



## *Librairie graphique pour le langage C++*

**Projet soutenu le 19 janvier 2024**

### **Membres du Groupe**

*Nadir MIR*  
*Thomas VENEROSO*  
*Keïlan VARON*  
*Dorian BUCCHIOTTY*  
*Jonathan DAGON*

**Projet effectué au département informatique de l'IUT de Metz**



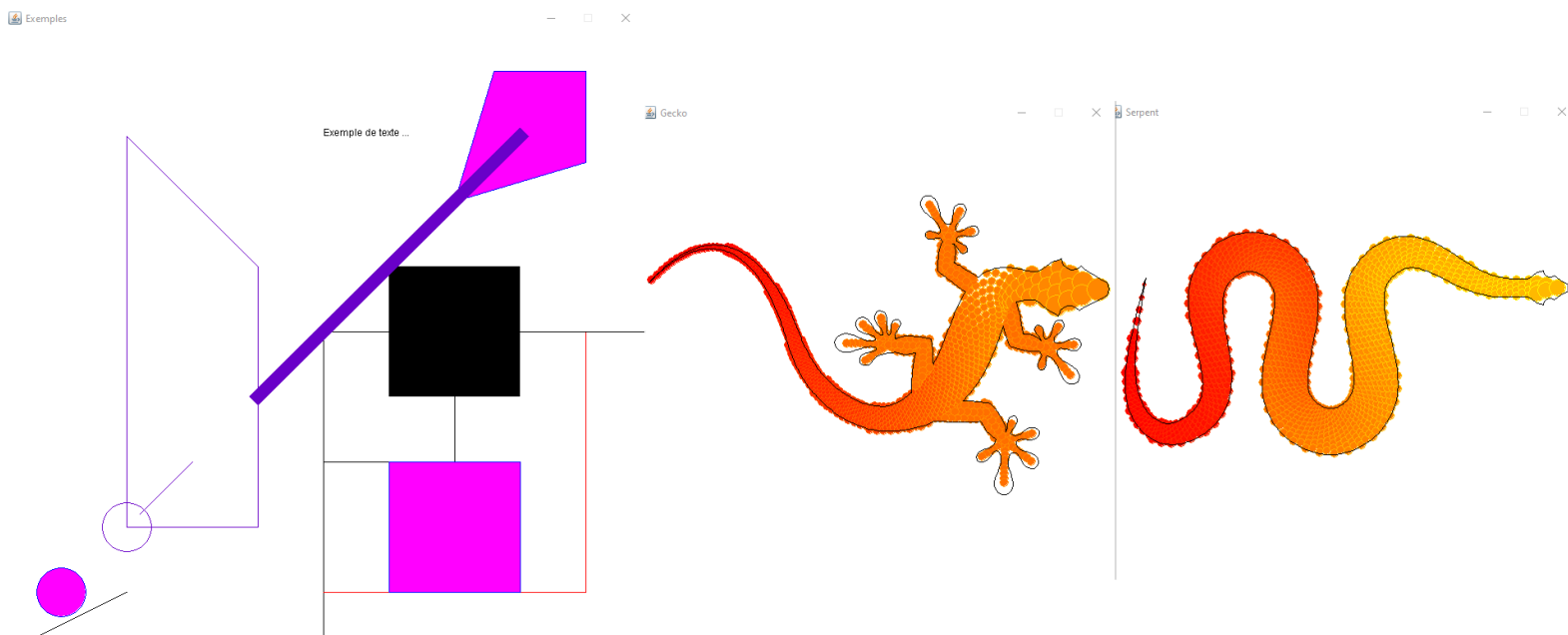
<b>Introduction.....</b>	<b>3-4</b>
• Sommaire .....	3
• Sujet du projet / Cahier des charges .....	4
<b>Développement.....</b>	<b>5-29</b>
• Gestion du projet .....	<b>5-7</b>
○ Répartition des tâches / heures .....	5-6
○ Pourcentage de travail .....	6
○ Diagramme de Gantt .....	7
• Analyse du sujet .....	<b>8-19</b>
○ Diagrammes .....	8-11
○ Outils utilisés .....	12-15
○ Nouveautés et problèmes potentiels .....	16-17
○ Solutions et méthodes .....	18-19
• Infrastructure .....	<b>20-29</b>
○ Caractéristiques du serveur .....	20
○ Outils utilisés .....	20
○ Difficultés rencontrées .....	21
○ Sécurité .....	21-23
○ Étapes de la configuration du projet (Windows).....	24-29
<b>Conclusion.....</b>	<b>30</b>
• Conclusion et perspectives .....	30

## Introduction

### Sujet du projet :

Le langage C++ est un excellent langage de programmation, largement utilisé pour développer des applications exigeant des calculs intensifs, tels que les jeux vidéo. Cependant, bien qu'il soit très utilisé, le langage ne dispose pas d'une librairie graphique intégrée (même si plusieurs projets indépendants existent). Notre objectif est de remédier à cette lacune en développant notre propre librairie graphique pour ce dernier.

Le langage JAVA offre des caractéristiques complémentaires à celles du langage C++. Malgré ses performances de calcul moins optimales, JAVA est doté d'un ensemble complet de librairies (graphiques, bases de données, cryptage, etc.). La librairie AWT, en particulier, permet de créer des dessins complexes (voir images ci-dessous) tout en restant simple à utiliser. Ainsi, notre projet est d'exploiter les fonctionnalités de la librairie AWT depuis un programme C++. La communication entre les deux langages sera établie sous la forme d'une application client-serveur TCP/IP, où le programme C++ agira en tant que client et le programme JAVA en tant que serveur.



### Cahier des charges :

Nous allons écrire une maquette minimaliste de cette librairie.

Le cahier des charges retenu est le suivant :

- Ouverture d'une fenêtre de dessin
- Tracé d'un segment en précisant couleur et épaisseur de tracé
- Tracé d'un cercle (avec couleur)
- Tracé d'un disque (avec couleur d'intérieur et couleur de bord)
- Tracé d'un polygone plein (avec couleur)
- Inscription d'un message
- La librairie graphique prendra en charge le changement des coordonnées entre monde et écran (en anglais : mapping coordinates to viewport).

## Développement

### Gestion du projet :

- Répartition des tâches / heures

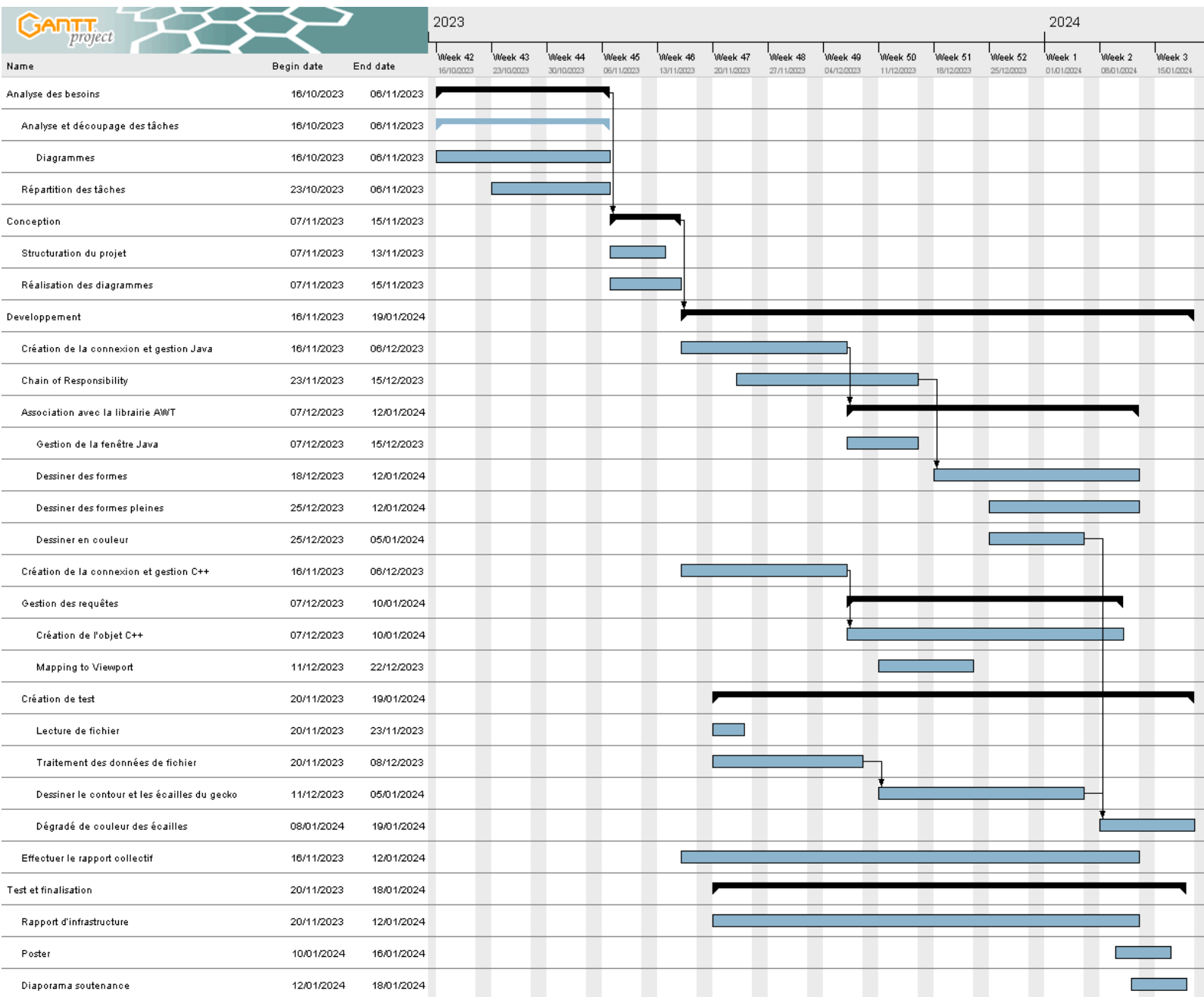
Noms	Tâches	État	Taux horaire
Dorian, Jonathan	Création de la connexion et gestion Java	Achevé ▾	7h   4h
Dorian, Jonathan	Création de la connexion et gestion C++	Achevé ▾	4h   6h
Dorian	Lecture de fichier	Achevé ▾	2h
Dorian	Traitement des données de fichier	Achevé ▾	2h
Dorian	Gestion de la fenêtre Java	Achevé ▾	12h
Dorian, Jonathan, Thomas	Dessiner des formes	Achevé ▾	10h   5h   1h
Dorian	Dessiner des formes pleines	Achevé ▾	6h
Dorian	Dessiner en couleur	Achevé ▾	5h
Dorian, Jonathan, Thomas	Gestion des requêtes	Achevé ▾	4h   4h   1h
Thomas	Chain of Responsibility	Achevé ▾	15h
Dorian	Mapping to Viewport	Achevé ▾	3h
Dorian, Jonathan, Thomas	Diagrammes	Achevé ▾	5h   4h   4h
Jonathan, Thomas, Nadir	Poster	Achevé ▾	1h   3h   1h
Nadir	Diaporama soutenance	Achevé ▾	2h

