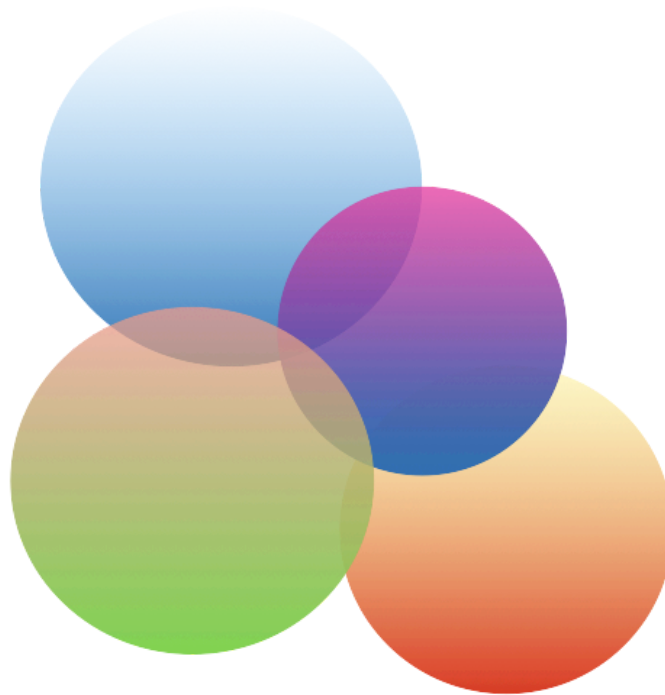


# Rapport de Projet



*Chromat'Ynk*

Projet réalisé en 2024-2025 par:

Gaétan RETEL - Matthias RIBEIRO - Adam SWICZKA - Farah MAHMOUD -  
Clément PRAUD



# Sommaire

<b>I. Introduction.....</b>	<b>3</b>
<b>II. Organisation de l'équipe.....</b>	<b>3</b>
Versionnement.....	3
Intégration Continue (CI).....	4
Répartition des tâches.....	5
<b>III. Architecture du projet.....</b>	<b>6</b>
Choix des technologies.....	6
Méthode de conception.....	6
Modélisation métier.....	6
Absence de méthode pas à pas.....	7
Impossibilité de supprimer des variables.....	8
Bonus.....	8
<b>IV. Présentation de l'application.....</b>	<b>9</b>
Menu.....	9
Fenêtre de dessin.....	9
Nos dessins.....	10

