Projet de session :
Développement d'une application *VirtuBois*

Livrable 2

présenté à

M. Jonathan Gaudreault

par

Équipe 02

matricule	*nom*	*signature*
111 174 511	Martine Deschênes	
111 175 430	Jordan Mayhue	
111 073 283	Yoan Chamberland	
111 178 727	Gabriel St-Pierre	

Université Laval

26 février 2019

Historique des versions		
version	*date*	*description*
#0	23 janvier 2019	Création du document et début de la réalisation du projet
#1	5 février 2019	Remise du Livrable 1
#2	6 février 2019	Début de la réalisation du Livrable 2
final	26 février 2019	Remise du Livrable 2

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Diagramme de classe de conception

1.1 Diagramme de classe de conception

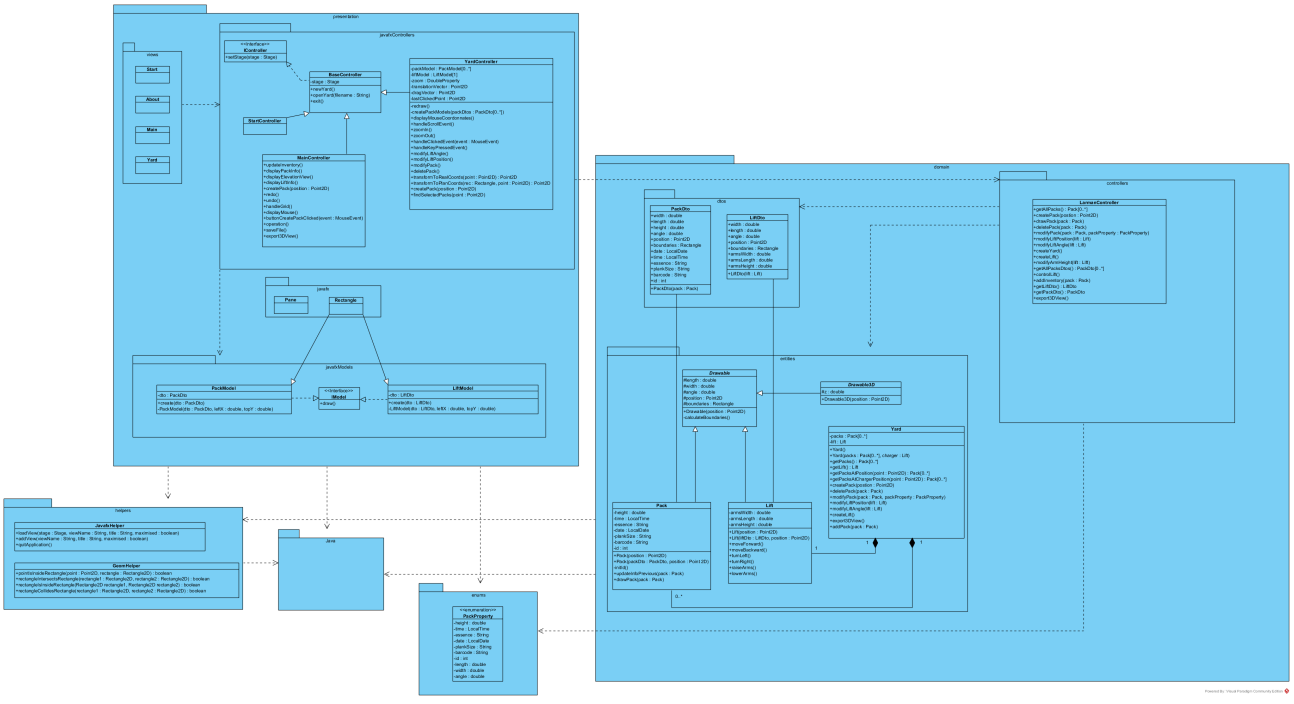


FIGURE 1.1 – Diagramme de classe de conception

1.2 Explication du digramme de classe de conception

1.2.1 Explication des classes du domaine

1.2.1.1 Contrôleur de Larman (LarmanController)

Le contrôleur de Larman est la passerelle entre l'interface utilisateur et le domaine de l'application. Ce contrôleur gère, à l'aide de diverses méthodes, toutes les interactions entre ces deux entités. Du côté de l'interface utilisateur (*presentation*), plusieurs contrôleurs JavaFX interagissent avec ce contrôleur pour faire le lien.

1.2.1.2 Dtos

Les attributs de chacune des classes (Pack et Lift) servent à contenir les données qui seront échangées entre l'interface et le domaine. Le contrôleur de Larman peut accéder à ces Dtos grâce aux méthodes *getLiftDto* et *getAllPacksDtos*. Les entités PackDto et LiftDto contiennent chacun un constructeur PackDto et LiftDto qui crée l'objet avec les données actuelles qu'elles contiennent.

1.2.1.3 Entités (Entities)

Les entités représentent les objets dans le domaine qui seront affichés à l'utilisateur. En effet, dans ce paquet, nous retrouvons les entités cour, paquet et chargeuse que l'utilisateur voit sur l'interface. Ce paquet contient aussi des entités de dessins afin de créer en 2D ou en 3D les entités de ce paquet. Finalement, il est possible de remarquer que chacune des entités sont liées entre elles. En effet, la cour contient les paquets et la chargeuse qui héritent eux-même de *Drawable*. *Drawable3D* hérite lui aussi de *Drawable*.

Pack
Cette classe représente les paquets de bois qui sont contenus dans la cour. Chaque paquet a pour attributs une hauteur bien définie, une date et une heure de création, une essence spécifique, la taille des planches contenues dans le paquet, un code à barre entré par l'utilisateur ainsi qu'un identifiant unique. Elle possède aussi des attributs qu'elle hérite de sa classe de base *drawable* permettant de lui spécifier une longueur, une largeur, un angle, une position dans la cour à bois ainsi qu'une frontière. On peut accéder aux attributs et les modifier à partir de *getters* et de *setters* qui seront implantés. Il est également à noter que le constructeur de la classe ne prend en paramètre qu'une coordonnée en x et y dans l'espace de la cours à bois. Cela est dû au fait que, par défaut, tous les attributs à l'exception de l'identifiant et de la date et l'heure seront initialisés aux valeurs du dernier paquet créé. Pour ce qui est de l'identifiant d'un paquet, il sera initialisé à partir de la méthode *initId*.

Lift
Cette classe gère tout ce qui touche à la chargeuse. Il ne peut exister plus d'une instance de cette classe, car il n'y a qu'une chargeuse dans la cour. Tout comme la classe Pack, elle hérite de la classe de base *drawable*. La chargeuse possède pour attributs la longueur de ses bras, la largeur de ses bras et la hauteur de ceux-ci. Elle hérite également des attributs *drawable*, soit une longueur, une largeur, un angle, une position dans la cour à bois ainsi qu'une frontière délimitant l'objet dans l'espace. On peut accéder aux attributs et les modifier à partir de *getters* et de *setters* qui seront implantés. Les méthodes *moveForward*, *moveBackward*, *turnLeft* et *turnRight*, permettront de déplacer la chargeuse dans la cour. Les méthodes *raiseArms* et *lowerArms* permettront, quant à elles, de modifier la hauteur des bras de la chargeuse.

Yard
Cette classe représente une cour à bois. Elle est composée d'une instance de *Lift* et d'instances de *Pack*. La classe implémente des méthodes permettant d'obtenir une liste des paquets présents dans la cour, mais également d'obtenir une liste des paquets présents à une position précise dans la cour. C'est aussi cette classe qui est en charge de créer les paquets, d'en modifier les différents attributs selon les instructions du contrôleur de Larman, mais également de créer une chargeuse et d'en modifier certaines propriétés.

Drawable
Drawable est une interface des entités qui soutient l'affichage des paquets et de la chargeuse à bois. En effet, *Pack* et *Lift*, deux des classes de ce paquet héritent de certains attributs essentiels au dessin dans l'interface des objets qui les composent. Il est possible d'y retrouver entres autres, la position, la longueur, l'angle, la largeur, etc. De plus, cette classe contient une méthode permettant de connaître les arêtes du rectangle qui compose le paquet. Elle contient aussi la méthode permettant de dessiner les objets.

Drawable3D

Drawable3D est très semblable à l'interface précédemment mentionnée. En effet, *Drawable3D*, qui hérite de *Drawable* permet tout simplement, grâce à sa méthode et à son attribut, de dessiner les objets en 3D lors de l'affichage en 3D de la cour à bois.

1.2.2 Helpers

Le paquet *Helpers* du diagramme contient toutes les fonctions utilitaires simples qui sont utilisées par le domaine et par l'interface utilisateur pour accomplir des tâches mathématiques ou pour ce qui à trait à la vue. Dans le premier cas, c'est la classe *GeomHelper* qui contient toutes les méthodes mathématiques pour, par exemple, savoir si les coordonnées d'un point sont dans un rectangle, ce qui remplacera l'algorithme implanté dans la partie 4 du rapport, si deux rectangles se touchent, etc. Les *JavafxHelpers*, quant à eux, aideront à l'affichage de l'application.

1.2.3 Java

Ce paquet est très simple, il s'agit du langage de programmation qui sera utilisé par le domaine, les *helpers* et l'interface utilisateur.

1.2.4 Enums

Le paquet *enums* contient la classe *PackProperty* de type *enumerate* qui énumère les attributs d'un paquet. Cette classe est utilisée à la fois par le contrôleur de Larman, dans le domaine, lors de la modification d'un des attributs d'un paquet ainsi que par l'interface utilisateur pour l'affichage à l'écran des attributs d'un paquet pour modification.

1.2.5 Presentation

1.2.5.1 views

Start

La classe *Start* du paquet *view* sert à l'affichage de la première fenêtre lorsque l'application est lancée. À son lancement, une fenêtre *pop-up* apparait à l'utilisateur pour qu'il crée une nouvelle cour, qu'il en charge une ou qu'il quitte tout simplement.

About

La classe *About* contient tant qu'à elle, un guide d'utilisation pour l'application.

Main

La classe *Main* de ce paquet contient toute l'information relative à la cour à bois. Elle contient tous les boutons pour le fonctionnement de la cour (grille magnétique, *undo*, *redo*, créer paquet, etc. Elle contient aussi toutes les zones d'informations pertinentes ainsi que la cour à bois où il est possible de voir la chargeuse en action et la disposition des paquets de bois.

1.2.5.2 javafxControllers

IController
Ce contrôleur de JavaFX de type *Interface* fera guise d'interface pour l'application. C'est le contrôleur qui gère l'affichage et dont le contrôleur *BaseController* dépend pour fonctionner correctement. En d'autres mots, il met l'interface en place.

StartController
Tant qu'à lui, le *StartController* ne contient aucune méthode en soi. Comme JavaFX est utilisé, le contrôleur est créé par défaut et est utilisé pour l'affichage de la fenêtre.

MainController
Le *MainController* gère tout ce qui touche l'affichage de l'information pertinente et des boutons. Ce contrôleur s'occupe de mettre à jour l'inventaire, lorsqu'un changement est fait à un paquet, il affiche toutes les informations sur le paquet et de la vue d'élévation lorsqu'un paquet est sélectionné et des informations de la chargeuse et de la souris en tout temps. En ce qui a trait aux boutons, il contrôle leur fonctionnement. Il s'assure que *undo, redo*, Créer Paquet et Grille Magnétique fonctionne de la bonne façon.

BaseController
Ce contrôleur, qui prend en argument l'objet créé par le *IControlller* s'assure du bon fonctionnement de la création, de l'ouverture ainsi que la fermeture de l'application. C'est de ce contrôleur qu'hérite le contrôleur *Main, Start et JavaFX*

YardController
Le *YardController* agit comme "afficheur de la vue en plan de la cour". Il contiendra l'ensemble des modèles JavaFX et sera responsable de la gestion des événements et de l'affichage de la cour. Ainsi, il communiquera très souvent avec le contrôleur de Larman afin d'obtenir la liste des paquets (sous format de DTOs) permettant de recréer chacun des modèles. Le reste de l'affichage de la fenêtre principale sera géré par le *MainController* afin d'alléger les classes.

1.2.5.3 javafx

Pane
Pane de JavaFX est l'équivalent de JPanel en Swing. Il s'agit d'une classe qui s'occupe de la disposition des éléments dans l'application. Il existe plusieurs types de Pane qui offre une expérience différente à l'utilisateur.

Rectangle
La classe *Rectangle* définie un rectangle avec une position et une forme prédifinie par l'utilisateur. Ici, il s'agit des paquets qui seront crées par l'utilisateur qui sont des objets de la classe *Rectangle*.

1.2.5.4 JavafxModels

Les modèles JavaFX (JavaFXModels), aussi appelés *objets JavaFX*, sont des classes dérivées de certains *Nodes* de la librairie. Ce sont des objets affichables et qui possèdent en plus leurs DTOs associés.

IModel
Le *IModel* est le gestionnaire de l'affichage des *Models*. Il agit comme interface pour les modèles.

PackModel
Le *PackModel* contient toutes les informations des paquets afin de facilement pouvoir les afficher à l'écran. Ce modèle contient donc les DTOs de la classe comme attribut. Il contient la méthode publique *create* qui crée le modèle et la méthode privée PackModel qui gère de façon individuelle ses propres évènements, allégeant ainsi grandement les contrôleurs JavaFX.

LiftModel
Le LiftModel contient toutes les informations de la chargeuse présente afin de facilité l'affichage à l'écran. Ce modèle contient donc les DTOs de la classe comme attribut. Il contient la méthode publique *create* qui crée le modèle de cet objet et la méthode privée *LiftModel* qui gère de façon individuelle ses propres évènements, allégeant ainsi grandement les contrôleurs JavaFX.

Chapitre 2

Architecture Logique

2.1 Diagramme de packages

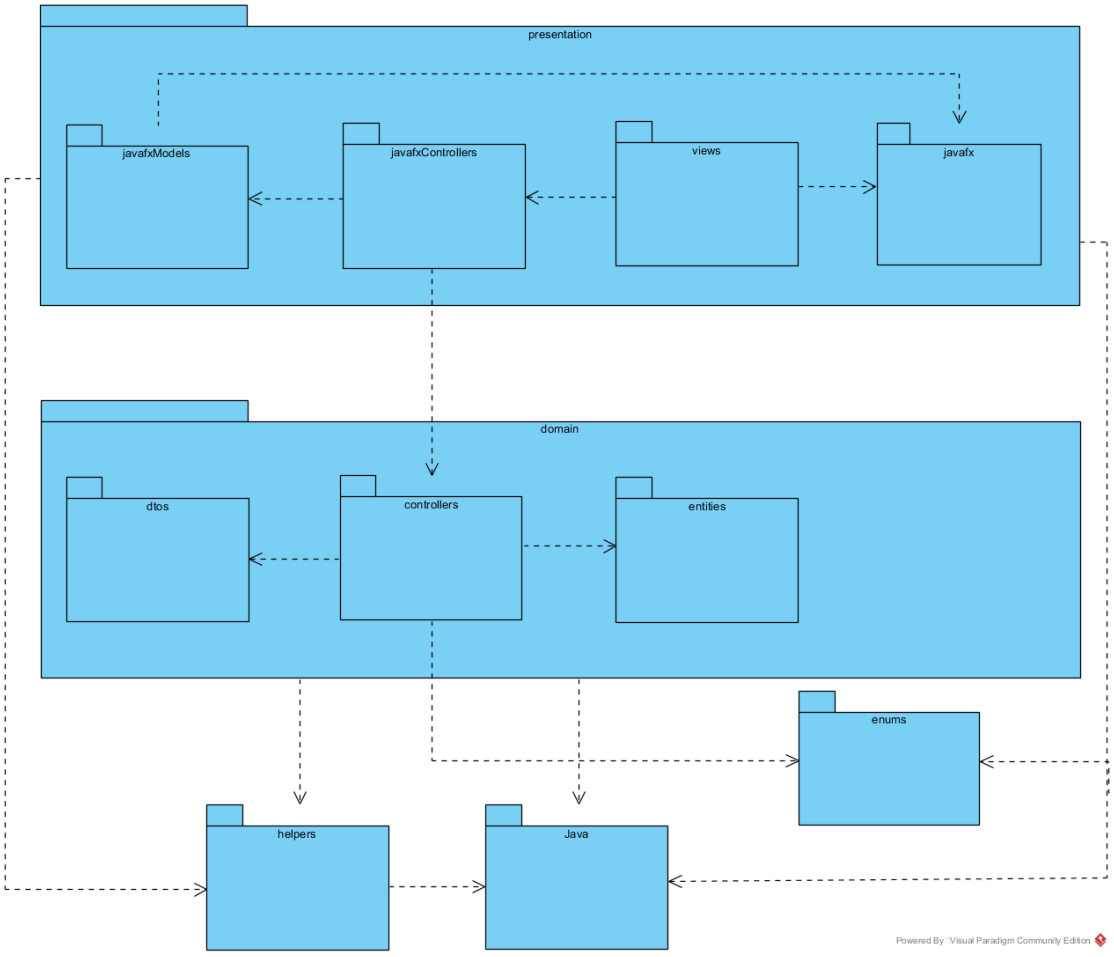


FIGURE 2.1 – Diagramme de packages

2.2 Texte explicatif

Notre architecture sera composée de deux couches principales : la couche de présentation et la couche du domaine. Celles-ci utiliseront fortement les différentes librairies de Java ainsi que les classes utilitaires (*helpers*) et les énumérations (*enums*).

