

EPIPAINTOR

par Fast Thinking

Rapport de soutenance : Soutenance 1.



MAHRAOUI Adam - HAMDOUN Zaky - VARLIETTE Robin - AZIKA Christian

Table des matières

1	Introduction	5
1.1	Le projet : Epipaintor	5
1.2	Le groupe : Fast-thinking	5
1.3	Présentation des membres	5
1.3.1	Adam Mahraoui	5
1.3.2	Christian Aziaka	5
1.3.3	Zaky Hamdoun	6
1.3.4	Robin Varliette	6
2	Rappel sur le projet	7
2.1	Idée	7
2.1.1	Nom	7
3	Le projet	8
3.1	Les outils	8
3.1.1	Le pinceau	8
3.1.2	Le seau de remplissage	9
3.1.3	Palette de couleur	10
3.2	Les filtres	11
3.2.1	Filtre Nuances de Gris : "Grayscale"	12
3.2.2	Filtre Sépia	12
3.2.3	Filtre Gaussien	13
3.2.4	Filtre Rouge	14
3.2.5	Filtre Binaire et Otsu (noir et blanc)	15
3.2.6	Filtre Binaire (colorisé)	16
3.2.7	Filtre Négatif ou Inversion des couleurs	16
3.2.8	Filtre Gamma	17
3.3	L'interface	18
3.4	Site Web	22
3.5	Répartitions des tâches	22
3.6	Planification	23
3.7	Ressenti personnel	24

3.7.1	Christian Aziaka	24
3.7.2	Zaky Hamdoun	24
3.7.3	Adam Mahraoui	24
3.7.4	Robin Varliette	24
3.8	Conclsuion	25

1 Introduction

1.1 Le projet : Epipaintor

Notre projet est simple : un photoshop intelligent, permettant d'aider les personnes étudiant ou travaillant à EPITA !

Derrière le doux nom d'**Epipaintor** résidera un utilitaire, permettant de dessiner ou modifier une image.

L'idée est d'implémenter tout un éventail de fonctionnalités graphique, utilisant chacun leur algorithmes propres, sur le modèle du logiciel Photoshop.

1.2 Le groupe : Fast-thinking

Notre groupe se nomme **Fast-Thinking** :

Il tire son nom de la vivacité d'esprit de ses membres, et de sa volonté à développer des projets ingénieux. Fast-Thinking est un groupe formé de 4 étudiants en deuxième année de l'école d'ingénieur EPITA, aux origines diverses (Italien, Algérien, Marocain, Français). La diversité culturelle est une des grandes forces de **Fast-Thinking** lui permettant d'avoir une combinaisons de raisonnements différents les uns des autres.

1.3 Présentation des membres

1.3.1 Adam Mahraoui

Étudiant à EPITA, je suis passionné par le monde de la programmation et du numérique en général. La curiosité est un trait de personnalité qui me caractérise bien et c'est pour cela que cette idée projet m'a plu. J'ai vu de plus en plus apparaître de nouvelles fonctionnalités incroyables sur les logiciels de montage photo en me demandant toujours comment cela fonctionnait mais je n'ai jamais vraiment penché sur le sujet. Grâce à ce projet je vais pouvoir enfin répondre à mes questions en renforçant mes compétences en C sur un projet concret.

1.3.2 Christian Aziaka

Bonjour, je suis Christian Aziaka et je suis passionné d'IoT, programmation, montage vidéo, d'UX, montage photo, de design et de production musicale.

1.3.3 Zaky Hamdoun

Passionné par l'informatique et la géopolitique, j'ai rejoint l'EPITA avec plus tard l'envie de devenir ingénieur en informatique spécialisé en cybersécurité, ou en big data et intelligence artificielle. Actuellement Trader pour Bright Future Investment Management, je m'occupe du développement d'algorithme de trading automatique. Cependant, mon penchant pour la géopolitique et les métiers de la défense font également que je suis intéressé par l'Open Source Intelligence (OSINT) et les renseignements, qui se résument très souvent à du traitement d'image. J'aime également les mathématiques et la philosophie.

1.3.4 Robin Varliette

Bonjour, et enchanté, à quiconque lisant ces lignes !

Je suis Robin Varliette et je suis le leader du groupe Fast-Thinking.

J'ai 20 ans, et je suis un passionné depuis toujours d'algorithmique, d'information, de mathématiques, ainsi que d'Intelligence Artificielle.

Amateur de sport, j'affectionne le volley, le handball, ou encore le badminton.

Ouvert à tous les horizons, j'adore découvrir de nouveaux univers par le voyage, ou bien la musique.

Enfin, j'ai pour hobby de partager mes passions en proposant mon aide en algorithmique ou en programmation aux élèves d'EPITA qui en ont besoin.

2 Rappel sur le projet

2.1 Idée

L'idée du projet était à l'origine de faire une intelligence artificielle : Un Chat-bot capable de converser avec un être humain. Elle était née du fait que les membres du groupe sont particulièrement attirés par les intelligences artificielles, eux-mêmes ayant pu en développer une dans le cadre leur projet effectué au semestre passé.

L'idée du Chat-Bot comme Intelligence Artificielle prends ses sources dans deux projets existants :

- ChatGPT, le chat bot aux performances fulgurantes développée par Open-AI, qui a émergée très récemment en 2022.