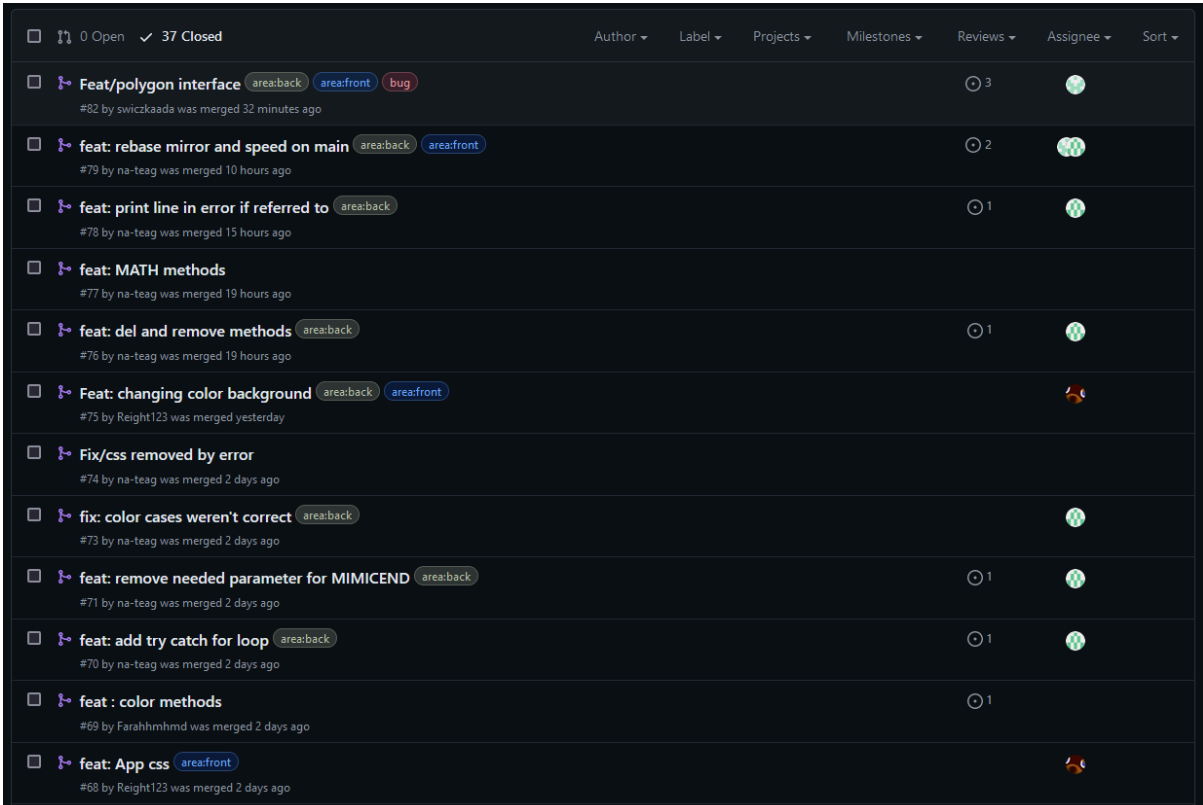
# Introduction

Pour le cours de Java de première année de cycle ingénieur de l'année 2023-2024, il nous a été proposé de réaliser un projet avec différentes contraintes listées dans un document PDF. Le choix du projet nous a été laissé. Nous avons choisi de partir sur Chromat'Ynk, un projet consistant en un programme qui va permettre de dessiner des formes quelconques sur un écran à partir d'un pseudo code. Dans ce projet, il a fallu donc créer un langage d'instruction, ainsi qu'un interpréteur de ce langage et l'affichage de ces instructions.

## Organisation de l'équipe

### Versionnement

Nous avons utilisé l'outil Git [hébergé sur Github](#) pour gérer les changements appliqués au projet et pour faciliter la collaboration entre les cinq membres du groupe. Chaque nouvelle fonctionnalité ou correction était développée dans une branche séparée, nommée selon les conventions "feat/<nom>" ou "fix/<nom>", fusionnée avec la branche principale "main" une fois la modification effectuée et passée en revue via une pull request.



<input type="checkbox"/>	0 Open	✓ 37 Closed	Author	Label	Projects	Milestones	Reviews	Assignee	Sort
<input type="checkbox"/>				Feat/polygon interface	area:back	area:front	bug	3	
#82 by swiczkaada was merged 32 minutes ago									
<input type="checkbox"/>				feat: rebase mirror and speed on main	area:back	area:front		2	
#79 by na-teag was merged 10 hours ago									
<input type="checkbox"/>				feat: print line in error if referred to	area:back			1	
#78 by na-teag was merged 15 hours ago									
<input type="checkbox"/>				feat: MATH methods					
#77 by na-teag was merged 19 hours ago									
<input type="checkbox"/>				feat: del and remove methods	area:back			1	
#76 by na-teag was merged 19 hours ago									
<input type="checkbox"/>				Feat: changing color background	area:back	area:front			
#75 by Reight123 was merged yesterday									
<input type="checkbox"/>				Fix/css removed by error					
#74 by na-teag was merged 2 days ago									
<input type="checkbox"/>				fix: color cases weren't correct	area:back				
#73 by na-teag was merged 2 days ago									
<input type="checkbox"/>				feat: remove needed parameter for MIMICEND	area:back			1	
#71 by na-teag was merged 2 days ago									
<input type="checkbox"/>				feat: add try catch for loop	area:back			1	
#70 by na-teag was merged 2 days ago									
<input type="checkbox"/>				feat : color methods				1	
#69 by Farahhmhd was merged 2 days ago									
<input type="checkbox"/>				feat: App css	area:front				
#68 by Reight123 was merged 2 days ago									

Pour les changements commits, nous avons adopté les conventions de [conventionalcommits.org](#) afin de standardiser la déclaration des différents changements et garder un historique consistant.

