Livrable 2

Travail présenté à Anthony Deschênes
Génie logiciel orienté objet

réalisé par
Sébastien Dubé(537 153 241), Antoine Morin(537 089 065), Adam Côté(537 145 036), Kamran Charles Nayebi(537 150 671), Louis-Etienne Messier(537 131 157) ;

13 octobre 2024

Table des matières

Table des figures

1 Classes de conception

1.1 Diagramme de classes de conception

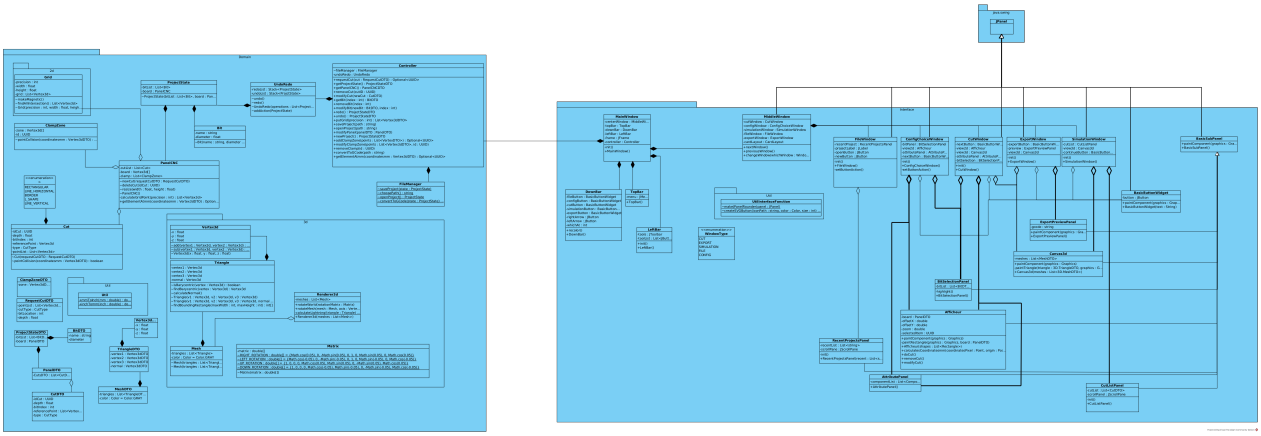

FIGURE 1 – Diagramme de classes de conception

1.2 Texte explicatif du diagramme de classes de conception

Il est possible de résumer l'architecture de l'application *FreeCarve* en 3 parties majeures : **le domaine**, **le contrôleur de Larman** (qui est inclus dans le domaine) et **l'interface graphique**.

1.2.1 Le domaine

Premièrement, **le domaine** constitue le regroupement des fonctionnalités concrètes du programme. Il est composé de classes pour encapsuler des concepts physiques comme la classe **Cut** (qui peut prendre différentes formes énoncées dans l'enum **CutType**), la classe **Bit**, la classe **ClampZone** ou la classe **PanelCNC**.

Le domaine possède aussi une classe **ProjectState** qui encapsule l'état actuel du projet, une composante nécessaire pour permettre la sauvegarde, la création et le chargement de projet gérée par la classe **FileManager**. La classe **UndoRedo** quant à elle gère le processus de retour en arrière et de retour en avant en guardant deux piles de variation du **ProjectState**.

Le domaine se charge aussi du calcul et du rendu de propriétés 2D comme la classe **Grid**.

Par ailleurs, **le domaine** contient aussi toutes les fonctionnalités et les classes en lien avec le rendu 3D : **Vertex3d**, **Triangle**, **Mesh**, **Matrix**, **Renderer3D**. Le coeur du complexe rendu 3D se produit donc dans le **domaine** et se fait simplement afficher dans **l'interface**.

Finalement, **le domaine** contient aussi le sous package **Util** qui renferme la classe utilitaire **Util**. La classe **Util** contient des fonctions généralement utiles comme des fonctions de conversions d'unité de mesure.

1.2.2 Le contrôleur de Larman

Deuxièmement, **le contrôleur de Larman** joue le rôle du médiateur entre **le domaine** et **l'interface graphique**. Il se charge de forcer la compartimentalisation des deux parties tout en leur permettant de communiquer à l'aide de DTO sécuritaire et de fonctions.