Noms	Tâches	État	Taux horaire
Jonathan	Dessiner le contour et les écailles du gecko	Achevé ▾	11h
Jonathan	Dégradé de couleur des écailles	Achevé ▾	5h
Dorian, Jonathan, Nadir , Keilan, Thomas	Effectuer le rapport collectif	Achevé ▾	4h   9h   3h   3h   3h
Jonathan, Dorian, Thomas	Rapport d'infrastructure	Achevé ▾	6h   2h   1h

**Pourcentage de répartition du travail dans le projet :**

- Dorian : 41%
- Jonathan : 35%
- Thomas : 18%
- Nadir : 4%
- Keilan : 2%

- Diagramme de Gantt

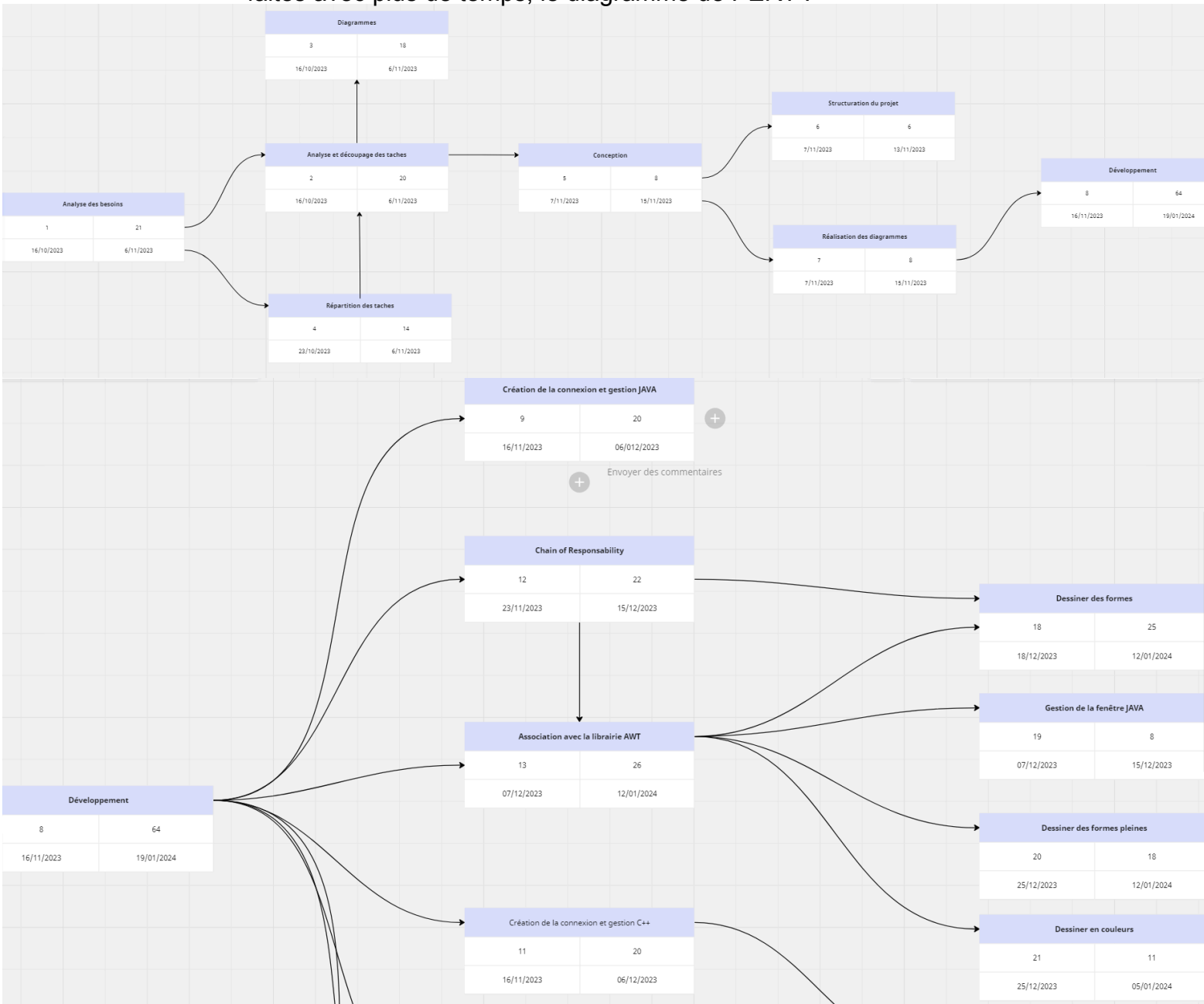


La gestion du projet a été efficace grâce à une répartition claire des tâches et à l'utilisation d'un diagramme de Gantt. Le suivi de ce dernier a permis de ne pas prendre de retard sur les tâches, et a également permis aux membres du groupe partiellement de travailler dans les temps de façon collective. Cette approche a permis d'assurer une coordination compétente entre les membres, ce qui a garanti ainsi le respect des délais et la réussite globale du projet. De plus, le respect des délais a permis d'ajouter d'autres fonctionnalités au projet, exemple : gérer la lecture d'un fichier de points, permettant par la suite de les dessiner sur la fenêtre Java.

## Analyse du sujet :

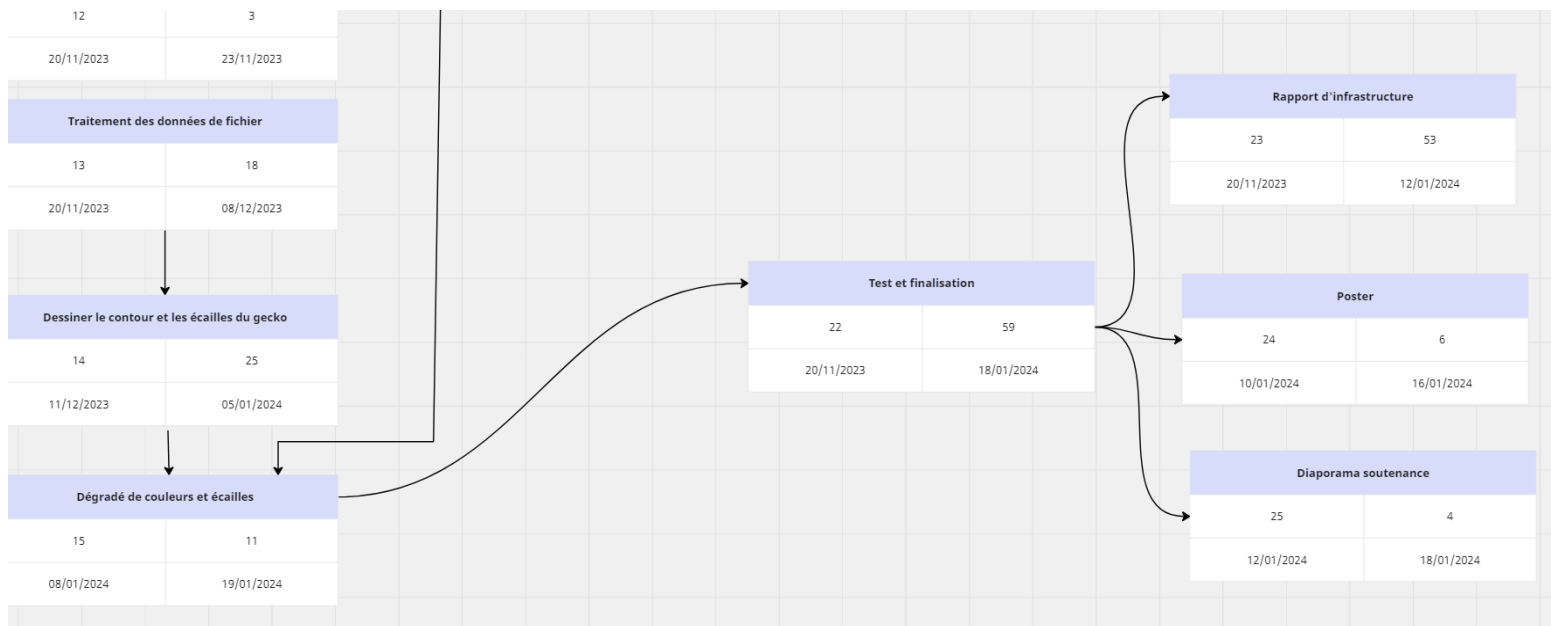
### ❖ Diagrammes

- Utile pour voir quelles tâches sont critiques et lesquelles peuvent être faites avec plus de temps, le diagramme de PERT :