- Tay.ai, le chat bot développé par Microsoft et utilisable sur Twitter, qui était une expérience de "Conversational Understanding" (en français : Apprentissage en conversation).

Cependant, pour des problèmes de complexité, et de barème, le projet n'a pas été accepté et à du être radicalement changé.

Le projet de faire un utilitaire sur la base de Photoshop nous a alors été proposé par notre professeur en charge des projets de 4ème semestre, et nous avons décidé de suivre ses recommandations afin de pouvoir proposer un projet qualitatif.

2.1.1 Nom

Pour commencer, il est important de rappeler qu'en tant qu'élèves d'EPITA, nous avons pour **devoir** d'honorer notre merveilleuse école. Qui serions nous sans elle, pas vrai ?

Il existe une règle implicite dans la nomenclature des choses à EPITA : Chaque nom doit avoir pour préfixe '**Epi**' que cela soit les associations, les sites internets, les organisations ou projets reliés à l'école ont presque tous comme préfixe '**Epi**'. Ainsi, nous suivons donc cette convention car nous portons les traditions de l'EPITA dans notre cœur.

Au départ, il était question d'appeler notre ChatBot '**EpiHelp**'.

Cependant, nous avons grandi avec le jeu '**Akinator**', un ChatBot inventé par l'ingénieur informatique français Arnaud Megret, capable de deviner un personnage en posant des questions à un utilisateur.

Cela étant dit, nous avons décidé de rendre honneur à cette prouesse technologique de l'époque en ajoutant '**nator**' à notre préfixe '**Epi**', formant ainsi le nom '**Epinator**'.

Le projet ayant changé en étant un utilitaire graphique, nous avons légèrement changé le nom du projet en remplaçant le **nator** en **paintor**, donnant **Epipaintor**.

3 Le projet

3.1 Les outils

3.1.1 Le pinceau

Quand on pense à un outil tel que paint, gimp ou photoshop, le premier outil qui nous vient à l'esprit est le pinceau. Il permet de manière intuitive d'écrire sur la zone de dessin en maintenant le clique gauche enfoncé.

Pour pouvoir obtenir tracé de crayon, on utilise dans un premier temps 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.

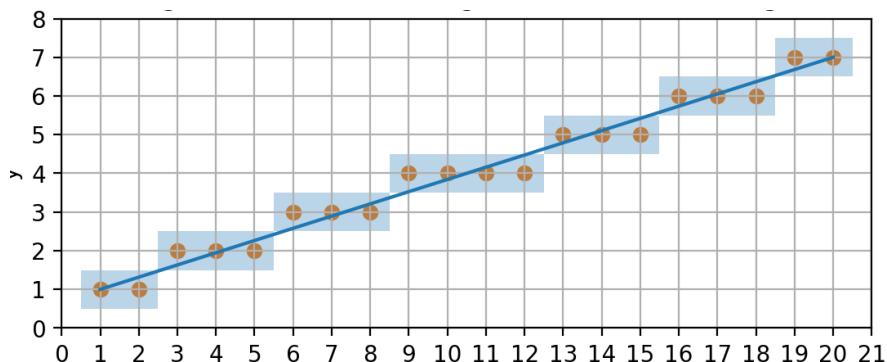


Fig 1 - Tracé d'un trait en utilisant l'algorithme de Bresenham

Pour finir, c'est avec une fonction qui enregistre les coordonnées d'un clic maintenu sur la page en continu que nous pouvons obtenir un résultat semblable à de vrais tracés fluides et précis.



Fig 2 - Résultat de l'utilisation du pinceau

3.1.2 Le seau de remplissage

L'outil Pot de Peinture, parfois également dénommé "seau", permet de modifier la couleur d'une zone en une autre en un seul clic. Ce dernier peut également permettre de remplir une zone à l'aide d'une texture. En l'occurrence, à l'état actuel, il ne permet que de remplir une zone avec une couleur. L'outil possède aussi un paramètre tolérance, permettant de spécifier dans quelles mesures les pixels doivent être sélectionnés. Pour implémenter le Pot de Peinture, on utilise l'algorithme "Flood Fill".

L'algorithme Flood Fill est un algorithme de remplissage utilisé en informatique graphique pour remplir une région close dans une image avec une couleur ou une texture. Il fonctionne en commençant par un point de départ (généralement un pixel) et en explorant les pixels adjacents jusqu'à ce que tous les pixels de la région soient remplis. L'algorithme examine chaque pixel adjacent à ceux qui ont déjà été remplis et décide s'il doit être rempli en fonction de certaines conditions. Cela peut être fait récursivement ou itérativement, selon l'implémentation choisie.

L'implémentation récursive de l'algorithme Flood Fill utilise la récursion pour remplir la région. À chaque étape, le pixel actuel est rempli et les pixels adjacents sont examinés pour voir s'ils doivent être remplis à leur tour. Cela se fait récursivement jusqu'à ce que tous les pixels de la région soient remplis. Cette implémentation est souvent plus simple à comprendre et à écrire, mais peut échouer si la profondeur de la récursion dépasse la limite maximale autorisée.