Le contrôleur de Larman est composé de d'attributs privés reliés au domaine comme le **FileManager** et le **UndoRedo** et possède une interface de fonction qui permet à **l'interface** de recevoir des DTO du **domaine**. Les DTO pouvant être reçus par **l'interface** sont : la classe **ClampZoneDTO**, la classe **ProjectStateDTO**, la classe **BitDTO**, la classe **ProjectStateDTO**, la classe **PanelDTO**, la classe **CutDTO**, la classe **VertexD3dTO**, la classe **TriangleDTO** et la classe **MeshDTO**

1.2.3 L'interface graphique

Troisièmement, **l'interface graphique** se charge d'afficher toutes les informations pertinentes au bon fonctionnement de *FreeCarve* par l'utilisateur. L'élément central du UI est la classe **MainWindow** qui possède un *JFrame* et qui encapsule la fenêtre de l'application. **MainWindow** est composé de :

- **Controlleur**, l'interface pour interagir avec le domaine et recevoir les DTO.
- **TopBar**, le *JMenu* ancré au haut de la fenêtre donnant accès à des fonctionnalités générales.
- **LeftBar**, le *JToolBar* ancré à la gauche de la fenêtre donnant accès à des outils pour éditer les coupes.
- **DownBar**, la timeline de *JButton* ancré au bas de la fenêtre donnant accès à la naviguation entre les menus des étapes à accomplir pour accomplir les coupes.
- **MiddleWindow**, la classe *JPanel* prenant le restant de l'espace de la fenêtre principale (le rectangle en haut à droite) pour afficher les fenêtres du menu sélectionné dans la **DownBar**.

Le **MiddleWindow** fait varier son contenu dépendemment du menu sélectionné dans le **DownBar**. Le **MiddleWindow** affiche en alternance :

- **FileWindow**, un *JPanel* encapsulant le fenêtre de gestion de projet initiale permettant de créer de nouveaux projets, d'ouvrir d'anciens projets et d'ouvrir des projets récents
- **ConfigChoiceWindow**, un *JPanel*, encapsulant la fenêtre de création du panneau de découpe et encapsulant le fenêtre de sélection des outils pour les coupes.
- **CutWindow**, un *JPanel*, encapsulant la fenêtre fonctionnelle de découpe du panneau, donnant accès à tous les outils, et toutes les découpes souhaités pour modifier le panneau.
- **SimulationWindow**, un *JPanel*, encapsulant un aperçu des coupes et du résultat final du panneau résultant des coupes.
- **ExportWindow**, un *JPanel*, encapsulant tous les paramètres d'exportations et un aperçu du *GCODE* résultant.

Chacune des *Window* que le **MainContentPane** affiche est elle-même composée de modules avec leur propre fonctionnnalité. Ces modules héritent de **BasicSubPanel** et sont :

— **RecentProjectsPane**, module présent seulement dans la **FileWindow**. Il affiche la liste de tous les projets récents et un aperçu du panneau.
— **AttributePane**, module très flexible qui affiche les paramètres relatifs au menu en place : positions x, y et profondeur des coupes pour le menu de découpage, ou encore les dimensions du panneau de départ souhaité.
— **BitSelectionPanel**, module qui affiche et permet de modifier les 12 outils de découpe que la CNC peut utiliser.
— **Afficheur**, module en charge de dessiner tous éléments non encapsulés de base dans Swing (donc la vue 2D du panneau).
— **CutListPanel**, module en charge d'offrir une liste modifiable de toutes les coupes effectuées jusqu'à maintenant sur le panneau.
— **Canvas3D**, module en charge d'afficher un aperçu 3D de la coupe du panneau.
— **SimulationPane**, module en charge d'afficher une animation des coupes selon la liste des coupes de **CultListPanel**.
— **ExportPreviewPane**, module en charge de permettre à l'utilisateur de modifier ses paramètres de *GCode* et de voir un aperçu du fichier exporté.

L'interface graphique possède aussi quelques classes utilitaires comme :
— **UtilInterfaceFonction**, une classe utilitaire pour faciliter la création de boutons.
— **BasicButtonWidget**, une template de class dont tous les boutons héritent.

