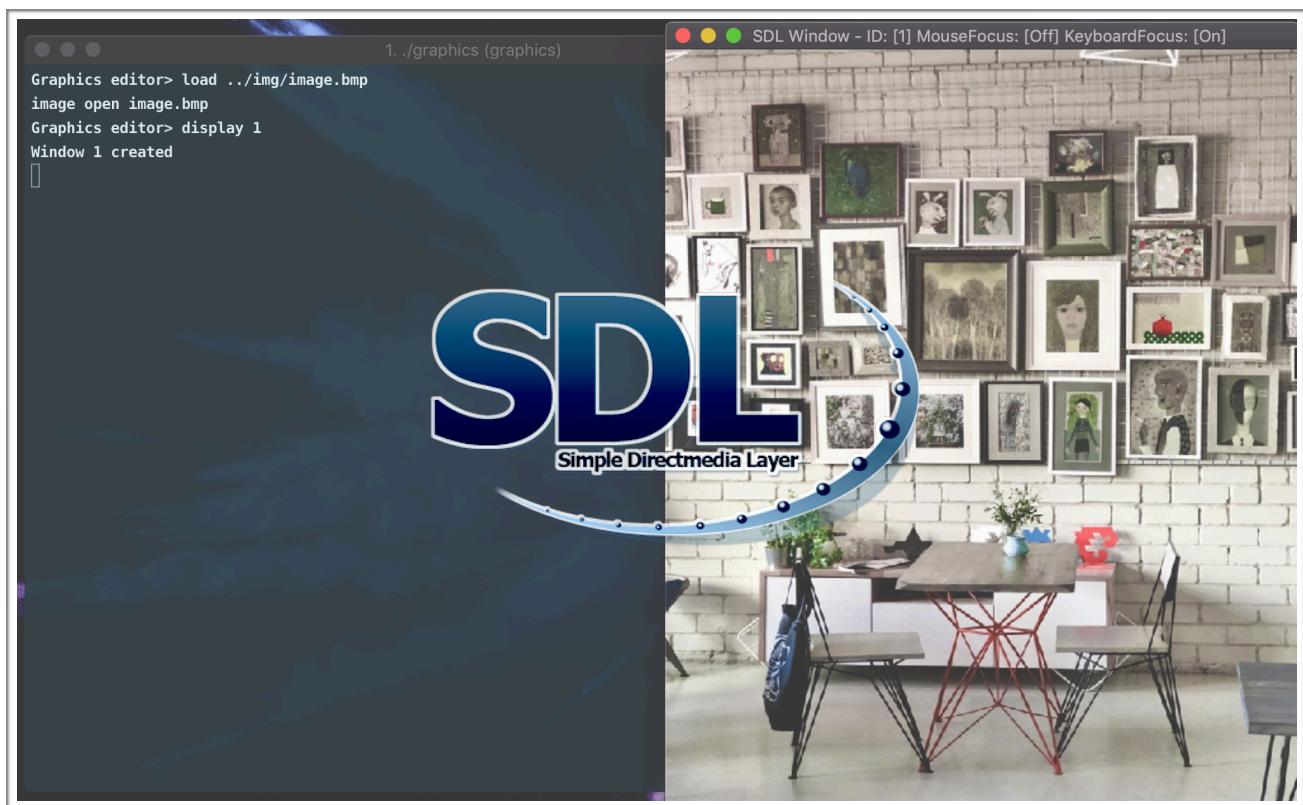


Projet de Conduite de Projet

Cimp 1 – Un éditeur d'images



Travail réalisé par :

- Chawki CHOUIB
- Antony FOURCHAULT
- Yaoyi LUO

Objectif

Le but de ce projet est d'implémenter en langage C un logiciel de manipulation d'images matricielles.

Les images matricielles sont stockées dans un ordinateur sous la forme d'une grille de pixels.