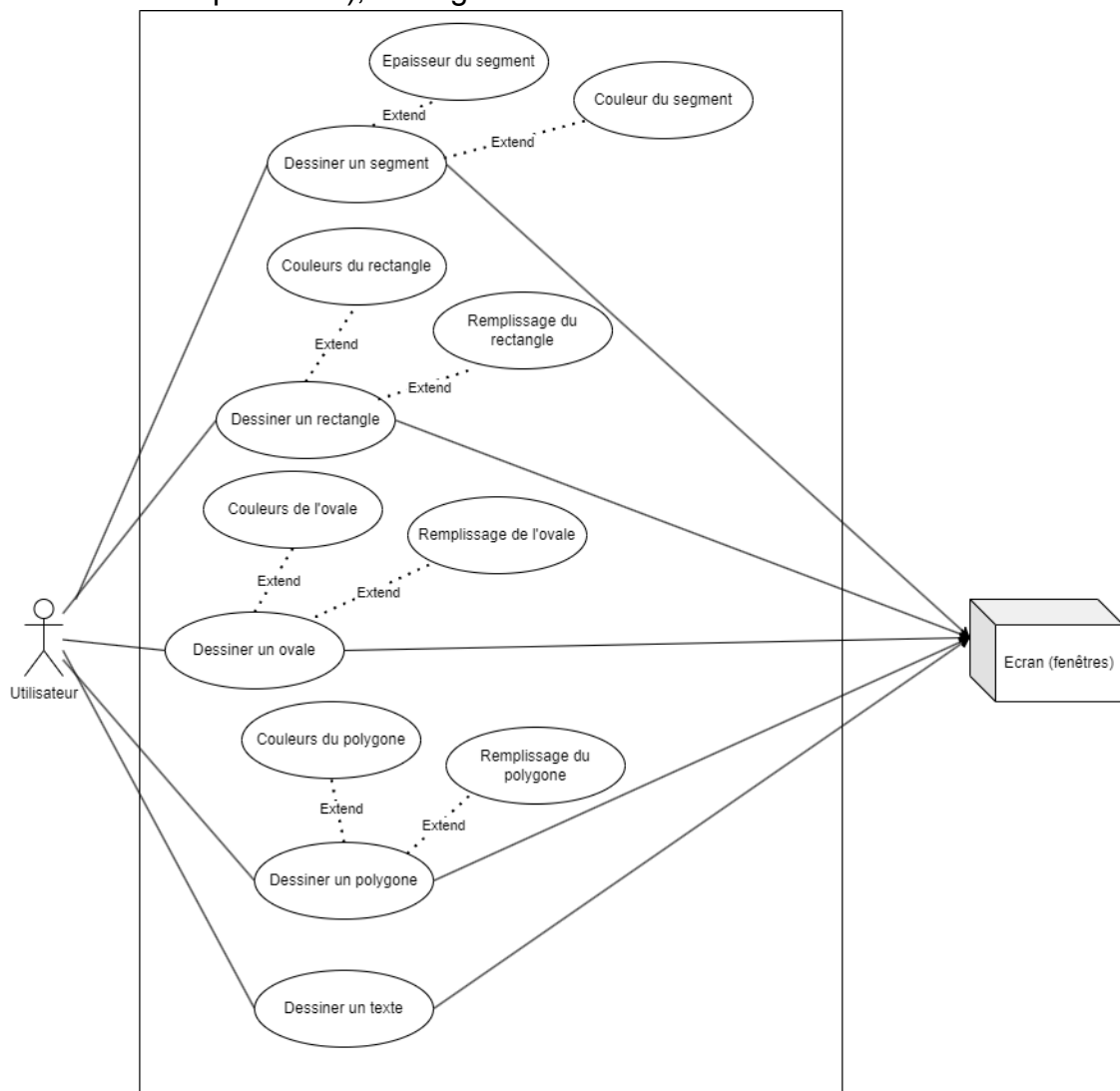




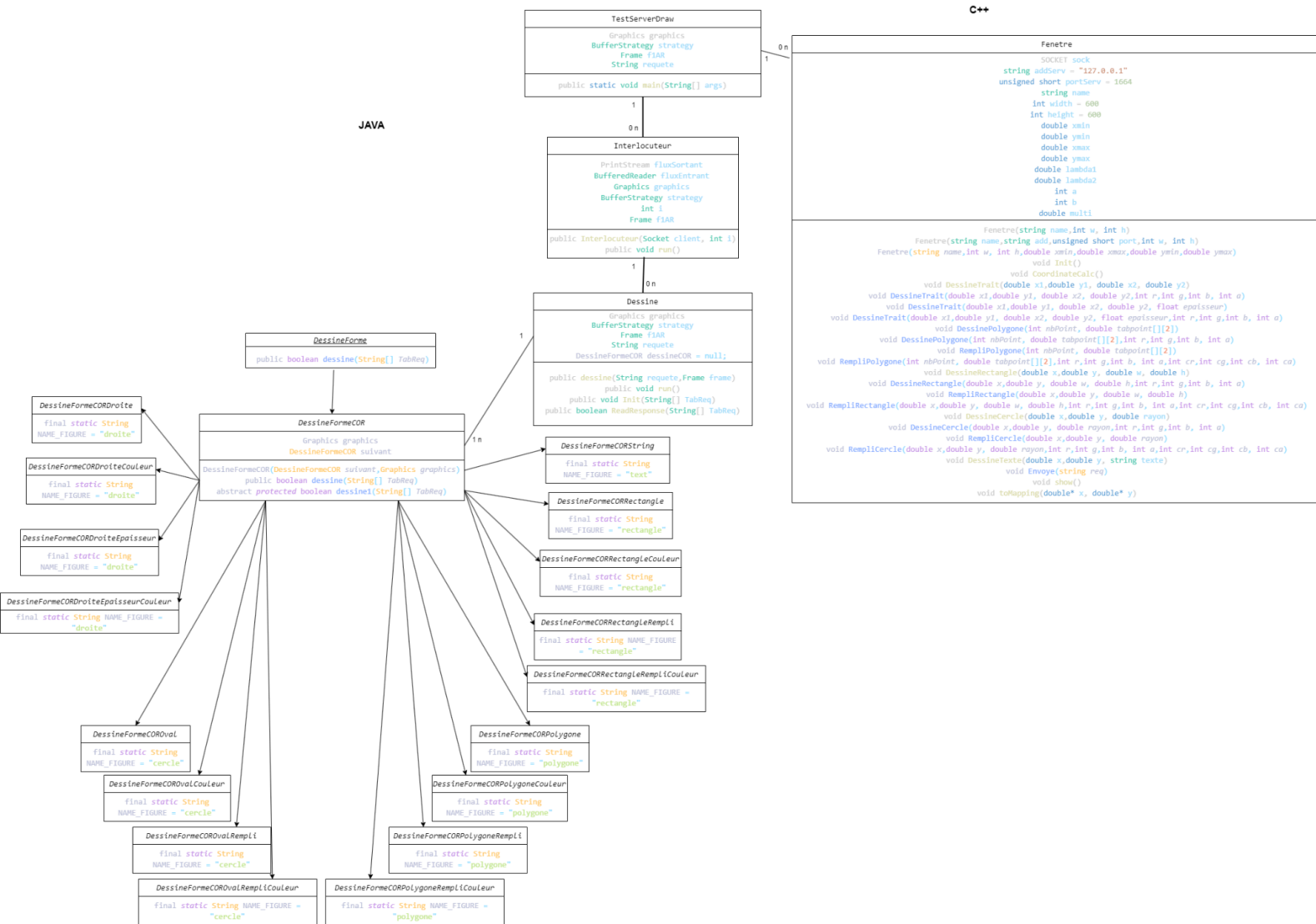




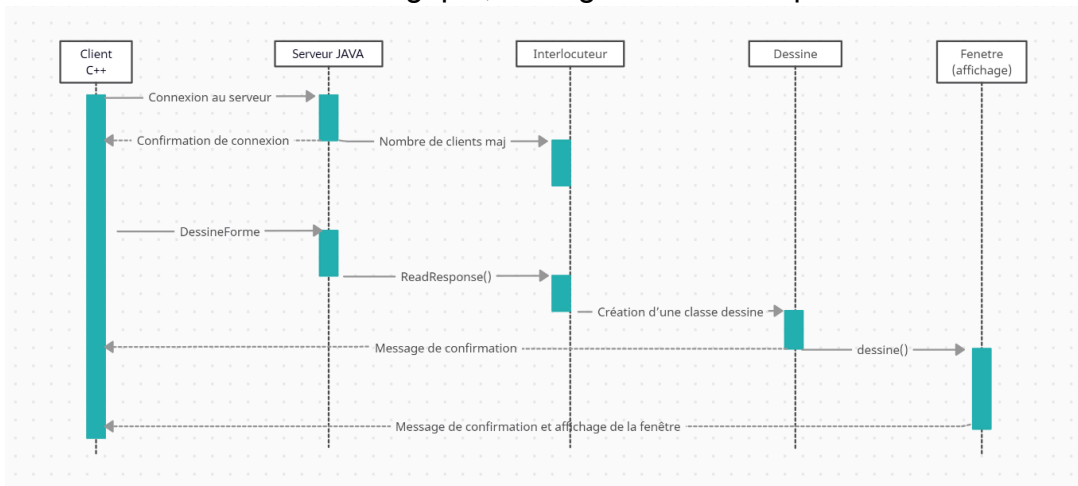
➤ Détail le fonctionnement du projet entre l'utilisateur et le serveur (les actions possibles), le diagramme de cas d'utilisation :



- Permet de visualiser la structure d'un projet en illustrant les classes, leurs attributs, et les relations entre elles, le diagramme de classes :



- Illustre la manière dont les différentes parties du projet interagissent entre elles dans un ordre chronologique, le diagramme de séquence :



## **Outils utilisés :**

Dans le cadre de ce projet, une liste d'outils a été faite pour garantir son efficacité. L'utilisation judicieuse de langages tels que Java et C++, associée à des bibliothèques graphiques telles que AWT, a permis de concrétiser l'avancée du projet. Ci-dessous, l'ensemble des outils utilisés qui ont permis le bon déroulement de ce dernier.

### ❖ Outils

#### ➤ **Langage Java**



Java, langage de programmation réputé, brille par sa portabilité via la JVM, sa gestion automatique de la mémoire, et une bibliothèque complète.

Son approche orientée objet simplifie le développement, créant des applications robustes et multiplateformes. Prisé pour son écosystème actif et les nombreux frameworks, notamment AWT.

#### ➤ **Langage C++**



Le C++, langage de programmation puissant, se distingue par sa performance, son contrôle direct sur la mémoire et son support complet de la programmation orientée objet. La portabilité est assurée, bien que moins transparente qu'avec Java.

Avec une syntaxe flexible, le C++ est utilisé pour des applications nécessitant une efficacité maximale, telles que les systèmes embarqués, les jeux vidéo et les logiciels critiques en termes de performance.

Sa popularité perdure grâce à une large communauté et à son utilisation dans des domaines divers, du développement système aux applications haute performance.

#### ➤ **Visual Studio Code**



Utilisé pour développer le code du projet. Également utilisé, avec des extensions, pour exécuter les bouts de code, c'est-à-dire, des extensions Java pour la compilation et démarrer le serveur, et des extensions C++ pour compiler les fichiers de calculs et démarrer le client.

➤ **MINGW pour la compilation C++**



Une adaptation des logiciels de développement et de compilation du GNU à la plate-forme Win32.

Utilisé pour installer le compilateur C++ sur la machine. Ce qui permet de compiler le code C++ du client permettant d'utiliser notre librairie graphique.

➤ **Java Development Kit (JDK)**



Un ensemble de bibliothèques logicielles de base du langage Java, tel que les outils avec lesquels le code Java peut être compilé.

Utilisé pour la compilation des classes à partir des fichiers Java établies sur Visual Studio Code, qui sont, essentielles à l'ouverture de la fenêtre de dessin.

➤ **Java Runtime (JRE)**








Une couche logicielle qui s'exécute sur le logiciel du système d'exploitation d'une machine et fournit les bibliothèques de classes et les autres ressources dont un programme Java a besoin pour fonctionner.

Utilisé pour ouvrir la fenêtre de dessin et dessiner dessus. Permettant le lancement du serveur java qui est essentiel au bon fonctionnement de la librairie graphique.

➤ **Github / Git**



 <b>DorianBucc</b> Gecko and snake add	823b687 · 2 hours ago	 <b>7 Commits</b>
 <b>projet</b>	Gecko and snake add	2 hours ago
 <b>src</b>	Correction Bug trait	yesterday
 <b>README.md</b>	Initial commit	last week

Un service de gestion de projet, utilisant le logiciel traitant et synchronisant des versions d'un projet qui se nomme Git. Assure par ailleurs un contrôle d'accès et des fonctionnalités destinées à la collaboration comme les demandes de fonctionnalités, la gestion de tâches.

Utilisé comme dépôt privé de l'équipe de projet, pour mettre à jour la programmation de cette application dans chaque partie du projet afin d'effectuer la synchronisation.

➤ **Mermaid / Draw.io**