C'est la raison pour laquelle a été utilisée l'implémentation par file. Dans ce cas, le processus de remplissage fonctionne de manière similaire, mais les pixels sont ajoutés à la fin de la file et retirés du début de la file, ce qui crée un algorithme de largeur d'abord. À chaque étape, le pixel en début de file est retiré et rempli. Les pixels adjacents sont alors examinés pour voir s'ils doivent être ajoutés à la file. Si un pixel doit être rempli, il est ajouté à la fin de la file. Ce processus se poursuit jusqu'à ce que la file soit vide.

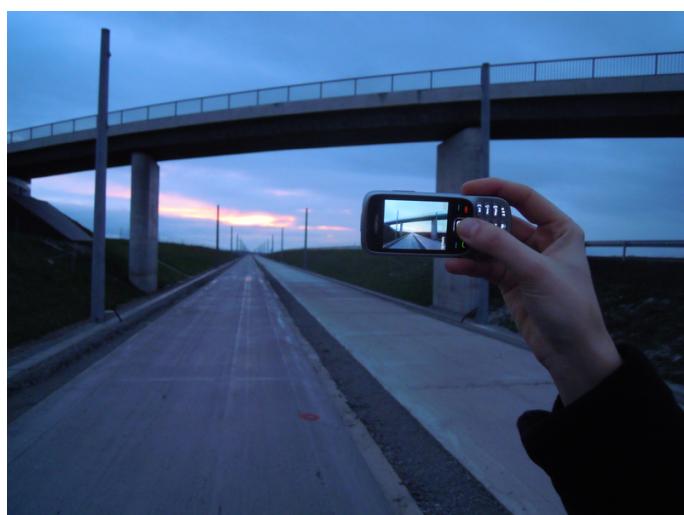


Fig 3 - Image de référence



Fig 4 - Image de référence après utilisation du seau

3.1.3 Palette de couleur

Les outils ne sont pas les seuls éléments indispensables à notre logiciel. En effet, il faut ajouter à cela une palette de couleur pour permettre à l'utilisateur de choisir les teintes qu'il veut utiliser sur la zone dessin.

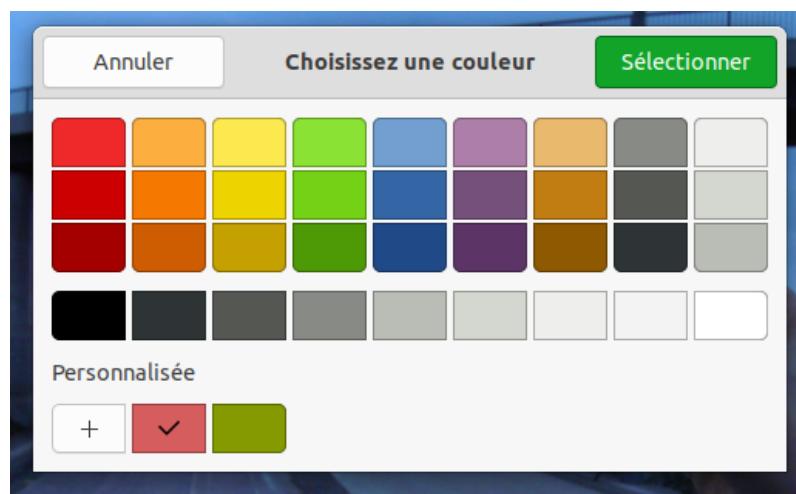


Fig 5 - Interface de la palette de couleur

En plus des couleurs prédéfinie, nous avons laissé le choix à l'utilisateur de pouvoir lui-même trouver la couleur qui lui conviendra parfaitement. Pour cela il peut utiliser un outil qui lui permet de choisir sur une teinte (faisant varier les valeurs R G B) puis de faire varier le niveau de noir et blanc.

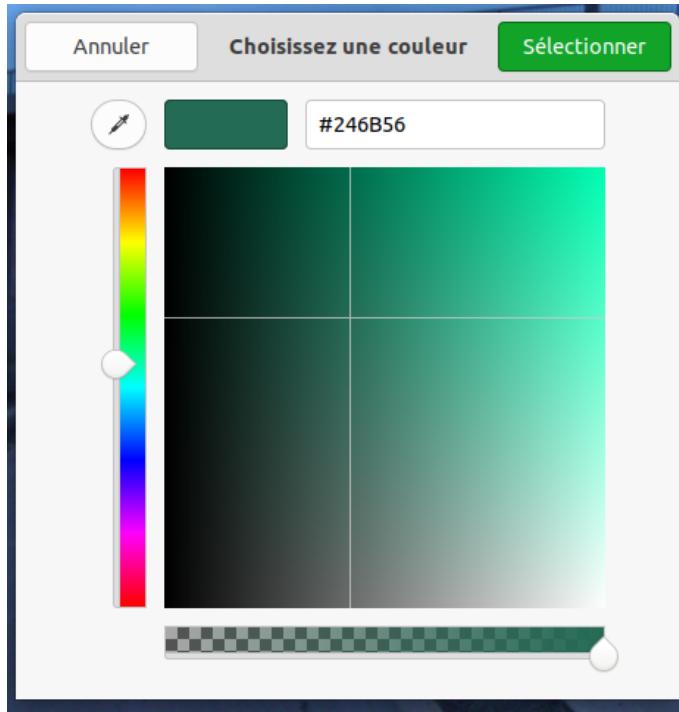


Fig 5.2 - Fenêtre de sélection d'une couleur personnalisée

Il peut aussi modifier l'indice de transparence de sa couleur mais aussi entrer directement la valeur de sa couleur en hexadécimal. En effet, chaque couleur peut-être symbolisé par un code de 6 caractères. Chaque doublet correspond respectivement au niveau de rouge, vert et bleu présent dans la couleur mais mis au format hexadécimal. Ce format permet alors de simplifié la représentation des couleurs de le rendre ainsi universel.