Ces pixels contiennent les informations de couleur et de luminosité de l'image. Les éditeurs d'images peuvent changer les pixels pour améliorer l'image de plusieurs façons.

Les pixels peuvent être modifiés en tant que groupe ou individuellement par les algorithmes sophistiqués des éditeurs d'images.

Compilation et utilisation

1. pour les tests

Aller dans le répertoire **test/** et exécuter la commande “**make**” puis la commande “**./graphics_test**”

2. pour une utilisation standard

Aller dans le répertoire **src/** et exécuter la commande “**make**” puis la commande “**./graphics**”, un fois le programme lancé taper la commande “**help**” qui vous affichera une aide

```
└ cd src
chawki@MacBook-Pro-de-chawki: ~/CLionProjects/raster_graphics_editor/src ] master • ④
make
make -C parsing
gcc -g -Wall -c `pkg-config sdl2 --cflags` `pkg-config SDL2_image --cflags` parsing.c
ar cru libparsing.a parsing.o
ranlib libparsing.a
make -C gestic
make -C my_struct_images
gcc -g -Wall -c `pkg-config sdl2 --cflags` `pkg-config SDL2_image --cflags` my_struct_images.c
ar cru libmy_struct_images.a my_struct_images.o
ranlib libmy_struct_images.a
make -C traitement_fenetre
gcc -g -Wall -c `pkg-config sdl2 --cflags` `pkg-config SDL2_image --cflags` traitement_fenetre.c
ar cru libtraitemen_fenetre.a traitement_fenetre.o
ranlib libtraitemen_fenetre.a
make -C traitement_image
gcc -g -Wall -c `pkg-config sdl2 --cflags` `pkg-config SDL2_image --cflags` traitement_image.c
ar cru libtraitemen_image.a traitement_image.o
ranlib libtraitemen_image.a
make graphics
gcc -c -g -Wall `pkg-config sdl2 --cflags` `pkg-config SDL2_image --cflags` main.c
gcc -g -Wall -o graphics main.o parsing/libparsing.a gestic/traitement_image/libtraitemen_image.o
g-config sdl2 --libs `pkg-config SDL2_image --libs` -lreadline
make clean
make -C parsing clean
rm -f libparsing.a parsing.o
make -C gestic clean
make -C my_struct_images clean
rm -f libmy_struct_images.a my_struct_images.o
make -C traitement_fenetre clean
rm -f libtraitemen_fenetre.a traitement_fenetre.o
make -C traitement_image clean
rm -f libtraitemen_image.a traitement_image.o
rm -f main.o
chawki@MacBook-Pro-de-chawki: ~/CLionProjects/raster_graphics_editor/src ] master • ④
[ ./graphics
Graphics editor> help

Liste des commandes disponibles :

- exit : quite l' application
- load pathfile : charge une image et lui donne un id
```

```
└ cd test
chawki@MacBook-Pro-de-chawki: ~/CLionProjects/raster_graphics_editor/test ] master • ④
make
make -C parsing
gcc -g -Wall -c `pkg-config sdl2 --cflags` `pkg-config SDL2_image --cflags` parsing.c
ar cru libparsing.a parsing.o
ranlib libparsing.a
make -C gestic
make -C my_struct_images
gcc -g -Wall -c `pkg-config sdl2 --cflags` `pkg-config SDL2_image --cflags` my_struct_images.c
ar cru libmy_struct_images.a my_struct_images.o
ranlib libmy_struct_images.a
make -C traitement_fenetre
gcc -g -Wall -c `pkg-config sdl2 --cflags` `pkg-config SDL2_image --cflags` traitement_fenetre.c
ar cru libtraitemen_fenetre.a traitement_fenetre.o
ranlib libtraitemen_fenetre.a
make -C traitement_image
gcc -g -Wall -c `pkg-config sdl2 --cflags` `pkg-config SDL2_image --cflags` traitement_image.c
ar cru libtraitemen_image.a traitement_image.o
ranlib libtraitemen_image.a
make graphics_test
gcc -c -g -Wall main.c `pkg-config sdl2 --cflags` `pkg-config SDL2_image --cflags`
gcc -c -g -Wall testes.c `pkg-config sdl2 --cflags` `pkg-config SDL2_image --cflags`
gcc -g -Wall -o graphics_test main.o testes.o ../src/parsing/libparsing.a ../src/gestic/traitement_
my_struct_images/libmy_struct_images.a `pkg-config sdl2 --libs` `pkg-config SDL2_image --libs` -lread
make clean
make -C .../src clean
make -C parsing clean
rm -f libparsing.a parsing.o
make -C gestic clean
make -C my_struct_images clean
rm -f libmy_struct_images.a my_struct_images.o
make -C traitement_fenetre clean
rm -f libtraitemen_fenetre.a traitement_fenetre.o
make -C traitement_image clean
rm -f libtraitemen_image.a traitement_image.o
rm -f main.o
rm -f main.o testes.o
chawki@MacBook-Pro-de-chawki: ~/CLionProjects/raster_graphics_editor/test ] master • ④
[ ./graphics_test

-----TEST-----
TEST Init sdl : OK
-----FIN TEST-----
```

