SOUTENANCE 1

2021-2022

Projet S4

On est là alors

Alexandre Devaux-Riviere Kaël Facon

Grégoire Vest Mathis Rabouille

Table des matières

1 Rappels - Projet Decolor

1.1 Notre Groupe

- Kaël Facon
- Grégoire Vest
- Mathis Rabouille
- Alexandre Devaux-Riviere (Chef de groupe)

Notre groupe est composé de 4 ex-lettons ayant dû rentrer en France suite au début du conflit entre la Russie et l'Ukraine. Nous nous sommes rassemblés au sein du groupe "On est là alors" car nous avions des idées en commun, nous voulions tous les quatre faire un logiciel en rapport avec le dessin et l'édition d'images. C'est donc dans cette optique que nous avons pensé à Decolor un logiciel linux.

1.2 Le sujet du projet

L'application (donc nommée Decolor) sera principalement orientée vers un logiciel très simple d'utilisation permettant à l'utilisateur de dessiner ou modifier une image afin de laisser libre cours à sa créativité. L'interface graphique de l'application, pour la partie outils, a une fonction qui permet de charger une image à partir d'un explorateur de fichier et de faire des modifications sur cette dernière ou de la manipuler. Par la suite, cette image peut être sauvegardée dans les fichiers locaux de l'ordinateur ou écrasée (au choix). Nous avons opté pour une interface sobre, ergonomique et surtout facile d'utilisation. L'intérêt algorithmique de notre projet se tourne vers l'imagerie, notamment vers le traitement et l'édition d'images comprenant la modification et le stockage de pixels.

1.3 Le choix des bibliothèques

1.3.1 Simple DirectMedia Layer (SDL) et (SDLgfx)

SDL :
Simple DirectMedia Layer est une bibliothèque de développement multiplateforme libre conçue pour fournir un accès de bas niveau au matériel graphique. Elle nous permet de travailler sur la manipulation d'images. Nous avons déjà pu utiliser cette bibliothèque lors de différents TPs ou encore lors de notre projet de troisième semestre qui consistait à résoudre un sudoku en faisant un OCR.

SDLgfx :
Le code de la bibliothèque SDLgfx fournit des routines de dessin de base et ajoute des fonctions utiles pour le zoom images et fait des traitements d'image de base sur des tableaux d'octets.

1.3.2 GIMP Toolkit - GTK

GTK est une boîte à outils multiplateforme gratuite et libre pour créer des interfaces utilisateur graphiques. Elle nous permettra de pouvoir réaliser une application simple et facile d'utilisation. Nous avons jugé que GTK serait l'outil parfait pour mener à bien nos ambitions.

2 Avancements - Projet Decolor

2.1 Structures LIFO / FIFO

Afin de mieux comprendre l'implémentation algorithmique de certains outils et de certaines fonctionnalités, nous introduisons ici les structures qui nous ont permis de les réaliser.

2.1.1 Outils - (FIFO)

Afin de manipuler certains outils comme celui du seau avec son parcours largeur de l'image, il a fallu créer une structure FIFO (une queue) que nous avons choisi circulaire et sans sentinelle pour une meilleure optimisation (liste chaînée circulaire).

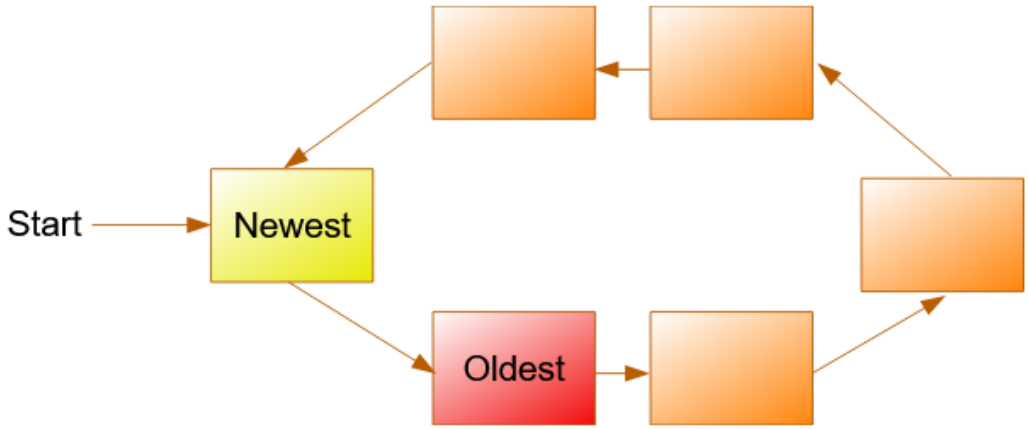

FIGURE 1 – Schéma de la structure FIFO choisie

2.1.2 Retour en arrière / Retour en avant - (FIFO + LIFO)

Dans le but de rajouter la fonctionnalité majeure pour notre logiciel que représente l'option de retour en avant / retour en arrière, il fallu créer une structure spéciale. En effet, le problème était de stocker dans une pile les modifications d'une image au fur et à mesure en enlevant les éléments du fond de la pile afin de limiter l'enregistrement des modifications (à 15 dans notre cas).

Nous avons de ce fait imaginé et implémenté la fusion d'une pile et d'une queue (FIFO + LIFO). De plus, du fait de l'interconnexion des options retour en arrière / avant, il fallait donc deux piles de même nature faites pour coïncider de tel sorte qu'un élément dépilé devait se retrouver dans la pile communiquante et être affiché.

Afin de simplifier l'explication de cette option, voici des schémas de la structure et des interactions que l'on peut y retrouver ("élément actuel" représente l'élément (image) affiché sur l'interface et "Data Element" représente un pointeur SDL_Surface) :

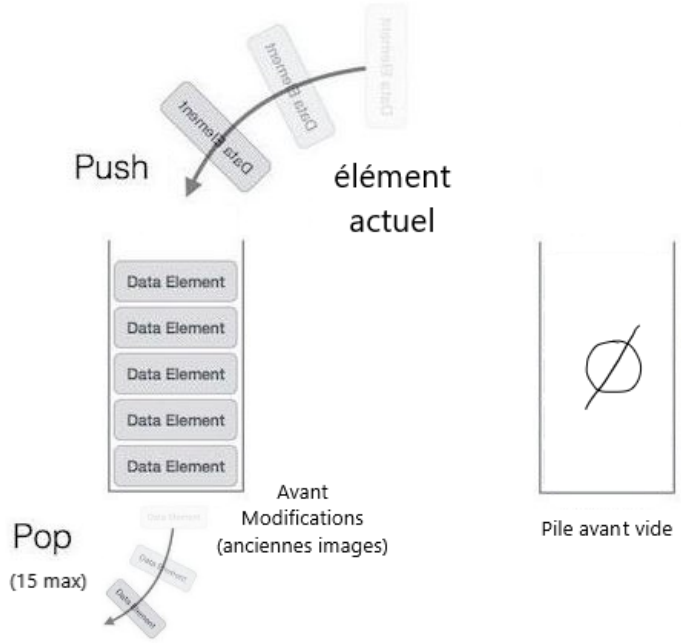

FIGURE 2 – Empilation des modifications de l'image

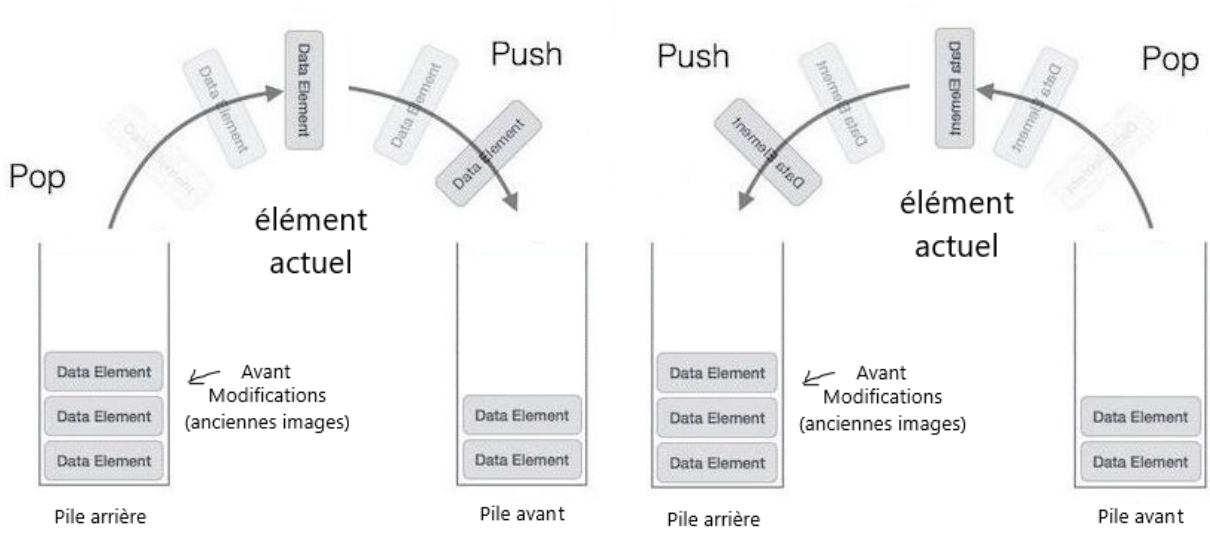

FIGURE 3 – Schéma du Retour en arrière (ctrl + Z) et Retour en avant (ctrl + Y)

Ainsi, pour résumer, à chaque modification de l'image, on empile celle-ci dans la pile arrière et on vide la pile avant. Lorsque l'on retourne en arrière, l'ancienne image s'il y en a une (celle en haut de la pile arrière) est affichée et l'image sur laquelle on était passe dans la pile avant. Lorsque l'on retourne en arrière, l'image actuelle est empilée dans la pile arrière et on affiche l'image dépilée de la pile avant.