2.2.1 Couche présentation

La couche de présentation contient tout ce qui concerne l'affichage. Elle contient d'abord certaines vues FXML (documents de style XML) qui décrivent le contenu principal des fenêtres (boutons, textes, etc.). Tous ces éléments visuels sont des objets appartenant à la librairie JavaFX. La plupart de ces vues sont rattachées à un contrôleur JavaFX dédié permettant de gérer l'affichage dynamique ainsi que les contrôles et les événements. Ces contrôleurs contiennent des modèles JavaFX (objets basés sur des éléments dessinables de la libraries JavaFX), basés sur les informations reçues du domaine, afin de gérer l'affichage des entités. Ce sont également ces contrôleurs qui communiqueront avec le contrôleur de Larman, présent dans le domaine, afin de demander les modifications nécessaires.

2.2.2 Couche domaine

La couche du domaine contient l'ensemble de la logique applicative. Elle commence par le contrôleur de Larman, dont le rôle est de recevoir l'ensemble des demandes de modifications venant de la couche de présentation. Ce contrôleur redirige ensuite les appels vers les entités du domaine (les « vrais » objets). Finalement, celui-ci retournera des DTOs (Data Tranfert Objects) afin de transmettre les informations des entités sans possibilité de directement accéder au domaine.

Chapitre 3

Diagrammes de séquence de conception

3.1 Déterminer les coordonnées d'un clic de souris et l'empilement à sélectionner

3.1.1 Diagramme

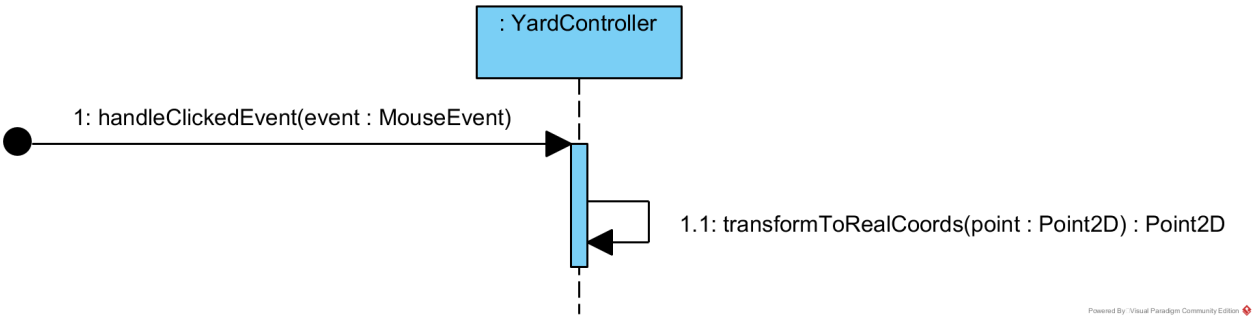


FIGURE 3.1 – Diagramme de séquence de conception - Déterminer les coordonnées d'un clic de souris

3.1.1.1 Texte explicatif

La séquence commence avec un clic de souris dans la fenêtre de la cour à bois qui sera pris en charge par l'entremise de la méthode *handleClickedEvent* du *YardController*. Cette méthode se chargera d'appeler une autre méthode définie dans le *YardController*, du nom de *transformToRealCoords*, qui est expliquée plus loin dans cette section. Grossièrement, cette dernière méthode se chargera de convertir les coordonnées d'affichage (celles du clic) en coordonnées réelles (celles du domaine).

La méthode *transformToRealCoords* prend en argument la position de la souris capté par

la méthode *handleClickedEvent*. Il est alors possible de retrouver la position du clic en pixel relative au coin supérieur gauche de la fenêtre avec des fonctions JavaFx. On doit d'abord inverser la coordonnée verticale puisque que celle-ci croit en descendant. En appliquant une translation aux coordonnées on peux recentrer le point central pour le positionner au milieu de la fenêtre. On doit aussi appliquer un vecteur de déplacement au point qui contient la position du centre de l'écran au moment du clic. Par exemple, si on décide de faire un *drag* sur l'écran, on devra stocker le vecteur total de ce déplacement et l'appliquer à la coordonnée cliquée. Ensuite, cette même fonction multipliera deux facteurs à cette position : un pour convertir les pixels en mètre et l'autre pour tenir compte du niveau de zoom de la fenêtre. Une fois ces transformations effectuées, la fonction retourne les coordonnées qui sont considérées réelles.

3.1.2 Diagramme

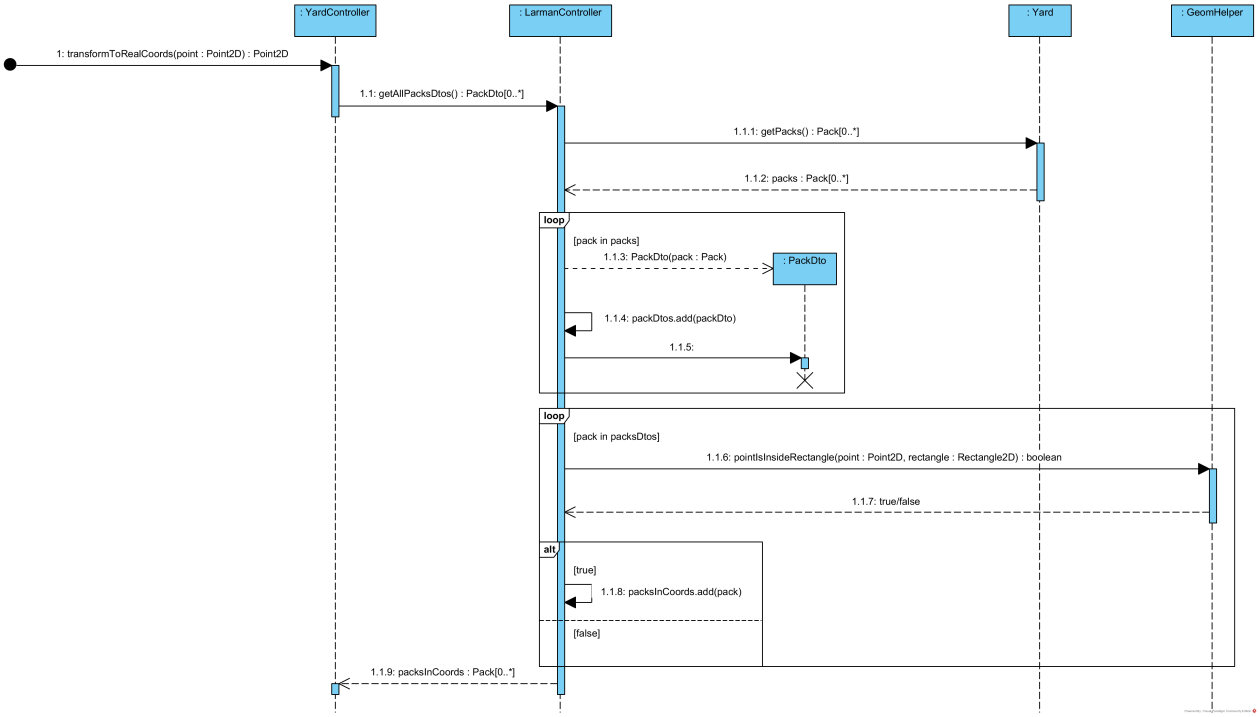


FIGURE 3.2 – Diagramme de conception - Déterminer les paquets qui constituent l'empilement selon un clic de souris

3.1.2.1 Texte explicatif

Cette séquence se produit tout de suite après la séquence *déterminer les coordonnées d'un clic de souris*, son but étant de déterminer s'il y a des paquets présents à la coordonnée du clic. Suite à la réception des coordonnées réelles de la cour par la méthode *transformToRealCoords* du *YardController*, celui-ci fait appel au *LarmanController* qui retournera une liste contenant avec tous les paquets contenus dans cours dans le bon ordre d'affichage. Ces paquets seront ensuite transformés en DTO pour être envoyés à la couche présentation plus tard. Par la suite, un tri sera effectué pour déterminer quels DTOs sont à garder à l'aide de la fonction *pointIsInsideRectangle* du package *helpers*. Lorsque le tri sera terminé, on renvoie la liste de DTOs qui connectent avec le point au *YardController*.

3.2 Créer un nouveau paquet

3.2.1 Diagramme

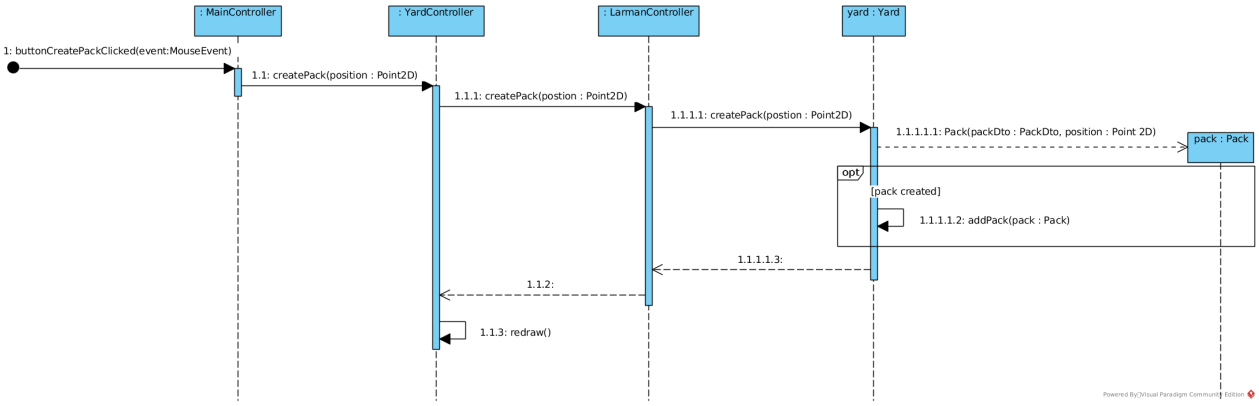


FIGURE 3.3 – Diagramme de séquence de conception - Créer un paquet

3.2.1.1 Texte explicatif

Création du paquet
La création du paquet débute lorsque le *MainController* reçoit le clic de la souris et appelle ainsi *buttonCreatedPackClicked*. Une suite d'appels est ensuite faite à la fonction *createPack* du prochain contrôleur jusqu'à l'entité yard de la classe *Yard*. Cet méthode prend en argument la position du paquet, qui correspond à l'endroit où l'utilisateur à cliquer pour la création du paquet.
Par la suite, la classe *Yard* envoie un appel au constructeur de la classe *Pack* pour qu'il crée un paquet avec des paramètres par défaut, contenu dans le *packDto*, et la postion du clic. Si un paquet a déjà été créé, les paramètres par défaut seront ceux du dernier paquet qui a été créé. Sinon, des paramètres par défaut initiaux sont attitrés à celui-ci.
Une fois le paquet crée sans erreur, le classe *Yard* ajoutera ce paquet dans la liste des paquets que contient la cour.

Affichage du paquet
Une fois l'objet pack créé, le *Yard* et le *LarmanController* envoie des messages vides à leur prédécesseurs. Le *YardController* réaffichera la cour, avec *redraw*, avec le nouveau paquet qui a été créé.

3.3 Afficher les éléments de la cour

3.3.1 Diagramme

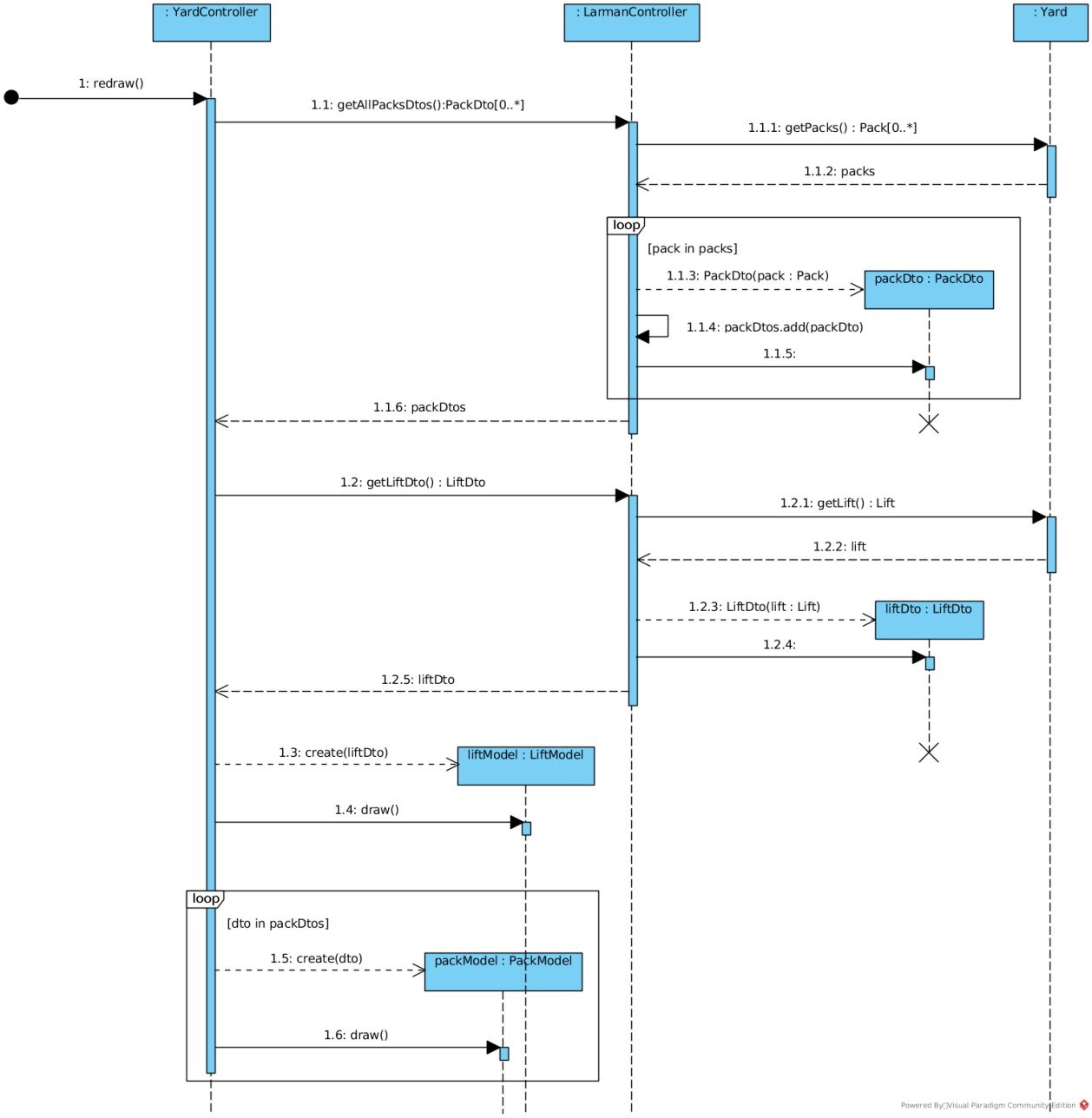


FIGURE 3.4 – Diagramme de séquence de conception - Afficher la cour

3.3.1.1 Texte explicatif

Lors d'une modification d'affichage comme le changement de propriété, la redimension, le zoom, bouger un paquet, etc., la fonction *redraw* du contrôleur JavaFX *YardController* est appelée afin de redessiner l'ensemble de la cours. Pour accomplir une telle tâche, le contrôleur JavaFX demande au contrôleur de Larman les DTOs représentant la chargeuse (Lift) et les paquets (Pack). Pour ce faire, le contôleur obtient d'abord les objets de la cour, puis les transforment en leur DTOs respectifs. Une fois les DTOs obtenus, le contrôleur JavaFX *YardController* supprimera toutes ses références aux modèles JavaFX, puis les recréera un à un en suivant l'ordre de la liste. L'ensemble des PackModel et le LiftModel ainsi générés seront par la suite dessinés et affichés à l'écran.