2 Architecture logique

2.1 Diagramme de packages

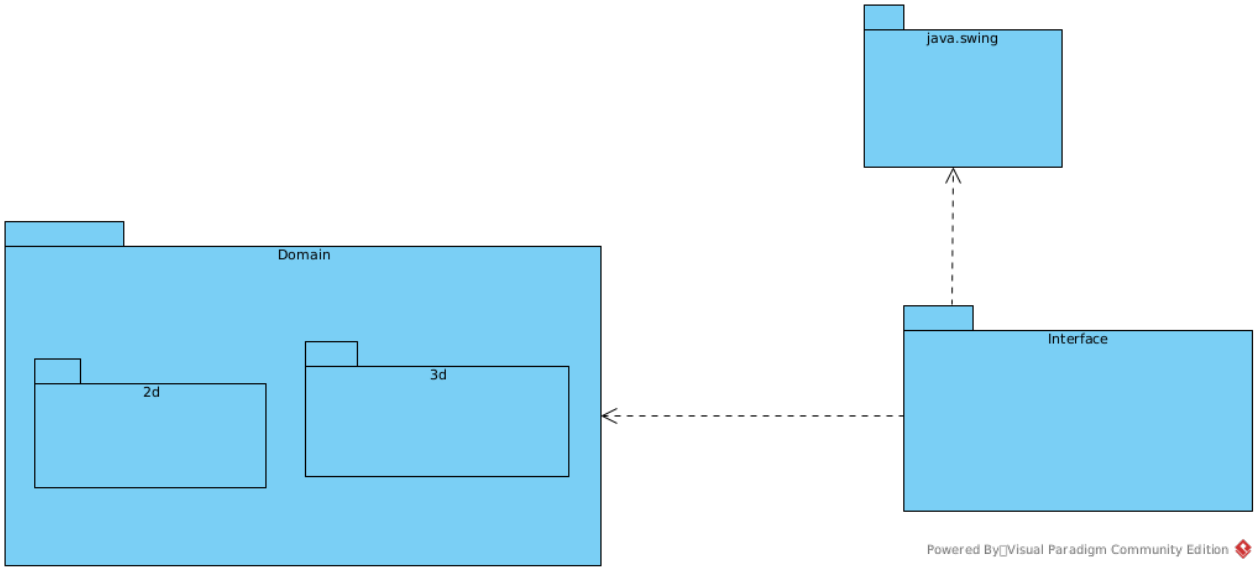

FIGURE 2 – Diagramme de packages

2.2 Texte explicatif du diagramme des packages

Le package **Interface** dépend du domaine (**Domain**) et de la librairie d'affichage graphique (**java.swing**). Il représente la couche d'affichage avec laquelle l'utilisateur va interagir et délègue toutes les tâches de validation et de préservation de l'état au domaine. Il se charge de faire la colle entre les besoins spécifiques de l'application et la librairie d'affichage **java.swing**.

Le package **Interface** se charge de valider des requêtes qui lui sont faites, de maintenir l'état de l'application et d'effectuer les opérations et calculs nécessaires à son bon fonctionnement.

Le package **java.swing** se charge d'offrir une grande variété de composantes pré-faites afin de pouvoir composer une interface graphique et se charge aussi de générer et distribuer les événements de souris et de clavier au package **Interface**.

Le sous-package **Domain.3d** se charge de garder l'état d'un monde 3d et de faire tous les calculs nécessaires à son affichage sauf le rendering lui-même, car il est dépendant de l'interface.

Idem pour le sous-package **Domain.2d** : il se charge de garder l'état d'un monde 2d et de faire tous les calculs nécessaires à son affichage sauf le rendering lui-même, car il est dépendant de l'interface.