De cette façon le retour en arrière et avant est optimisé et fonctionne parfaitement comme le fait un logiciel de type Paint3D.

2.2 Edition d'image - SDL

2.2.1 Algorithmes derrière les outils

Le crayon :

La création du crayon passe par trois étapes et possède plusieurs options (la couleur et la taille). Pour recréer un tracé de crayon, il y a, tout d'abord, une fonction qui s'occupe de dessiner un point de la couleur et de la taille sélectionnée en forme de losange.

Dans un second temps, une fonction s'occupe de dessiner un segment, continuité de points créés par la première fonction, entre deux coordonnées de l'image grâce à l'algorithme de Bresenham.

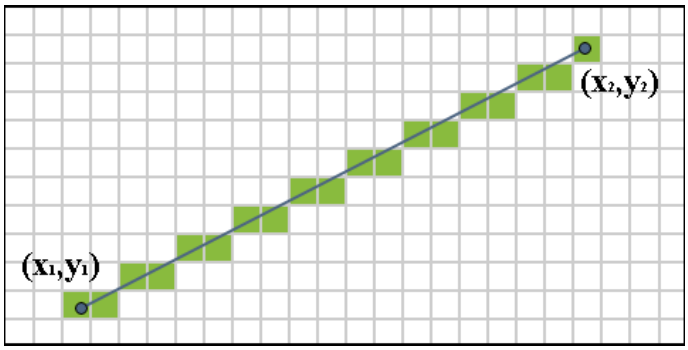

FIGURE 4 – Algorithme de Bresenham - Traçage d'un segment

La troisième partie du crayon se trouve au niveau de l'interface GTK avec une fonction qui, en continu, enregistre les coordonnées d'un clic maintenu sur la page et est expliqué dans les parties de ce rapport de soutenance concernant notre interface.

L'ensemble de ces fonctions contribue à créer un pinceau plutôt fluide et précis.

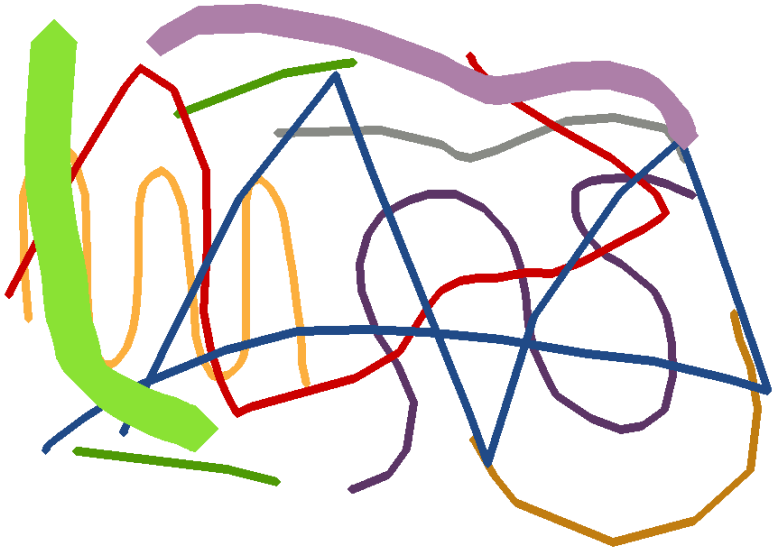

FIGURE 5 – Crayon Fonctionnel avec option couleur et taille

Le seau :

Le seau, outil emblématique d'un logiciel d'édition d'image, se fait par un parcours largeur du graphe qu'est l'image (pointeur SDL_Surface). Ce parcours utilise une structure FIFO comme introduite plus tôt. L'endroit du clic sur l'interface est envoyé dans la fonction du seau, et la couleur du pixel correspondant à ces coordonnées est enregistrée. Par la suite, l'ensemble des pixels aux alentours qui sont de la même couleur que celle enregistrée rentrent dans le parcours largeur et sont modifiés par la suite.

Le seau possède aussi un seuil de sensibilité afin d'inclure plus ou moins de variance de couleur de pixels à changer.

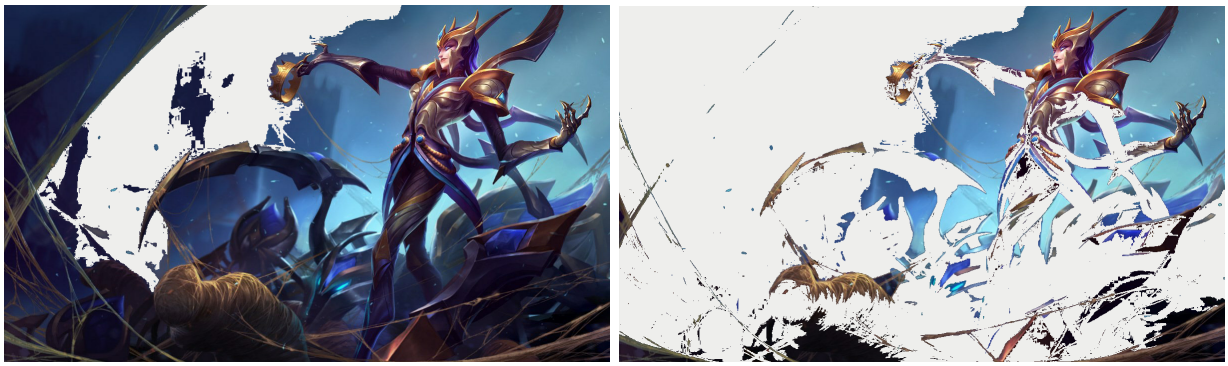

FIGURE 6 – Application du seau sur une image avec un seuil de 10% et 30%

FIGURE 7 – Application du seau sur une image avec un seuil de 60% et 90%

FIGURE 8 – Application du seau sur une image avec un seuil de 100%

La pipette :

Nous n'avons pas eu à implémenter la pipette. En effet, elle est déjà disponible directement sur l'interface avec GTK. Plus d'informations sont donnés dans la suite de notre rapport.

La gomme blanche :

Outil essentiel pour dessiner, la gomme utilise exactement le même algorithme que celui du crayon sauf qu'à la différence de celui-ci, la couleur sera toujours le blanc !

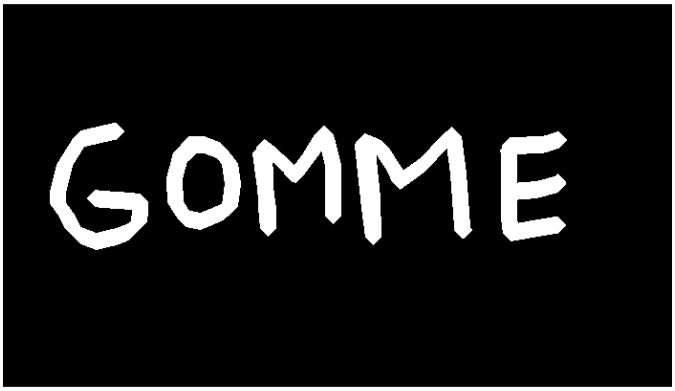

FIGURE 9 – Gomme fonctionnelle avec taille réglable

La gomme qui restore l'image :

Cette gomme reprend le fonctionnement de l'autre gomme sauf qu'à la place de placer du blanc sur les pixels des traits, on place les pixels aux même coordonnées de l'image précédemment chargée (blanc par défaut). Grâce à cet outil il est beaucoup plus simple de dessiner sans se préoccuper des potentielles erreurs de dépassement sur l'image modifiée.

FIGURE 10 – Gomme d'annulation fonctionnelle avec taille réglable

2.2.2 Formes 2D

Le segment :

Pour réaliser ce segment, on reprend un algorithme précédemment utilisé pour le crayon sauf qu'à défaut de tracer des segments en continu, seulement un segment entre deux coordonnées sera tracé sur l'image grâce à l'algorithme de Bresenham précédemment expliqué (coordonnées du début du clic et du relâchement du clic).