Chapitre 4

Pseudo-code d'un algorithme

Algorithm 1 Déterminer si un point x,y (en mètres) se trouve à l'intérieur d'un périmètre défini par un rectangle

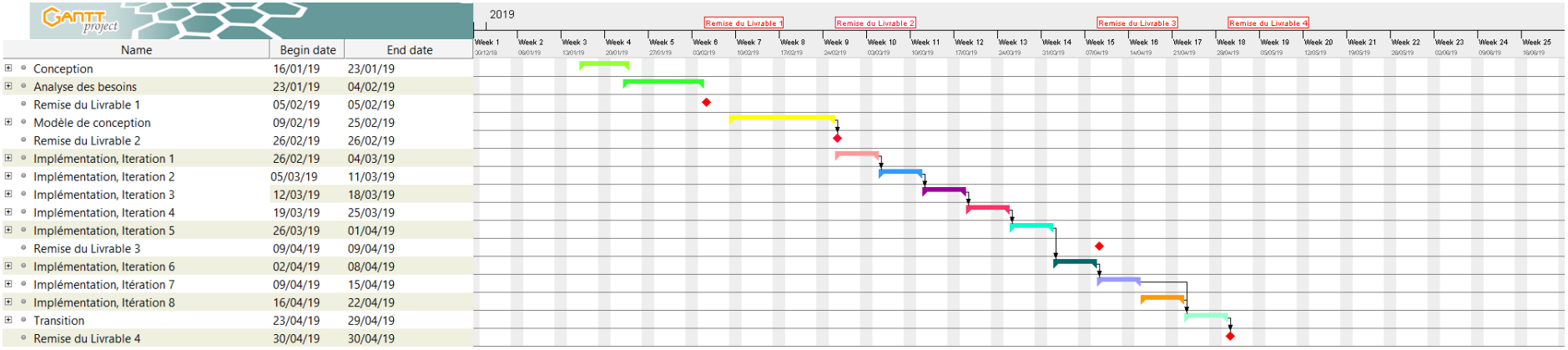
```
procédure COORDONNEEINTERIEUR(coordonneeXY)
n ← 0
sommetRectangle ← Package.getCoordonneesSommet()
for coordonneesSommet in sommetRectangle do
  if n = sommetRectangle.size() - 1 then
    areaTemp ← helpers.areaTriangle(coordonneeXY, coordonnesSommet, coordonnees-
    Sommet[0]
  else
    areaTemp ← helpers.areaTriangle(coordonneeXY, coordonnesSommet, coordonnees-
    Sommet[n+1]
  end if
  areaTotal ← areaTotal + areaTemp
end for
if areaTotal est égal à Package.getArea() then

  return vrai
else

  return faux
end if
fin procédure
```

Cet algorithme fait appel à la méthode *getCoordonneesSommet* de la classe *Pack*. Cette méthode retourne les sommets, qui forme le rectangle vu de haut, d'un paquet. Pour se faire, une méthode (coordonneesRectangle) de la classe prend en argument les coordonnées centrales (x_2, y_2) ainsi que la longueur, la largueur et l'angle d'un paquet, calcule les coordonnées

et les ajoute au conteneur sommetRectangle. Ainsi, lorsque que l'on déplace la chargeuse, il sera possible de savoir en tout temps si les bras de celle-ci se trouvent à l'intérieur d'un paquet pour un éventuel déplacement. Sinon, cet algorithme devient utile lorsque l'utilisateur crée un paquet pour afficher la vue d'élévation, pour déplacer un ou des paquets, pour le supprimer, etc.
Cependant, JavaFX offre une une méthode (*rectangle.contains(point)*) qui remplacera cet algorithme dans le code qui sera produit.

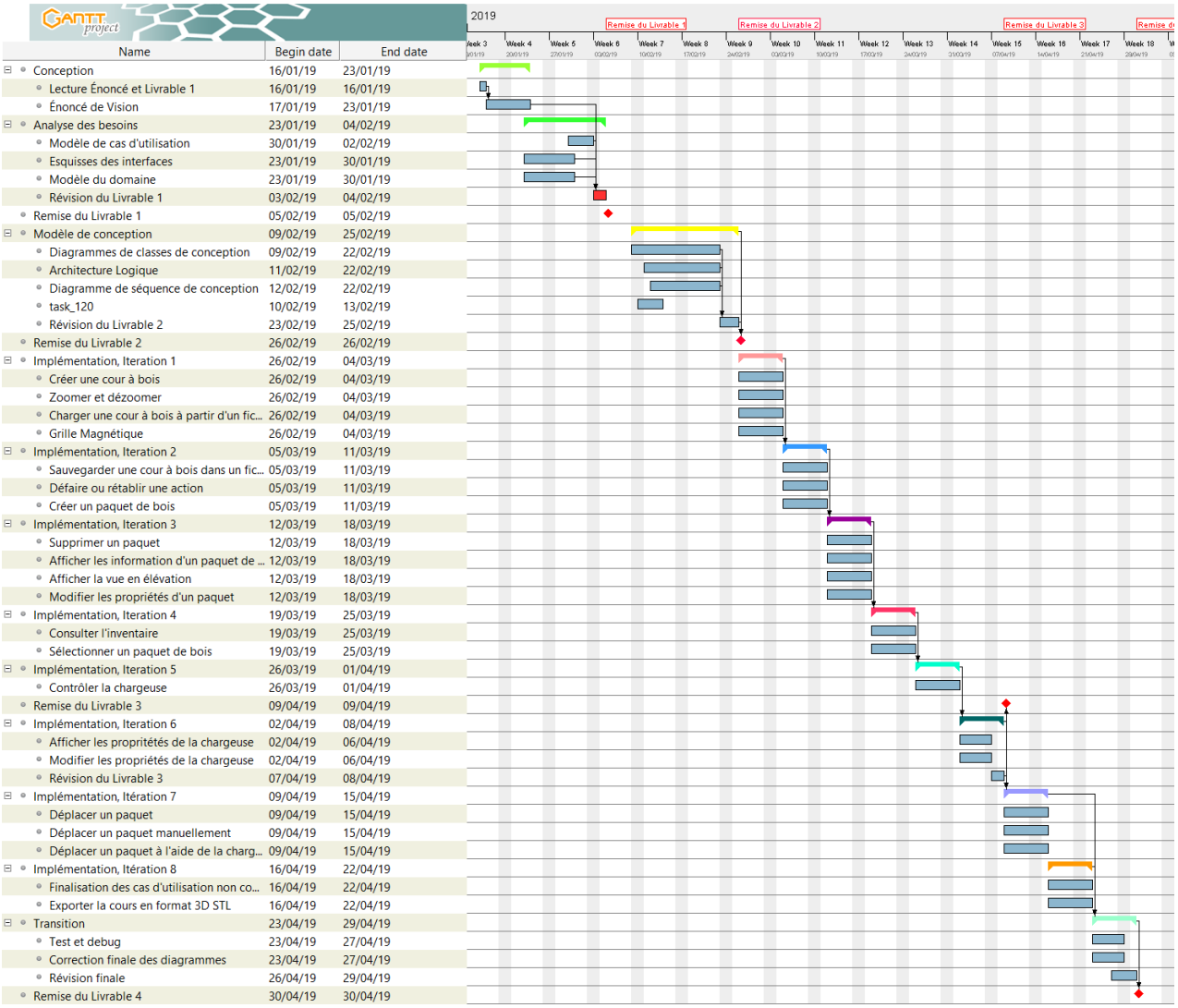
Chapitre 5

Diagramme de Gantt modifié



Name	Begin date	End date
Conception	16/01/19	23/01/19
Analyse des besoins	23/01/19	04/02/19
Remise du Livrable 1	05/02/19	05/02/19
Modèle de conception	09/02/19	25/02/19
Remise du Livrable 2	26/02/19	26/02/19
Implémentation, Iteration 1	26/02/19	04/03/19
Implémentation, Iteration 2	05/03/19	11/03/19
Implémentation, Iteration 3	12/03/19	18/03/19
Implémentation, Iteration 4	19/03/19	25/03/19
Implémentation, Iteration 5	26/03/19	01/04/19
Remise du Livrable 3	09/04/19	09/04/19
Implémentation, Iteration 6	02/04/19	08/04/19
Implémentation, Itération 7	09/04/19	15/04/19
Implémentation, Itération 8	16/04/19	22/04/19
Transition	23/04/19	29/04/19
Remise du Livrable 4	30/04/19	30/04/19

FIGURE 5.1 – Diagramme de Gantt complet modifié



Name	Begin date	End date
Conception	16/01/19	23/01/19
Lecture Énoncé et Livrable 1	16/01/19	16/01/19
Énoncé de Vision	17/01/19	23/01/19
Analyse des besoins	23/01/19	04/02/19
Modèle de cas d'utilisation	30/01/19	02/02/19
Esquisses des interfaces	23/01/19	30/01/19
Modèle du domaine	23/01/19	30/01/19
Révision du Livrable 1	03/02/19	04/02/19
Remise du Livrable 1	05/02/19	05/02/19
Modèle de conception	09/02/19	25/02/19
Diagrammes de classes de conception	09/02/19	22/02/19
Architecture Logique	11/02/19	22/02/19
Diagramme de séquence de conception	12/02/19	22/02/19
task_120	10/02/19	13/02/19
Révision du Livrable 2	23/02/19	25/02/19
Remise du Livrable 2	26/02/19	26/02/19
Implémentation, Iteration 1	26/02/19	04/03/19
Créer une cour à bois	26/02/19	04/03/19
Zoomer et dézoomer	26/02/19	04/03/19
Charger une cour à bois à partir d'un fic...	26/02/19	04/03/19
Grille Magnétique	26/02/19	04/03/19
Implémentation, Iteration 2	05/03/19	11/03/19
Sauvegarder une cour à bois dans un fic...	05/03/19	11/03/19
Défaire ou rétablir une action	05/03/19	11/03/19
Créer un paquet de bois	05/03/19	11/03/19
Implémentation, Iteration 3	12/03/19	18/03/19
Supprimer un paquet	12/03/19	18/03/19
Afficher les information d'un paquet de ...	12/03/19	18/03/19
Afficher la vue en élévation	12/03/19	18/03/19
Modifier les propriétés d'un paquet	12/03/19	18/03/19
Implémentation, Iteration 4	19/03/19	25/03/19
Consulter l'inventaire	19/03/19	25/03/19
Sélectionner un paquet de bois	19/03/19	25/03/19
Implémentation, Iteration 5	26/03/19	01/04/19
Contrôler la chargeuse	26/03/19	01/04/19
Remise du Livrable 3	09/04/19	09/04/19
Implémentation, Iteration 6	02/04/19	08/04/19
Afficher les propriétés de la chargeuse	02/04/19	06/04/19
Modifier les propriétés de la chargeuse	02/04/19	06/04/19
Révision du Livrable 3	07/04/19	08/04/19
Implémentation, Itération 7	09/04/19	15/04/19
Déplacer un paquet	09/04/19	15/04/19
Déplacer un paquet manuellement	09/04/19	15/04/19
Déplacer un paquet à l'aide de la charg...	09/04/19	15/04/19
Implémentation, Itération 8	16/04/19	22/04/19
Finalisation des cas d'utilisation non co...	16/04/19	22/04/19
Exporter la cours en format 3D STL	16/04/19	22/04/19
Transition	23/04/19	29/04/19
Test et debug	23/04/19	27/04/19
Correction finale des diagrammes	23/04/19	27/04/19
Révision finale	26/04/19	29/04/19
Remise du Livrable 4	30/04/19	30/04/19

FIGURE 5.2 – Diagramme de Gantt complet et modifié incluant les tâches détaillées

Chapitre 6

Contribution des membres de l'équipe

Diagramme de classe de conception : L'équipe

Architecture Logique : Gabriel

Diagrammes de séquence de conception : L'équipe

Pseudo-code d'un Algorithme : Yoan

Diagramme de Gantt modifié : Yoan

Révision du document : L'équipe

Protoype d'interface : Jordan et Gabriel

Annexe A

Énoncé de vision

A.1 Introduction

L'objectif principal de ce projet est de créer une application, nommée *VirtuBois*. En démarrant cette application, l'utilisateur pourra créer une ou plusieurs cours à bois, dans différents projets séparés. Chaque projet représente une cours à bois illimitée qu'il peut zoomer ou dézoomer à l'infini. Dans chaque projet, l'utilisateur peut créer plusieurs paquets de planches de bois qui ne contient qu'un type de planche. Il peut les empiler un sur l'autre, manuellement ou avec une chargeuse qu'il aura créée, les dépiler ou changer la dimension ou l'orientation d'un paquet en modifiant les valeurs dans la fenêtre à cet effet.

A.2 Analyse de marché

Présentement, rien de tel ne peut être trouvé sur le marché. Ainsi, implanter une application web qui permet de simuler des cours à bois pourrait grandement aider les entreprises qui ont des cours à bois à mieux les gérer. Plusieurs plans pourraient être simulés et ainsi, à la réception du bois, l'entreposage peut se faire plus facilement. En implantant une application simple, la tâche est facilement réalisable, l'utilisateur aura donc une expérience agréable et il aura accès à toutes les fonctionnalités de base d'une cours à bois.

A.3 Fonctionnalités principales

L'application devra fournir quelques fonctionnalités principales pour répondre aux besoins de l'usager. Voici ces fonctionnalités :

- Définir des paquets de bois qui ne contiennent qu'un seul type de bois selon une dimension et une orientation choisies par l'utilisateur.
- Permettre à différents paquets d'être empilés et dépilés de façon manuelle (fenêtre sur les renseignements du paquet) et avec la chargeuse.
- Afficher la vue d'élévation d'une pile.

— Créer une chargeuse à bras pour déplacer les paquets.
— Permettre à la chargeuse de se déplacer avec les flèches du clavier et spécifiant certains comportements associés à différentes touches du clavier.

Cette liste est sujette au changement tout au long du projet.

A.4 Spécifications supplémentaires

Le projet comporte quelques spécifications qui méritent d'être énoncés. Premièrement, ce projet se réalisera à l'aide de Java 8-10 et aucune autre librairie externe ne peut être utilisée sauf sous autorisation. Sinon, l'interface graphique du projet sera implantée grâce au générateur d'interface graphique JavaFX de IntelliJ.

A.5 Échéanciers

Le projet, qui sera échelonné du 14 janvier 2019 au 30 avril 2019, se divise en quatre livrables différents. Ceux-ci devront être remis respectivement le 5 février, le 26 février, le 9 avril ainsi que le 30 avril. Dès le livrable 3, l'implémentation du code demandera plusieurs itérations. Ces itérations auront plusieurs objectifs précis qui demanderont le travail de chacun. Nous avons déterminé qu'il y aura sept itérations d'une durée d'une semaine débutant le 26 février et qui terminerait le 23 février. La dernière semaine sera utilisée pour réviser la totalité du projet.