Des éditeurs de graphiques. Principalement basés sur le web, avec des fonctionnalités hors ligne supplémentaires activées par des applications de bureau. L'ensemble des fonctionnalités est axé sur l'utilisation dans la conception de l'interface utilisateur et de l'expérience utilisateur, en mettant l'accent sur la collaboration en temps réel.

Utilisé pour créer et développer les différents diagrammes. Permettant de trouver la cohérence sur les liens entre les différentes classes.

### ➤ **GanttProject**



Un logiciel libre de gestion de projet qui permet d'éditer un diagramme de Gantt. Un outil de gestion de projet utilisé pour planifier et suivre les activités et les tâches du projet.

L'outil fournit des informations visuelles sur le calendrier du projet, les délais et les responsabilités de chaque membre de l'équipe.

Les tâches sont représentées sous forme de graphiques composés de barres horizontales qui indiquent la durée.

### ➤ **Discord**



Sur l'aspect discussion, et partage de fichiers, nous nous sommes tournés vers Discord.

Cette application a paru comme une évidence pour nous, au vu de la grande connaissance et de l'utilisation quasiment quotidienne de celles-ci, en plus d'une utilisation dans notre cursus du BUT.

### ➤ **Google Docs**



Utilisé pour créer un rapport en équipe et rassemblé toutes les idées sur le même rapport.

Ci-dessus ont été détaillés les outils essentiels pour la mise en œuvre réussie du projet. Ces outils, soigneusement sélectionnés et expliqués en profondeur, ont joué un rôle crucial dans le développement de l'application. Leur utilisation a permis une gestion optimale du code, une collaboration efficace et la résolution des divers défis techniques.

## **Nouveautés et problèmes potentiels :**

Nouveautés :

1. Java / C++
2. AWT
3. Socket Réseaux
4. Mapping
5. Chain of Responsibility

Dans le cadre de ce projet, certaines nouveautés ont généré quelques “problèmes”. Le premier de cette liste a été le calcul des points sur la fenêtre Java. Il a été identifié que ce point nécessitait une certaine attention. Le deuxième, plus visuel, était lié à la redimension de la fenêtre et à la disparition potentielle des parties des dessins. Les détails de ces problèmes sont exposés ci-dessous.

### 1. Problème : Calculer les points sur la fenêtre Java

Le premier problème était spécifique au calcul des points dans la fenêtre Java, c'est-à-dire bien calculer où les points devraient être sur la fenêtre sans saccager le dessin.

Détail :

- Certains points, destinés à être affichés dans la fenêtre graphique, se retrouvaient en dehors des limites prévues du cadre de la fenêtre Java. Ils dépassaient les bords définis de cette dernière. Les conséquences visuelles de ce problème étaient visibles lorsque des dessins complexes étaient en cours d'exécution.

### 2. Problème : Redimensionner la fenêtre et redessiner par-dessus

Le deuxième problème était associé à la redimension de la fenêtre et au comportement du dessin sur cette dernière.

Détail :

- En redimensionnant la fenêtre Java, le dessin existant n'était pas adapté à la nouvelle taille. Il arrivait donc, dans la majorité des cas, que le dessin soit déformé. Dans certains cas, la redimension pouvait entraîner l'effacement complet du dessin.



### 3. Problème : Dessiner sur la fenêtre java au moment demandé

Le troisième problème était spécifique au fonction permettant de dessiner sur la fenêtre Java, c'est-à-dire pouvoir décider de dessiner sur la fenêtre au moment demandé.

Détail :

- Certaines fonctions de la librairie AWT, destinées à dessiner sur la fenêtre graphique, se dessinent dans un laps de temps incertain. Le problème est qu'on ne veut pas attendre que la page décide toute seule du moment où elle dessine. Ce qui ne conviendrait absolument pas au cahier des charges d'une librairie graphique.

### 4. Problème : Perte des requêtes sur la socket

L'avant dernier problème était que certaines requêtes n'arrivaient pas au serveur java au début et à la fin.

Détail :

- Le problème apparaissait qu'à certains moments, les quelques premières requêtes du début disparaissaient seulement sur la première fenêtre. Contrairement aux requêtes à la fin, elles disparaissaient qu'à une condition, si les requêtes étaient envoyées trop rapidement.

