télécom saint-étienne
école d'ingénieurs nouvelles technologies

# Color your mood

Étudiantes en Design :

- MAHDI Inès
- SAHRAOUI Nassima
- TAMARZIST Syrine

Étudiants Télécom :

- LACLAVERIE Pierre
- BIRON Grégoire
- ALEB Adam
- DEGRANGE Alexis
- RAYNAL Julien

# Table des matières

# I/ Introduction

Ce projet a été initié pour nous faire prendre en main l'organisation d'un projet via la méthodologie Agile.

Durant tout ce projet, notre Product Owner était Mme Bertrand et notre Scrum master était Mme Laclau.

Nous avions aussi M Chevalier en tant qu'expert Java et M Ducottet en tant qu'expert Image.

Nous avons travaillé de pair avec 3 masters designs sur ce projet.

Ce projet à pour but de créer une image unique à chaque utilisateur. L'image est construite à partir des mouvements que celui-ci effectue devant la caméra.

# II/ Périmètre du projet

## Cahier des charges

L'objectif de notre projet est de transformer des mouvements effectués par l'utilisateur en une image propre et unique à chaque passage.

Pour se faire, nous avons découpé notre projet en 3 parties :

- La captation des mouvements
- Le traitement de données
- La retransmission de l'image

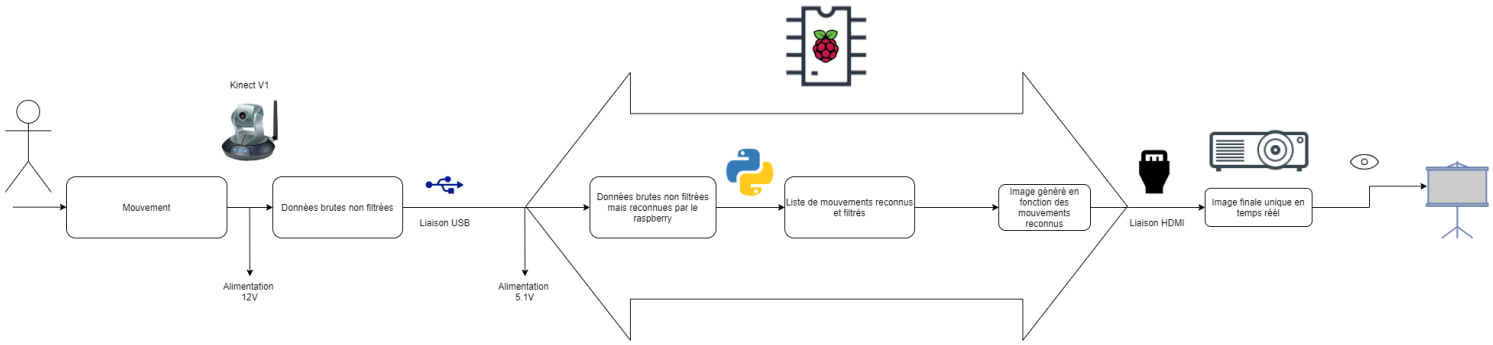



Schéma synthétisant notre cahier des charges

Pour ce qui concerne la partie de captation des mouvements, nous avons choisi de commander un Kinect qui nous sert de caméra.

La Kinect est donc connectée à la NUC Intel, et se charge d'effectuer les traitements et les calculs. L'ensemble de notre code est donc exécuté sur la NUC Intel sous une distribution Linux car ce sont des distributions gratuites. Puisque Python est déjà installer sur les distributions Linux, nous avons choisi d'utiliser ce langage de programmation. De plus, Python est un langage assez simple d'utilisation pour le traitement d'image.

Le traitement sur Python se décompose en 2 parties. La première partie concerne le traitement des images d'entrée envoyées par la Kinect. Dans cette partie, il s'agit de détecter les mains et d'envoyer des informations à la deuxième partie du programme qui concerne l'ajout des calques.

Enfin, l'image est retransmise via un câble HDMI et projeté avec un vidéoprojecteur en temps réel.

Nous avons aussi ajouté une partie permettant de récupérer l'image générée. Lorsque le programme se termine un code unique est généré et transmis à l'utilisateur. Par la suite, l'utilisateur peut se connecter à un site web, entrer son unique code et télécharger son image. Nous avons choisi de passer par cette méthode pour respecter les gestes barrières.