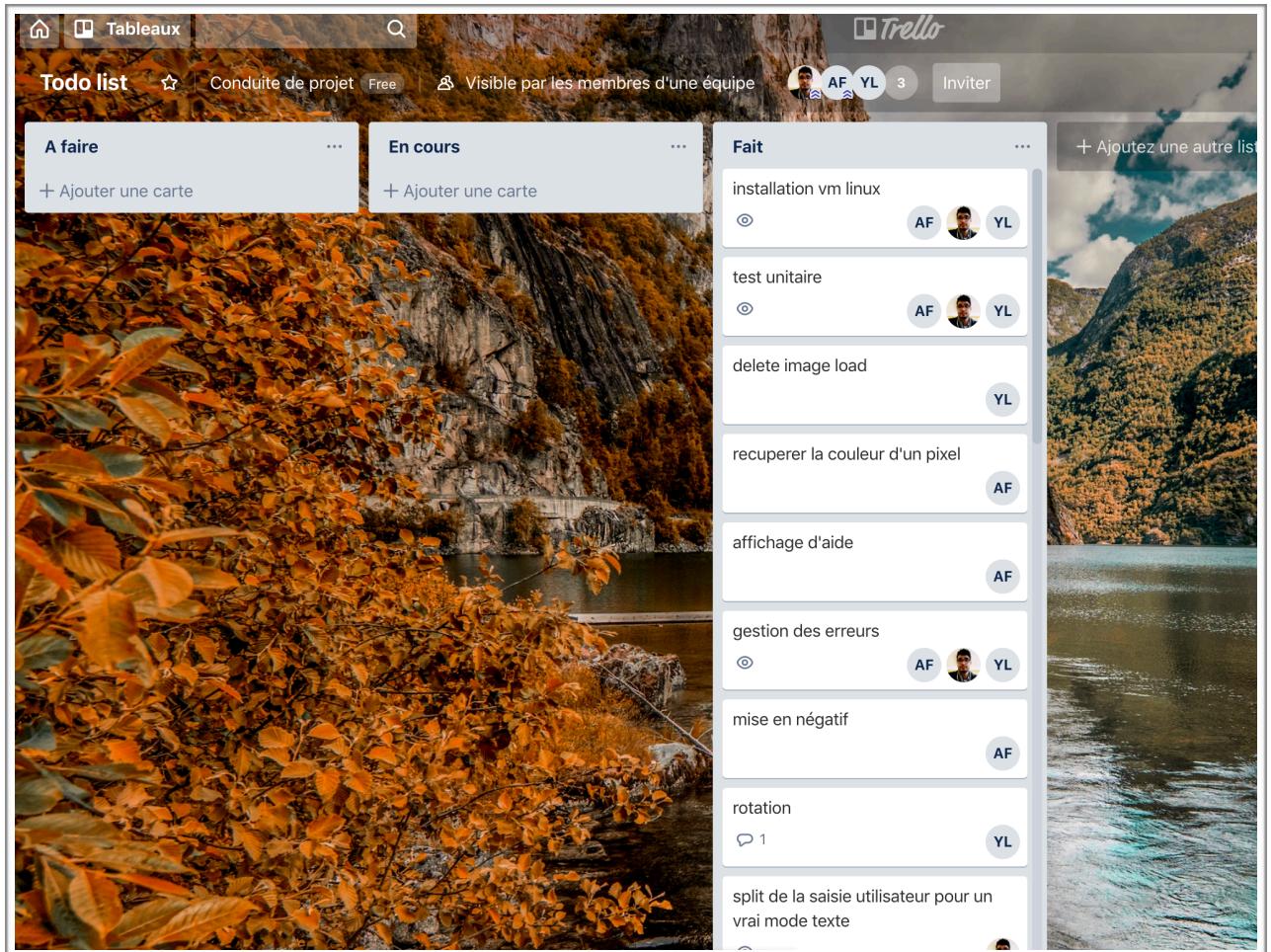
Organisation du travail

Afin de faciliter notre organisation, nous avons utilisé **trello**, un logiciel permettant de diviser les tâches d'un projet, ainsi