3.2 Les filtres

Évidemment, tout logiciel de traitement d'image ne serait pas complet s'il ne possédait pas une multitude de filtres permettant de modifier les images de manière prédéfinies. Un filtre de modification d'image est un algorithme utilisé en traitement d'image pour modifier les caractéristiques visuelles d'une image, telles que la couleur, la luminosité, le contraste, la netteté, etc. Les filtres de modification d'image sont largement utilisés en photographie, en design graphique et dans d'autres domaines pour améliorer la qualité d'une image ou en changer l'apparence.

Les filtres de modification d'image peuvent être implémentés de différentes manières, mais le principe général est de manipuler les valeurs des pixels d'une image en utilisant une formule mathématique spécifique. Par exemple, un filtre de luminosité ajustera la valeur des pixels d'une image pour augmenter ou diminuer la luminosité globale de l'image. Un filtre de flou généralement utilisera une moyenne pondérée des pixels autour d'un pixel donné pour lisser les bords et les transitions dans l'image.

3.2.1 Filtre Nuances de Gris : "Grayscale"

Pour convertir une image infographique couleur en niveau de gris il faut remplacer, pour chaque pixel les trois valeurs représentant les niveaux de rouge, de vert et de bleu, par une seule valeur représentant la luminosité. Pour cela, l'utilisation d'une formule mathématique calculant la moyenne des composantes rouges, vertes et bleues de chaque pixel est nécessaire. Celle-ci fut définie de la manière suivante :

$$gray = 0.4 * red + 0.35 * green + 0.25 * blue$$

Fig 6 : formule de calcul du niveau de gris.

Ces coefficients spécifiques sont utilisés afin de quand même rendre l'image visible vis-à-vis de l'utilisateur.



Fig 7 - Application du filtre nuance de gris

3.2.2 Filtre Sépia

Le filtre sépia est un filtre de modification d'image qui donne à une image une apparence d'antan en utilisant des couleurs brunes et dorées. Cette technique a été utilisée à l'origine pour simuler l'apparence des images en noir et blanc sur du papier photo jauni avec le temps.

Pour appliquer un filtre sépia à une image, les valeurs de couleur de chaque pixel sont modifiées pour donner une apparence d'image ancienne. Cela peut impliquer la modification des valeurs de rouge, de vert et de bleu pour produire des couleurs brunes et dorées. Par exemple, pour chaque pixel, la valeur de rouge peut être augmentée pour donner une apparence dorée, tandis que la valeur de bleu peut être diminuée pour donner une apparence brune.

Le filtre sépia est souvent utilisé pour donner une apparence vintage à des photos récentes. Il peut également être utilisé pour simuler l'apparence des photos anciennes

ou pour donner une apparence distincte à une image. Il est disponible dans de nombreux logiciels de traitement d'image, y compris les applications en ligne, les applications mobiles et les logiciels de bureau.

La fonction de conversion en pixel sépia est alors définie telle que :

$$\text{newRed} = \min(255, 0.393 * \text{red} + 0.769 * \text{green} + 0.189 * \text{blue})$$

$$\text{newGreen} = \min(255, 0.349 * \text{red} + 0.686 * \text{green} + 0.168 * \text{blue})$$

$$\text{newBlue} = \min(255, 0.272 * \text{red} + 0.534 * \text{green} + 0.131 * \text{blue})$$

Fig 8 : formule de calcul du niveau de gris.



Fig 9 - Application du filtre Sepia

3.2.3 Filtre Gaussien

Le filtre flou gaussien est un filtre de modification d'image qui est utilisé pour adoucir les détails d'une image et lisser les transitions entre les couleurs et les textures. Il est particulièrement utile pour supprimer les bruits indésirables dans une image, tels que le bruit numérique ou le grain filmique.

Le filtre flou gaussien est basé sur la théorie statistique et utilise une fonction de densité gaussienne pour déterminer la pondération des pixels voisins d'un pixel donné. En d'autres termes, pour chaque pixel, la valeur moyenne des pixels environnants est calculée en utilisant des coefficients pondérés pour déterminer la valeur finale du pixel. Les coefficients pondérés sont déterminés en utilisant une fonction gaussienne pour donner plus de poids aux pixels les plus proches du pixel en question et moins de poids aux pixels plus éloignés.

Le filtre flou gaussien est un filtre de flou linéaire qui peut être utilisé pour flouter une image entière ou des parties sélectionnées de l'image. Il peut également être utilisé en combinaison avec d'autres filtres pour produire des effets plus complexes.

Il est disponible dans de nombreux logiciels de traitement d'image, y compris les applications en ligne, les applications mobiles et les logiciels de bureau.

$$(R, G, B) = \left(\frac{\sum_{n=0}^n r}{n}, \frac{\sum_{n=0}^n g}{n}, \frac{\sum_{n=0}^n b}{n} \right)$$

Fig 10 : formule de calcul du flou gaussien.