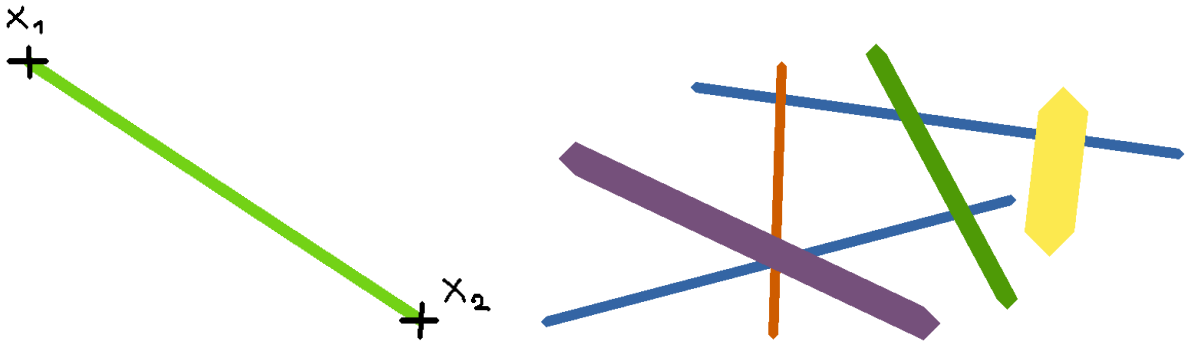

FIGURE 11 – Forme Segment avec couleur et épaisseur modifiable (Exemples X1 vers X2)

Le rectangle :

Afin de concevoir le rectangle vide, les coordonnées au clic puis au relâchement du clic sont enregistrées et un rectangle qui est intérieur à ces deux coordonnées est créé. Cela se fait grâce à l'aide d'une fonction de tracés verticaux et d'une fonction de tracés horizontaux appelées deux fois respectivement pour former les quatre cotés de la forme. De cette façon, l'utilisation de cet outil se fait de façon plus précise. La taille et la couleur sont aussi des options modifiables.

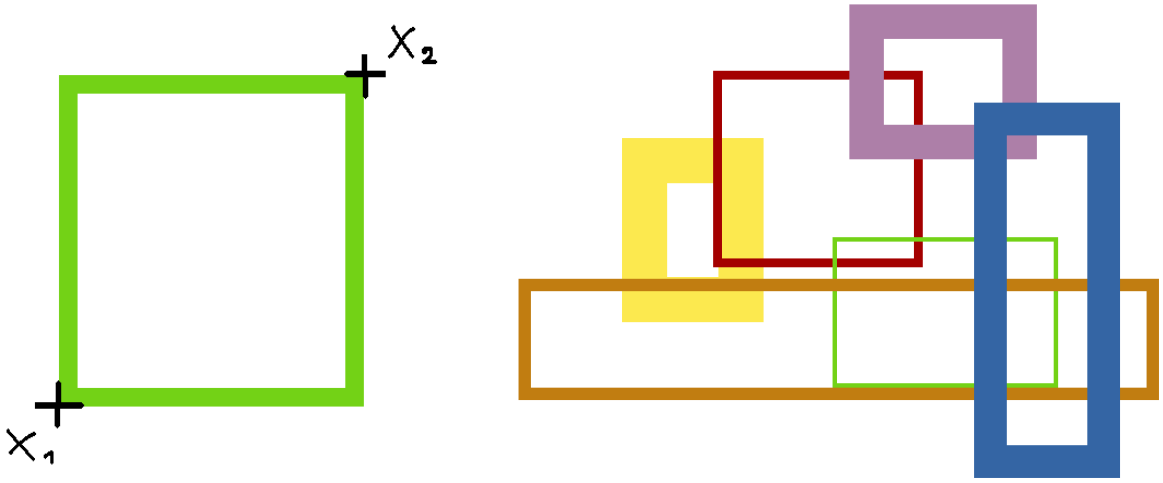

FIGURE 12 – Forme Rectangulaire avec couleur et épaisseur modifiable (Exemples X1 vers X2)

Le triangle :

Pour la conception du triangle vide, les coordonnées au clic puis au relâchement du clic sont enregistrées et un triangle qui est intérieur à ces deux coordonnées est créé. Certains calculs sont faits pour obtenir les coordonnées des sommets de celui-ci en fonction du sens des deux coordonnées X1 et X2 pour finalement obtenir les points A et B. Par la suite, trois segments sont formés entre X1 et A, A et B, ainsi que B et X1. De cette façon, l'utilisation de cet outil se fait de façon plus précise. La taille et la couleur sont aussi des options modifiables.

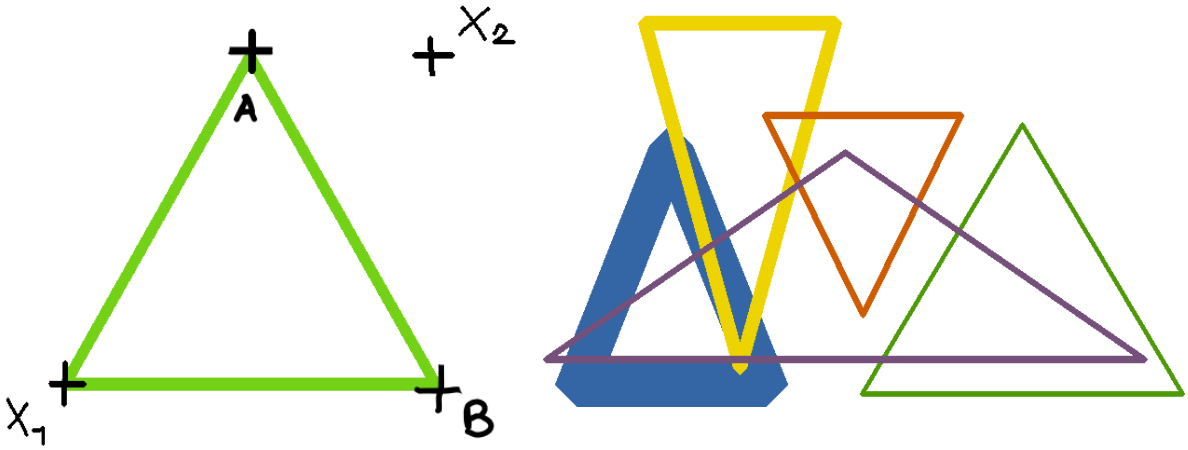

FIGURE 13 – Forme Triangulaire avec couleur et épaisseur modifiable (Exemples X1 vers X2)

Le cercle :

Pour la conception du cercle vide, les coordonnées au clic puis au relâchement du clic sont enregistrées et un cercle qui est intérieur à ces deux coordonnées est créé (de centre X1 et de rayon X1 - X2). Par la suite, huit arcs de cercles sont formés par symétrie centrale (X1) et se lient pour former le cercle entier grâce à l'algorithme du cercle de Bresenham. De cette façon, l'utilisation de cet outil se fait de façon plus précise et optimisée. La taille et la couleur sont aussi des options modifiables.

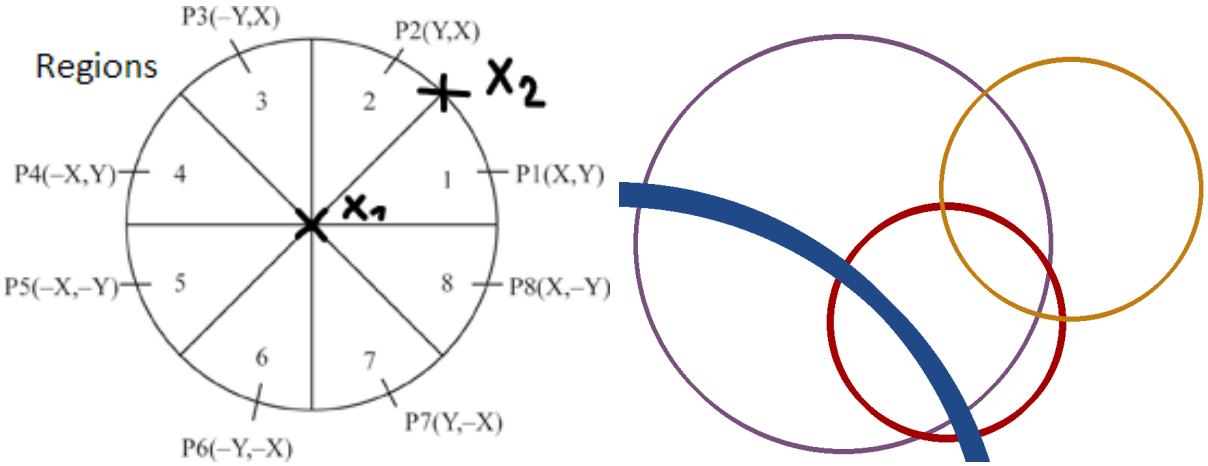

FIGURE 14 – Forme Circulaire de Bresenham avec couleur et épaisseur modifiable (Exemples X1 vers X2)

Formes 2D remplies

Les formes existent aussi de façon « remplies » mais les fonctions ne sont pas encore optimisées donc on ne peut pas encore accéder à cela depuis l'interface.

2.2.3 Les filtres

Une des fonctionnalités « basiques » d'un logiciel de dessin/retouche photo et d'avoir quelques filtres simples pour changer les couleurs/tons de l'image rapidement et efficacement. Nous avons donc implémenté quelques filtres qui permettent de modifier les images que l'utilisateur a créé :

— **Filtre de nuance de gris** : le plus basique existant, les pixels de l'image passent tous par la formule « 0.3*r + 0.59*g + 0.11*b » pour avoir une image qui reste dans les mêmes tons mais en gris.