feat: print line in error if referred to (#78)	Verified	cfe682a		
na-teag committed 15 hours ago				
feat: print line in error if referred to		08237b0		
na-teag committed 15 hours ago				
feat: MATH methods (#77)	Verified	9b0c66c		
na-teag committed 19 hours ago				
feat: MATH methods		0b10ffc		
na-teag committed 19 hours ago				
feat: del and remove methods (#76)	Verified	00dd0cc		
na-teag committed 19 hours ago				
feat: del and remove methods		83a48d9		
na-teag committed 19 hours ago				
Feat: changing color background	Verified	a88dcc3		
Reight123 committed yesterday				
dev: button for redraw + css		e671821		
Reight123 committed yesterday				
dev: button for color background of drawing		9d9d014		
Reight123 committed yesterday				
dev: comment		9ed9ec7		
Reight123 committed yesterday				
dev: comment		2c11187		
Reight123 committed yesterday				
dev: fix mooving drawing page		3833448		
Reight123 committed yesterday				
fix/css removed by error (#74)	Verified	d0c7b7f		
na-teag committed 2 days ago				

## Intégration Continue (CI)

Pour assurer le bon fonctionnement du projet final et se prémunir du manque de temps, nous avons réparti nos tâches grâce à un planning.

Projet Java   Visible par l'espace de travail  Tableau				
13 Mai	14 Mai - 19 Mai	20 Mai - 22 Mai	23 Mai - 26 Mai	
Matthias 1/1	Les Méthodes 1/2 19/21	Les Méthodes 2/2 2/2	Peaufinage + Bonus	
Clément 1/1	+ Ajouter une carte	+ Ajouter une carte	+ Ajouter une carte	
Gaétan 1/1				
Adam 1/1				
Farah 1/1				
+ Ajouter une carte				






# Répartition des tâches

Nous avons décidé de séparer notre équipe en plusieurs parties :

- Clément PRAUD, chargés du menu et de la fenêtre de dessin (Java/JavaFX/CSS)
- Gaétan RETEL, chargés de l'interpréteur (Java)
- Adam SWICZKA, Matthias RIBEIRO et Farah MAHMOUD de la gestion du curseur et des méthodes de dessin (Java/JavaFX)

Chaque repository Github possède un système de "issues", c'est-à-dire de problèmes/tâches à faire, assignables à une ou plusieurs personnes, associées à des catégories. Le cycle habituel de la résolution d'un bug/l'implémentation d'une fonctionnalité était donc:

- S'assigner à l'issue correspondante
- Créer une nouvelle branche
- Implémenter la fonctionnalité/résoudre le bug
- Créer une pull request pour fusionner les changements avec la branche principale

<input type="checkbox"/>	<input checked="" type="radio"/> Add colour gestion by name in interpreter <span>area:back</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> Update the class diagram and use case diagram <span>documentation</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> write the report <span>documentation</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> write the README <span>documentation</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> load instruction in app when opening a txt file <span>area:back</span> <span>area:front</span> <span>bonus</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> update command if already launched to save execution time <span>area:back</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> write the doc for the whole project <span>documentation</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> Option speed <span>area:back</span> <span>area:front</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> comment and document the pointer class <span>documentation</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> Add option to skip incorrect command to interpreter <span>area:back</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> Bonus: game : try to reproduce a given drawing <span>area:back</span> <span>area:front</span> <span>bonus</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> Method MIRROR <span>area:back</span> <span>area:front</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> Method REMOVE and DEL <span>area:back</span> <span>area:front</span>	
<input type="checkbox"/>	<input checked="" type="radio"/> Method COLOR <span>area:back</span> <span>area:front</span>	

# Architecture du projet

## Choix des technologies

Le projet donné comprenait certains aspects techniques :

- gestion des dépendances
- exécution des différents fichiers en synchronisé

C'est pourquoi nous avons choisi d'utiliser Maven qui gère la mise en place des éléments techniques cités, nous permettant de nous concentrer sur le cahier des charges associé à l'énoncé de notre projet. Maven consiste en un outil de gestion et d'automatisation de production des projets logiciels Java.