Avec n correspondant au nombre de pixel choisi dans les paramètres de la fonction, le rayon du carré autour du pixel concerné.

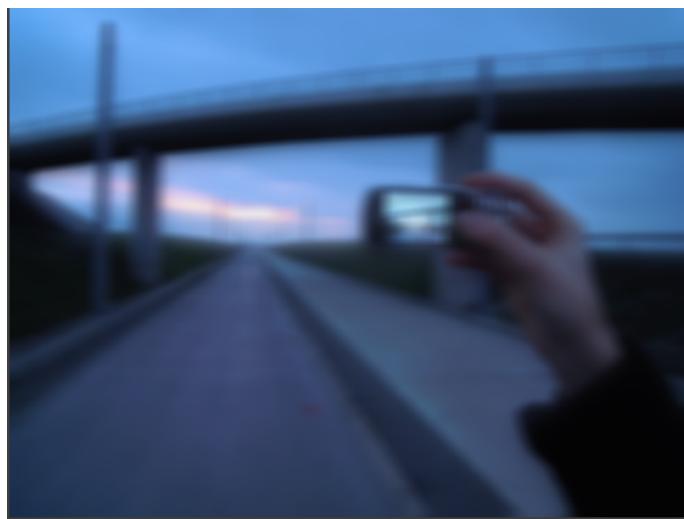


Fig 11 - Application du flou Gaussien

3.2.4 Filtre Rouge

Le filtre de lumière rouge est un filtre de modification d'image qui accentue la composante rouge de chaque pixel dans une image. Cela signifie que les parties de l'image qui étaient déjà rouges deviennent plus rouges, tandis que les autres parties sont modifiées pour paraître plus rouges.

L'application du filtre de lumière rouge est réalisée en augmentant la valeur de la composante rouge de chaque pixel, tout en laissant les autres composantes inchangées ou en les modifiant moins. Cela peut être réalisé en multipliant la composante rouge par un facteur d'amplification, tout en maintenant les autres composantes inchangées ou en les multipliant par un facteur plus faible.

Le filtre de lumière rouge peut être utilisé pour une variété de raisons, telles que la correction de couleurs pour les images en sous-exposition ou la visualisation de structures en relief dans les images médicales. Il peut également être utilisé pour des raisons esthétiques, pour ajouter une touche de couleur rouge à une image.



Fig 12 - Application du filtre rouge

3.2.5 Filtre Binaire et Otsu (noir et blanc)

Le thresholding d'image est une technique de traitement d'image qui consiste à convertir une image en noir et blanc en se basant sur une valeur de seuil. Cette valeur de seuil détermine les pixels de l'image qui seront considérés comme blancs et ceux qui seront considérés comme noirs.

Lors du thresholding, la luminosité de chaque pixel est comparée à la valeur de seuil. Si la luminosité d'un pixel est supérieure à la valeur de seuil, ce pixel est considéré comme blanc. Sinon, il est considéré comme noir. En fin de compte, cela permet de segmenter l'image en deux parties distinctes : les zones blanches et les zones noires.

Le thresholding d'image est utilisé pour un certain nombre d'applications, telles que la reconnaissance de formes, la numérisation de documents, la segmentation de l'image et la détection de bords. Il est également souvent utilisé en combinaison avec d'autres techniques de traitement d'image pour résoudre des problèmes plus complexes.

Il existe plusieurs méthodes pour déterminer la valeur de seuil, telles que le thresholding global, le thresholding local et le thresholding adaptatif. Chacun de ces méthodes a ses propres avantages et inconvénients en fonction de la qualité de l'image d'entrée et de la complexité du problème à résoudre. Le choix de la méthode dépend donc des besoins spécifiques de chaque application.

En l'occurrence, ont été implémentées une version manuelle, mais également une version automatique du thresholding basé sur les formules d'Otsu.

La valeur de seuil n'est pas fixe. Elle dépend de l'image traitée. Le calcul consiste à prendre les valeurs des pixels voisins à notre pixel traités afin de calculer la valeur de t tel que la variance reste identique le plus souvent, puis nous renvoyons cette valeur comme étant le seuil. Deux méthodes nous permettent de calculer cela :

$$\sigma^2(t) = \omega_{bg}(t)\sigma_{bg}^2(t) + \omega_{fg}(t)\sigma_{fg}^2(t)$$

Fig 13 : formule probabiliste de la variance.

$$\sigma^2(t) = \frac{\sum(x_i - \bar{x})^2}{N - 1}$$

Fig 14 : formule par somme de la variance.

Il est ensuite possible de calculer la variance pour différentes valeurs de t. La variance sera alors seuillée étant donnée que pour un certain $t > \text{threshold}$, celle-ci ne changera plus. Le t deviendra alors le seuil à utiliser dans le cadre de la binarisation. Cependant, il est important de noter que la binarisation d'Otsu ne fonctionne pas avec toutes les images.