3 Diagrammes de séquence de conception (*DSC*)

3.1 Déterminer l'élément sur lequel a lieu un clic de souris

3.1.1 Diagramme #1

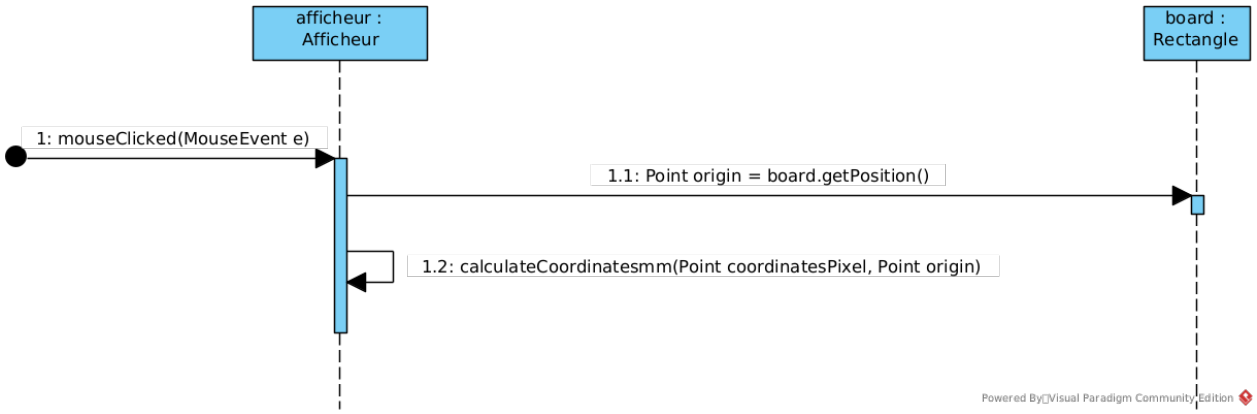

FIGURE 3 – DSC d'un clic de souris

L'événement mouse clicked de l'interface MouseListener de Java est appelé lors d'un clic de souris sur le JPannel afficheur. Nous pouvons récupérer la position de la souris en pixel relativement au JPanel sous forme d'un objet Point. Le point (0, 0) se trouvera toujours dans le coin inférieur gauche du board, donc le programme doit récupérer cette information pour effectuer son calcul. L'objet afficheur a comme variables privées des décalages en X et Y (offsetX et offsetY), la hauteur du JPanel ainsi qu'une variable zoom qui sert à gérer l'affichage des éléments dans le JPanel avec le zoom en fonction de la position de la souris. Ces variables sont utilisées dans la fonction calculateCoordinatesmm pour déterminer, à l'aide d'une équation mathématique, les coordonnées en mm à partir des coordonnées en pixel. Les coordonnées en mm sont donc calculées directement dans afficheur