### 5. Problème : Comment organiser et compiler le projet pour une utilisation simple et intuitive

Le dernier et l'un des plus gros problèmes à été l'organisation et la compilation du projet, car cela devait garantir une expérience utilisateur fluide, simple et optimale.

Détail :

- Le problème crucial résidait dans l'organisation et la compilation du projet, nécessaire pour assurer une expérience utilisateur optimale. La création de fichiers et de classes manquait de structuration, ne favorisant pas une utilisation simple et intuitive.

Ci-dessus ont été listés les plus gros problèmes survenus durant ce projet et la nouveauté découverte. Comme cela à été expliqué, on a dû faire face à de nombreuses heures de réflexion afin de pallier ces contraintes. Cependant et fort heureusement, des solutions ont été trouvées, ce qui a grandement résolu les complications qu'on a rencontré et nous a permis d'avancer de façon bien plus efficace. Les solutions ayant été trouvées seront listées dans la prochaine page, avec également, les méthodes qui nous ont permis d'y parvenir.

## **Solutions et méthodes :**

Plusieurs approches ont été explorées pour résoudre les problèmes identifiés. Cependant, afin d'optimiser la compréhension et d'assurer une efficacité maximale, une seule solution a été retenue pour chaque problème. Les solutions ainsi que le détail des solutions sont expliqués ci-dessous.

### **1. Solution : Passage des coordonnées du monde à l'écran (Mapping)**

La solution a été trouvée à l'aide d'un peu de recherche, et d'aide extérieure au projet. Cela nous a donc permis de trouver quelque chose qui soit à la fois viable, simple et efficace en toutes circonstances.

Détail :

- La première solution a impliqué la mise en œuvre d'une technique de mapping pour résoudre le problème de calcul des points sur la fenêtre Java. Cette méthode a consisté à établir une certaine liaison entre les coordonnées du monde virtuel et celles de l'écran Java. Cela a nécessité de faire des calculs bien précis en prémisses, afin de convertir les valeurs réelles données, en valeur pixels sur la fenêtre. Le mécanisme de lecture en temps réel de la fenêtre Java a été intégré en temps que solution additionnelle pour assurer une meilleure adaptation, garantissant ainsi que les points restent dans les limites de la fenêtre.

### **2. Solution : Redimensionner la fenêtre et redessiner par-dessus**

La solution pour ce problème a été un peu plus fastidieuse. Certes, elle a été trouvée rapidement, mais plus difficile à mettre en œuvre, car gérer la taille de la fenêtre, et redessiner le dessin aux nouvelles dimensions, et ce, en temps réel, a été plus dur que prévu.

Détail :

- Le deuxième problème, lié à la redimension de la fenêtre et à la disparition potentielle des parties des dessins, a été résolu en introduisant une méthode de redimensionnement fixe. C'est-à-dire que cela a permis qu'à l'initialisation de la fenêtre, cette dernière ne soit pas redimensionnable, évitant ainsi les déformations indésirables. La lecture de la taille de la fenêtre est donc constante, donc le dessin n'a pas besoin d'être redessiné et est toujours adapté aux dimensions initiales.

### 3. Solution : Dessiner sur la fenêtre java au moment voulu

Résoudre ce problème a exigé une approche minutieuse et rapide. La solution a été identifiée assez facilement, bien que sa mise en œuvre ait nécessité des ajustements.

Détail :

- Le troisième problème a été résolu en adoptant une stratégie de recherche associée à une collaboration externe au projet. Cette démarche a abouti à une solution viable, simple et efficace pour garantir un dessin optimal au moment souhaité qui dessinait au moment voulu et non au moment où le java le voulait.

### 4. Solution : Perte des requêtes sur la socket

La solution pour ce problème a été assez longue à trouver et elle s'est découverte un peu à force d'essayer.

Détail :

- Le quatrième problème, des plus importants, s'est résolu par le constat d'un ajout qui permettait de mettre en pause le programme avant le début de l'envoi des requêtes. Nous avons donc augmenté certains délais pour corriger cela, ce qui a résolu 50% du problème. Ensuite, à force de tests et de réflexion, nous avons constaté que le problème venait du fait que le C s'exécute rapidement et se ferme trop tôt, fermant ainsi les threads permettant l'envoi de requêtes. Suite à cela, le programme devra attendre un certain délai avant de se terminer.

### 5. Solution : Comment organiser et compiler le projet pour une utilisation simple et intuitive

Pour résoudre ce problème, nous avons essayé et comparé à d'autres librairies graphiques.

Détail :

- La solution a été trouvée en cherchant la meilleure méthode pour ajouter la bibliothèque, en réfléchissant à la façon la plus intuitive et simple de l'organiser et de la compiler dans le projet. Le résultat a été d'ajouter un simple argument lors de la compilation et d'inclure la bibliothèque dans le code.

Ci-dessus ont été énoncées les solutions adoptées pour surmonter les problèmes rencontrés dans le projet. Ces solutions, expliquées en détail, ont été essentielles pour résoudre les problèmes identifiés, permettant ainsi une progression plus fluide et efficace du projet.

## **Infrastructure :**

- Caractéristiques du serveur

- **RAM 4 Go**

La RAM est amplement suffisante pour notre application. Avec seulement 38% utilisés par Windows et environ 50 Mo utilisés par notre application, cette configuration garantit une gestion optimale des ressources.

- **Processeur Intel Xeon Bronze 3106 1.7GHz :**

Le processeur répond largement aux besoins de notre application. Avec une utilisation inférieure à 1% tant par Windows que par notre application, la fréquence et la puissance de calcul sont amplement adéquates.

- **90 Go de stockage :**

Le stockage de 90 Go est bien au-delà des exigences de notre projet qui ne nécessite qu'environ 1 Mo. Cette capacité assure une marge confortable pour les données du projet.

- Outils utilisés

Le serveur utilise une panoplie d'outils pour assurer le bon fonctionnement de l'application. Parmi eux figurent Java, C++, Visual Studio Code, MINGW, Java Runtime (JRE), et Github/Git.

- **Langage Java :** Java, reconnu pour sa portabilité et sa gestion automatique de la mémoire, est utilisé pour développer des applications robustes et multiplateformes. Sa popularité s'appuie sur un écosystème actif et divers frameworks tels qu'AWT.
- **Langage C++ :** Le C++, puissant et offrant un contrôle direct sur la mémoire, est privilégié pour des applications nécessitant une efficacité maximale. Sa syntaxe flexible en fait un choix idéal pour les systèmes embarqués, les jeux vidéo et les logiciels critiques en termes de performance.
- **Visual Studio Code :** Utilisé pour le développement, avec des extensions Java et C++ facilitant la compilation et le démarrage du serveur et du client.
- **MINGW :** Adaptation des logiciels de développement GNU pour Win32, MINGW permet l'installation du compilateur C++ sur la machine, essentiel pour compiler le code client.
- **Java Runtime (JRE) :** Couche logicielle fournissant les bibliothèques de classes nécessaires à l'exécution du programme Java. Essentiel pour ouvrir la fenêtre de dessin et lancer le serveur Java.
- **Github / Git :** Service de gestion de projet utilisant Git, il assure un contrôle d'accès et des fonctionnalités de collaboration. Utilisé comme dépôt privé pour la synchronisation du code de l'application.