A.6 Faisabilité et risques du projet

Ce projet comporte certainement quelques risques, mais ceux-ci seront très bien gérables si tous les membres de l'équipe mettent un minimum d'effort pour accomplir la tâche à venir. En effet, il n'y a qu'un membre de l'équipe qui a de l'expérience avec Java et avec l'outil de création d'interface Java FX. Cependant, les membres de l'équipe sont motivés et prêts à mettre les efforts nécessaires pour livrer un travail de qualité avec Java FX et d'apprendre, du même coup, à utiliser cet outil. Aussi, certains membres de l'équipe n'ont jamais monté et réalisé un tel projet dans son entièreté. Malgré ces quelques risques, le projet est tout de même réalisable et faisable dans son ensemble.

Annexe B

Modèle du domaine

B.1 Diagramme du modèle du domaine

B.2 Texte explicatif

Utilisateur : Il s'agit tout simplement de l'utilisateur du logiciel. Cet utilisateur peut créer une cour à bois, dans un projet distinct, dans lequel il crée différents paquets de bois qu'il peut déplacer de manière manuelle ou avec une chargeuse à bras. Il peut aussi modifier, à sa guise, les attributs des paquets comme ses dimensions, sa localisation, son orientation, etc.

Cour : Ceci représente en quelque sorte un projet : un objet contenant toutes les informations pour une cour à bois. Il est possible d'avoir plusieurs cours qui contiennent des informations différentes les unes des autres. Il faudra cependant charger des fichiers indépendants pour avoir accès à l'une ou l'autre des cours. Chaque cour possède des dimensions infinies, c'est-à-dire qu'on peut se déplacer indéfiniment dans une cour sans jamais atteindre de limite. De plus, une cour possède une instance de la classe Chargeuse et autant d'instances de la classe Pile que l'utilisateur souhaite.

Paquet : Ce sont les paquets de bois qui sont contenus dans la cour. Ils ont des propriétés qui permettent de les diférencier les uns des autres, soit un code barre, une largeur, une longueur, une hauteur, une date et heure de production ainsi qu'un type (p.ex. 2x6 Érable). Tous les paquets ont la forme d'un prisme à base rectangulaire. L'utilisateur peut les déplacer dans la cour en tout temps, soit manuellement ou à l'aide de la chargeuse. Également, l'utilisateur peut modifier l'angle et les attributs des paquets en tout temps. On peut retrouver l'emplacement d'un paquet avec une simple recherche par code à barre dans l'inventaire.

Pile : Une pile est constituée d'au moins un paquet. C'est ce qui représente la position de un ou plusieurs paquet superposés à partir de coordonnées. L'utilisateur peut, en tout temps, connaître le nombre de paquets contenu dans la pile, l'ordre de ces paquets dans

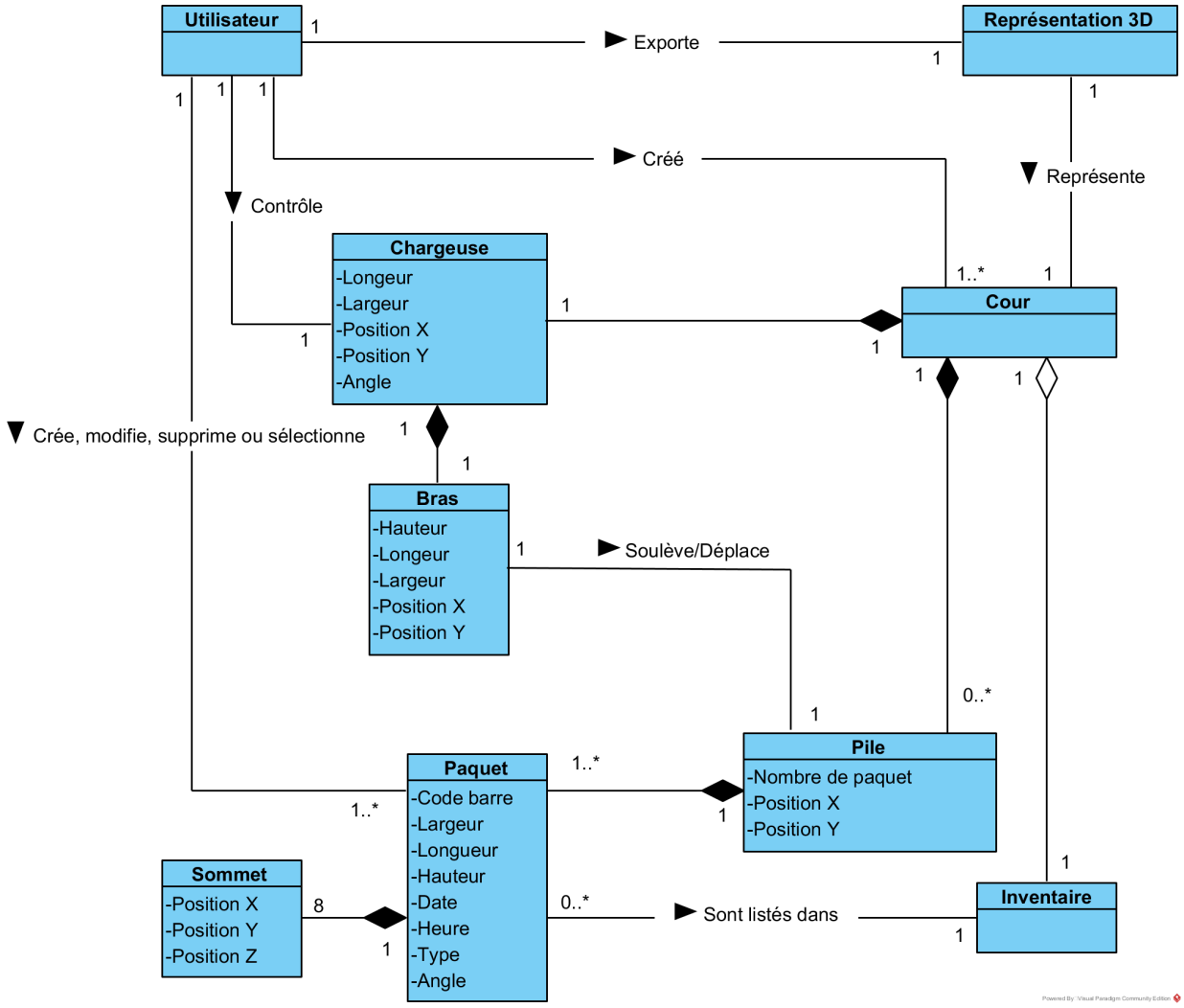


FIGURE B.1 – Diagramme des classes conceptuelles

la pile ainsi que les différentes informations sur chacun des paquets. Ceci est contenu dans la vue d'élévation. Il peut aussi utiliser la chargeuse pour déplacer une partie des paquets de la pile ou encore la pile au complet, tout dépendant de la hauteur des bras de la chargeuse.

Inventaire : Il s'agit d'une liste de tous les paquets présents dans la cour à bois. Grâce à l'inventaire, un utilisateur pourra en tout temps rechercher un paquet par code à barre pour retrouver son emplacement dans la cour et connaître ses propriétés respectives.

Chargeuse : C'est une entité qui permet de se déplacer dans la cour et de déplacer les paquets d'une pile à une autre. Elle contient des coordonnées qui permettent de la localiser. En tout temps, l'utilisateur peut prendre le contrôle de la chargeuse simplement en cliquant dessus. Il peut ainsi la déplacer dans la cour à travers les différentes piles de paquets. La

chargeuse est composé d'un bras pour lui permettre d'effectuer son travail.

Bras : C'est ce qui soulève les paquets. On utilise des coordonnées décalées de celles de la chargeuse. Grâce à certaines touches du clavier, les bras peuvent monter ou descendre. Ainsi, les coordonnées horizontales des bras sont dépendantes des coordonnées horizontales de la chargeuse tandis que les coordonnées verticales sont indépendantes de la chargeuse.

Coordonnées : Il s'agit de coordonnées en trois dimensions représentant une position en x, y et z de plusieurs objets dans la cour à bois. Chaque coordonnée peut être modifiée avec le clavier ou la souris ou lors de déplacement avec la chargeuse.

Sommet : Il s'agit d'une coordonnée utilisée pour représenter un des coins du polyèdre qui représente un paquet.

Prisme Rectangulaire : C'est tout simplement la forme que prend un paquet dans l'espace. Il est utilisé pour connaître la position des paquets dans la cour.

Annexe C

Modèle des cas d'utilisation

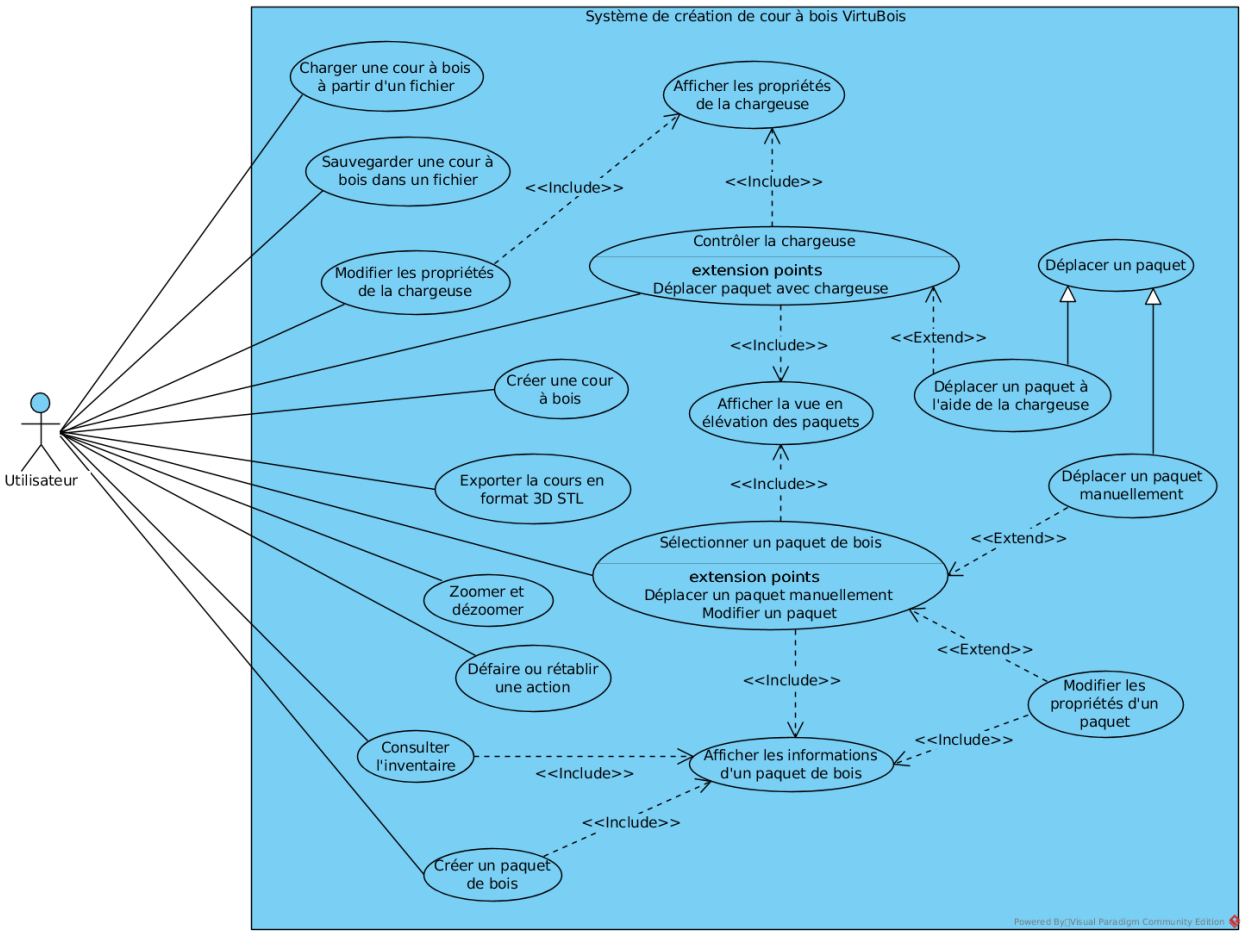


FIGURE C.1 – Diagramme des cas d'utilisation

C.1 Cas principaux

C.1.1 Charger une cour à bois

Cas d'utilisation :	Charger une cour à bois à partir d'un fichier
Système :	Système de création de cour à bois *VirtuBois*
Acteur(s) :	Utilisateur
Partie prenante et intérêts :	Utilisateur : Charger une cour à bois existante pour consultation ou modification.
Précondition(s) :	L'application est lancée ou l'utilisateur travaille déjà sur un autre projet
Garanties en cas de succès :	La cour à bois est créer dans un projet distinct et l'utilisateur peut débuter la création de la cour à bois.
Scénario principal :	1. L'utilisateur démarre l'application. 2. L'utilisateur décide de charger un projet. 3. Présentation d'une fênetre permettant de naviguer dans les dossiers de l'ordinateur de l'utilisateur. 4. Le système valide le format du fichier sélectionné par l'utilisateur. 5. Le système charge les informations du projet existant. 6. Affichage de la cour à bois. 7. L'utilisateur peut consulter ou modifier sa cour à bois.
Scénario(s) alternatif(s) :	Ligne 4 : Le format du fichier n'est pas supporté par l'application. Un message d'erreur est affiché à l'utilisateur. Ligne 5 : Le fichier contient des informations illisibles par l'application. Ces informations ne sont donc pas affichées à l'utilisateur. Un fichier totalement corrompu déclenche un message d'erreur.

TABLEAU C.1 – Cas d'utilisation pour charger une cour à bois

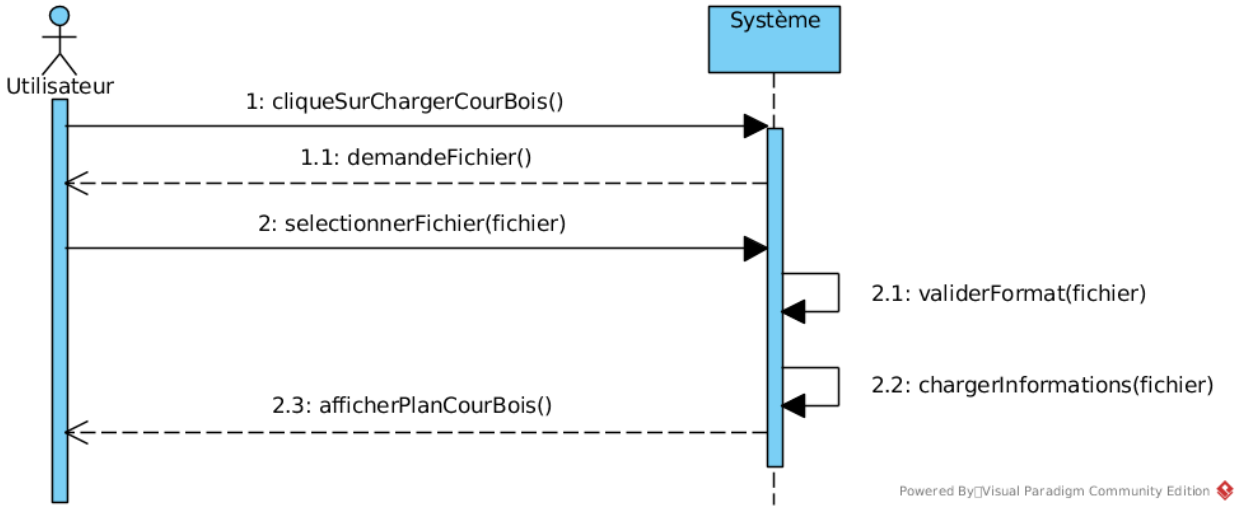


FIGURE C.2 – Diagramme de séquence système pour charger une cour à bois

C.1.2 Sauvegarder une cour à bois

Cas d'utilisation :	Sauvegarder une cour à bois dans un fichier
Système :	Système de création de cour à bois *VirtuBois*
Acteur(s) :	Utilisateur
Partie prenante et intérêts :	Utilisateur : Sauvegarder un projet en cours pour conserver le travail accompli ou pour y revenir plus tard pour consultation ou modification.
Précondition(s) :	L'application contient un projet de cour à bois ouvert.
Garanties en cas de succès :	La cour à bois est sauvegardée dans un fichier sur l'ordinateur de l'utilisateur.
Scénario principal :	1. L'utilisateur décide de sauvegarder son projet et ouvre le menu à cet effet. 2. Présentation d'une fenêtre permettant de naviguer dans les dossiers de l'ordinateur de l'uilisateur et permettant la modification du nom au besoin. 3. L'utilisateur décide de l'endroit de sauvegarde. 4. Le système valide l'endroit de sauvegarde et le nom du fichier. 5. L'utilisateur peut continuer la création de sa cour à bois ou quitter le projet sans risque de perdre de l'information.
Scénario(s) alternatif(s) :	Ligne 4 : Le nouveau nom entré par l'utilisateur est le nom d'un projet déjà existant ou l'endroit de sauvegarde est invalide. Un message d'erreur est affiché à l'utilisateur.