chacun de nous 3 n'avait plus qu'à choisir l'une de ces tâche et y indiquer son état d'avancement.

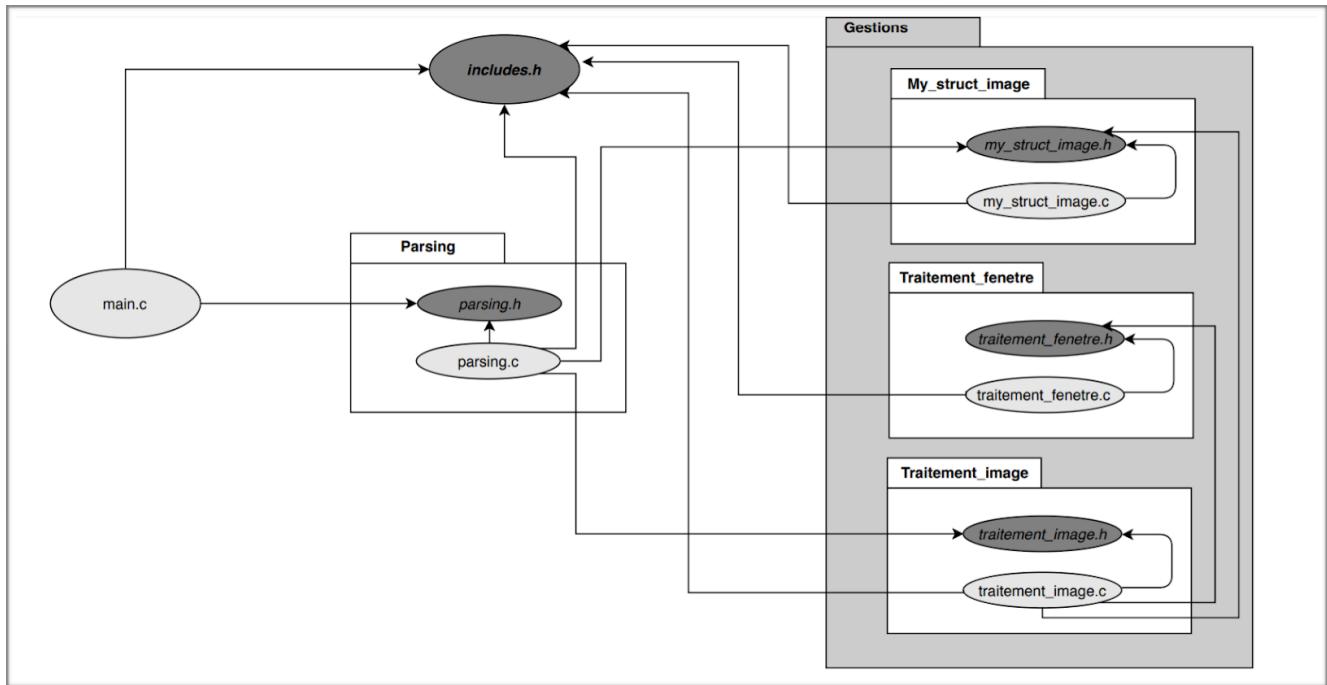
lien: <https://trello.com/b/aITDfMnb/todo-list>