FIGURE 15 – Image de base

FIGURE 16 – Filtre gris

— **Filtre négatif** : les couleurs deviennent leur opposé. Il suffit pour cela, pour chaque pixel, de modifier sa couleur en les soustrayant à 255 (la valeur maximale).

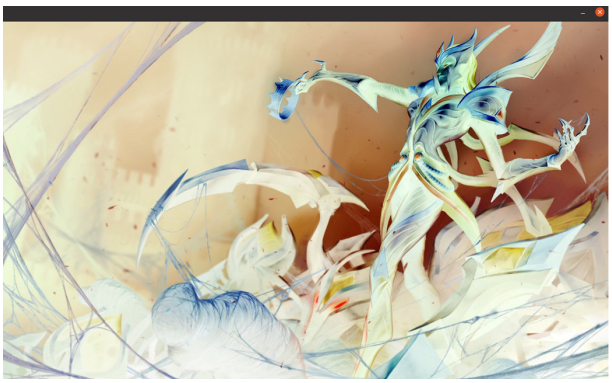

FIGURE 17 – Filtre négatif

— **Filtre de couleur customisable** : comme nous n'avions pas trop d'idée de filtre à faire sans que ça nous prenne trop de temps, nous avons créé un filtre avec plusieurs paramètres pour customiser le choix du rendu. Ce filtre prend en paramètre : la surface évidemment, un booléen constant pour chaque composante de la couleur pour savoir celles qu'on gardera, une valeur entre 0 et 255 pour savoir à quel seuil les composantes qu'on ne garde pas devront être. Par exemple, sur cette image on garde le rouge et on met le vert et le bleu à 0 :

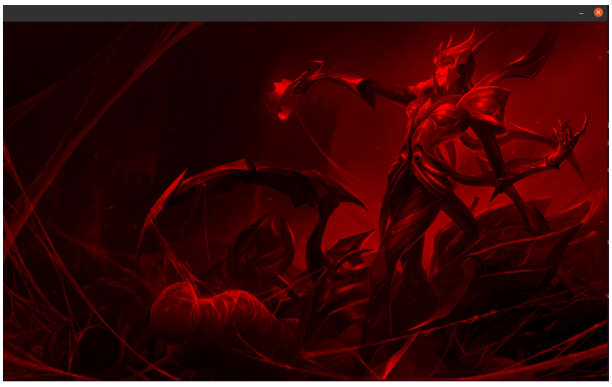

FIGURE 18 – Filtre rouge

Sur celle-ci, on garde le vert et le bleu et on fixe le rouge à 100 :

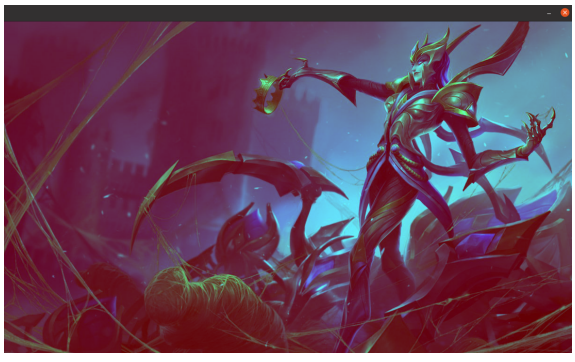

FIGURE 19 – Filtre "bleu-vert"

Et, évidemment, on peut cumuler ces filtres basiques pour faire des filtres plus complexes.

2.2.4 Transformation sur l'image

On a pu implémenter quelques transformations simples sur les images. Ces fonctionnalités ne sont pas encore toutes disponibles sur l'interface mais elles existent cependant dans notre code. On peut compter :

— **Le rognage** : l'utilisateur devrait choisir le x et le y du départ et de l'arrivée, pour effectuer le rognage sur toute la zone en dehors.

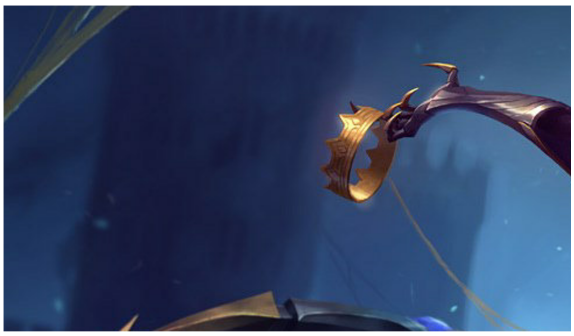

FIGURE 20 – Image rognée

— **L'inversion** : l'utilisateur pourra aussi choisir de faire une symétrie axiale de l'image (plus communément appelé « outil miroir ») horizontalement ou verticalement.

FIGURE 21 – Image inversée

— **La rotation** : L'utilisateur pourra simplement effectuer la symétrie centrale en choisissant l'angle qu'il voudra.

FIGURE 22 – Image tournée de 3 degrés

— **L'agrandissement/rétrécissement** : L'utilisateur pourra agrandir ou rétrécir l'image.

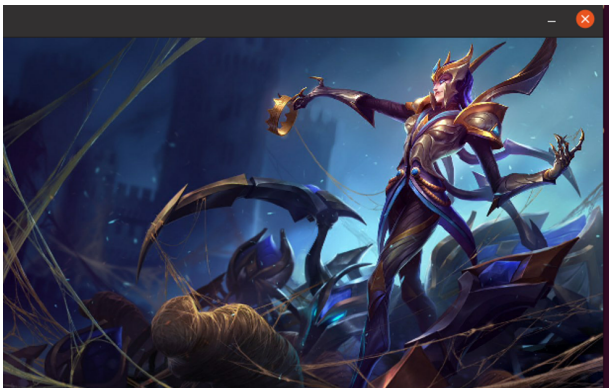

FIGURE 23 – Image rétrécie (elle est plus pixelisée)

2.3 Interface - GTK

Premiers prototypes :

Nous avons commencé le travail pour cette soutenance à partir de rien, nous avions besoin d'un début d'interface pour implémenter les premières choses basiques du logiciel comme le fait de pouvoir charger une image et de la sauvegarder. Nous avons donc utilisé gtk 3.0 pour créer une simple interface répondant aux problème que nous avions. Le fait d'avoir travaillé sur gtk lors du projet d'OCR du semestre dernier m'a beaucoup servi dans ce cas. Je savais pertinemment que cette interface était simple voir même trop simple mais je savais aussi que nous allions en changer pour une version finale plus complète.

FIGURE 24 – Premier prototype d'interface

Ensuite, lorsque nous avons eu quelques outils de base prêts, Kaël a créé l'architecture de l'interface à l'aide de Glade, logiciel que nous avions découvert lors d'un TP de programmation du semestre dernier. Une fois ceci fait nous avons pu commencer à structurer le code de l'interface où nous avons commencé par lier les éléments de l'interface aux fonctions que les autres membres avait créés sur SDL.

FIGURE 25 – Prototype de l'interface graphique actuelle par Kaël et Alexandre

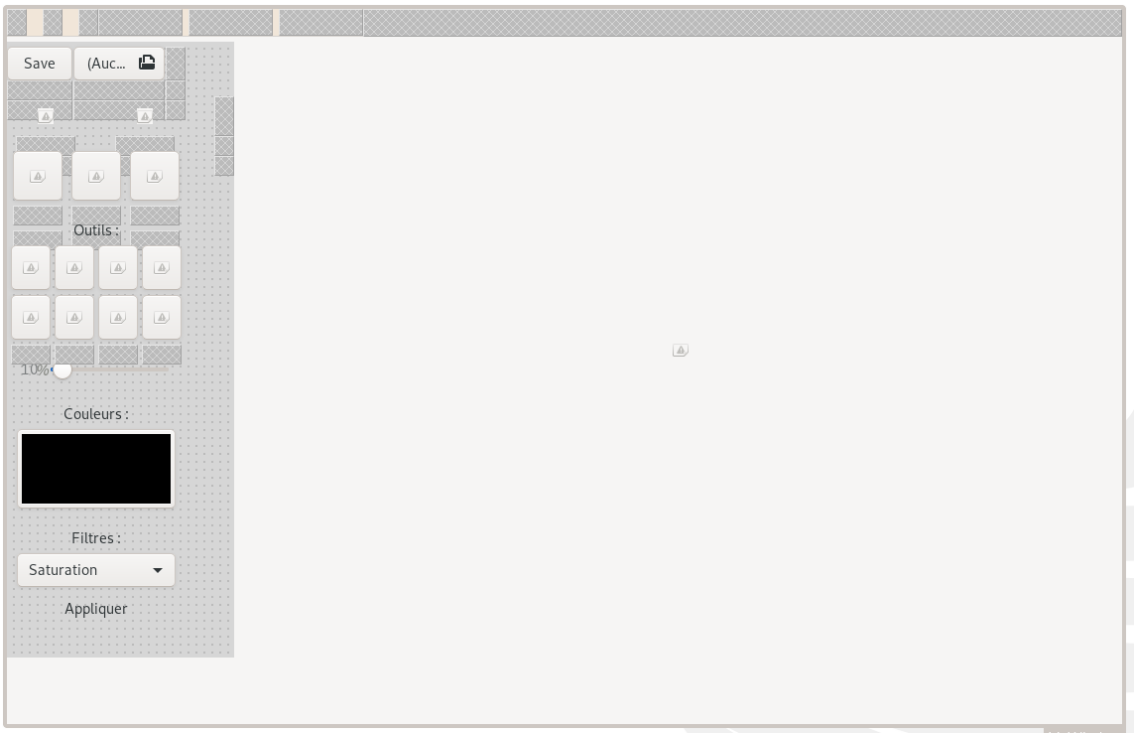

FIGURE 26 – Prototype de l'interface actuelle sur Glade

2.4 Ergonomie de l'interface

Notre interface est pensée pour être ergonomique à souhait !
En effet ce qui est le plus accessible se trouve au milieu de l'écran c'est pourquoi nous avons décidé d'y placer l'outil de sélection de couleur qui est, pour sûr, un élément majeure de n'importe quelle application de dessin ! Ce dernier nous permet également de séparer les outils classiques et les filtres, de manière à bien distinguer les deux !