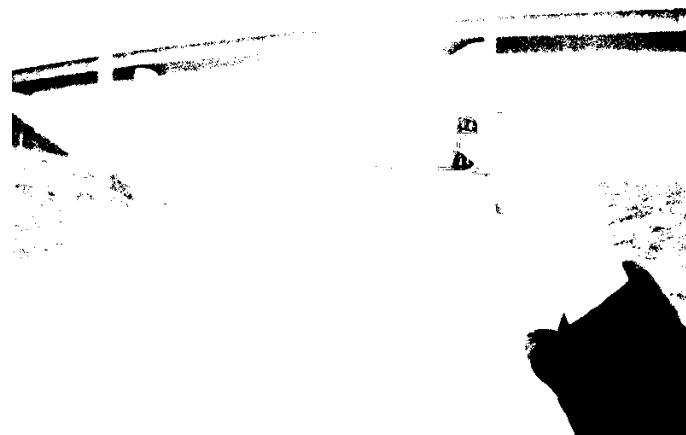


Fig 15 - Application du filtre binaire(noir ou blanc)

3.2.6 Filtre Binaire (colorisé)

Fonctionnant exactement de la même manière que pour le filtre Binaire classique, ce dernier permet de spécifier deux autres couleurs, au lieu du noir et du blanc.

3.2.7 Filtre Négatif ou Inversion des couleurs

Le filtre d'inversion des couleurs est un filtre de modification d'image qui inverse les couleurs d'une image. Cela signifie que les couleurs claires deviennent sombres, et vice versa.

L'inversion des couleurs est réalisée en inversant les valeurs des composantes de couleur de chaque pixel. Par exemple, si un pixel était originellement de couleur rouge, il deviendrait vert après l'inversion des couleurs. Les valeurs de chaque composante de couleur peuvent être inversées en les soustrayant de leur valeur maximale (par exemple, 255 pour les couleurs codées sur 8 bits).

Le filtre d'inversion des couleurs est souvent utilisé pour des raisons esthétiques ou pour des applications spécifiques, telles que la correction de couleurs inversées dans des images ou la visualisation de structures en relief dans des images médicales.

$$\text{newRed} = 255 - \text{red}$$

$$\text{newGreen} = 255 - \text{green}$$

$$\text{newBlue} = 255 - \text{blue}$$

Fig 16 : formule de calcul des valeurs R, G, B inversées.



Fig 17 - Application du filtre négatif

3.2.8 Filtre Gamma

Le filtre d'augmentation du gamma est un filtre de modification d'image qui permet de contrôler la luminosité d'une image. Il est souvent utilisé pour corriger l'exposition d'une image, pour la rendre plus sombre ou plus claire selon les besoins.

Le gamma est une valeur qui détermine comment les valeurs de luminosité d'une image sont interprétées et affichées. Une augmentation du gamma signifie qu'il y aura une amplification des valeurs de luminosité plus faibles et une atténuation des valeurs de luminosité plus élevées. Inversement, une diminution du gamma aura pour effet d'atténuer les valeurs de luminosité plus faibles et d'amplifier les valeurs de luminosité plus élevées.

L'augmentation du gamma peut être réglée en utilisant un curseur ou en entrée une valeur numérique spécifique. Il est important de noter que des valeurs de gamma extrêmement élevées ou extrêmement faibles peuvent rendre l'image peu naturelle ou même illisible. Par conséquent, il est souvent nécessaire de trouver un équilibre entre l'amélioration de la luminosité de l'image et la préservation de son aspect naturel.

L'augmentation du gamma se fait par l'intermédiaire d'une fonction $g(x)$ définie en l'occurrence par :

$$g(x) = 255 * \left(\frac{x}{255}\right)^n$$

Fig 18 : formule de l'augmentation du gamma.

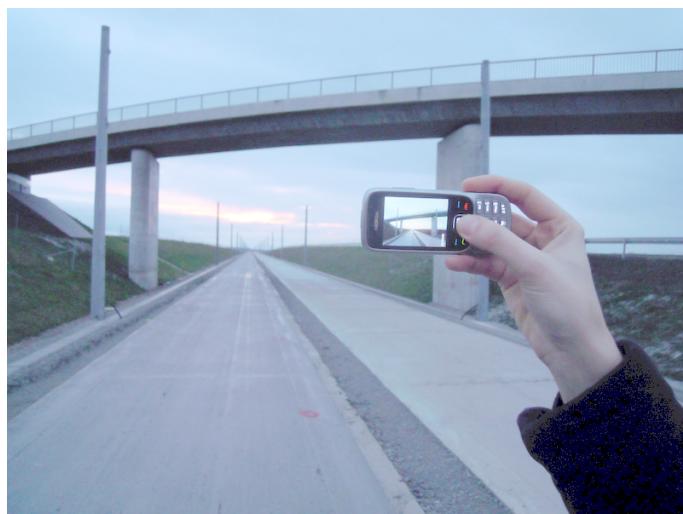


Fig 8 - Application du filtre gamma

3.3 L'interface

Pour que notre projet soit réellement utilisable et qu'il paraisse moins austère pour les utilisateurs lambda nous avons choisi de réaliser une interface avec la librairie GTK3 en l'associant à l'outil Glade.

La librairie GTK est composée d'un grand nombre de widget qui sont utilisables pour pouvoir modéliser notre interface. Pour pouvoir rendre l'utilisation de GTK plus facile et surtout beaucoup plus visuel le logiciel Glade nous permet de désigner en temps réel le visuel de notre interface.

En effet, Glade permet de générer des fichiers d'interface en .glade qui sont utilisables par GTK. Dans notre code principal il faut penser à bien définir chaque

widget avec un nom, qui devra alors être indiqué sur GTK pour ensuite pouvoir les utiliser pour lancer des fonctions par exemple.