Nous avons commencer par créer un prototype commun qui permettait de garder une image en mémoire à l'aide de la librairie SDL, puis nous avons créer plusieurs branches à notre projet afin d'y implémenter les fonctionnalités chacun de notre coté.

Une fois toutes les fonctionnalité écrites (voir readme du dépôt) nous avons mergé nos travaux et écrit des tests afin de s'assurer du bon fonctionnement de notre programme.



Graphe de conception



Les fonctions de notre programme sont divisé en 2 modules:

1. le module **Parsing** qui se charge de traiter les saisies utilisateurs afin d'y exécuter ou non les fonctions souhaitées par celui-ci.
2. le module **Gestion** qui est divisé en 3 sous modules :
 - le sous module **My_struct_image** qui permet d'enregistrer dans une structure les images que l'utilisateur souhaite modifier et de pour le programme afin qu'il puisse les traiter
 - le sous module **Traitement_fenetre** qui permet de configurer et gérer les événements des fenêtres qui serviront à afficher les images
 - le sous module **Traitement_image** qui permet d'appliquer des modifications à une image et de sauvegarder ces modifications

Fonctionnalités

- **exit** : quitte l'application.
- **load pathfile** : charge une image et lui donne un id.
- **delete id** : décharge l'image lié à id.
- **save id path [type/png/bmp]** : sauvegarde l'image lié à id.
- **show** : affiche toutes les informations des images chargés.
- **display id** : affiche l'image lié à id.
- **symv id** : applique une symétrie verticale à l'image lié à id.
- **symh id** : applique une symétrie horizontale à l'image lié à id.
- **rotate id nb** : effectue nb rotation vers la gauche à l'image lié à id.
- **drawzone id [width_origin] [height_origin] [width_end height_end]** : modifie la zone de dessin l'image lié à id en fonction du point supérieur gauche (width_origin, height_origin) et du point inférieur bas (width_end, height_end).
- **resize id width height** : modifie la résolution de l'image lié à id en fonction de width et height.
- **select id [width_origin=(default:0)] [height_origin=(default:0)] [width_end=(default:max)] [height_end=(default:max)]** : sélectionne une partie de l'image lié à id en fonction du point supérieur gauche (width_origin, height_origin) et du point inférieur bas (width_end, height_end) afin d'effectuer les action suivantes :
 - **cut [reverse=1]**: remplace la zone par du noir ou du blanc si l'option reverse est utilisé.
 - **copy position_x position_y** : copie la zone dans une autre zone d'origine le point supérieur gauche (position_x, position_y).
 - **fill r g b** : rempli la zone d'un mélange de couleur rgb, attention r, g et b doit être compris entre 0 et 255.
 - **switch r g b nr ng nb t** : recherche dans la zone les pixels ayant pour couleur un mélange de couleur r g b avec une tolérance -/+ t et les modifies par une mélange de couleur nr ng nb.
 - **neg** : applique une filtre négatif sur la zone.
 - **bw** : applique une filtre noir et blanc sur la zone.
 - **grey** : applique une filtre gris sur la zone.
 - **exit** : désélectionné la zone.