Nous avons regroupé les boutons utilitaires permettant la gestion de l'image, les boutons de "Sauvegarde" et de "Chargement", les boutons "Retour arrière" et de "Retour avant" et les boutons "Zone de Sélection", "Zone de Texte" et "Rogner".

Situé en dessous de ces outils de gestion nous avons mis les outils de dessin classique ("Pinceau", "Seau", "Gomme") sur la première ligne et les outils géométriques ("Segment", "Rectangle", "Triangle", "Cercle") sur la seconde ligne.

On retrouve en dessous du bouton sélection de couleur un panneau déroulant correspondant à la zone de sélection des différents filtres. Une fois le filtre sélectionner il suffit d'appuyer sur le bouton "Appliquer", un jeu d'enfant !

2.5 Icônes de l'interface

Pour illustrer les différents outils de l'application il fallait créer des icônes les représentants. Pour rappel voici les différents outils pour l'instant présents dans l'application :

- Pinceau
- Seau
- Gomme
- Gomme d'annulation
- Segment
- Rectangle
- Triangle
- Cercle
- Rogner
- Zone de Sélection
- Zone de Texte
- Retour arrière/ Retour avant

Nous avons décidé de créer les icônes par nous même. Cela représentait deux avantages certains, éviter des problèmes de droit d'auteur tout en nous permettant d'avoir une unicité dans le style des icônes ! Pour le style des icônes nous avons opté pour des icônes de dimensions de 20x20 pixels sur fonds transparents, toutes avec un même schéma de couleur : l'outil en NoirBlanc et ce que l'outil nous permet de faire sur la zone de dessin en dégradé de couleur (rouge et bleu).

2.5.1 Implémentation des outils dans l'interface

Nous avons d'abord commencer par implémenter les 8 outils de bases dans l'interface, pour ce faire il nous a fallu trouver un moyen de gérer le clic et les coordonnées de la souris, nous savions que cela serait nécessaire, j'avais donc déjà créé un prototype de fonction s'occupant du clic simple. Maintenant que je savais ce dont j'avais besoin, nous avons créé une fonction qui met a jour en temps réel la position de la souris et l'enregistre à l'appui et au relâchement du clic. Ceci fait nous pouvions implémenter les outils, certains ayant simplement besoin des coordonnées de l'appui du clic par exemple le seau et certains ayant besoin des coordonnées de l'appui et du relâchement comme les formes.

FIGURE 27 – Outils

La deuxième chose que nous avons décidé d'implémenter est le bouton de sélection des couleurs et également le sélecteur permettant de choisir l'épaisseur des outils. Pour ce faire nous avons utilisé deux variables globales tenue à jour par des fonctions liées aux signaux respectifs du bouton et du sélecteur.

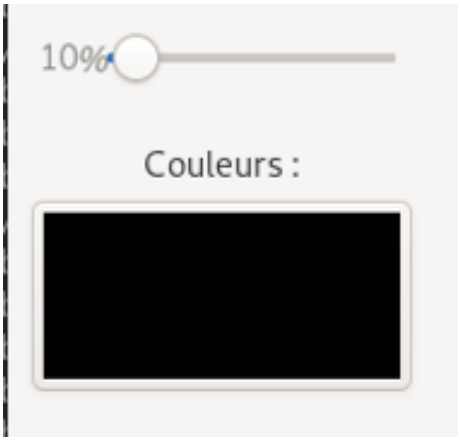

FIGURE 28 – Les outil en question

Le plus dur a été d'implémenter le pinceau à l'aide de la fonction traçant des segments entre deux points car nous n'avions aucune idée de comment faire. Pour implémenter cet outil j'ai d'abord eu beaucoup de mal et nous avons essayé de suivre des tutoriels mais finalement une idée simple et efficace nous est venue, simplement garder en mémoire la dernière position de la souris et la relier à la nouvelle par un segment lorsque l'outil du pinceau est choisi et que le clic est enfoncé. Implémenter ceci n'a pas été très compliqué mais l'outil du pinceau reste améliorable.

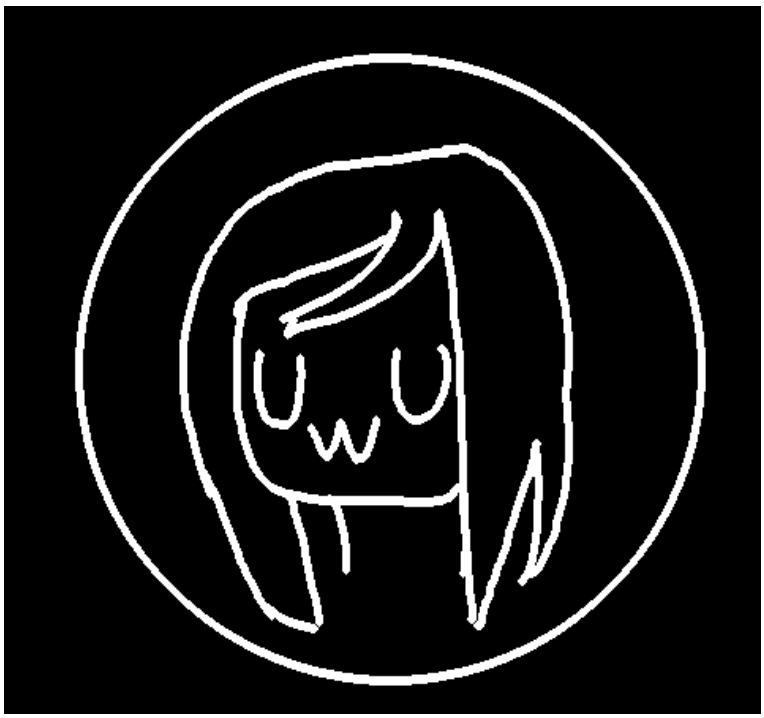

FIGURE 29 – Exemple des possibilités du pinceau

Pour savoir à tout moment quel outil est sélectionné nous avons décidé d'utiliser une variable globale qui serait modifiée à l'appui des boutons de sélection des outils.

2.5.2 Mise à jour de l'affichage

Nous avons eu aussi beaucoup de problèmes pour l'affichage, nous voulions d'abord utiliser une gtk image et une SDL Surface reliées par la fonction update qui utiliserais un fichier temporaire pour passer de gtk image à SDL Surface pour y appliquer les modifications souhaitées puis de nouveau faire la conversion dans le sens inverse pour mettre à jour l'affichage mais avec cette solution il était impossible de gérer la souris et ses déplacements et clics. Pour remédier à ça nous avons utilisé un autre module de gtk, la DrawingArea avec celui-ci nous avons pu efficacement gérer la souris à l'aide d'une Event Box gérant les signaux liés au déplacements et au clic de la souris. Il nous fallait donc juste une nouvelle fonction de mise à jour de l'affichage et pour ce faire j'ai remplacé celle associée au signal «draw» qui est appelée à chaque fois que la zone de dessin est affichée, ou que nous la déclenchons.

Dans cette fonction qui est une de celle qui m'a pris le plus de temps j'ai gardé le principe de fichier temporaire mais pour simplifier les choses nous avons fait le choix de créer une SDL Surface au lancement du logiciel que nous gardons jusqu'à sa fermeture. Toutes les modifications créées par les différents outils sont fait sur cette SDL Surface et ensuite nous appelons la fonction de mise a jour de l'affichage qui va faire le parallèle entre SDL et GTK grâce a un fichier temporaire unique.

2.6 Redimension de la fenêtre et centrage de l'image

Afin de rendre l'utilisation de l'application plus agréable nous avons fait en sorte que les dimensions et la disposition de la fenêtre s'adapte en fonction de l'image téléchargée.

Pour cela nous agrandissons la taille de la fenêtre si l'image téléchargée est plus grande que les dimensions 1280x850 pixels.