Pour pouvoir lancer des fonctions, nous utilisons certains signaux (cliques simples par exemple ou alors clique maintenue) qui peuvent être détectés et utiliser pour des actions précises.

Ces deux outils utilisés ensemble permettant de réaliser des interfaces épurées et simples idéales pour ce genre de projet où le côté pratique est plus important que le visuel.

Notre logiciel possède une interface intuitive et donc simple à comprendre pour n'importe quel utilisateur. Elle se compose :

D'une barre latérale gauche qui contient tous les outils disponibles ainsi que le choix de la couleur.

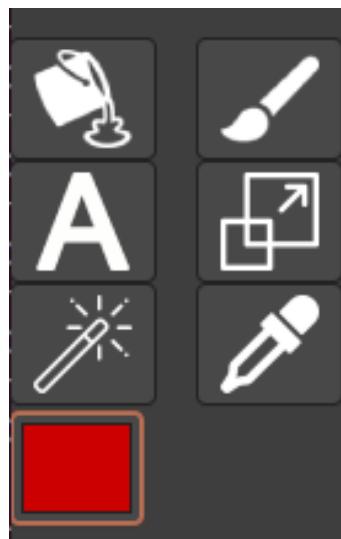


Fig 19 - Barre latérale gauche (outils)

Sur le coin supérieur gauche on retrouve 3 sous-menus. Ces derniers permettent d'afficher plus d'options lorsque l'on passe la souris dessus ce qui permet de gagner l'espace sur la zone de dessin.

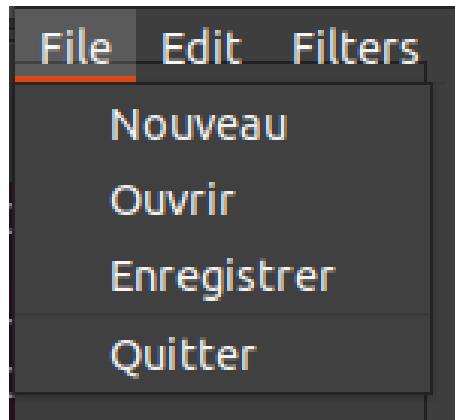


Fig 20 - Menu "File"

File : ce menu permet d'ouvrir un image présente dans les fichiers de l'utilisateur, enregistrer dans le fichier "save" la production faite et quitter l'application. Il permet aussi grâce au bouton "Nouveau" d'ouvrir une nouvelle zone de dessin

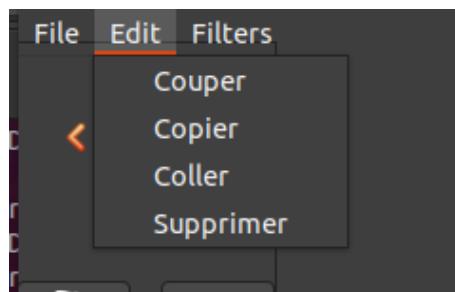


Fig 21 - Menu "Edit"

Edit : il permettra de couper, copier, coller ou supprimer la zone sélectionnée par l'outil de selection ou baguette magique.

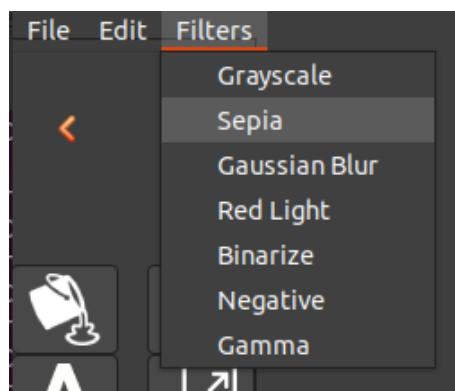


Fig 22 - Menu "Filters"

Filters : Il permet d'appliquer à toute la zone de dessin les différents filtres proposés.