## Méthode de conception

Nous avons adopté une approche centrée sur le Domain Driven Development (DDD).

## Modélisation métier

Le Domain Driven Development consiste à commencer par représenter le domaine métier et de construire le reste du code par dessus. Cette approche est intéressante car elle nous permet de nous concentrer sur l'aspect fonctionnel du projet et de ne pas accorder trop de responsabilités à la couche technique. Nous avons donc modélisé notre métier en créant les différentes relations. Ces relations sont modélisées par un diagramme de classe ainsi qu'un diagramme des cas d'utilisation.

## Choix techniques

Pour ce projet, plutôt que de construire un nouveau langage de programmation, nous avons développé un interpréteur qui traduit le pseudo code entré par l'utilisateur en java. Ce code java est ensuite compilé et exécuté.

Ce choix assure une gestion des erreurs au niveau de la partie exécution du code de l'utilisateur très robuste, car cette partie s'exécute complètement séparément du code principal de l'application. En effet, ce code sera exécuté dans un autre thread, et par une commande dans un terminal différent. Ainsi, lorsque l'utilisateur rentre un code incorrect, une partie des erreurs est gérée par l'interpréteur et le fichier App, et le reste est envoyé directement à java. De cette façon, si un problème survient et que le code crash ou ne compile pas, le programme principal récupère simplement l'erreur fournie par java, et l'affiche à l'utilisateur.

# Limitations fonctionnelles

## Absence de méthode pas à pas

Dans notre projet, l'utilisateur doit pouvoir modifier la vitesse d'exécution et accéder à une vitesse pas à pas. Ainsi, dans le but d'atteindre cet objectif, nous nous sommes concentrés, dans un premier temps, sur `forward` et `backward` pour appliquer dès le début une vitesse au tracé. De nombreux problèmes ont surgi :

Java est un langage de programmation synchrone : il exécute toutes les instructions avant de renvoyer l'affichage graphique. Par conséquent, lorsque la fenêtre est chargée, le dessin est fini et il n'y a aucune animation. On ne peut donc pas répondre au cahier des charges.

**1ère tentative** : utiliser `Thread` ou `TimeUnit.sleep` pour mettre des pauses entre chaque pixel. Le problème est qu'il n'y a aucun changement et les instructions se chargent après le `Thread`.

**2ème tentative** : à l'aide de la documentation Java, nous avons tenté différentes approches pour transformer la méthode `forward` en asynchrone, mais toutes furent des échecs.

**3ème tentative** : utiliser `TranslateTransition`. Cette bibliothèque JavaFX est très utilisée pour les animations et définit des durées aux transitions. Malheureusement, elle ne permet pas de tracer et semble ne fonctionner que dans la méthode `start()`.

**1er succès** : en consultant la documentation, nous sommes tombés sur `AnimationTimer`, qui permet aux actions d'être asynchrones. Ainsi, elle met à jour la position pixel par pixel et trace entre les deux. Malgré un effet de tracé continu, la méthode asynchrone n'attend pas sa fin pour charger les instructions suivantes. En conséquence, les positions sont incorrectes et les tracés sont en simultané.

Pour résoudre cet obstacle, nous avons divisé la méthode `fwd` en deux. `Forward` calcule la position finale et appelle la méthode asynchrone pour le tracé. Cela permet de ne pas croiser les deux et de provoquer des erreurs d'affichage. Toutefois, les tracés se font tous en même temps.

**2ème succès** : ne pouvant pas réaliser des actions pas à pas, nous avons tout de même réussi à modifier la vitesse d'exécution grâce à `TimeUnit.sleep()` dans la méthode asynchrone. Si la vitesse augmente, le temps de sommeil diminue proportionnellement.

Pour modifier la vitesse, il faut soit passer par le **curseur de l'interface** ou écrire sur la **1er ligne du fichier `.txt`** `"SPEED [VALUE]"`.

**Bilan** : nous ne sommes pas parvenus à créer la méthode pas à pas jusqu'au dernier moment. Aucune information ne pouvait nous aider sur Internet et, d'après l'un de nos camarades, la seule manière est de passer par du **bytecode**. Il aurait donc fallu revoir toute notre structure en milieu de projet, ce qui est une mauvaise décision par rapport au temps qu'ils nous restaient.