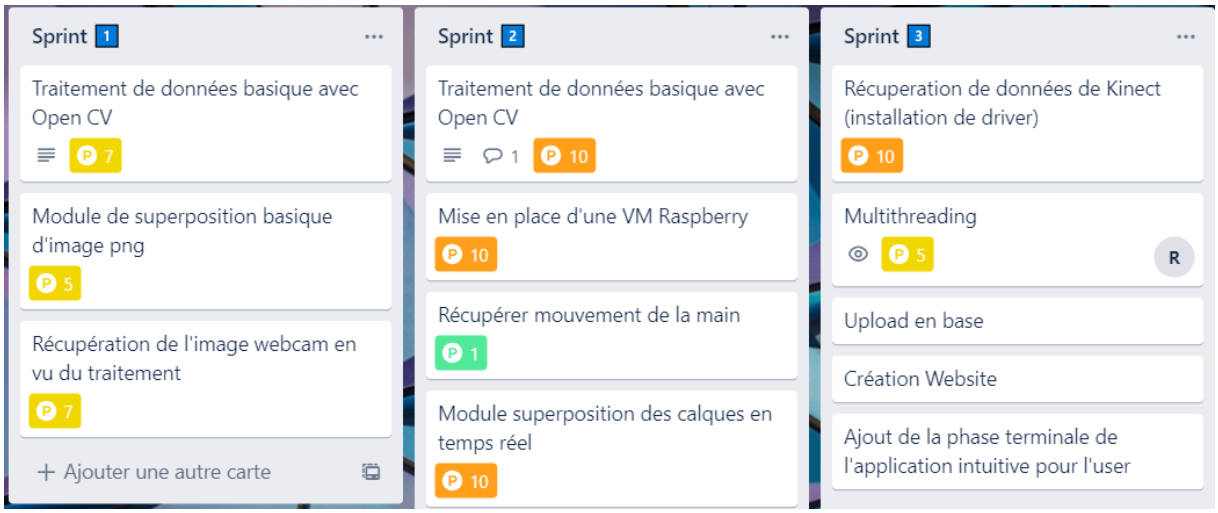
## Planning

Chaque sprint durait deux semaines. Nous avons organisé nos trois sprints de la manière suivante :

- Sprint 1 : Nous avons géré les différentes parties séparément.
  Alexis et Pierre se sont chargés de récupérer les mouvements des mains des utilisateurs grâce à la bibliothèque OpenCV. Grégoire et Julien se sont chargés d'afficher les calques dans une nouvelle fenêtre et Adam s'occupait de la Kinect, pour ce qui concerne l'installation des drivers …
  Nous avons terminé ce sprint dans les temps.

- Sprint 2 : Ce sprint concerne la fusion des différentes parties.
  On à tous travaillé sur le lien des deux parties précédentes. À la fin de ce sprint, nous avions donc un logiciel non optimisé mais fonctionnel. Nous l'avons terminé en avance, et nous avons pu commencer le sprint 3 avec un peu d'avance.

- Sprint 3 : Ce sprint est lié à l'optimisation de notre logiciel.
  En effet, à la fin du sprint 2, notre logiciel nécessitait beaucoup de ressources et était peu optimisé (inclusion de boucle infini dans une boucle infinie). Nous avons fait le choix de passer sur du multithreading ce qui a grandement amélioré les performances de celui-ci. De plus, nous avons créé le site web pour récupérer les images générées dans ce sprint.




Nous avons choisi d'organiser les sprints de la manière décrite plus haut pour pouvoir rendre un livrable réaliste et réalisable. Ainsi, nous avions beaucoup de temps pour faire l'essentiel et il nous restait du temps à la fin pour pouvoir améliorer le logiciel. Nous avons réalisé l'essentiel en premier pour dégrossir le sujet et nous avons affiné notre travail au fur et à mesure des sprints.

# Organisation

## Répartition du travail

Dès le début, nous avons travaillé de pair avec les étudiantes en Master Design. Ainsi, nous nous étions mis en accord avec celles-ci sur le livrable final du projet. Nous avons pu délimiter le périmètre du projet.

Nous avons dès le début établi une liste de matériel nécessaire pour le bon déroulement du projet. Les éléments principaux étant un NUC Intel, un vidéoprojecteur, la Kinect et la connectique.

Nous nous sommes répartis le travail, pour pouvoir avancer par groupe. En effet, les master Design travaillaient sur les calques pendant que nous élaborions le logiciel sous Python. Au sein du projet Python, Alexis et Pierre ont travaillé sur la partie de la captation de la main, Adam s'est occupé de mettre en place la Kinect et a généré le site Web, et Grégoire et Julien ont travaillé sur l'affichage des calques.

Pour lier les différentes parties, nous nous sommes répartis les tâches de sorte que tous le monde travaille. Par la suite, nous avons souhaité améliorer le logiciel, nous sommes donc passé à du multithreading.

## Trello

Pour pouvoir se répartir efficacement les tâches, nous sommes passé par un Trello. Nous avons fait des réunions chaque semaine pour connaître l'avancement de chacun et pouvoir se répartir les tâches efficacement. Chaque tâche était attribuée à un membre du groupe et permettait d'avoir une vue globale sur l'avancement du projet.