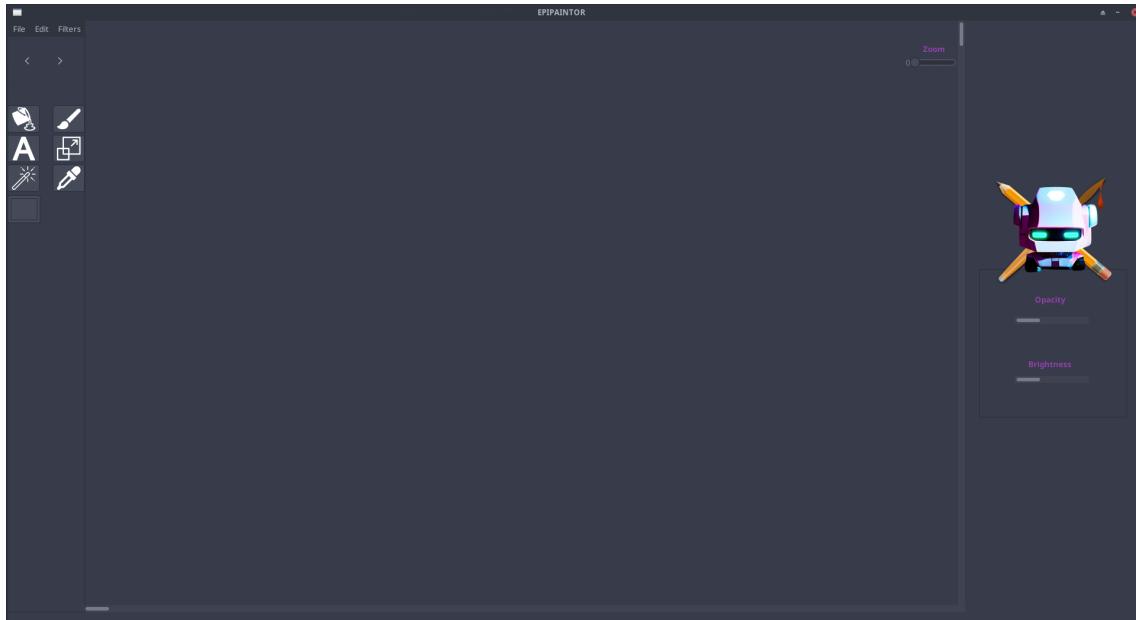


Fig 23 - Interface générale

3.4 Site Web

3.5 Répartitions des tâches

Tâches	Adam	Zaky	Christian	Robin
SDL				
Gestion des images			R	S
Gestion des couleurs		S	R	
Gestion de la structure des images	S	R		
Filtres			S	R
GTK - Interface				
Menu utilisateur	S			R
Développement des outils		S		R
Ouverture / Sauvegarde de fichiers			R	S
Zone de dessin	R	S		
Pop-ups			R	S
Structure				
Design de l'application		S	R	
Architecture des fichiers sources	S			R
Raccord des parties (Compilation, makefile)	R	S		
Présentation				
Site internet			R	
Documents latex	S	R		

TABLE 1 – Répartition des Tâches - S :Suppléant - R : Responsable

3.6 Planification

Tâches	Soutenance 1	Soutenance 2	Soutenance 3
GTK - Interface			
Menu utilisateur	100%	100%	100%
Développement des outils	50%	75%	100%
Ouverture / Sauvegarde de fichiers	100%	100%	100%
Zone de dessin	50%	80%	100%
Pop-ups	45%	100%	100%
SDL			
Gestion des images	50%	85%	100%
Gestion des couleurs	70%	95%	100%
Gestion de la structure des images	70%	100%	100%
Filtres	50%	750%	100%
Interface graphique			
Design de l'application	40%	70%	100%
Architecture des fichiers sources	70%	100%	100%
Raccord des parties	30%	60%	100%

TABLE 2 – Planification de l'avancement des tâches

3.7 Ressenti personnel

3.7.1 Christian Aziaka

Cette première étape de notre projet a été intense. Bien qu'originellement mal-partis, nous avons réussi à rattraper notre retard en un week-end. Ayant déjà des connaissances en matière d'interface, j'ai pu retrouver mes repères pendant le développement de ce projet. On a pu rencontrer des embûches lors de notre parcours, des bugs frustrants qui nous ont fait jusqu'à réinitialiser notre système d'exploitation quelques jours avant le rendu de soutenance, mais à force de volonté et de détermination, nous avons surmonté ces épreuves afin de délivrer un exécutable dont nous sommes fiers.

3.7.2 Zaky Hamdoun

Ce projet est intéressant, et il nous permet de créer une version de Photoshop qui s'adapte à nos besoins, ni trop compliquée, ni trop simpliste. Toutefois, je trouve que l'aspect algorithmique de notre projet est assez pauvre, et peut être amélioré.

3.7.3 Adam Mahraoui

Ce début de projet a été très intéressant, c'est un univers que je n'avais jamais encore manipulé donc il y a eu une partie importante de documentation. Pour la prochaine soutenance j'espère pouvoir avancer plus sur le projet et ainsi rendre un exécutable plus complet avec plus de fonctionnalités

3.7.4 Robin Varliette

Le travail que j'ai pu effectuer pour la première soutenance a été formateur d'un point de vu de l'interface. J'ai pu me rendre compte que des parties du projet que j'avais estimée simple se sont avérées complexes, comme par exemple relier des algorithmes de modifications d'images, qui marchaient récursivement pour de petites images mais qui faisaient des stack overflow pour de plus grandes tailles. La première soutenance a permis à l'équipe de se familiariser avec git et github, ce qui nous permettra d'être d'autant plus productif quant aux prochaines soutenances.

3.8 Conclusion

Lors de cette première partie du projet, une cohésion et forme d'entraide entre chaque membre a été vite mise en place. Une bonne entente règne dans ce groupe et nos objectifs ont été réalisés. Tout au long de ce premier rapport de soutenance, nous vous avons déroulé les avancées de notre projet, point par point, de manière à expliciter chaque partie et faire un point objectif sur ce qu'il nous reste à faire, ce que nous avons déjà réussi à faire et les premiers rendus de ce défi que nous nous sommes posé. Il nous reste du chemin à parcourir, mais nous avons réussi à avancer comme nous le souhaitions sans nous mettre en retard, et nous sommes sur la bonne voie pour réaliser le meilleur projet possible avec les échéances que nous avons à respecter et celles que nous nous sommes imposées. La communication au sein du groupe est fluide, claire et efficace, de manière à ce que chacun puisse avancer au mieux sur son travail tout en ayant les dernières informations sur l'avancée des autres.