## Impossibilité de supprimer des variables en java

Dans le cahier des charges, les fonctions DEL et REMOVE doivent supprimer respectivement des variables de type primitifs et des instances de pointeurs. Or, en Java il semblerait qu'il ne soit pas possible d'effectuer ces opérations. Pour contourner le problème, l'interpréteur tient une liste des variables qui ont été "supprimées" par l'utilisateur, et si il essaie de le ré-instancier, l'interpréteur ne fera qu'assigner une nouvelle valeur, et donc java ne produira pas d'erreur.

Pour autant, une variable sera donc toujours existante et accessible après la commande DEL ou REMOVE. Cependant, après la commande REMOVE, la variable de type Pointer pointera sur null, pour simuler au mieux la suppression.

Il n'est en revanche pas possible de faire de même pour les variables de type primitif, car elles ne peuvent pas prendre la valeur null, et les classes enveloppantes (wrapper classes) qui sont des classes qui remplacent les types primitifs en autorisant la valeur null ne s'utilisent pas exactement de la même manière, et ne sont donc pas viables.

## Bonus

Nous avons intégré dans ce projet plusieurs fonctionnalités qui n'étaient pas demandées, principalement pour des raisons de praticité.

La commande COLOR accepte par exemple un nom de couleur en anglais, plutôt que de couleurs disponibles forcer l'utilisation de code hexadécimale ou RGB. Cette liste est définie selon les 147 couleurs prises en compte dans la classe [Color](#) de java 21.

L'utilisateur peut aussi choisir lorsqu'un dessin est affiché de redessiner le dessin sans recharger le code. Il peut également choisir de, changer la couleur du fond d'écran du dessin.

La commande TURN est accompagnée des commandes TURNL et TURNR afin de spécifier clairement le sens de rotation.

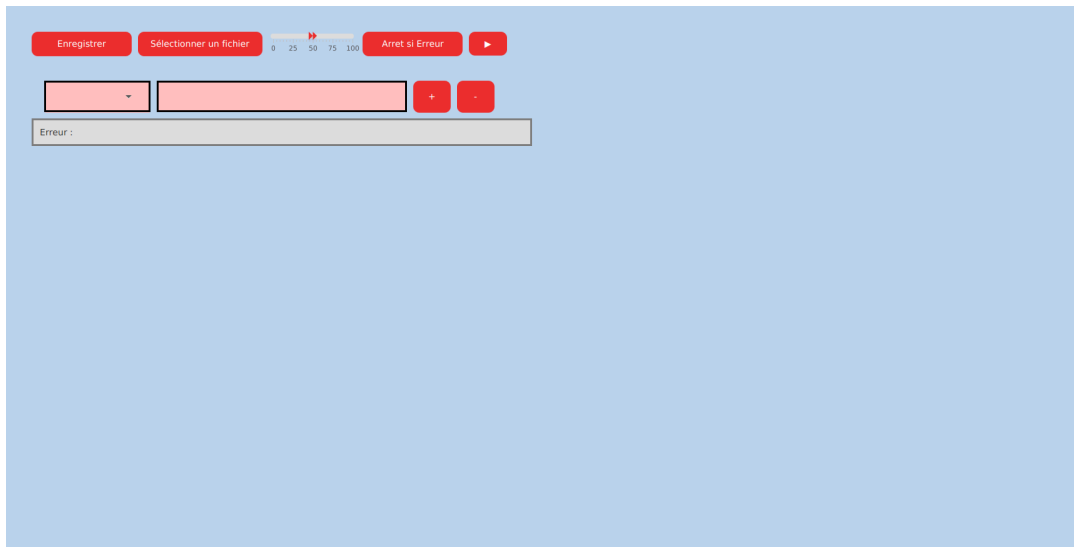
Une fonction MATH est disponible pour faire des calculs ou des instructions qui ne requiert pas l'utilisation de méthode, chose qui n'est pas permise dans le cahier des charges initial.