TABLEAU C.2 – Cas d'utilisation pour sauvegarder une cour à bois

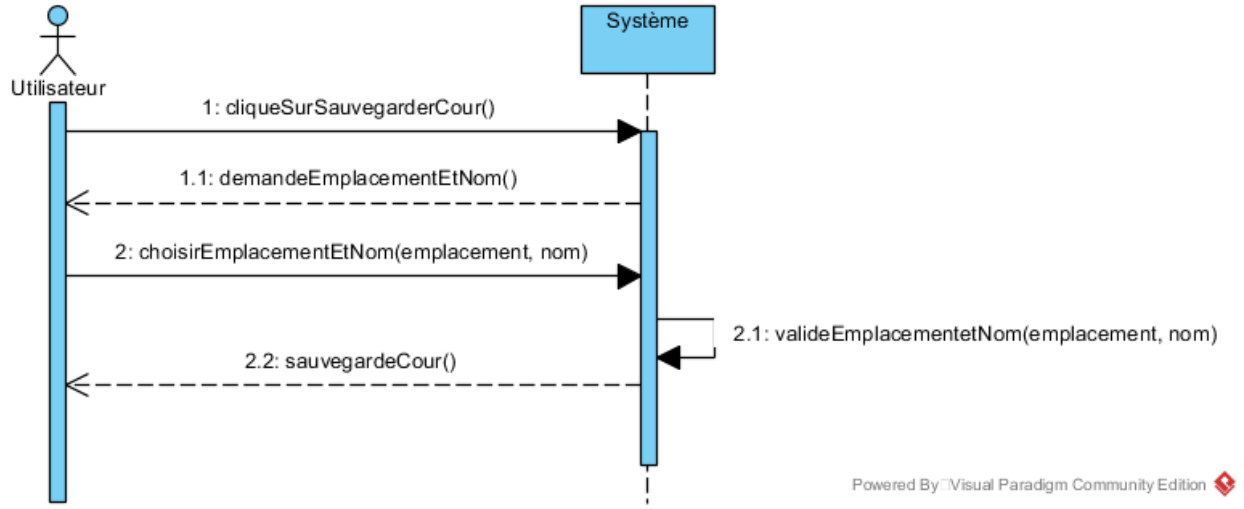


FIGURE C.3 – Diagramme de séquence système pour sauvegarder une cour à bois

C.1.3 Créer un paquet

Cas d'utilisation :	Créer un paquet de bois
Système :	Système de création de cour à bois *VirtuBois*
Acteur(s) :	Utilisateur
Partie prenante et intérêts :	Utilisateur : Créer un ou plusieurs paquets de bois pour aménager la cour à bois.
Précondition(s) :	L'application contient une cour à bois vide ou non.
Garanties en cas de succès :	Le paquet de bois est placé dans la cour à bois. Il contient des coordonnées et une série de caractéristiques spécifiques à lui-même.
Scénario principal :	1. L'utilisateur clique sur le bouton de création de paquets. 2. Présentation d'une fenêtre demandant des caractéristiques spécifiques pour le paquet. 3. Le système valide les informations entrées par l'utilisateur. 4. Fermeture de la fênetre. 5. L'utilisateur peut continuer l'aménagement de sa cour.
Scénario(s) alternatif(s) :	Ligne 3 : Les caractéristiques spécifiques du paquet ont un format invalide. Un message d'erreur est lancé.

TABLEAU C.3 – Cas d'utilisation pour créer un paquet de bois

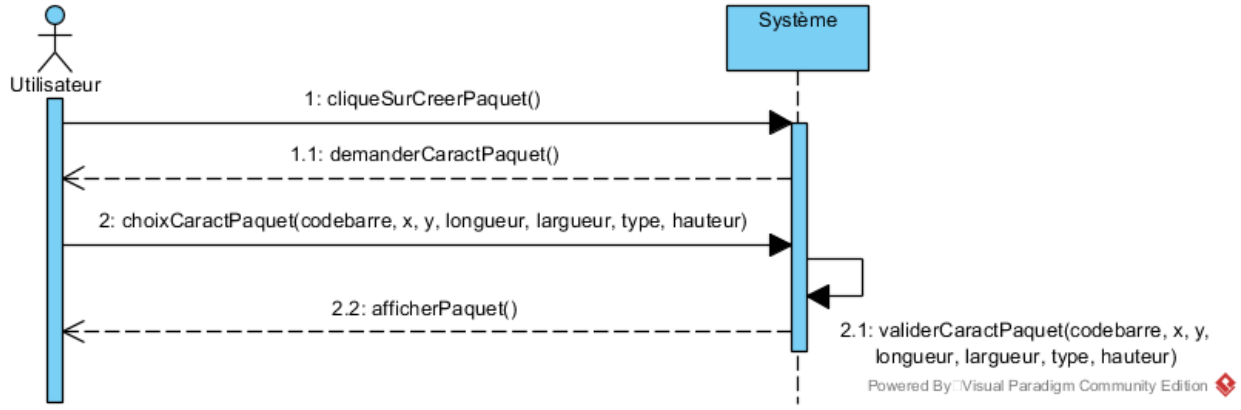


FIGURE C.4 – Diagramme de séquence système pour créer un paquet de bois

C.1.4 Créer une cour

Cas d'utilisation :	Créer une cour à bois vierge
Système :	Système de création de cour à bois *VirtuBois*
Acteur(s) :	Utilisateur
Partie prenante et intérêts :	Utilisateur : Créer une cour à bois vierge pour commencer l'aménagement de celle-ci.
Précondition(s) :	L'application est lancée.
Garanties en cas de succès :	La cour à bois est créée dans un projet distinct et l'utilisateur peut commencer à y ajouter des paquets de bois.
Scénario principal :	1. L'utilisateur démarre l'application 2. L'utilisateur décide de créer un nouveau projet. 3. Présentation d'une fênetre demandant le nom du projet. 4. Le système valide le nom du projet. 5. Affichage de la nouvelle cour à bois. 6. L'utilisateur peut continuer l'aménagement de sa cour.
Scénario(s) alternatif(s) :	Ligne 4 : Le nom du projet lors de la création de la cour à bois existe déjà. Un message d'erreur est lancé.

TABLEAU C.4 – Cas d'utilisation pour créer une cour à bois

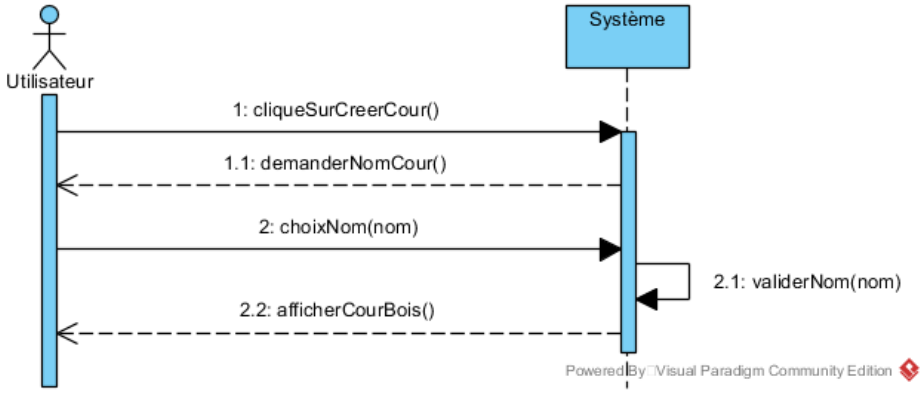


FIGURE C.5 – Diagramme de séquence système pour créer une cour à bois

C.1.5 Sélectionner un paquet

Cas d'utilisation :	Sélectionner un paquet de bois
Système :	Système de création de cour à bois *VirtuBois*
Acteur(s) :	Utilisateur
Partie prenante et intérêts :	Utilisateur : Sélectionner un paquet afin d'en voir les informations.
Précondition(s) :	Au moins un paquet se retrouve sur la cour.
Garanties en cas de succès :	Le paquet est marqué comme sélectionné et les actions étendues sont effectuées.
Scénario principal :	1. L'utilisateur clique sur un paquet. 2. Le paquet est marqué comme sélectionné. 3. Les informations du paquet sont affichées.
Scénario(s) alternatif(s) :	Ligne 1 : L'utilisateur clique sur une pile. La vue en élévation est affichée. L'utilisateur sélectionne un paquet dans la vue en élévation. Le scénario continue à la ligne 2.

TABLEAU C.5 – Cas d'utilisation pour sélectionner un paquet

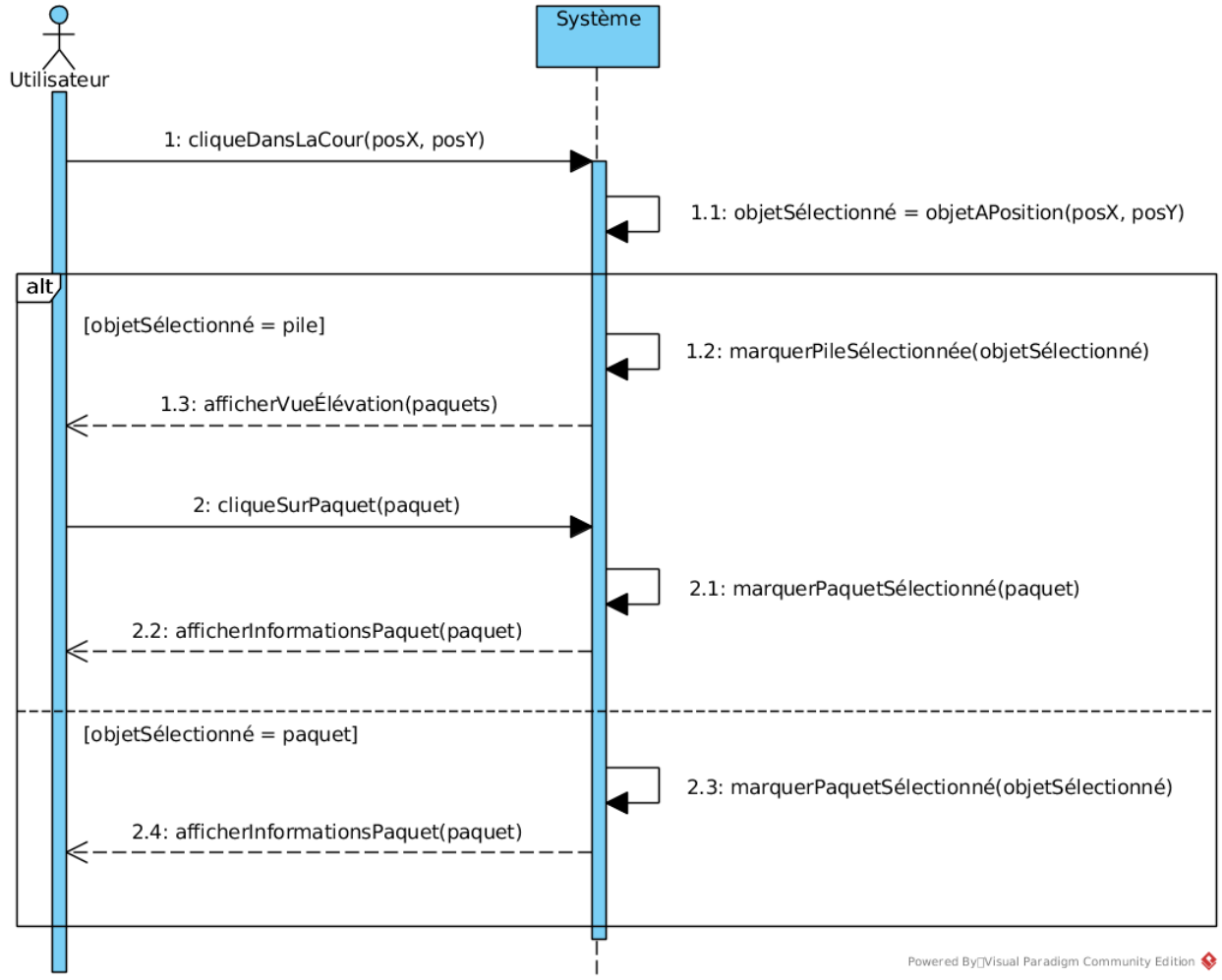


FIGURE C.6 – Diagramme de séquence système pour sélectionner un paquet

C.1.6 Modifier les propriétés d'un paquet

Cas d'utilisation :	Modifier les propriétés d'un paquet de bois
Système :	Système de création de cour à bois *VirtuBois*
Acteur(s) :	Utilisateur
Partie prenante et intérêts :	Utilisateur : Modifier les propriétés d'un paquet de bois et les enregistrer.
Précondition(s) :	Un paquet est sélectionné.
Garanties en cas de succès :	Les modifications effectuées sont enregistrées correctement.
Scénario principal :	1. Modifie le champ d'une propriété. 2. Valide la modification. 3. Enregistre la modification 4. Met à jour l'affichage.
Scénario(s) alternatif(s) :	Ligne 2 : La validation échoue. La modification est annulée et un message d'erreur est affiché. Le scénario continue à la ligne 4.