A l'inverse si l'image est plus petite que les dimensions précédemment citées, l'image sera recentrée au milieu de la zone de dessin.

2.6.1 Conclusion sur l'interface

Au final pour cette soutenance nous avons une interface qui permet de bien utiliser les outils développés par les autres membre du groupe. Malgré quelques problèmes rencontrés lors du développement, comme celui de la gestion de la souris, du choix entre GTK Image et GTK Drawing Area, tous les problèmes on été réglés rapidement et de façon efficace.

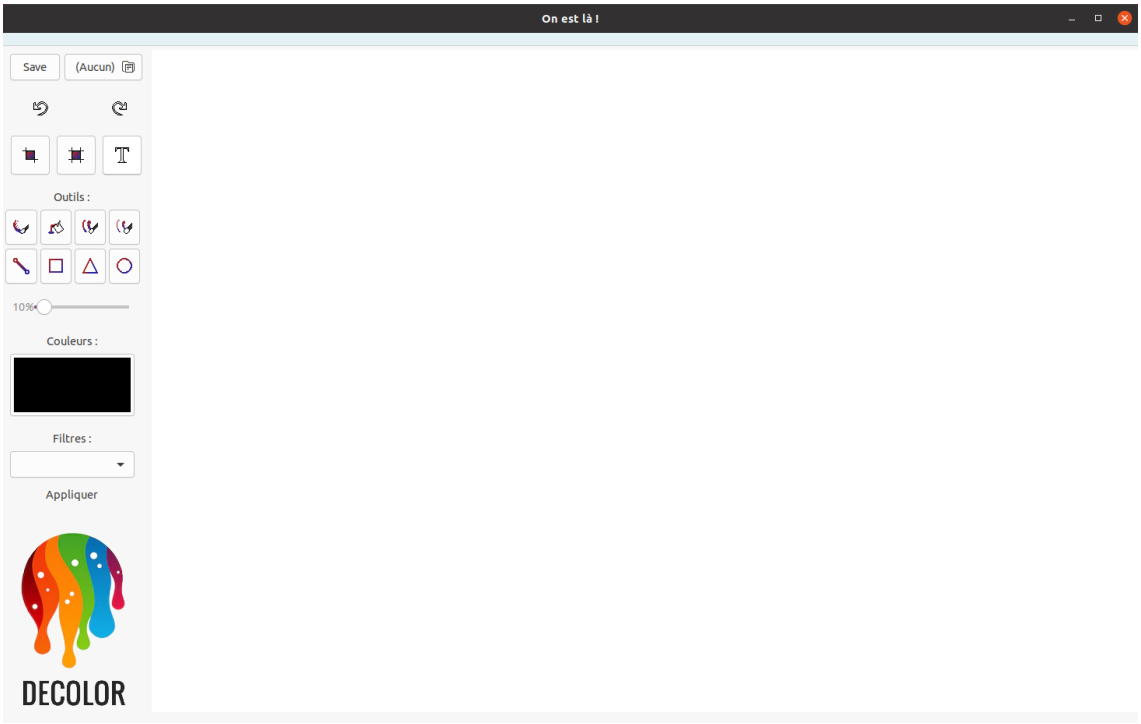

FIGURE 30 – Interface actuelle du logiciel

2.7 Lancement de l'application

Comment utiliser Decolor ?
— Utiliser la commande make sans aucun argument.
— Taper ensuite ./decolor.
— Une interface va maintenant apparaître et vous permettre de modifier et de créer des images.

Il est possible de taper 'make clean' pour nettoyer chaque fichier poubelle et fichier compilé. Voici par ailleurs nos options pour compiler notre projet.

```
CC = gcc

CPPFLAGS = `pkg-config gtk+-3.0 --cflags sdl` -MMD
CFLAGS = -Werror -Wextra -std=c99 -g -Wall
LDLIBS = `pkg-config gtk+-3.0 --libs sdl` -lSDL_image -lSDL_gfx -lm
```

FIGURE 31 – Options pour make de notre projet

2.8 Objectifs et projections pour la deuxième soutenance

Pour la prochaine soutenance, nous souhaitons faire au mieux pour intégrer les fonctions pour couper l'image, changer sa taille, intégrer du texte, ajouter des formes pleines, ajouter des filtres, améliorer la graphisme général de l'interface avec un fichier CSS, implémenter des calques facilitant les retouches de l'image et peut-être inclure le transparent comme couleur pour l'ensemble des outils. D'autres ajouts au fur et à mesure de l'avancement du projet seront aussi étudiés et envisageables.

3 Site internet du logiciel

Le site Web de notre projet est avant tout le site Web de notre projet (OELA - Decolor). Ceux-ci sont respectivement représentés au travers de différents articles ainsi que liens de téléchargements et autres interactions dans le but d'attirer les visiteurs à télécharger l'application.

La structure du site Web est entièrement terminée mais sera mis à jour suite aux nouveaux ajouts.

Voici le lien pour aller le consulter, **`https://topagrume.github.io/decolor_web/accueil.html`**

Possédant déjà un compte *GitHub*, nous nous sommes tournés vers ce support pour héberger le site et le mettre à jour automatiquement au fil des nouvelles versions envoyées vers la sauvegarde distante. A l'aide de nos connaissances en HTML (pour la mise en forme des pages), CSS (pour le style et l'apparence des pages) et JavaScript (pour ce qui est du code à exécuter côté client surtout les effets dans notre cas), des forums en ligne ainsi que d'exemple d'organisation, de répartitions du texte, d'effets et d'autres outils disponibles librement sur Bootstrap et d'autres espaces de libre-échange, nous avons pu réaliser ce site.

Nous nous sommes basés sur le thème de notre logo Decolor pour l'apparence du site qui allie des couleurs vives.Le langage CSS n'étant pas toujours évident, cela est passé par beaucoup de tâtonnements, de découvertes puis aussi d'exploration de documents sur Internet, mais nous sommes aujourd'hui très satisfaits du résultat qui s'adapte de plus à de nombreuses tailles d'écran mis à part l'arbre d'historique de notre projet avec les écrans de téléphones. La navigation est alors agréable, intuitive et plus aisée.

3.1 La structure du site

Il a tout d'abord fallu réfléchir à la structure du site, son arborescence, l'organisation de chacune des pages, la répartition des images, du texte, des liens, d'un historique de projet, de texte et de statistiques. Nous avons décidé de choisir une structure simple composée d'un en-tête dans lequel on retrouve le nom de notre groupe, le nom et le lien vers les différentes pages du site : (Projet, Avancées, Documentation, Équipe) ainsi qu'un bouton « télécharger ». Puis dans la suite de la structure un corps de texte qui laisse libre recours à nos choix et explications et enfin un pied de page destiné à n'être qu'un simple décor.

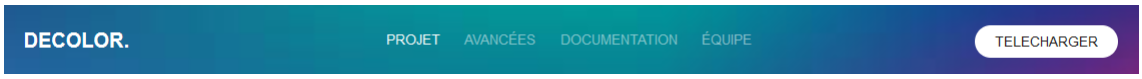

FIGURE 32 – Structure du site

3.2 Page d'accueil

Bien en évidence, nous avons opté pour directement placer un bouton afin de renvoyer vers un lien de téléchargement du logiciel en version normale/lite. Dans le corps de la page, nous avons opté pour une présentation brève de notre logiciel, et de ses qualités, répartie en différents onglets mis en parallèles avec des images représentatives, puis de nos valeurs en tant que développeur, une présentation brève de l'intérêt algorithmique et de son fonctionnement et enfin un petit paragraphe sur notre politique ainsi que le choix de notre Logo.

Plus bas, il nous a semblé important présenter les différents points d'avancement par des images représentatives explicites.

Enfin en pied de page, nous avons implémenté des statiques à titre purement informatif et esthétique sur le nombre de nos commit, d'ajouts réussies, d'étapes réussies et enfin le nombre d'heures qui ont été dédiées au projet dans son ensemble.

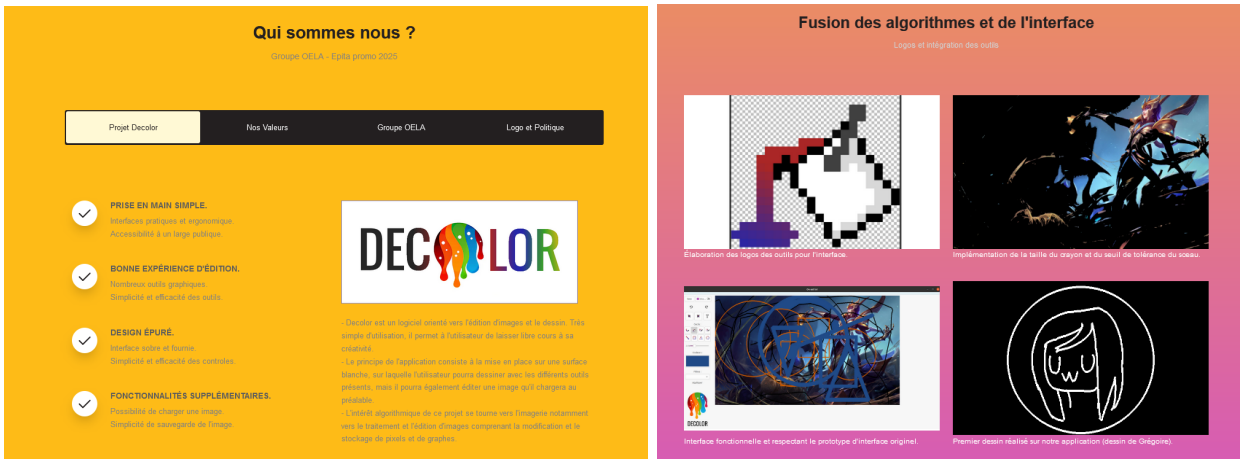

FIGURE 33 – Présentation du projet et des ajouts

3.3 Page d'avancements

Nous avons ici opté pour présenter les différentes étapes clés du projet passées tout au long de notre semestre (mars à juin) avec notamment de nombreux détails sur l'état de notre avancée, de nos choix, de nos réorientations, de notre productivité, de nos implémentations et étapes passées.