De plus, lorsque quelqu'un avait une idée en tête ou découvrait une erreur de fonctionnalité, un ticket Trello en informait l'utilisateur ou alors un nouveau ticket était créé.

Grâce à ce Trello, nous avons facilement pu organiser notre méthodologie Agile et rester aisément dans les temps sans accumuler de retard.

https://trello.com/invite/b/g7OeHmEa/e3402c2f09e6b1835a5e154a88d10633/scrum

## Gitlab

En parallèle du Trello, nous avons mis notre code en commun pour pouvoir aisément obtenir les modifications effectuées par d'autres membres du groupe. Pendant longtemps, nous avons eu deux branches principales, la branche « master », qui était la version sûre et stabilisée de notre code et la branche « dev » qui était une version plus améliorée que master mais non définitive et pas forcément stable. Nous essayions de mettre à jour master assez fréquemment.

Lorsqu'un membre du groupe voulait effectuer des modifications, il créait une branche à sont nom en se basant sur dev, et il faisait ses modifications en local avant de les pousser sur sa branche en ligne. Par la suite, il fusionnait sa branche avec la branche dev.

Pour sécuriser les branches, nous avons choisi de mettre un seul membre du groupe en tant que « maintainer » et tous les autres membres en tant que développeurs. Ainsi, seul le maintainer pouvait effectuer les fusions de branche notamment de dev sur master.

https://code.telecomste.fr/prinfo/design-4

### Réunions ScrumMaster, ProductOwner et expert

À chaque fin de sprint, nous prenions rendez-vous avec notre ScrumMaster pour pouvoir s'orienter facilement dans la réalisation du projet et pour être sûr que nous n'étions pas hors-sujet.

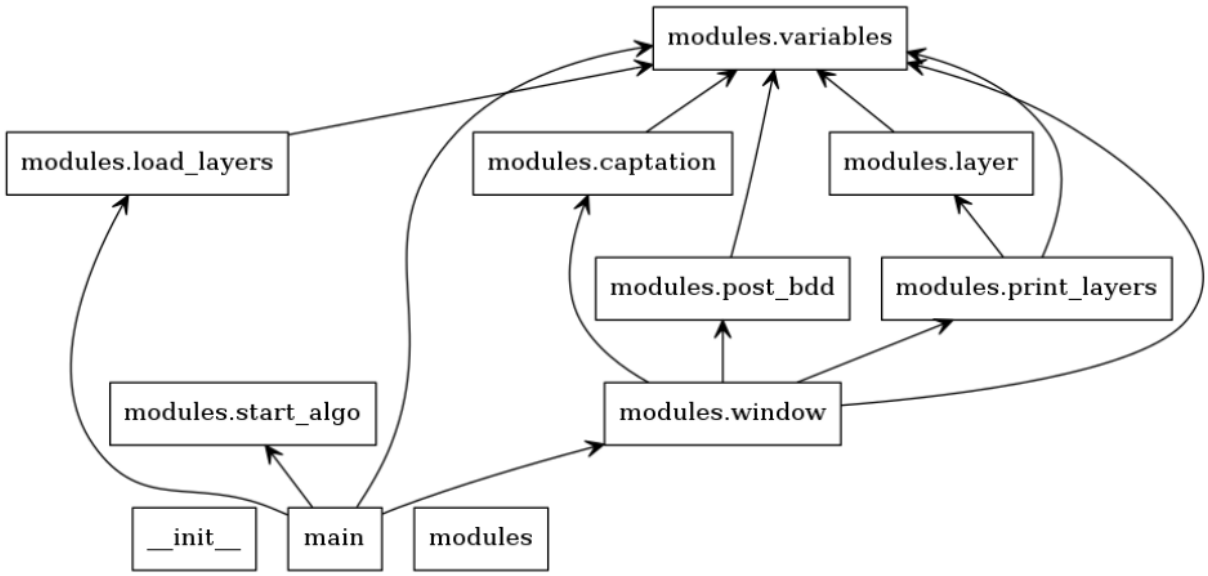
Nous avions réunions toutes les 2 semaines avec notre ProductOwner. Durant ces réunions, nous présentions l'avancement du projet et nous écoutions les retours émis par le ProductOwner. Si des corrections étaient à apporter au projet, alors celles-ci étaient mises en priorités avant de commencer le nouveau sprint.

Enfin, lorsque nous avions un problème technique, nous prenions réunion individuellement ou en groupe avec notre expert.