TABLEAU C.6 – Cas d'utilisation pour modifier les propriétés d'un paquet de bois

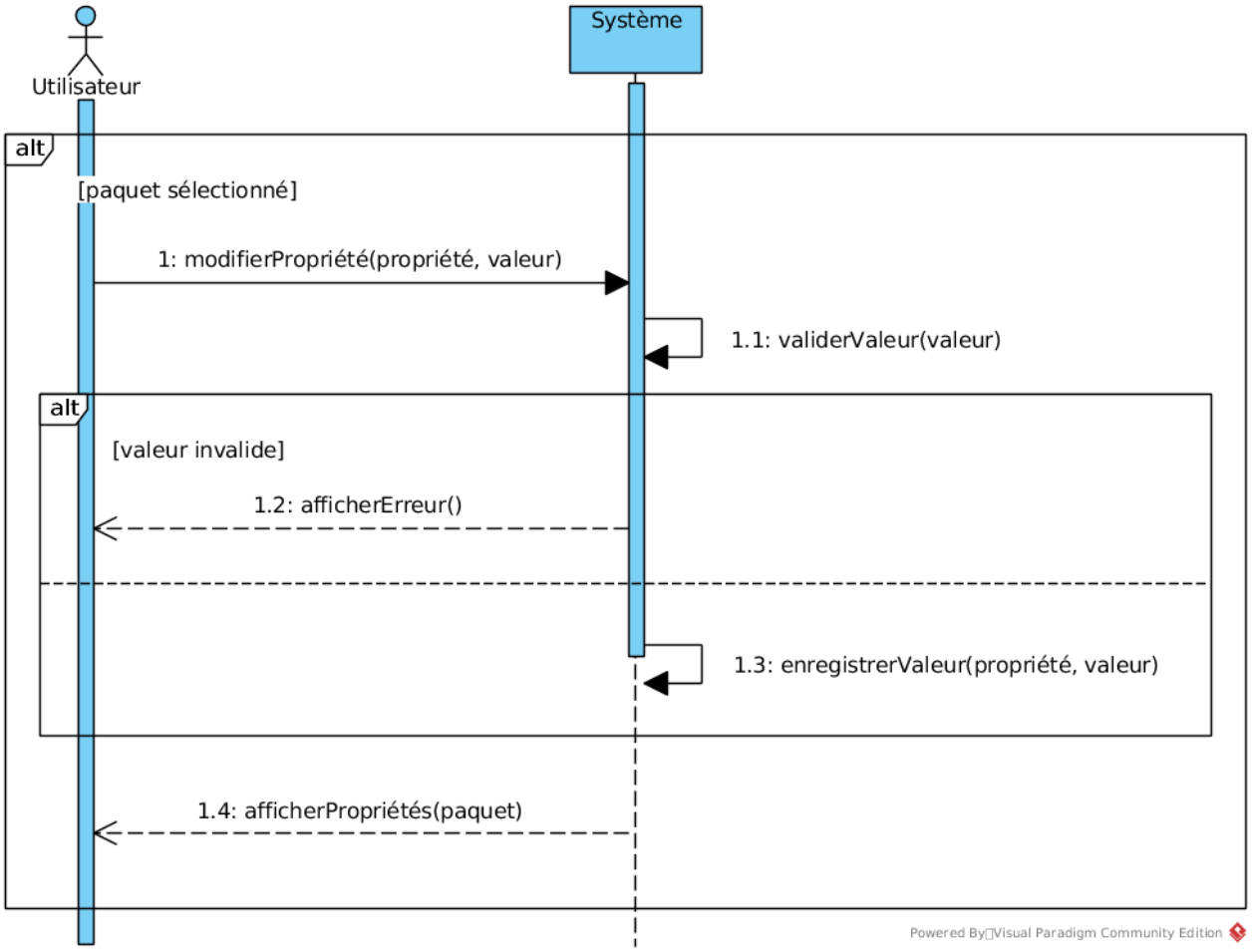


FIGURE C.7 – Diagramme de séquence système pour modifier les propriétés d'un paquet

C.1.7 Contrôler la chargeuse

Cas d'utilisation :	Contrôler la chargeuse
Système :	Système de création de cour à bois *VirtuBois*
Acteur(s) :	Utilisateur
Partie prenante et intérêts :	Utilisateur : Contrôler la chargeuse afin de pouvoir déplacer des paquets.
Précondition(s) :	La cour est créée et la chargeuse est déposée dans la cour.
Garanties en cas de succès :	La chargeuse se déplace selon les commandes envoyées.
Scénario principal :	1. L'utilisateur appuie sur la flèche du haut. 2. La chargeuse se déplace vers l'avant. 3. L'utilisateur cesse d'appuyer sur la touche. 4. La chargeuse cesse de bouger.
Scénario(s) alternatif(s) :	Ligne 1 : L'utilisateur appuie sur la flèche du bas. Ligne 2 : La chargeuse recule. Le scénario continue à la ligne 3. --- Ligne 1 : L'utilisateur appuie sur la flèche de gauche. Ligne 2 : La chargeuse tourne dans le sens anti-horaire. Le scénario continue à la ligne 3. --- Ligne 1 : L'utilisateur appuie sur la flèche de droite. Ligne 2 : La chargeuse tourne dans le sens horaire. Le scénario continue à la ligne 3. --- Ligne 1 : L'utilisateur appuie sur les touches contrôle et flèche du haut. Ligne 2 : Les bras de la chargeuse s'élèvent. Le scénario continue à la ligne 3. --- Ligne 1 : L'utilisateur appuie sur les touches contrôle et flèche du bas. Ligne 2 : Les bras de la chargeuse descendent. Le scénario continue à la ligne 3. --- Ligne 1 : L'utilisateur appuie sur les touches contrôle et espace. Ligne 2 : La chargeuse avance jusqu'à ce qu'elle atteigne un paquet. Le scénario continue à la ligne 3. --- Ligne 1 : L'utilisateur appuie sur la touche d'effacement. Ligne 2 : La chargeuse ramasse les paquets plus hauts que ses bras et en avant d'elle, puis la vue en élévation des paquets chargés est affichée. Le scénario continue à la ligne 3. --- Ligne 1 : L'utilisateur appuie sur la touche entrée. Ligne 2 : La chargeuse décharge les paquets préalablement ramassés. Le scénario continue à la ligne 3. --- En tout temps : si la chargeuse s'apprête à heurter un obstacle, alors elle ne sera pas déplacée.

TABLEAU C.7 – Cas d'utilisation pour contrôler la chargeuse

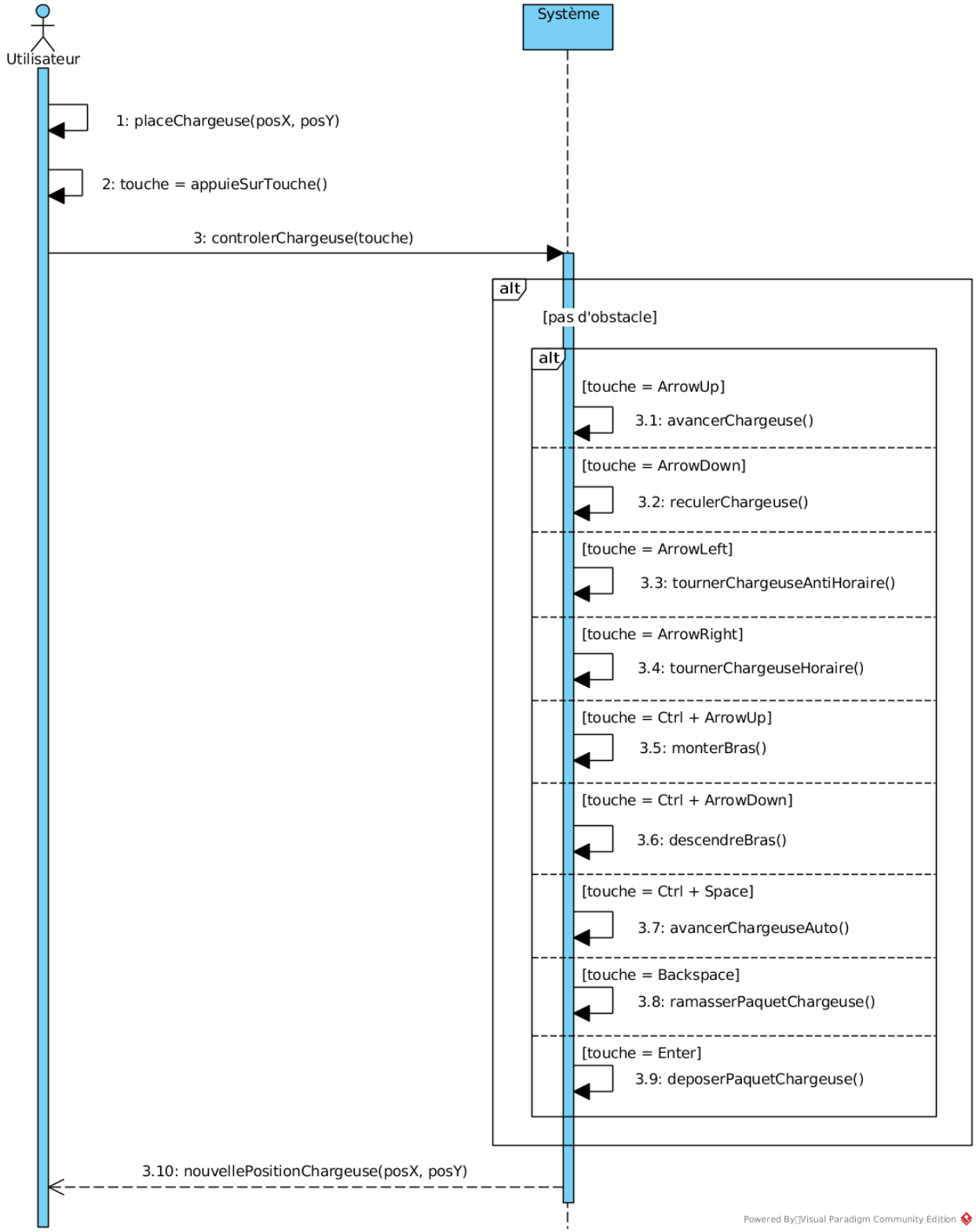


FIGURE C.8 – Diagramme de séquence système pour contrôler la chargeuse

C.1.8 Modifier les propriétés de la chargeuse

Cas d'utilisation :	Modifier les propriétés de la chargeuse
Système :	Système de création de cour à bois *VirtuBois*
Acteur(s) :	Utilisateur
Partie prenante et intérêts :	Utilisateur : Modifier les propriétés de la chargeuse et les enregistrer.
Précondition(s) :	La chargeuse est présente sur la cour.
Garanties en cas de succès :	Les modifications effectuées sont enregistrées correctement.
Scénario principal :	1. Modifie le champ d'une propriété. 2. Valide la modification. 3. Enregistre la modification 4. Met à jour l'affichage.
Scénario(s) alternatif(s) :	Ligne 2 : La validation échoue. La modification est annulée et un message d'erreur est affiché. Le scénario continue à la ligne 4.

TABLEAU C.8 – Cas d'utilisation pour modifier les propriétés de la chargeuse

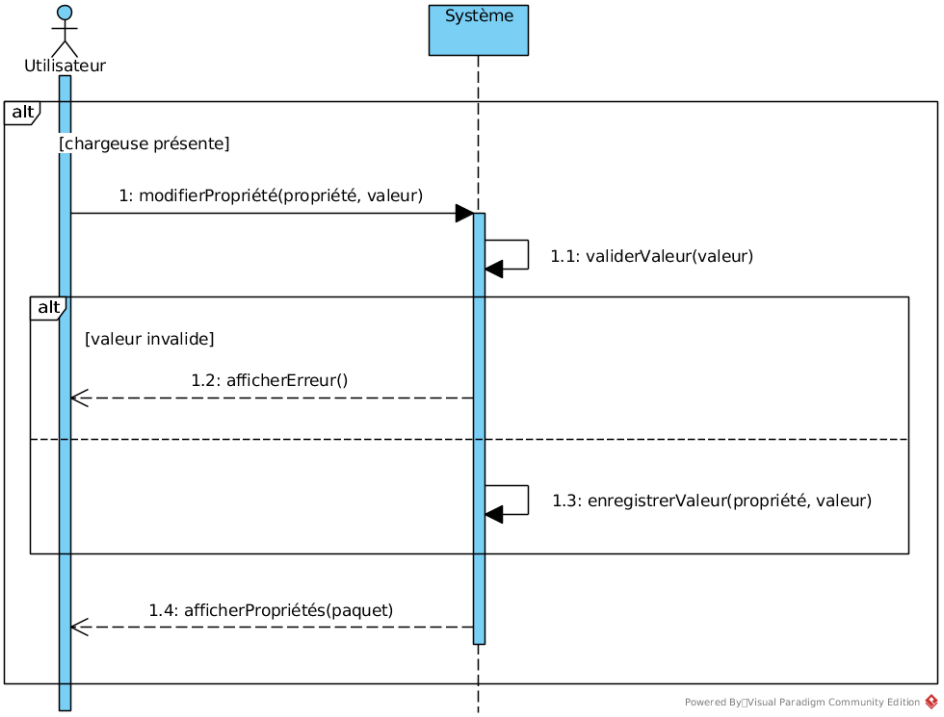


FIGURE C.9 – Diagramme de séquence système pour modifier les propriétés de la chargeuse

C.1.9 Déplacer un paquet à l'aide de la chargeuse

Cas d'utilisation :	Déplacer un paquet à l'aide de la chargeuse
Système :	Système de création de cour à bois *VirtuBois*
Acteur(s) :	Utilisateur
Partie prenante et intérêts :	Utilisateur : Voir et entretenir les déplacements effectués au sein de la cour.
Précondition(s) :	Au moins un paquet a été créé.
Garanties en cas de succès :	Le paquet est déplacé à l'endroit désiré.
Scénario principal :	1. L'utilisateur déplace la chargeuse vers le paquet désiré. 2. L'utilisateur appuie sur la touche permettant de soulever le paquet. 3. Le paquet est soulevé. 4. L'utilisateur déplace le paquet à l'aide de la chargeuse. 5. La position du paquet est affichée. 6. L'utilisateur appuie sur la touche permettant de déposer le paquet. 7. Le paquet est déposé.
Scénario(s) alternatif(s) :	Ligne 3 : Si plusieurs paquets sont empilés, la partie de la pile au dessus du paquet soulevé sera emportée avec celui-ci.

TABLEAU C.9 – Cas d'utilisation pour déplacer un paquet à l'aide de la chargeuse

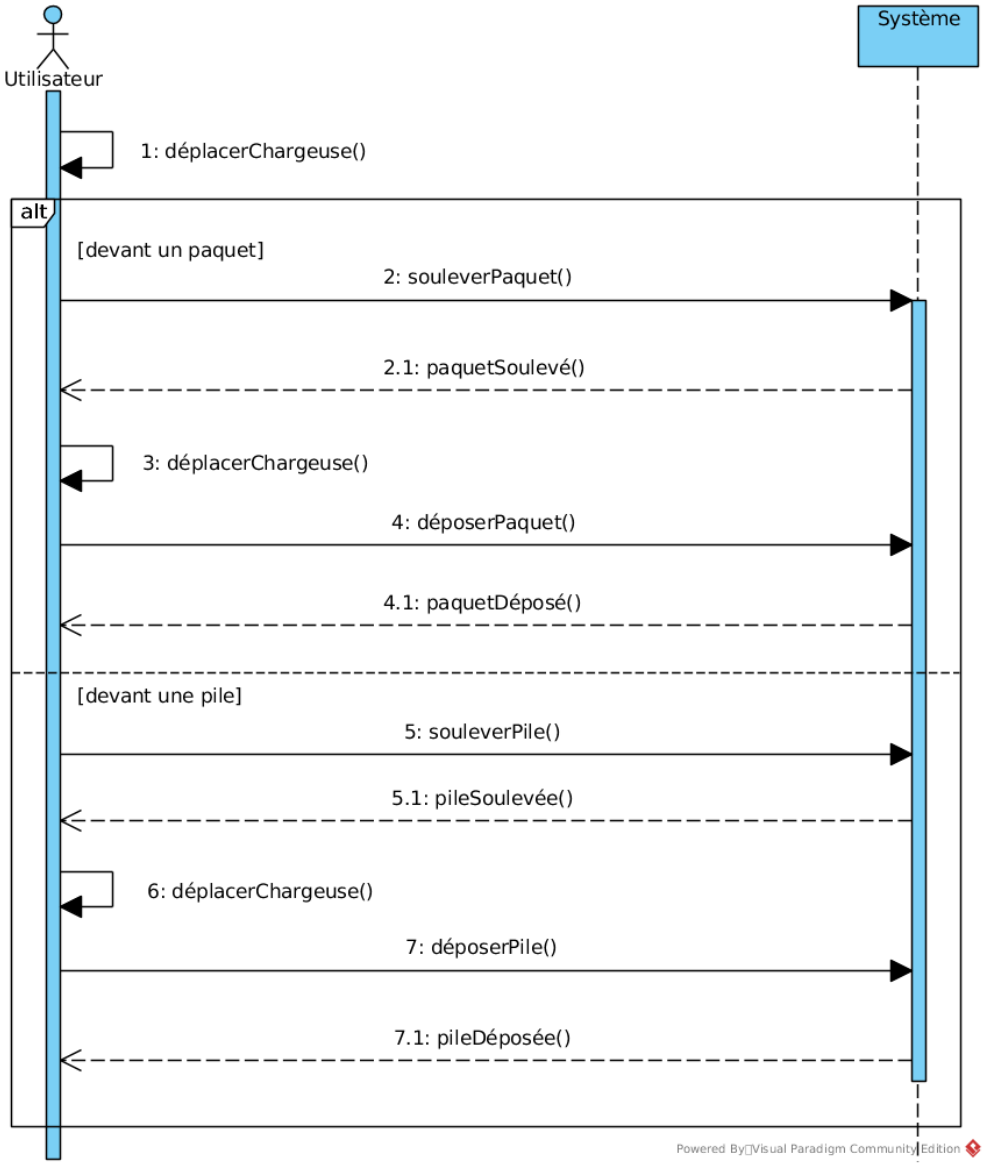


FIGURE C.10 – Diagramme de séquence système pour déplacer un paquet à l'aide de la chargeuse

C.1.10 Déplacer un paquet manuellement

Cas d'utilisation :	Déplacer un paquet manuellement
Système :	Système de création de cour à bois *VirtuBois*
Acteur(s) :	Utilisateur
Partie prenante et intérêts :	Utilisateur : Voir et entretenir les déplacements effectués au sein de la cour.
Précondition(s) :	Un paquet ou une pile est sélectionné.
Garanties en cas de succès :	Le paquet ou la pile est déplacé à l'endroit désiré.
Scénario principal :	1. Déplace le paquet à l'endroit désiré. 2. Valide la nouvelle position 3. Enregistre la nouvelle position 4. Affiche la nouvelle position
Scénario(s) alternatif(s) :	Ligne 1 : C'est une pile qui est sélectionnée. C'est alors la pile que l'on déplace. Ligne 2 : La validation échoue. Un message d'erreur est affiché.