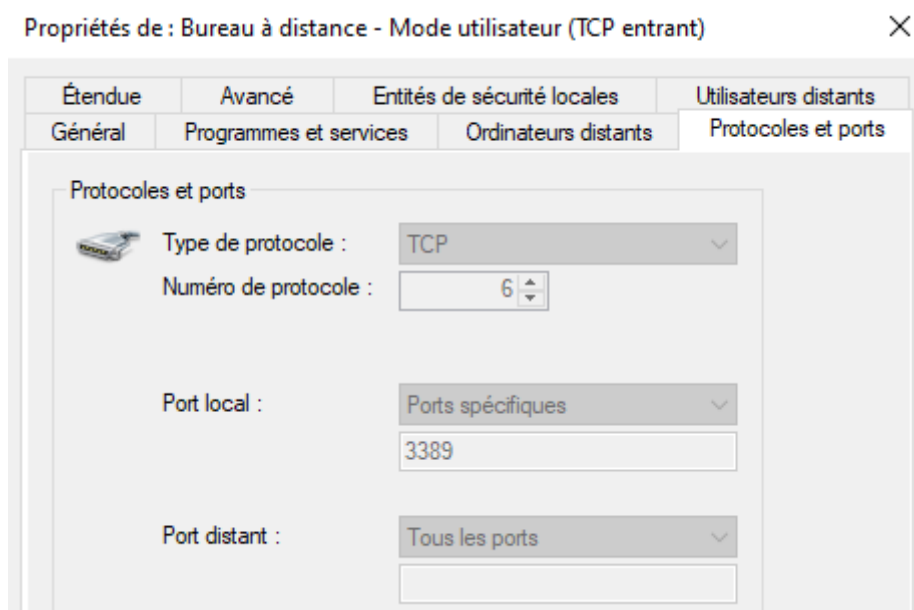
- Difficultés rencontrés

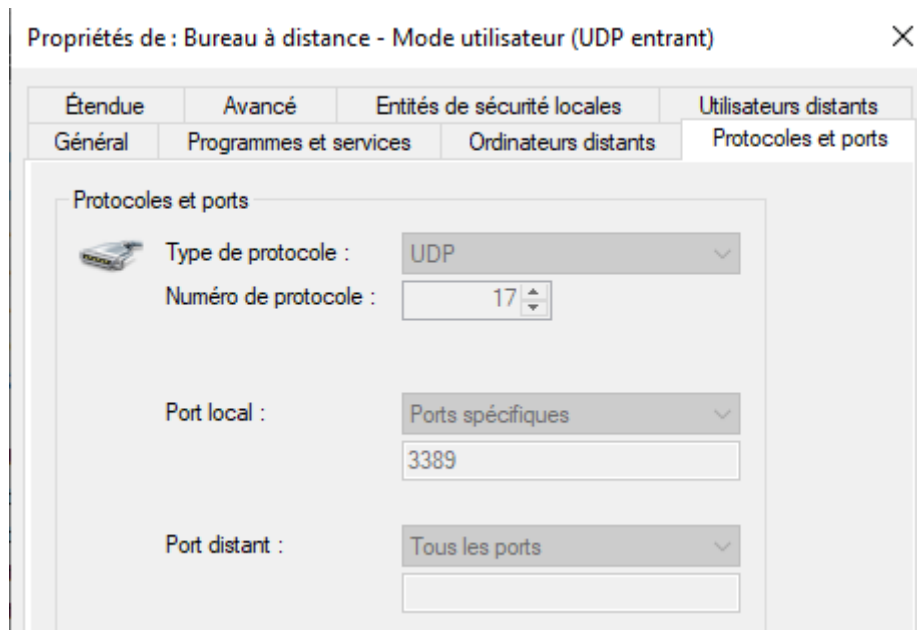
Aucune difficulté majeure n'a été relevée, car notre projet est conçu de manière intégrée, évitant ainsi des complications liées à la séparation des parties serveur et client. Selon notre réflexion, ce projet est exclusivement dédié aux développeurs, il ne sera pas probable qu'un utilisateur modifie l'application.

- Sécurité

La connexion à distance via le Bureau à distance (port 3389, TCP/UDP) demeure un pilier essentiel. Cette approche assure une surveillance en temps réel du rendu graphique sur notre machine Windows Server, favorisant la collaboration et la sécurité des données. En intégrant cette solution, nous garantissons un accès fluide et sécurisé aux fonctionnalités de notre bibliothèque graphique. Cette stratégie renforce la base de notre infrastructure, optimisant l'expérience utilisateur tout en préservant l'intégrité des données, alignant ainsi notre projet avec les normes de sécurité les plus strictes.

Nous mettons toutes les mesures pour renforcer à la fois la machine Windows Server et le système Bureau à distance (Remote Desktop). Au niveau de la machine, des mises à jour régulières sont impératives, garantissant l'application des derniers correctifs de sécurité. L'utilisation d'un logiciel antivirus et antimalware maintenu à jour est cruciale pour détecter et éliminer toute menace potentielle. La configuration d'un pare-feu rigoureux et l'application de politiques de groupe permettent de restreindre l'accès et de renforcer la sécurité du système.





Notre application utilise le port 1664 pour connecter le serveur JAVA et le client C++. Ce port est bien fermé dans le pare-feu de Windows. Aucune attaque est donc possible, on peut aussi vérifier qu'il est impossible de s'y connecter avec la commande telnet après l'installation de son package :

Marche à suivre pour installer telnet :

- 1) Ouvrir le powershell
- 2) Exécuter cette commande : `Install-WindowsFeature -Name Telnet-Client`
- 3) `Install-WindowsFeature -Name Telnet-Server`

Vous pouvez à présent utiliser la commande telnet : `telnet [options] [port]`

Exemple de commande :

- `telnet 8.8.8.8 443` (serveur google pour le https)
- `telnet 8.8.8.8 80` (serveur google pour le http)

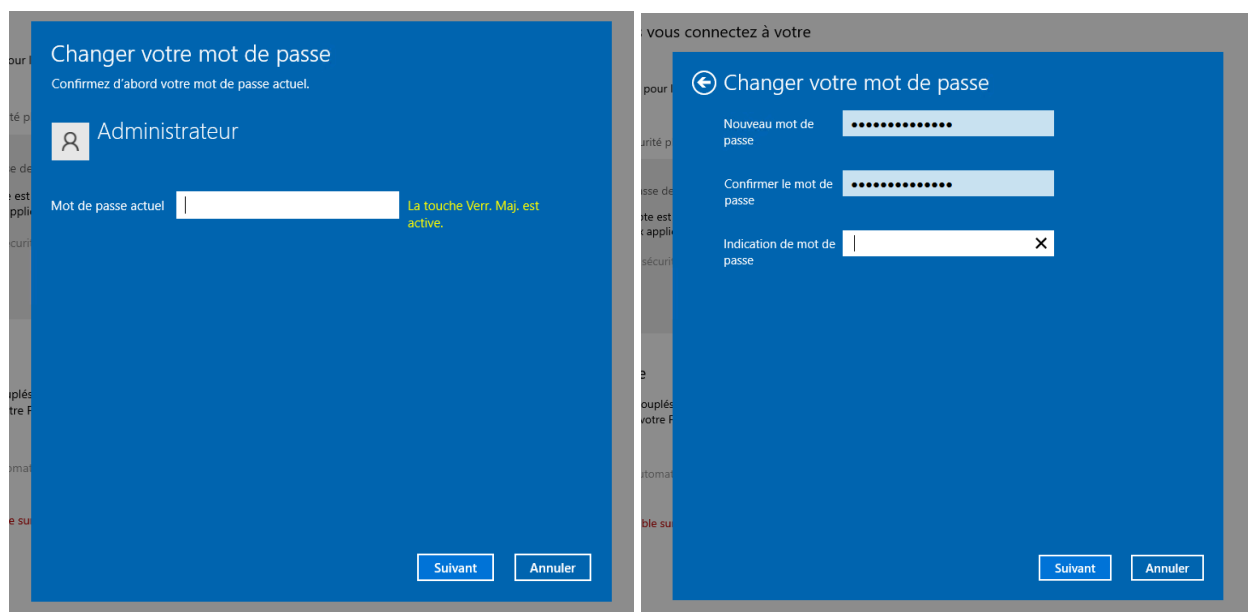
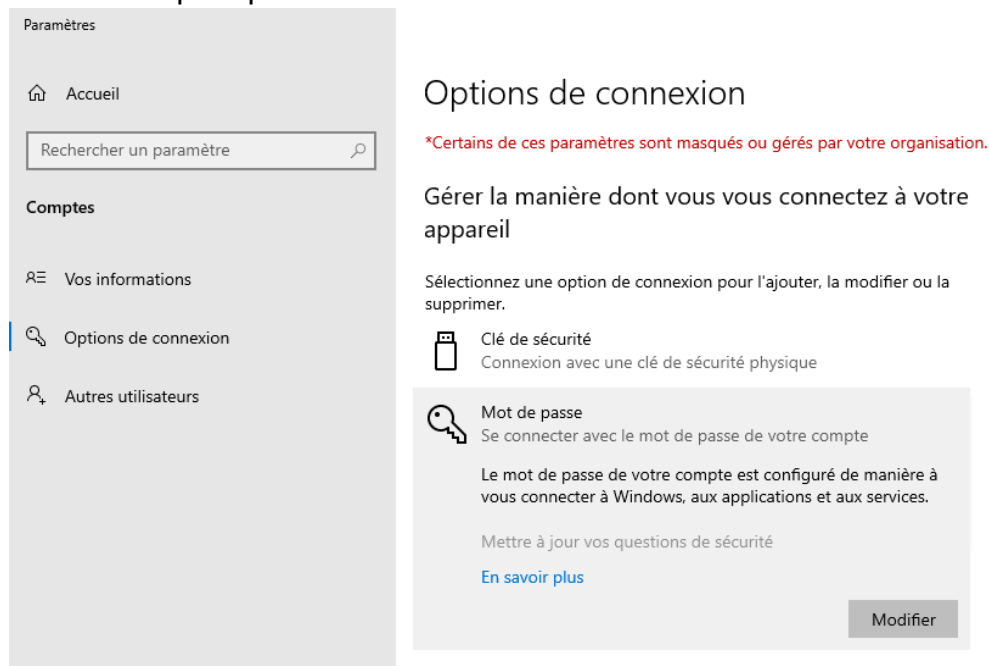
Résultat de la commande telnet avec l'adresse ip du serveur et le port de notre application :

```
PS C:\Users\Administrateur> telnet 100.74.7.88 1664
Connexion à 100.74.7.88...Impossible d'ouvrir une connexion à l'hôte, sur le port 1664: Échec lors de la connexion
PS C:\Users\Administrateur> _
```

On remarque qu'il est impossible de s'y connecter car le port est fermé.

Nous avons aussi modifié le mot de passe de base du serveur.