3.1.2 Diagramme #2

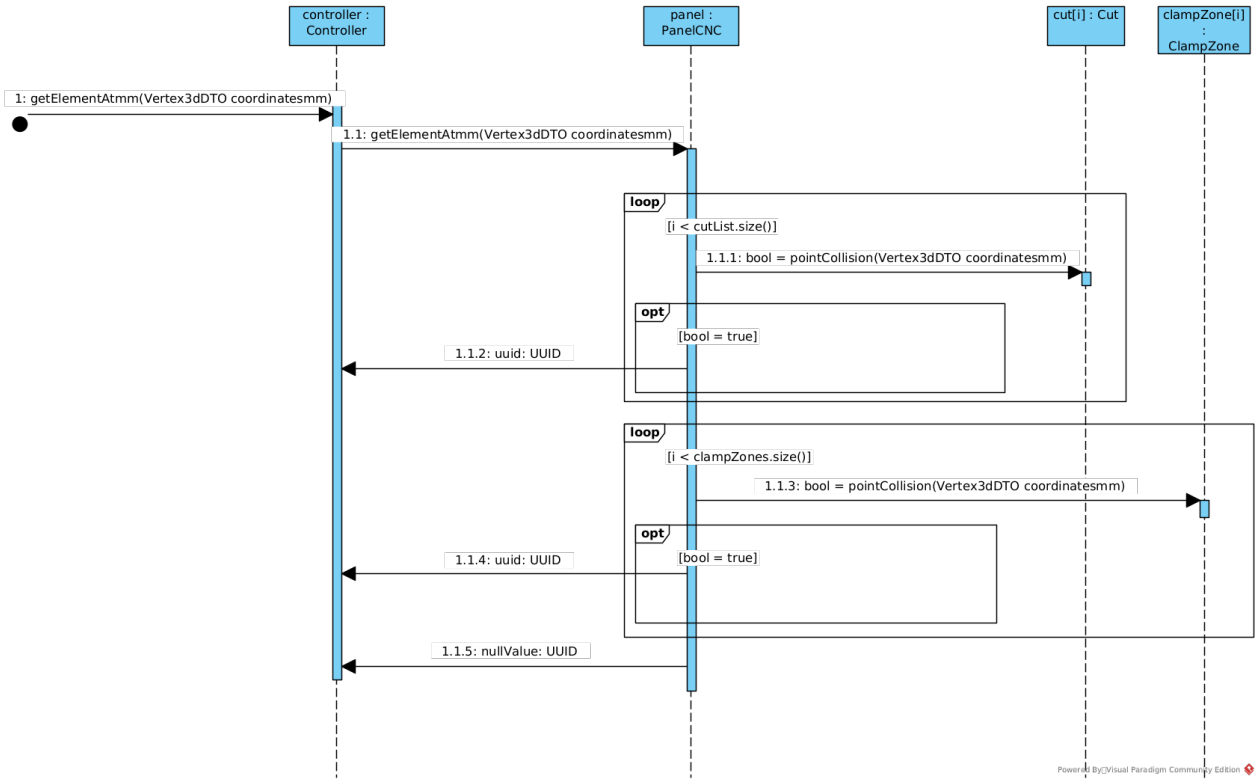

FIGURE 4 – DSC pour déterminer l'élément à la position en mm

Le contrôleur va recevoir un point de coordonnées en millimètres et va déterminer quel élement du panneau est en collision avec ce point. Le panneau va recevoir le point en mm et itérer à travers sa liste de coupes et sa liste de zones interdites pour trouver un élément qui est en collision avec le point. La fonction pointCollision vérifie à l'aide d'opérations mathématiques si un point est en collision avec l'élément en fonction de sa position, largeur et hauteur. Dès qu'un élément est en collision avec le point, son UUID est retourné pour pouvoir l'identifier par la suite. Si aucun élément est en collision avec le point, la fonction retourne null.