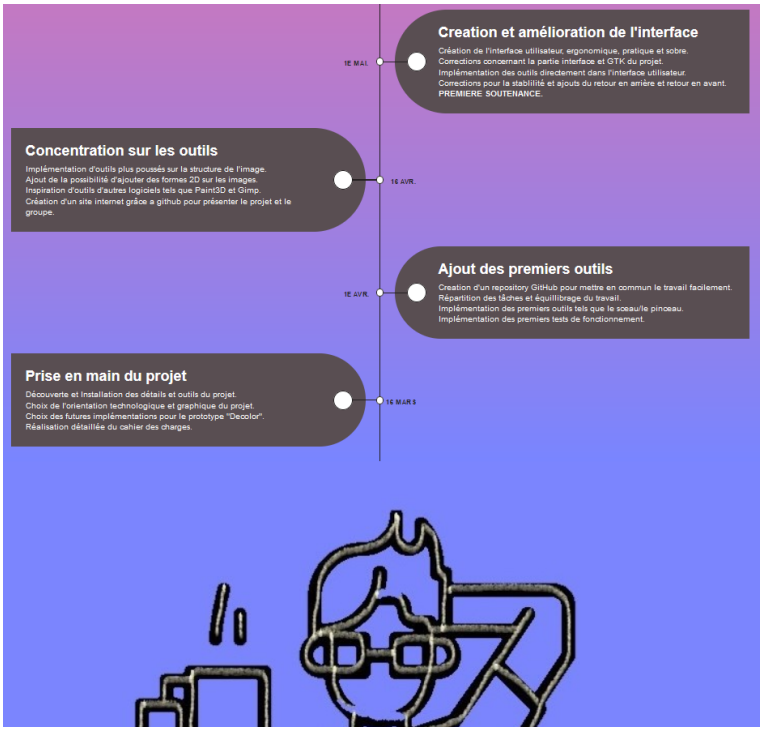

FIGURE 34 – Page d'avancements en fonction du calendrier

3.4 Page de documentation

Dans cette partie nous avons rassemblé l'ensemble des documents qui nous ont été demandés durant l'avancement de notre projet à savoir notre cahier des charges, première et deuxième soutenance. Elles

sont présentées avec leurs différentes caractéristiques avec un bouton pour les télécharger.

Dans la lignée des présentations, nous avons rassemblé l'ensemble des logiciels et des outils qui nous ont permis de faire aboutir notre projet.

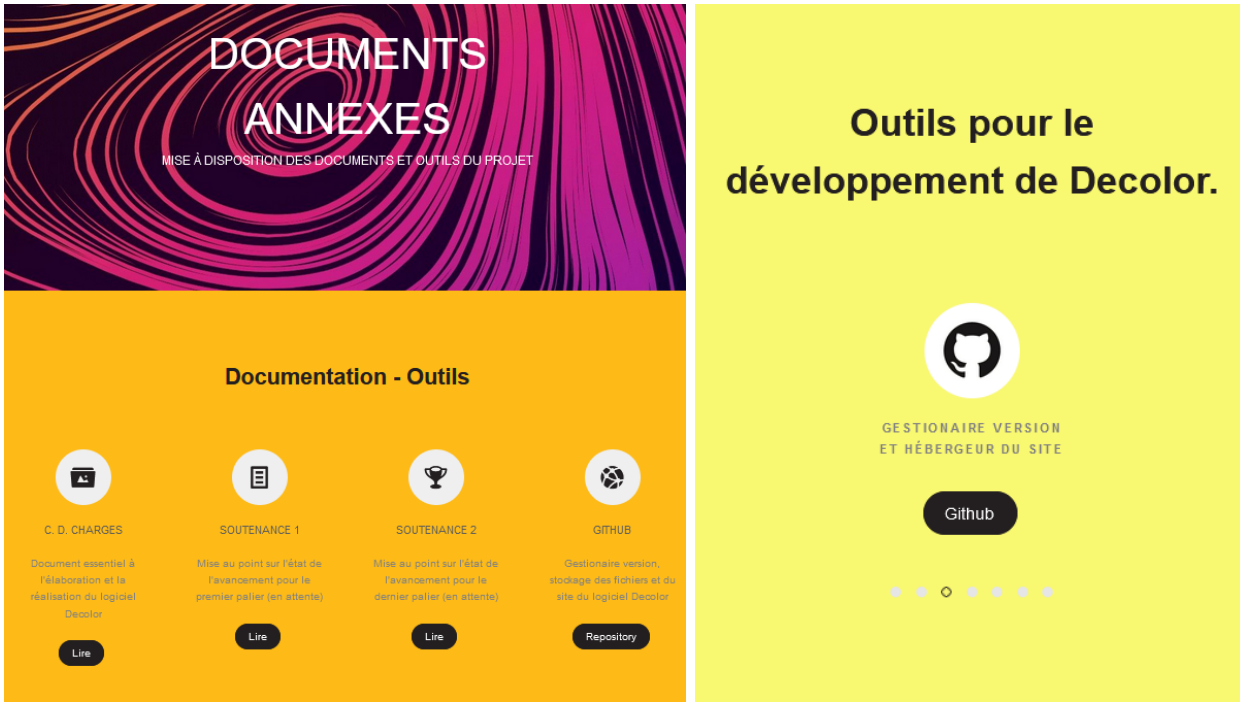

FIGURE 35 – Page documentations et outils

3.5 Page de l'équipe

Il nous a semblé important de présenter les différents membres du projet individuellement avec notamment leurs domaines de travail. Les liens vers le Twitter et le GitHub de chaque membre est renseigné sous chaque personne. Nous avons aussi simplement disposé quatre moyens pour pouvoir nous contacter ou qui concernent la partie communication de notre projet.

FIGURE 36 – Présentation de l'équipe

3.6 Page de téléchargement

Lorsque le bouton télécharger est activé, un ".tar.gz" correspondant à la version choisie de notre projet est téléchargé. Après l'avoir décompressé, il est fonctionnel. Une version du projet en version lite (sans fichiers de test/inutiles) et une version normale sont actuellement disponibles au téléchargement.