Voici les étapes que nous avons suivi :



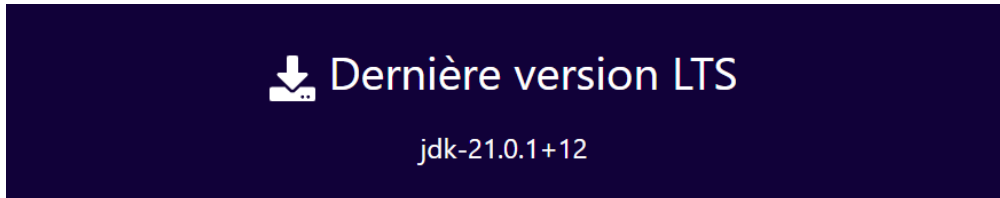
Nous avons opté pour un mot de passe cochant toutes les cases pour avoir la meilleure sécurité possible c'est-à-dire :

- Une ou plusieurs majuscule
- + de 8 caractères
- des caractères spéciaux
- des chiffres

## Étapes de Configuration (Environnement Windows) :

### 1. Installer le Java Development Kit (JDK) :

- Téléchargez et installez JDK à partir de <https://adoptium.net/fr/>



### 2. Installer Java :

- Suivez les instructions d'installation disponibles sur <https://www.oracle.com/java/technologies/downloads/#jdk21-windows>

x64 Installer

163.82 MB

[https://download.oracle.com/java/21/latest/jdk-21\\_windows-x64\\_bin.exe](https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe) (sha256)

### 3. Installer G++ via MINGW :

- Téléchargez et installez G++ depuis <https://sourceforge.net/projects/mingw/files/Installer/>



### 4. Installer Visual Studio Code :

- Téléchargez et installez Visual Studio Code depuis <https://code.visualstudio.com/download>



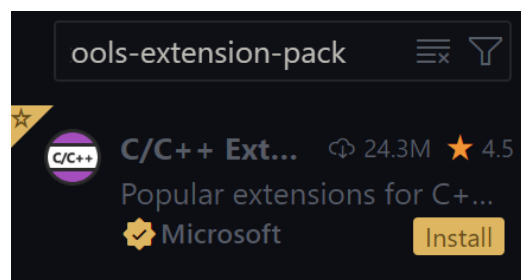
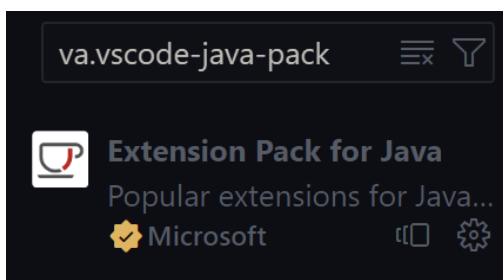


## 5. Installer les extensions Java/C++ pour Visual Studio Code :

- Ouvrez Visual Studio Code
- Recherchez et installez l'extension java :
  1. Une fois Visual Studio Code ouvert, recherchez ce symbole sur le côté gauche de l'écran :



2. Puis, tapez ceci « **vscjava.vscode-java-pack** » et « **ms-vscode.cpptools-extension-pack** » dans la barre de recherche et installez ces deux extensions:
- 3.



## 6. Installer Git :


- Téléchargez et installez Git à partir de <http://git-scm.com/download/win>.

**64-bit Git for Windows Setup.**

- Appuyez simplement sur « **Next** » jusqu'à ce que l'installation démarre, et appuyez sur « **Finish** » une fois cette dernière terminée.

## 7. Clônage du projet avec Git :

- Une fois Gît installé, ouvrez « **Git CMD** »

 Administrateur : Git CMD

```
C:\Users\Administrateur>_
```

- Tapez dans la fenêtre de commandes « **cd Documents** »

```
C:\Users\Administrateur>cd Documents
```

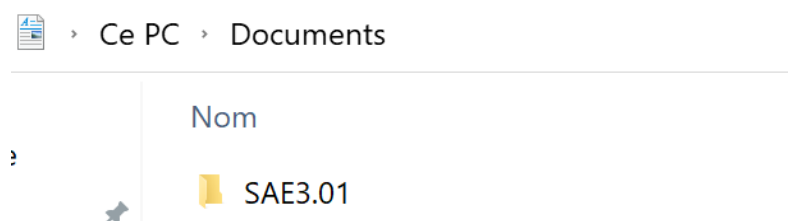
```
C:\Users\Administrateur\Documents>_
```

- Puis, tapez « **git clone https://github.com/DorianBucc/SAE3.01.git** »

```
C:\Users\Administrateur\Documents>git clone https://github.com/DorianBucc/SAE3.01.git
Cloning into 'SAE3.01'...
remote: Enumerating objects: 89, done.
remote: Counting objects: 100% (89/89), done.
remote: Compressing objects: 100% (52/52), done.
Receiving objects: 100% (89/89), 124.47 KiB | 8.89 MiB/s, done.
Resolving deltas: 100% (35/35), done.
C:\Users\Administrateur\Documents>
```

## 8. Vérification du clonage :

- Vous pouvez désormais ouvrir votre explorateur de fichiers, et aller dans « **Documents** ». Vous devriez avoir désormais ceci :



## 9. Vous avez le choix :

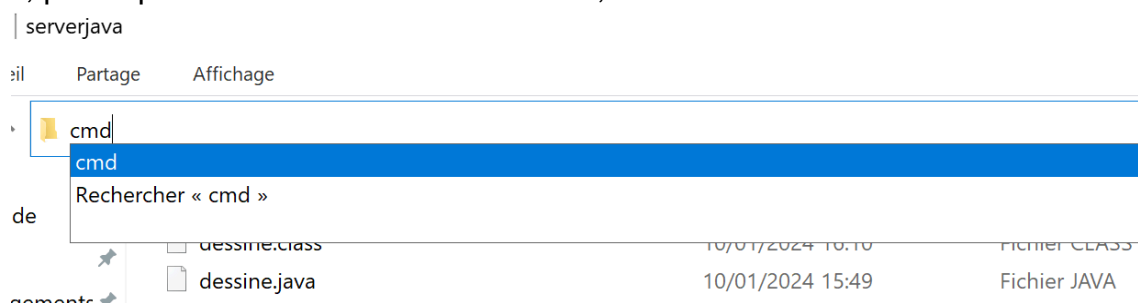
- Maintenant que vous avez toutes les installations nécessaires au fonctionnement du projet, vous avez 2 possibilités :
  - Vous pouvez naviguer dans le dossier affiché ci-dessus, aller dans « **projet** », puis double cliquez sur le fichier « **a.exe** » afin que le programme s'exécute automatiquement

OU (non obligatoire)

- Vous pouvez démarrer le serveur et le client manuellement, en suivant les étapes ci-dessous :

## 10. Mise à jour des fichiers Java (serveur) :

- Naviguez dans le dossier comme précédemment, allez dans « **src** », puis tapez dans la barre de recherches, la commande « **cmd** »



- Une fenêtre de commande devrait s'ouvrir.

```
C:\Users\Administrateur\Documents\SAE3.01\src\serverjava>
```

### 11. Lancer le serveur :

- Exécutez la commande « **javac \*.java** » pour mettre tous les fichiers à jour avec votre version java

```
C:\Users\Administrateur\Documents\SAE3.01\src\serverjava>javac *.java  
C:\Users\Administrateur\Documents\SAE3.01\src\serverjava>_
```

- Puis, exécutez la commande « **java -classpath .. serverjava.TestServerDraw** »

```
\SAE3.01\src\serverjava>java -classpath .. serverjava.TestServerDraw
```

### 12. Vérifier le bon fonctionnement du serveur :

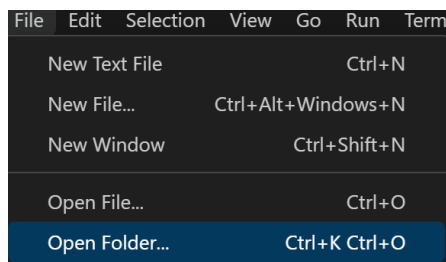
- Si tout est correct, une fenêtre Java s'ouvrira, et la console affichera "Serveur JavaDraw prêt"

```
C:\Users\Administrateur\Documents\SAE3.01\src\serverjava>java -classpath .. serverjava.TestServerDraw  
Server JavaDraw prêt : ServerSocket[addr=0.0.0.0/0.0.0.0,localport=1664]
```

### 13. Configuration des tests dans Visual Studio Code :

#### a. Ouvrir le Projet dans Visual Studio Code :

- Lancez Visual Studio Code
- Accédez au dossier « clientc++ » en ouvrant le projet

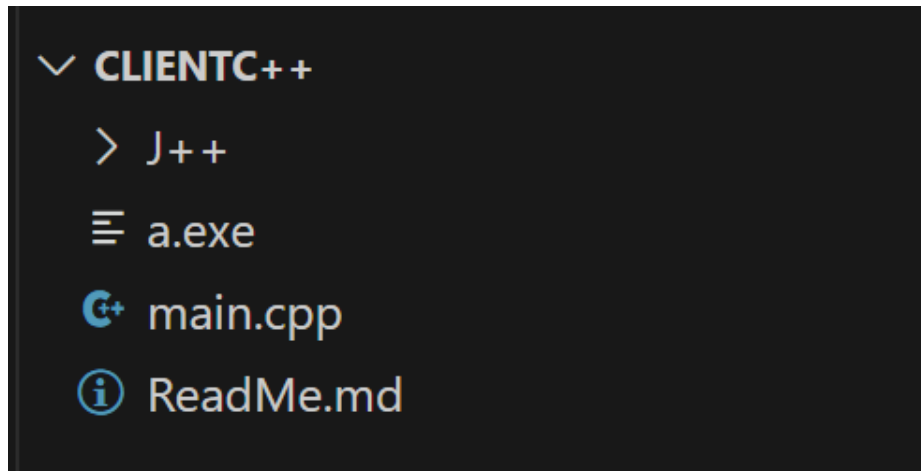


- Puis, appuyez sur « sélectionner le dossier » une fois avoir cliqué sur « clientc++ ».