TABLEAU C.10 – Cas d'utilisation pour déplacer un paquet manuellement

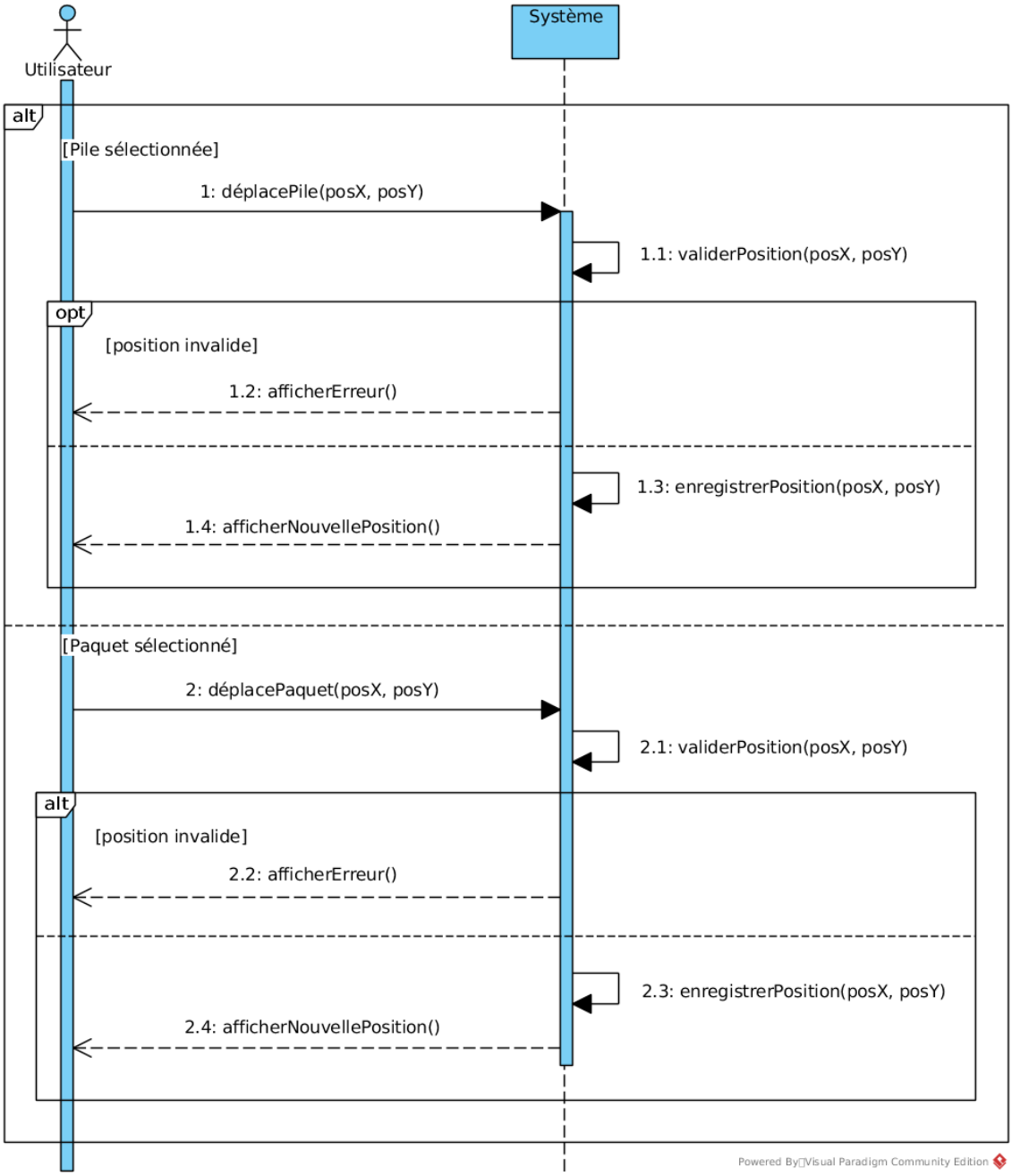


FIGURE C.11 – Diagramme de séquence système pour déplacer un paquet manuellement

C.2 Cas secondaires

C.2.1 Afficher les propriétés d'un paquet

Cas d'utilisation :	**Afficher les propriétés d'un paquet**
Acteur :	Utilisateur du logiciel
Type :	Secondaire
Description :	Lorsqu'un paquet est sélectionné, l'ensemble des propriétés d'un paquet (code à barre, type, position, etc.) sont affichées dans une région de l'interface prévue à cet effet. Cet affichage comprend des champs que l'utilisateur peut modifier en tout temps.

Tableau C.11 – Cas d'utilisation pour afficher les propriétés d'un paquet

C.2.2 Afficher les propriétés de la chargeuse

Cas d'utilisation :	**Afficher les propriétés de la chargeuse**
Acteur :	Utilisateur du logiciel
Type :	Secondaire
Description :	En tout temps, on peut voir les propriétés de la chargeuse dans une des sous-fenêtres de la fenêtre principale. On peut voir la position en x et y, la rotation de la chargeuse et l'élévation du bras de chargement. Les informations sont modifiées en temps réel dans le cas d'un déplacement de la chargeuse.

Tableau C.12 – Cas d'utilisation pour afficher les propriétés d'une chargeuse

C.2.3 Afficher la vue d'élévation

Cas d'utilisation :	**Afficher la vue d'élévation**
Acteur :	Utilisateur du logiciel
Type :	Secondaire
Description :	L'utilisateur peut en tout temps cliquer sur un paquet pour que la fenêtre lui affiche la vue d'élévation des paquets. Cette vue affiche les niveaux des paquets. Ainsi, si la pile en comporte trois, l'utilisateur y verra trois niveaux.

Tableau C.13 – Cas d'utilisation pour afficher la vue d'élévation

C.2.4 Consulter l'inventaire

Cas d'utilisation :	**Consulter l'inventaire**
Acteur :	Utilisateur du logiciel
Type :	Secondaire
Description :	Une section de l'application est réservée à l'affichage de l'inventaire. Cette section se présente sous forme d'une liste de paquets présents dans la cour à bois. L'utilisateur peut en tout temps consulter l'inventaire des paquets avec la barre déroulante. De plus, il peut effectuer une recherche par code à barre pour retrouver un paquet spécifique. S'il sélectionne un paquet dans la liste, les informations sur celui-ci sont affichées dans la section réservée aux informations d'un paquet.

Tableau C.14 – Cas d'utilisation pour consulter l'inventaire

C.2.5 Zoomer et dézoomer

Cas d'utilisation :	**Zoomer et dézoomer**
Acteur :	Utilisateur du logiciel
Type :	Secondaire
Description :	L'utilisateur a la possibilité de zoomer et de dézoomer à l'infini peu importe où il se situe dans la cour à bois. Il n'a qu'à cliquer sur l'une des deux loupes (situées dans l'un des coins de la section d'affichage de la cour à bois) selon l'action qu'il souhaite posée.

Tableau C.15 – Cas d'utilisation pour zoomer et dézoomer

C.2.6 Défaire et rétablir une action

Cas d'utilisation :	**Défaire ou rétablir une action**
Acteur :	Utilisateur du logiciel
Type :	Secondaire
Description :	L'utilisateur peut en tout temps défaire ou rétablir une action posée par celui-ci. Le système sera responsable de défaire ou rétablir son action, selon le choix de l'utilisateur.

Tableau C.16 – Cas d'utilisation pour défaire ou rétablir une action

C.2.7 Exporter la cour en format 3D STL

Cas d'utilisation :	**Exporter la cour en format 3D STL**
Acteur :	Utilisateur du logiciel
Type :	Secondaire
Description :	L'utilisateur peut, en cliquant sur le bouton à cet effet, exporter la cour à bois dans un format 3D standard (STL). On y verra donc les paquets empilés les uns sur les autres et ce, sous tous les angles voulus.

Tableau C.17 – Cas d'utilisation pour exporter la cour en format 3D STL

Annexe D

Esquisses des interfaces utilisateur

D.1 Interface utilisateur sans plan de la cour de bois

Il s'agit de l'interface utilisateur lorsque le programme est lancé. Seul le bouton pour la gestion de fichier est activé. À ce moment, l'utilisateur a deux choix : il peut soit créer un nouveau projet ou encore charger un projet existant. Il n'y a aucun contenu dans les sections sous la barre de tâches, car aucun projet n'est réellement ouvert à ce stade.

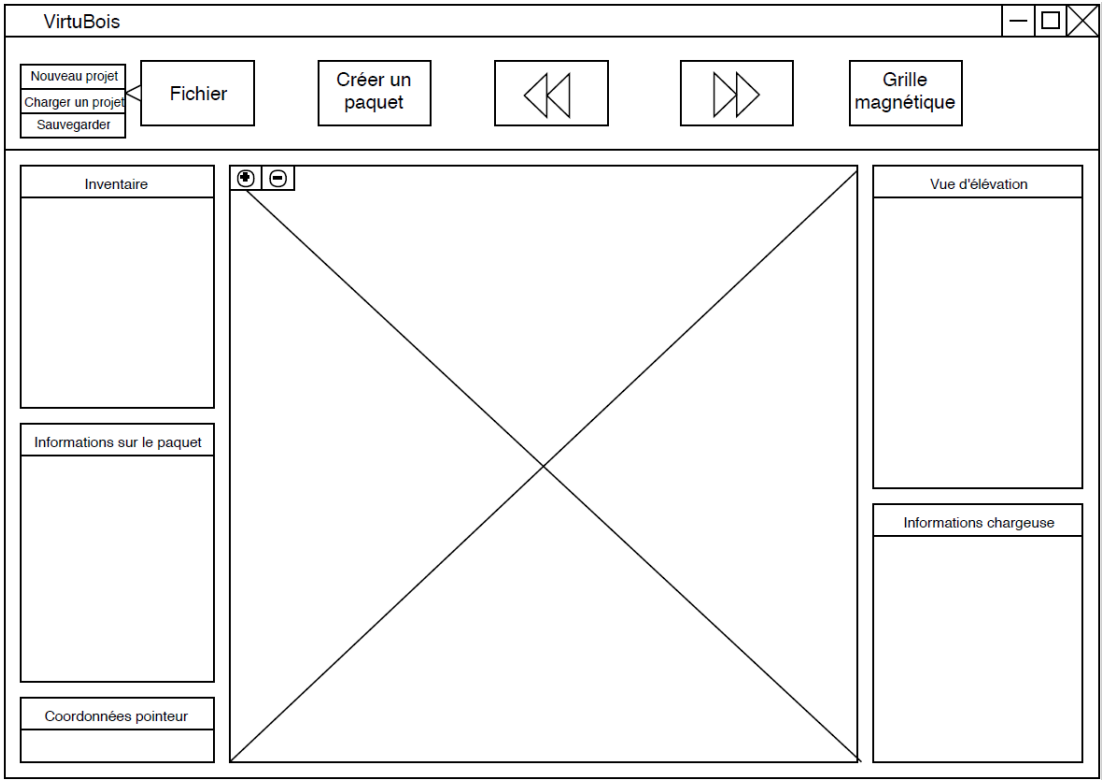


FIGURE D.1 – Interface utilisateur sans plan de la cour de bois

D.2 Interface utilisateur avec plan de cour à la création d'un nouveau projet

Lors de la création d'un nouveau projet, le contenu et les boutons de l'application deviennent actifs. La cour à bois est affichée et ne contient, au départ, que la chargeuse centrée dans la cour. À ce stade, il est donc possible pour l'utilisateur de créer différents paquets et de déplacer la chargeuse. Les informations affichées à l'intérieur des boîtes placées de part et d'autre du plan de la cour à bois devraient se mettre à jour automatiquement en fonction des actions de l'utilisateur.

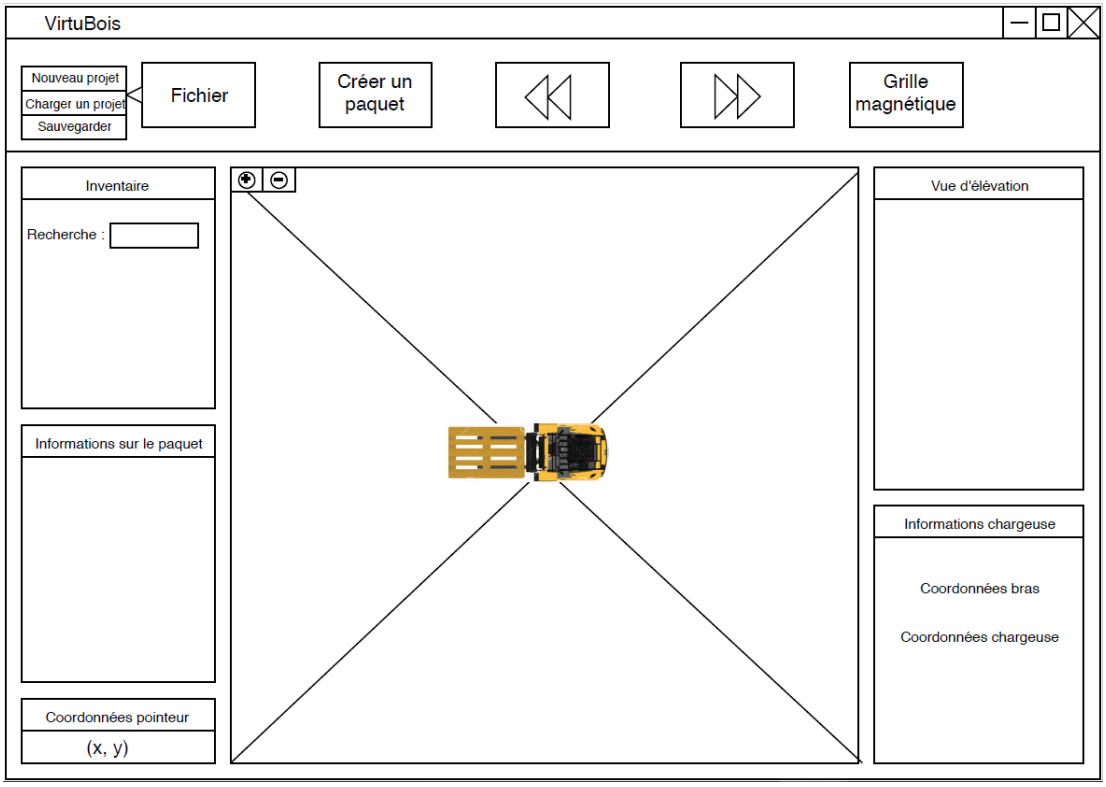


FIGURE D.2 – Interface utilisateur avec plan de cour à la création d'un nouveau projet

D.3 Interface utilisateur lors de la création d'un paquet

L'utilisateur peut créer un nombre infini de paquets à partir d'un bouton de la barre des tâches. Àprès avoir cliqué sur ce bouton, une fenêtre permettant de paramétriser le paquet à créer s'ouvre. L'utiliseur remplit les différents champs et le paquet apparait par la suite dans le plan de la cour à bois à partir des informations entrées par l'utilisateur.

