d'effectuer des transformations plus élaborées, [10] par exemple :
— modification de la balance/de la palette des couleurs,
— rotation d'angle arbitraire,
— effets artistiques divers (flou, pixellisation, déformations,...).

9. https://en.wikipedia.org/wiki/Alpha_compositing#Alpha_blending
10. L'idée de cette extension est non pas de recourir une fois de plus à une librairie externe ou de recopier des informations trouvées sur le web, mais bien de relever le défi consistant à trouver le moyen d'implémenter le mieux possible une transformation non triviale et/ou originale avec vos propres idées.

4 Évaluation de votre projet

La notation prendra en compte tous les points suivants.

Utilisation des outils standard. Les cours magistraux présenteront tout au long du semestre des outils standard de développement (Makefile, git, etc.) qui devront être *effectivement* utilisés pendant le projet. Cette utilisation va de pair avec une séparation des différentes parties du projet en modules et librairies séparées.

Par exemple, une utilisation frauduleuse de *Makefile* pourrait être un appel unique à un script externe.

Modularité et tests. Les différents modules du programme devront être séparés en différents fichiers sources (`.c`) et en fichiers d'en-tête (`.h`) utilisés à bon escient. Vous préparerez aussi des tests unitaires de chaque fonction importante sur des exemples choisis, et des tests d'acceptation des fonctionnalités globales du projet. Ces tests seront inclus dans votre dépôt git.

Contrôle continu. Le travail doit être réalisé tout au long du semestre grâce à un répertoire *git* pour chaque trinôme.

1. Votre répertoire doit être *privé*, mais dès les premières semaines, vous donnerez accès à ce répertoire à tous les enseignants. Ceci permettra notamment de garder la trace et les dates de contributions de chaque étudiant.
2. Vous pouvez choisir entre le service d'hébergement GitLab offert par l'université [11] ou tout autre service tiers, toujours en respectant les consignes du point précédent.
3. Nous vous demandons de gérer en continu les modifications de votre répertoire et d'effectuer systématiquement des revues de code à l'intérieur de votre trinôme :
   — avec GitLab, à l'aide de *merge requests* [12]
   — avec GitHub, à l'aide de *pull requests* [13]

Journal. Chaque trinôme tiendra à jour un *journal de développement* inclus dans son répertoire git. Chaque semaine, vous indiquerez dans votre journal vos objectifs pour la semaine à venir, et vos résultats de la semaine passée. Dans la mesure du possible, faites en sorte que chaque commit corresponde à une tâche précise et soit effectué par la personne ayant effectué la tâche. Un dépôt git n'est pas limité à du code, n'hésitez pas à y inclure d'autres documents qui nous permettront de suivre votre travail, par exemple : des courts résumés de prise de décision, des notes sur le fonctionnement des bibliothèques que vous allez utiliser, la structure que vous projetez pour vos modules, etc.

Lisibilité et clarté. Il est fortement recommandé d'indenter systématiquement, d'éviter les lignes trop longues (80 caractères), de choisir des noms parlants pour les variables et fonctions, et de commenter le code souvent, mais en restant concis. Vous supprimerez également toute duplication de code, suivant le principe DRY [1] ("Don't Repeat yourself").

11. http://moule.informatique.univ-paris-diderot.fr:8080/
12. https://docs.gitlab.com/ee/gitlab-basics/add-merge-request.html
13. https://help.github.com/articles/about-pull-requests/

Références

[1] Don't Repeat Yourself. https://en.wikipedia.org/wiki/Don%27t_repeat_yourself.

[2] SDL-Wiki. http://wiki.libsdl.org/FrontPage.

[3] SDL2 tutorial. https://www.willusher.io/pages/sdl2/.