3.2 Ajout d'une coupe rectangulaire

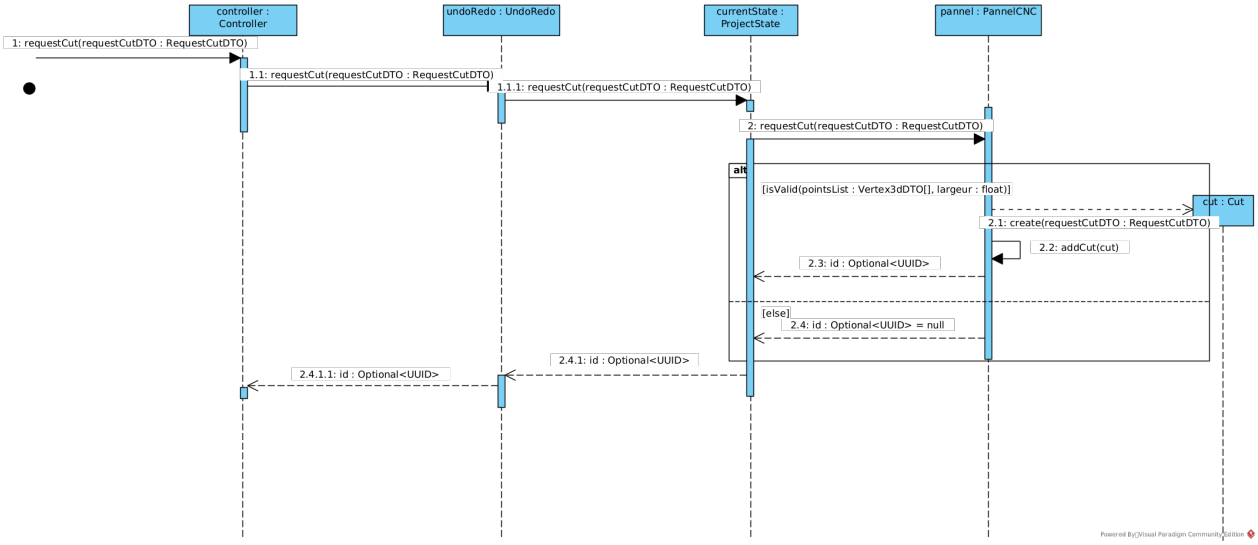

FIGURE 5 – DSC de l'ajout d'une coupe rectangulaire

Afin de pouvoir créer une coupe rectangulaire, l'utilisateur devra choisir le type de la coupe, l'index de la mèche qu'il souhaite utilisé, la profondeur de la coupe, un point où la coupe va commencer et le point finale. Ces informations seront alors transmises au contrôleur, sous la forme d'un objet transférable afin de réaliser l'action souhaité. Tout d'abord, le contrôleur déterminera qu'il doit effectuer une coupe parce que l'interface appellera la méthode requestCut() présente dans le contrôleur. Après avoir reçu la requête, le contrôleur va alors distribué la tâche à un instance de la classe UndoRedo possédant une instance de ProjectState. Cette instance de ProjectState enverra la requête à une instance de PanelCNC afin qu'elle gère la création de la coupe. Pour qu'une coupe soit valide, on doit effectuer une vérification que les points la constituants n'entrent pas en collision avec une zone de coupe interdite. Si la coupe est valide, nous allons la créer, l'ajouter à la liste des coupes du panneau et renvoyer un numéro d'identifiant unique qui correspondra à cette coupe. Si elle n'est pas valide, on renvoie un identifiant équivalent à nulle pour dire qu'aucune coupe n'ait été créée.

3.3 Affichage de la vue

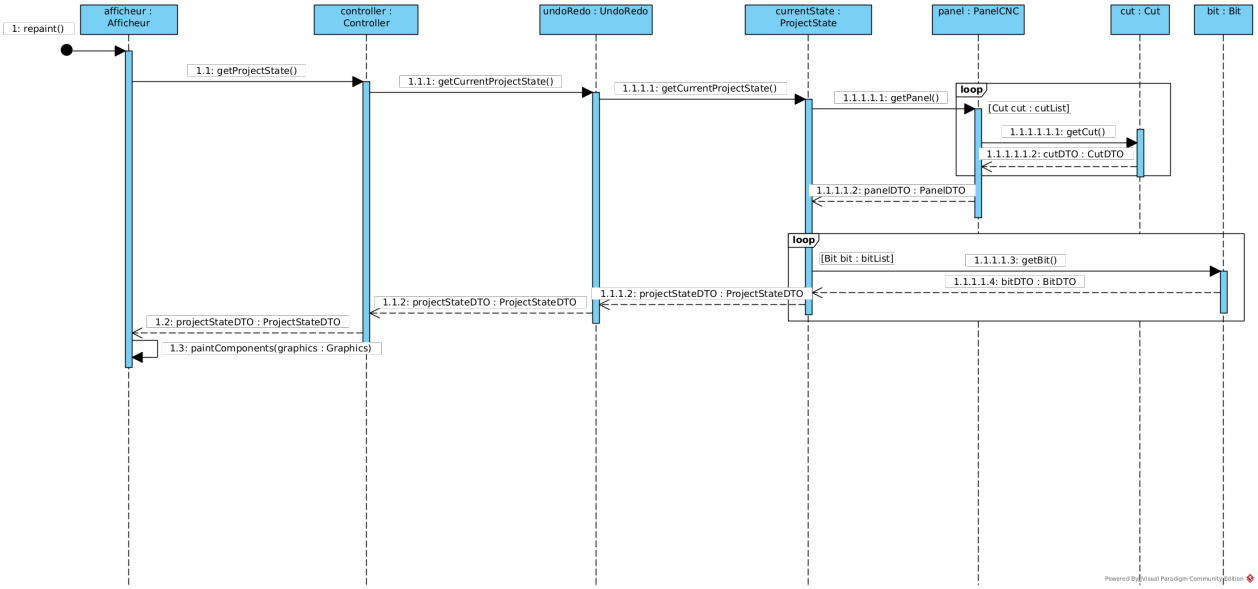

FIGURE 6 – DSC de l'affichage de la vue

Lorsqu'on veut rafraichir l'affichage de la vue après une modification, on effectue un appel à la méthode repaint() présente dans l'afficheur. Cette méthode va effectuer un appel au contrôleur pour avoir l'état du projet afin de récupérer la liste des coupes actuelles et la liste des mèches présentes. Afin de récupérer ces informations, le contrôleur doit demandé à l'instance de la classe UndoRedo afin qu'elle demande à son instance de ProjectState les informations. Cette instance va alors propager la demande afin d'avoir les informations nécessaires pour constituer un objet de la classe ProjectStateDTO et transmettre cette information plus haut pour l'acheminer à l'interface. Une fois que l'interface reçoit l'information, afficheur va alors appelé sa méthode paintRectangle avec un object de type Graphics qui va s'occuper de redessiner l'interface avec les nouveaux changements.

4 Diagramme de Gantt

Le diagramme de Gantt suivant permet de structurer la conception du projet en associant des tâches spécifiques à accomplir à chaque itération (période de 2 semaines). Le Gantt contient aussi les dates clés de remise des livrables.