## III/ Organisation des fichiers

# IV/ Diagramme UML – Packages

## Diagramme de Packages

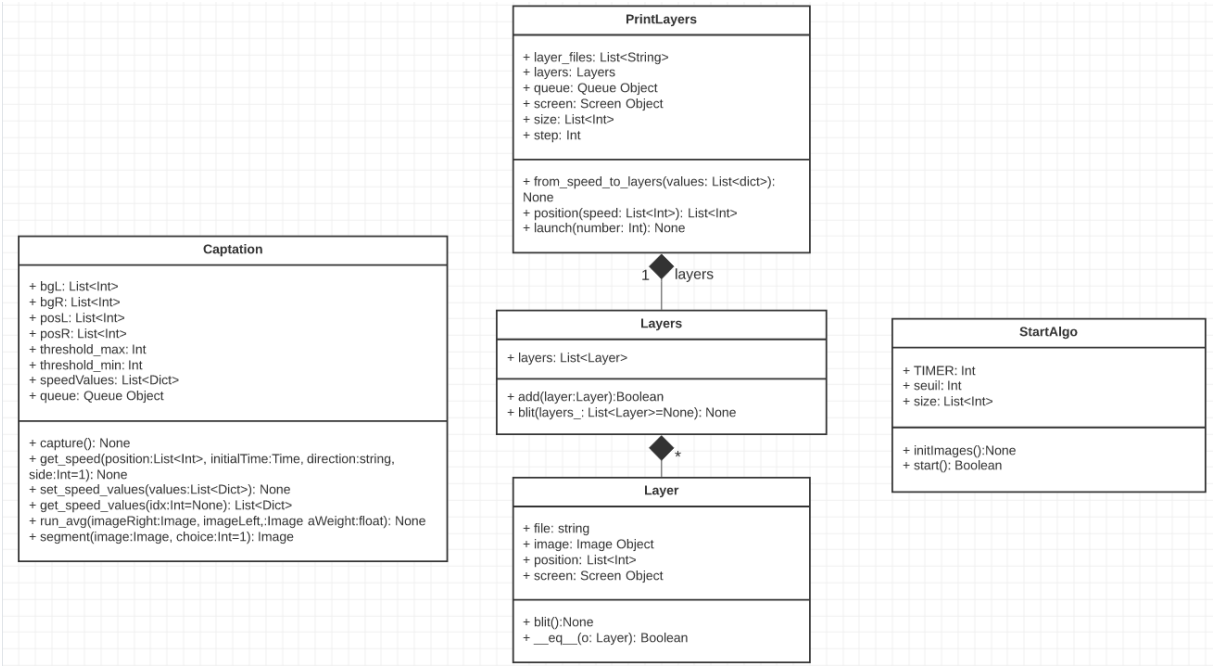



Le point d'entrée du programme est le fichier script « main.py ». Par la suite, le script « start_algo.py » est appelé pour initialiser les fenêtres explicatives. Les fichiers sont chargés par l'appel du module « modules.load_layers ». Enfin, le fichier « modules.windows » lance le corps principal du programme. C'est-à-dire que les différents threads sont lancés dans ce script. Les modules « modules.print_layers » et « modules.captation » sont lancés de manière concurrentielle et communiquent entre eux jusqu'à la fin du process lancé. À la fin des deux sous processus, le module « module.post_bdd » est appelé pour envoyer les images en base de données.

Le module « module.variables » contient l'ensemble des variables partagées nécessaires au bon fonctionnement du programme.

## Diagramme UML

Une partie de notre programme est organisé sous forme de classe même si cela n'est pas très optimisé en Python. Cette implémentation nous permettait de stocker des valeurs aisément dans une classe et d'y accéder facilement grâce à un attribut.




La classe « StartAlgo » est une classe qui permet de lancer les fenêtre utilisateurs du début de programme.

La classe « Layer », correspond à un objet de calques et permet de paramétrer et d'afficher un calque sur l'écran.

La classe « Layers » s'apparente à une collection de Layer. On peut aisément via cette classe imprimer tous les calques (et présent dans la classe) à l'écran. De plus, on peut facilement ajouter un calque à cette classe.

La classe « PrintLayers » possède l'attribut layers de type « Layers » qui peut être vu comme une collection de « Layer ». Ainsi, à chaque ajout de calque, la classe Layers est appelé et on lui ajoute le calque correspondant.

La classe « Captation » est la classe qui s'occupe de récupérer les mouvements utilisateurs. Grâce à la bibliothèque OpenCV, les mouvements sont récupérés et transmis sous forme de dictionnaire à la classe « PrintLayers » par l'intermédiaire d'une Queue.

En effet, les classes « Captation » et « PrintLayers » sont lancées de manière concurrentielle. On fait du multithreading, sur les deux classes. Les informations collectées dans la classe « Captation » sont mises en bout de Queue, et elles sont lus par la classe « PrintLayers ».