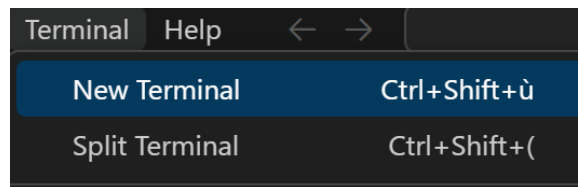


**b. Exécuter le programme :**

- Une fois le dossier ouvert, vous devriez avoir quelque chose comme ceci :



- Ouvrez désormais une console en ouvrant un nouveau terminal, comme ci-dessous :



**c. Configurer les fichiers :**

- Une fois le terminal ouvert, tapez « **g++ \*.cpp .\J++\\*.cpp -lwsck32** », si vous avez correctement suivi les étapes d'installation de G++ via une vidéo d'aide, vous ne devriez pas avoir de problème lors de l'exécution de cette commande. Si toutefois, vous avez une erreur, revoyez cette vidéo (en anglais), qui explique comment bien installer g++ :

[https://www.youtube.com/watch?v=sXW2VLrQ3Bs&t=270s&ab\\_channel=ProgrammingKnowledge2](https://www.youtube.com/watch?v=sXW2VLrQ3Bs&t=270s&ab_channel=ProgrammingKnowledge2)

```
\SAE3.01\src\clientc++> g++ *.cpp .\J++\*.cpp -lwsck32
```

**d. Configurer l'adresse du serveur :**

- Tapez ensuite, « **.\a.exe** »

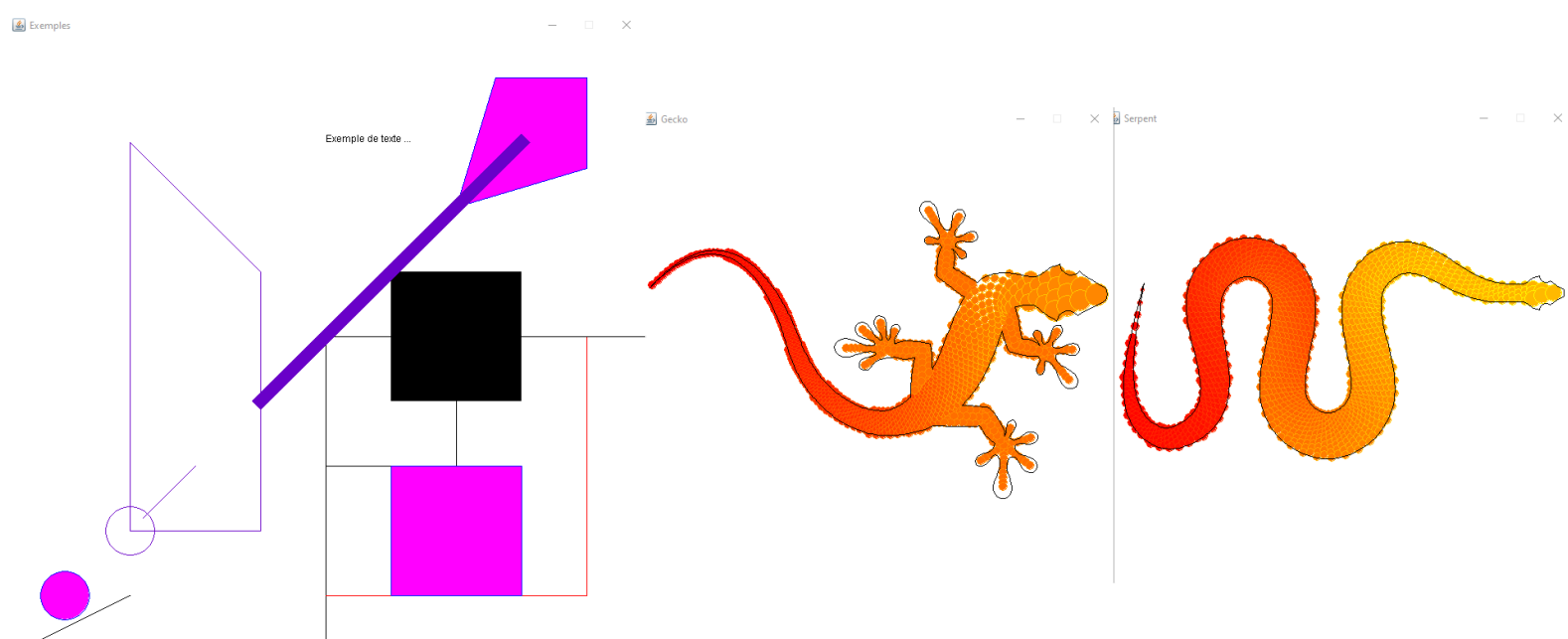
```
PS C:\Users\Administrateur\Documents\SAE3.01\src\clientc++> .\a.exe
```

#### e. Validation de la connexion :

- Si toutes les étapes sont correctes, le « cmd » vu précédemment, affichera ceci :

```
C:\Users\Administrateur\Documents\SAE3.01\src\serverjava>java -classpath .. serverjava.TestServerDraw
Server JavaDraw prêt : ServerSocket[addr=0.0.0.0/0.0.0.0,localport=1664]
Connexion numero 1 réussie
Le clientb n°1, Requete = window,800,800,Exemple
Le clientb n°1, Requete = droite,0,800,160,720
Le clientb n°1, Requete = droite,640,160,320,480,15,108,0,200,255
Le clientb n°1, Requete = rectangle,400,400,160,160
Le clientb n°1, Requete = rectangle,400,400,320,320,255,0,0,255
Le clientb n°1, Requete = rectangle,400,400,480,480
Le clientb n°1, Requete = rectangle,480,320,160,160,fill
Le clientb n°1, Requete = rectangle,480,560,160,160,255,0,255,255,0,0,255,255,fill
Le clientb n°1, Requete = cercle,130,610,60,60,108,0,200,255
Le clientb n°1, Requete = cercle,50,690,60,60,255,0,255,255,0,0,255,255,fill
Le clientb n°1, Requete = polygone,4,160,640,320,640,320,160,160,108,0,200,255
Le clientb n°1, Requete = polygone,4,720,80,720,192,560,240,608,80,255,0,255,255,0,0,255,255,fill
Le clientb n°1, Requete = text,Texte ...,400,160
Le clientb n°1, Requete = droite,176,624,240,560,108,0,200,255
```

- Et, une fenêtre Java devrait également s'ouvrir, affichant un ensemble de formes géométriques avec de la couleur, une fenêtre gecko, puis une fenêtre serpent une fois le gecko terminé :



#### f. Finalisation :

- Si jusqu'à maintenant, toutes les étapes ont correctement été effectuées, vous devriez avoir une fenêtre Java avec un ensemble de formes et de couleurs, deux images d'animaux (un gecko et un serpent), ainsi que leurs dégradés, comme l'image ci-dessus.
- Félicitations, vous avez désormais créé un client-serveur C++/Java sur votre machine, capable d'interpréter et dessiner.

## **Conclusion et perspectives :**

Par rapport aux objectifs initiaux qui ont été fixés, nous sommes pour le moment arrivés à un résultat qui nous satisfait amplement, bien que l'on puisse faire quelques améliorations telles que l'ajout d'outils, d'assets ou de presets pour faciliter et améliorer encore plus les interactions utilisateur. Pour ce qui est de l'infrastructure, cela a été une opportunité énorme pour nous d'apprendre plus en détail sur ce qui se fait en environnement professionnel, et cela a été très instructif, étant donné que nous estimions au départ cette partie comme mineure par rapport à l'ensemble du projet.

De plus, une nouvelle perspective tournée vers la sécurité a dû être prise en compte. C'est une composante du projet à laquelle on se confronte moins durant le cursus scolaire, mais qui ici était quelque chose de très important, puisque le projet a vocation à être utilisé par d'autres programmeurs. Il fallait qu'il y ait le moins de failles possibles afin de garantir aux futurs utilisateurs une confiance envers le logiciel.

Ce projet tutoré nous a montré les différentes composantes et problématiques d'un projet en entreprise pouvant se présenter dans notre future vie professionnelle. En plus d'avoir développé des connaissances et compétences tout au long du parcours, avec des technologies et des concepts qui nous seront sûrement très utiles par la suite. Des attentions toutes particulières ont donc été portées sur la sécurité et les techniques de développement afin d'optimiser et implémenter de futures améliorations au code.

Tous ces points énumérés ne peuvent que nous apporter un plus dans notre avenir professionnel. Pour terminer, nous concluons sur le fait que ce projet nous a grandement intéressé et que l'on est fier du chemin parcouru ainsi que du résultat final.

Et qu'il a grandement contribué à nous faire grandir en tant que développeur informatique.