FIGURE 7 – Liste des itérations 1 à 4

FIGURE 8 – Liste des dernières itérations et des dates de remise

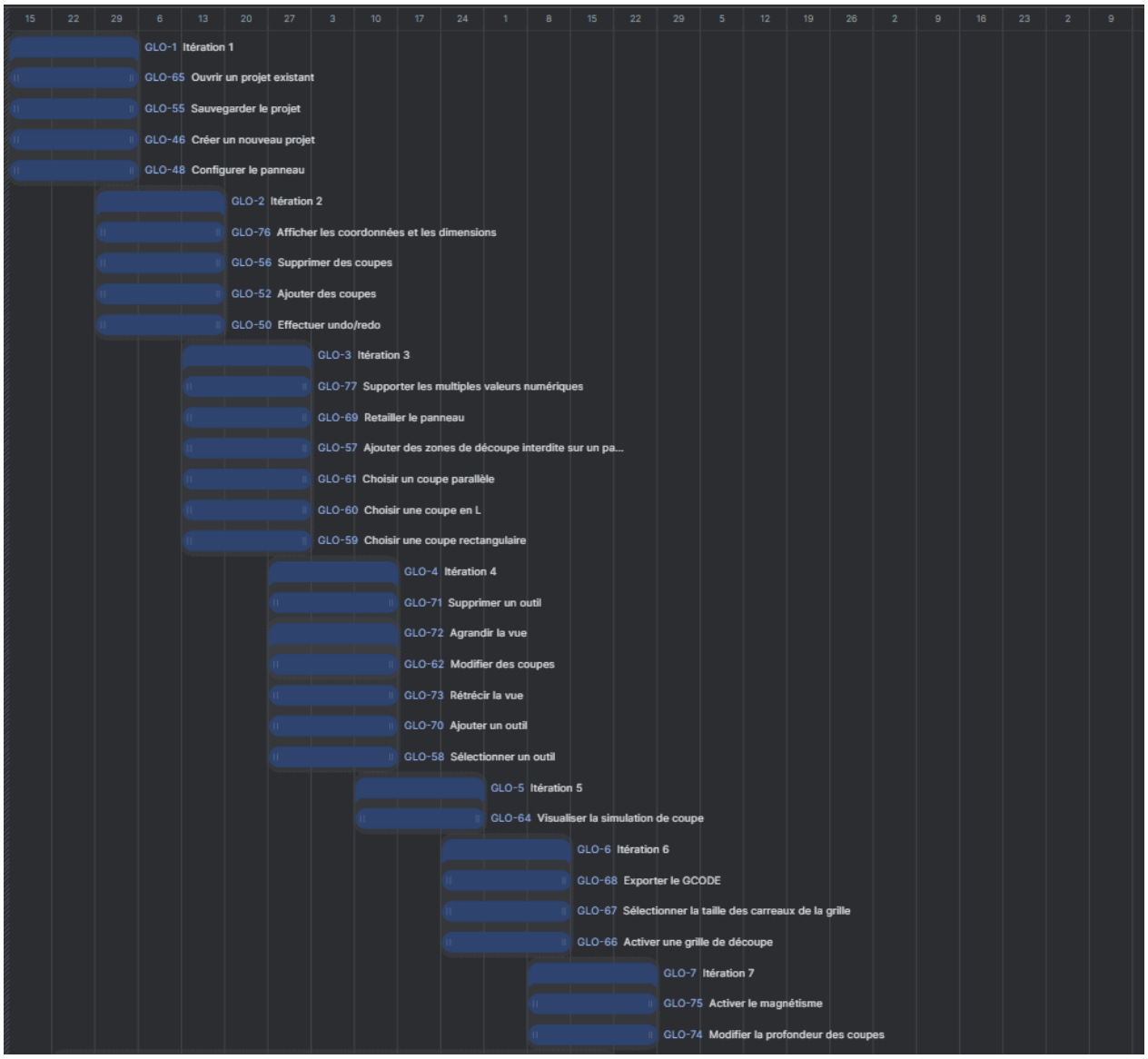

FIGURE 9 – Ligne du temps des itérations

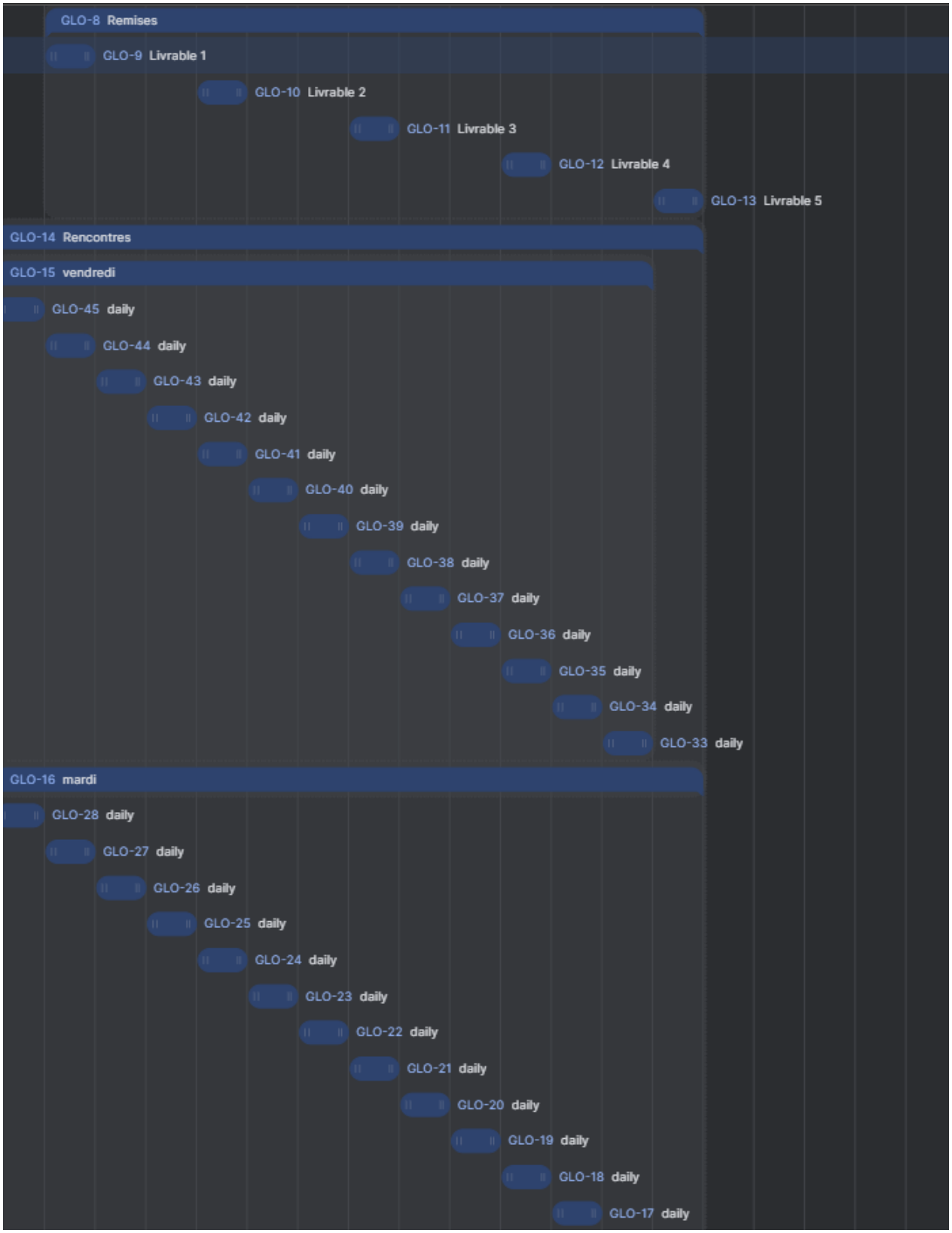

FIGURE 10 – Ligne du temps des remises et des rencontres

5 Contribution des membres de l'équipe

Nom des membres	**Contribution**
Louis-Etienne Messier	Mise à jour du diagramme de Gantt et conception du *DCC*/texte explicatif du *DCC*
Adam Côté	Réalisation du *DCC* et code du squelette du UI
Kamran Charles Nayebi	Réalisation du diagramme de packages et formatage des git, pdf et jar pour la remise
Antoine Morin	Réalisation des diagrammes de séquence conceptuel 3.2 et 3.3
Sébastien Dubé	Réalisation des diagrammes de séquence conceptuel 3.1.1 et 3.1.2