L'utilisateur dispose également de commandes lui permettant de dessiner des formes géométriques pré enregistrés, à savoir des cercles, des croix, des triangles, des rectangles et des carrés. Ces formes peuvent être remplies ou non.

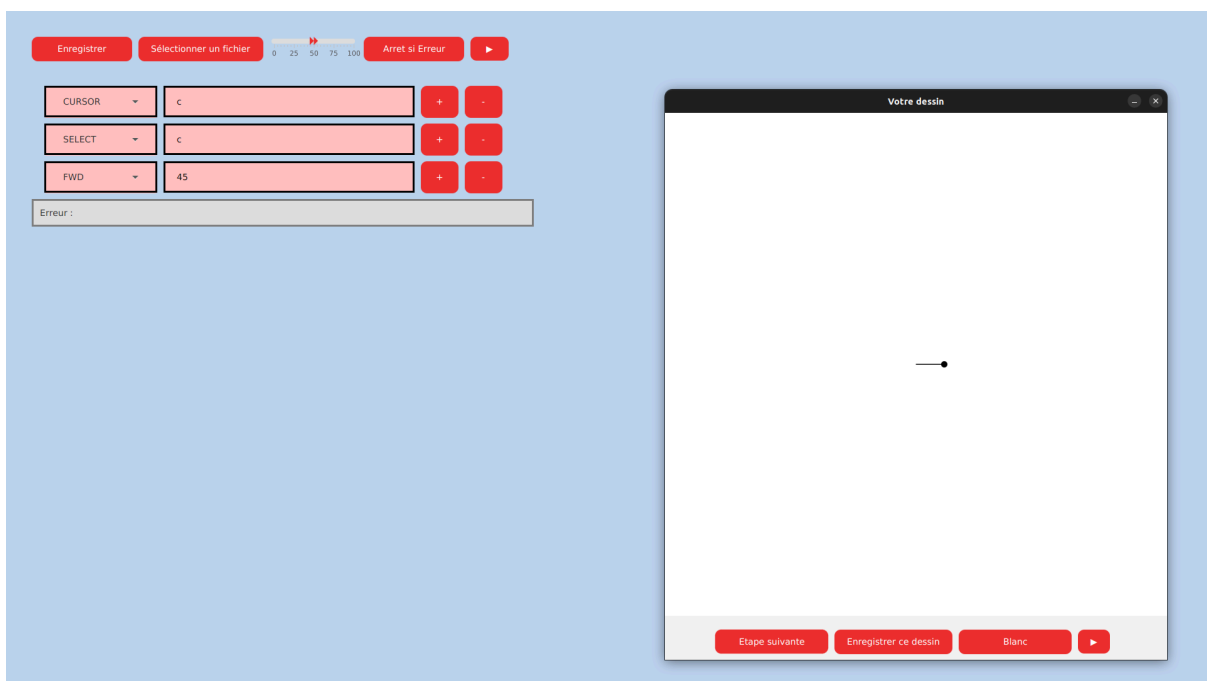


# Présentation de l'application

## Menu



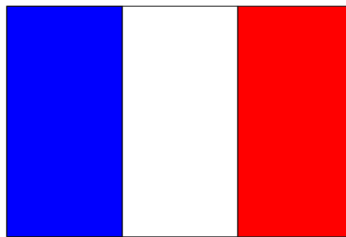
## Fenêtre de dessin



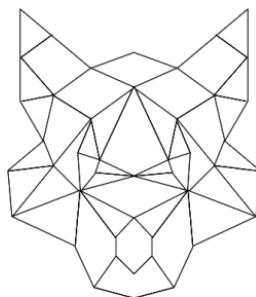
## Nos dessins



Nous avons dessiné Mario grâce aux méthodes supplémentaires que nous avons créées. Celles-ci prennent en entrée des longueurs ou des rayons et permettent de dessiner des carrés, des triangles, des rectangles, des croix et des cercles vides ou pleins.



Pour le drapeau français nous avons utilisé la méthode mimic : ici un mimic dans un mimic pour réaliser en simultanée trois rectangles.



Enfin, pour le loup, nous sommes passés par la méthode mirror (symétrie axiale) pour éviter de répéter les instructions en sens inverse.