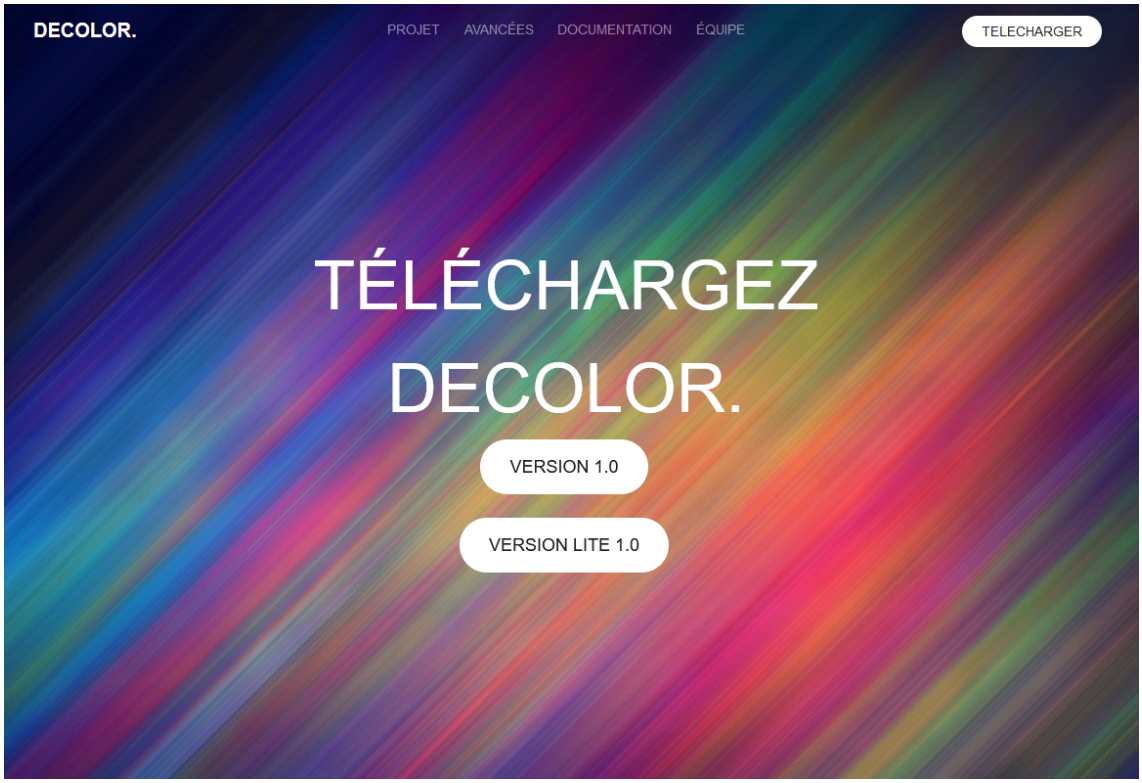

FIGURE 37 – Page de téléchargement

4 Découpage du projet

4.1 Tableau de répartition des tâches

	Alexandre	Kaël	Grégoire	Mathis
GTK - Interface		**Supp**		**Resp**
Interface graphique globale		X		*
Implémentation des outils	*	*	*	X
Chargement et enregistrement des fichiers				X
Implémentation de l'environnement de travail	*	*		X
Fenêtres informartives		*		X
SDL	**CoResp**		**CoResp**	
Création des outils	X		*	
Création des formes 2D	X		*	
Gestion des images	*		X	
Edition de la structure des images	*	*	X	
Application de filtres		*	X	*
Structure du Projet	**Supp**	**Resp**		
Ergonomie de l'application	*	X		
Compilation des fichiers (Makefile, headers...)	X	*	*	*
Communication	**Resp**	**Supp**		
Site Web	X	*		
Rendus Overleaf	X	*	*	*

Légende :

X = Responsable, * = Suppléant

4.2 Prévisions soutenances

Soutenance / Tâche	Prévisions Soutenance 1	Accompli Soutenance 1	Soutenance 2
GTK - Interface	70	80	100
SDL	50	70	100
Structure du Projet	60	70	100
Communication	80	80	100

TABLE 1 – Tableau des soutenances (en %)

5 Développement sur les tâches réparties et travail à venir

5.1 Schémas Github - Contributions de chacun

Pour une raison non connue, les contributions en lignes ajoutées et supprimées de Grégoire n'apparaissent pas. Cependant il est possible de voir son nombre de commit dans la deuxième photo (Il est en 3e position).

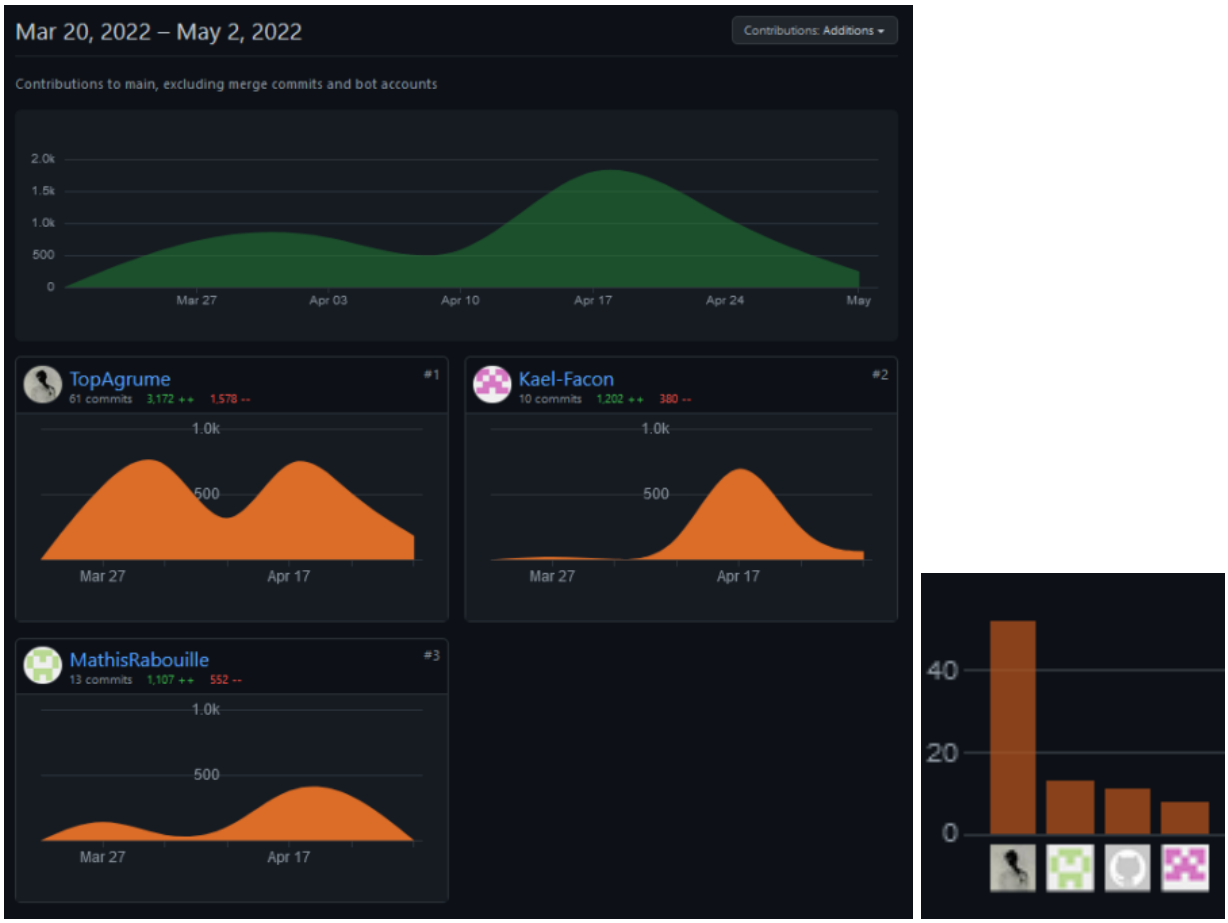

FIGURE 38 – Contribution GitHub avec respectivement Alexandre, Mathis, Grégoire et Kaël

5.2 alexandre.devaux-riviere

Au cours de cette première période travail, mon rôle en tant que chef de projet à été d'organiser et motiver mon équipe. Premièrement j'ai pu créer le répertoire du projet a l'aide de *Github* qui nous a vraiment été très utile. Puis, j'ai pris la décision de m'occuper principalement de nos outils et algorithmes de modification de l'image tels que le seau, les formes 2D, le site web du projet, les options de retour en arrière / retour en avant, l'ergonomie de l'interface et dans une moindre mesure du pinceau.

Même si mes responsabilités étaient plus importantes pour la partie SDL de notre projet et de notre site, j'ai aussi eu la chance de pouvoir toucher à l'ensemble des domaines de notre projet avec la liaison GTK / SDL et les différents ajouts sur l'interface avec GTK.

La cohésion du groupe, aidée par l'amitié qui nous lie, a fait de ce cette première partie de projet une réussite. Par la suite, je compte m'occuper des calques, avec Grégoire, par l'implémentation de nouvelles structures et aussi de la création de nouveaux outils et de certaines corrections sur la partie GTK.

5.3 kael.facon

Personnellement j'ai été surpris par la vitesse à laquelle le projet s'est développé et l'ampleur qu'il a pris !

Nous avons étonnamment de l'avance sur nos prévisions ce qui laisse un champ libre à notre imagination quant aux prochains ajouts pour le projet !
Je pense que la bonne cohésion d'équipe, la bonne répartition des tâches et la motivation des différents membres vis-à-vis du projet, a aidé à son développement rapide et efficace ! Sans compter sur notre chef de projet qui était investit et savait où mettre les échéances afin que le projet avance sans discontinuité.

Pour la suite de ce projet et la prochaine soutenance je prévois d'aider Mathis dans l'implémentation de nouvelles fonctionnalités ainsi que d'embellir l'interface grâce a un fichier CSS.

5.4 gregoire2.vest

De mon côté, je vais devoir faire un gros travail sur les calques, je pense que ça sera un bon défi algorithmique. Cependant, l'implémentation des calques va être assez difficile à mettre en place dans l'interface donc je cherche des solutions pour rendre cela plus simple.

5.5 mathis.rabouille

La suite du projet pour moi va consister à continuer d'implémenter toutes les nouvelles fonctionnalités que nous allons créer dans l'interface. Comme écrit ci-dessus il faudra intégrer les filtres sur l'interface. Je vais également aider sur d'autres parties car la majeure partie de mon travail est déjà faite pour l'interface, je vais, pour finir, chercher des moyens d'améliorer l'interface pour la rendre plus agréable a utiliser.

6 Nous contacter

— Alexandre Devaux-Rivière : alexandre.devaux-riviere@epita.fr
— Kaël : kael.facon@epita.fr
— Grégoire: gregoire2.vest@epita.fr
— Mathis : mathis.rabouille@epita.fr