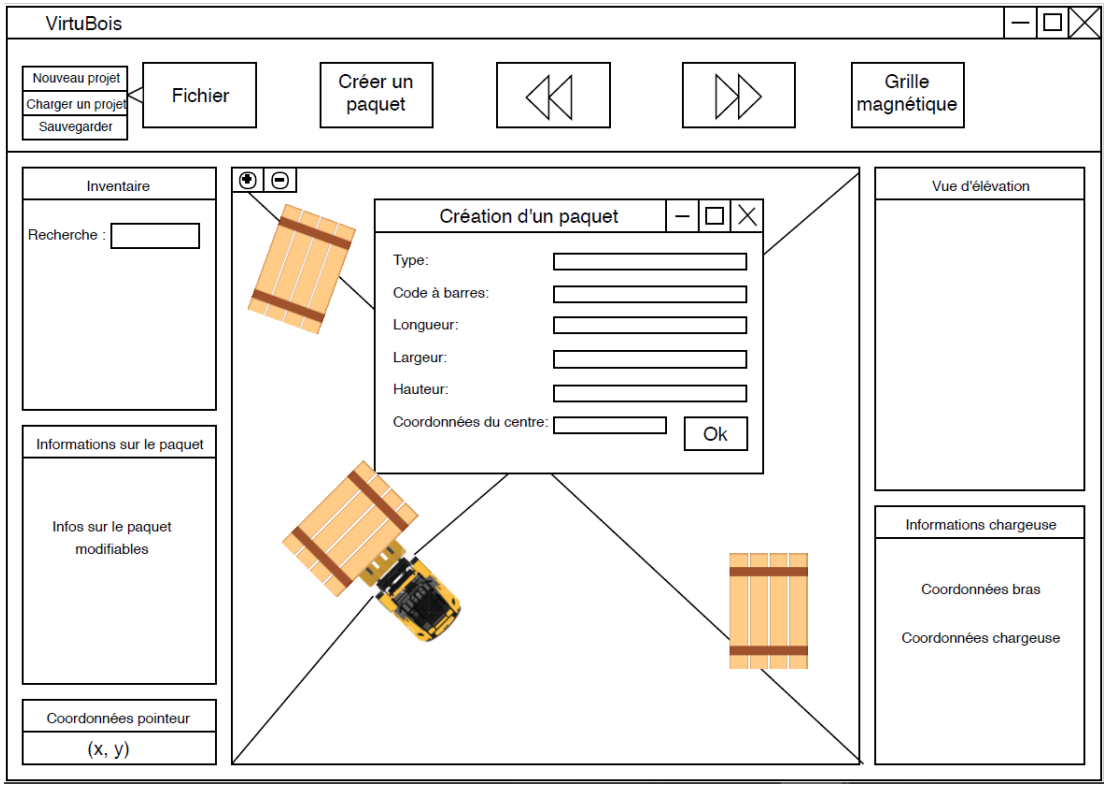


FIGURE D.3 – Interface utilisateur lors de la création d'un paquet

D.4 Interface utilisateur lors la sélection d'une pile de paquets

Lors de la sélection d'une pile par l'utilisateur, le système envoie une vue d'élévation de la pile dans la section correspondante. L'utilisateur peut donc voir les composantes de la pile. Par la suite, via cette section, il peut sélectionner le paquet dont il veut connaitre les informations qui seront affichées dans la section des informations sur le paquet.

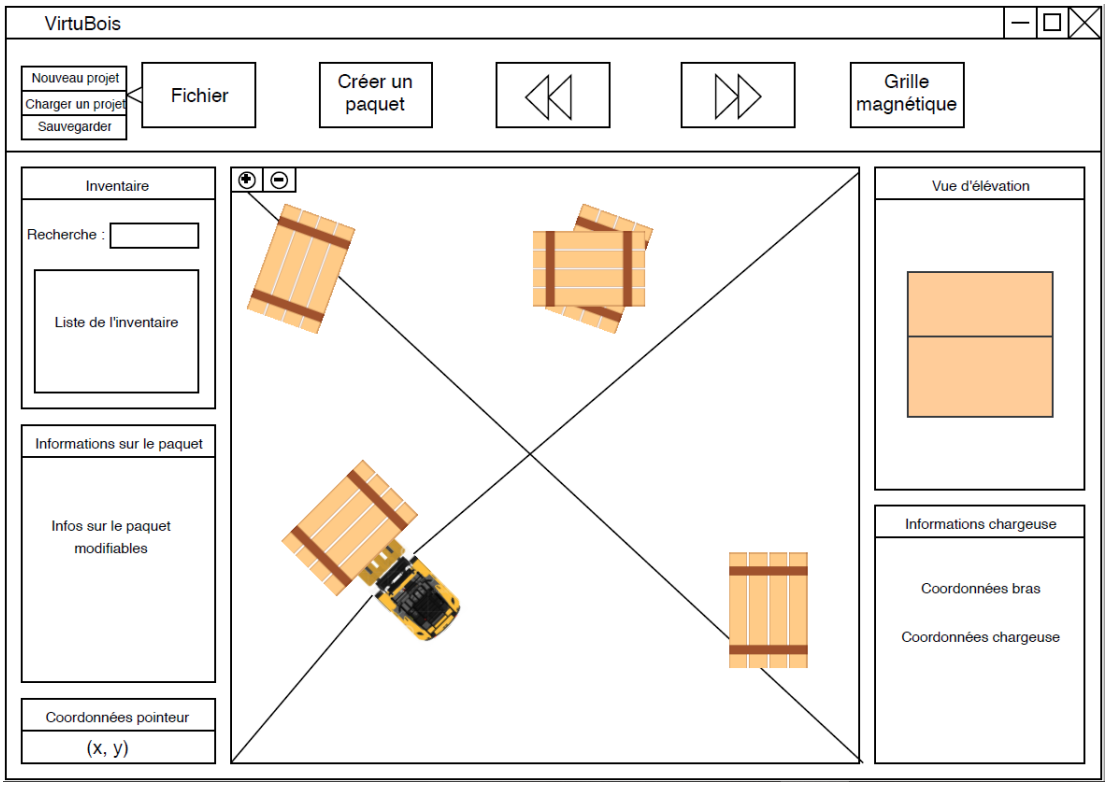
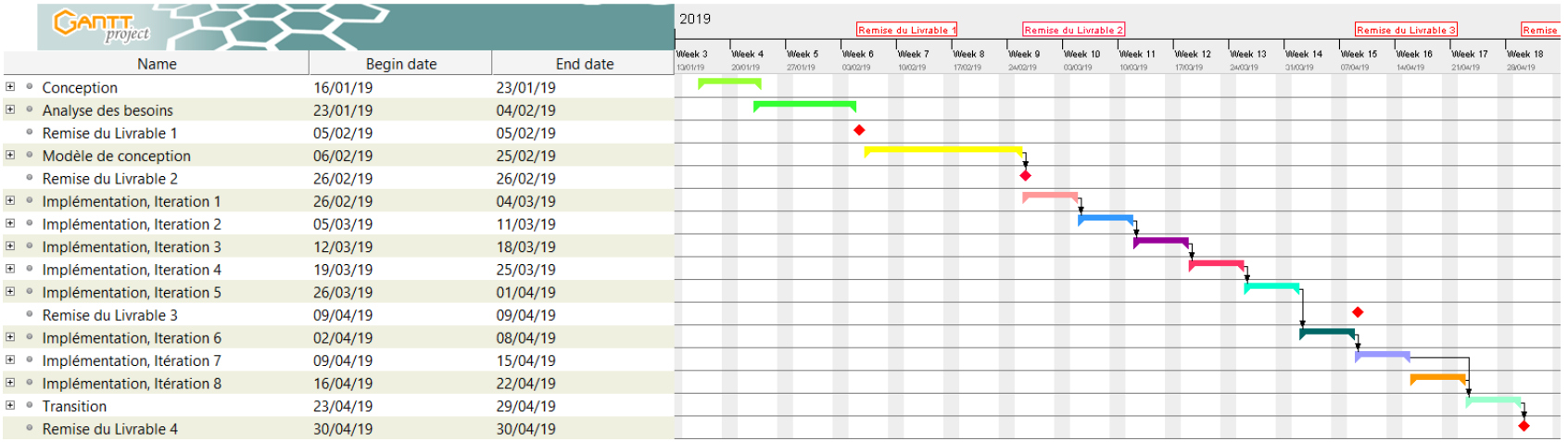


FIGURE D.4 – Interface utilisateur lors la sélection d'une pile de paquets

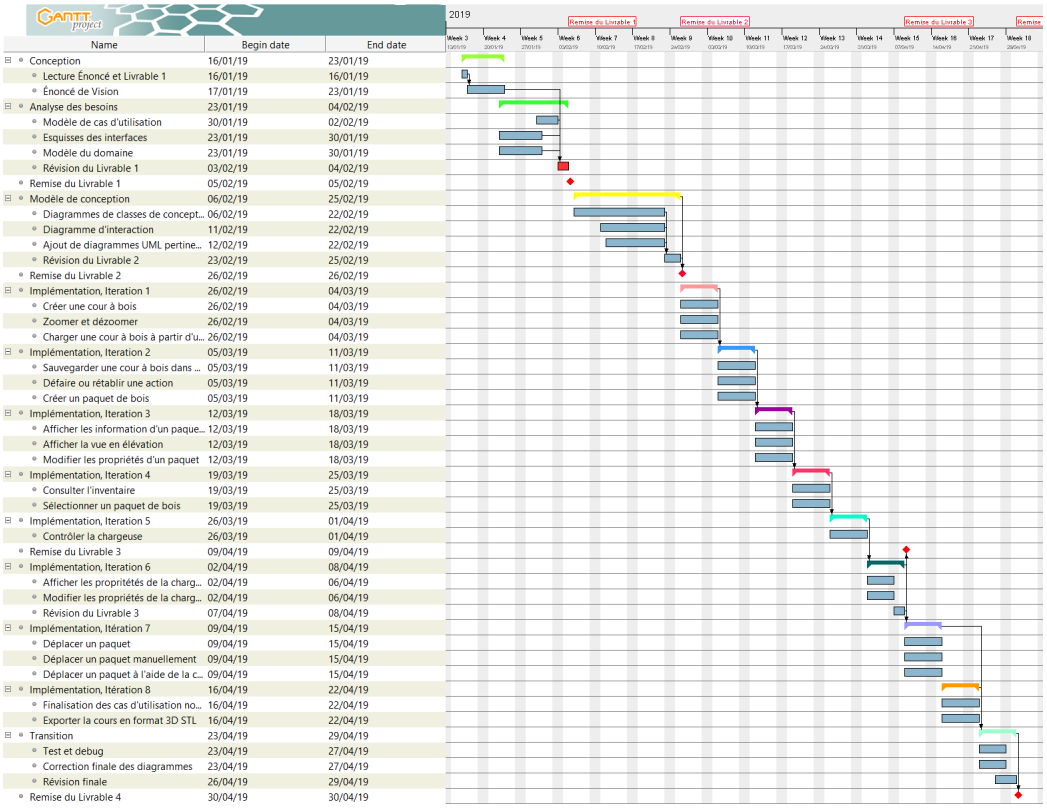
Annexe E

Diagramme de Gantt



Name	Begin date	End date
Conception	16/01/19	23/01/19
Analyse des besoins	23/01/19	04/02/19
Remise du Livrable 1	05/02/19	05/02/19
Modèle de conception	06/02/19	25/02/19
Remise du Livrable 2	26/02/19	26/02/19
Implémentation, Iteration 1	26/02/19	04/03/19
Implémentation, Iteration 2	05/03/19	11/03/19
Implémentation, Iteration 3	12/03/19	18/03/19
Implémentation, Iteration 4	19/03/19	25/03/19
Implémentation, Iteration 5	26/03/19	01/04/19
Remise du Livrable 3	09/04/19	09/04/19
Implémentation, Iteration 6	02/04/19	08/04/19
Implémentation, Itération 7	09/04/19	15/04/19
Implémentation, Itération 8	16/04/19	22/04/19
Transition	23/04/19	29/04/19
Remise du Livrable 4	30/04/19	30/04/19

FIGURE E.1 – Diagramme de Gantt complet



Name	Begin date	End date
Conception	16/01/19	23/01/19
Lecture Énoncé et Livrable 1	16/01/19	16/01/19
Énoncé de Vision	17/01/19	23/01/19
Analyse des besoins	23/01/19	04/02/19
Modèle de cas d'utilisation	30/01/19	02/02/19
Esquisses des interfaces	23/01/19	30/01/19
Modèle du domaine	23/01/19	30/01/19
Révision du Livrable 1	03/02/19	04/02/19
Remise du Livrable 1	05/02/19	05/02/19
Modèle de conception	06/02/19	25/02/19
Diagrammes de classes de concept...	06/02/19	22/02/19
Diagramme d'interaction	11/02/19	22/02/19
Ajout de diagrammes UML pertine...	12/02/19	22/02/19
Révision du Livrable 2	23/02/19	25/02/19
Remise du Livrable 2	26/02/19	26/02/19
Implémentation, Iteration 1	26/02/19	04/03/19
Créer une cour à bois	26/02/19	04/03/19
Zoomer et dézoomer	26/02/19	04/03/19
Charger une cour à bois à partir d'u...	26/02/19	04/03/19
Implémentation, Iteration 2	05/03/19	11/03/19
Sauvegarder une cour à bois dans ...	05/03/19	11/03/19
Défaire ou rétablir une action	05/03/19	11/03/19
Créer un paquet de bois	05/03/19	11/03/19
Implémentation, Iteration 3	12/03/19	18/03/19
Afficher les information d'un paque...	12/03/19	18/03/19
Afficher la vue en élévation	12/03/19	18/03/19
Modifier les propriétés d'un paquet	12/03/19	18/03/19
Implémentation, Iteration 4	19/03/19	25/03/19
Consulter l'inventaire	19/03/19	25/03/19
Sélectionner un paquet de bois	19/03/19	25/03/19
Implémentation, Iteration 5	26/03/19	01/04/19
Contrôler la chargeuse	26/03/19	01/04/19
Remise du Livrable 3	09/04/19	09/04/19
Implémentation, Iteration 6	02/04/19	08/04/19
Afficher les propriétés de la charg...	02/04/19	06/04/19
Modifier les propriétés de la charg...	02/04/19	06/04/19
Révision du Livrable 3	07/04/19	08/04/19
Implémentation, Itération 7	09/04/19	15/04/19
Déplacer un paquet	09/04/19	15/04/19
Déplacer un paquet manuellement	09/04/19	15/04/19
Déplacer un paquet à l'aide de la c...	09/04/19	15/04/19
Implémentation, Itération 8	16/04/19	22/04/19
Finalisation des cas d'utilisation no...	16/04/19	22/04/19
Exporter la cours en format 3D STL	16/04/19	22/04/19
Transition	23/04/19	29/04/19
Test et debug	23/04/19	27/04/19
Correction finale des diagrammes	23/04/19	27/04/19
Révision finale	26/04/19	29/04/19
Remise du Livrable 4	30/04/19	30/04/19

FIGURE E.2 – Diagramme de Gantt complet incluant les tâches détaillées

Annexe F

Contribution des membres de l'équipe

Énoncé de vision : Yoan

Modèle du domaine :

1. Diagramme des classes conceptuelles : Jordan
2. Texte explicatif : Jordan et Martine

Modèle des cas d'utilisation :

1. Diagramme des cas d'utilisation : Gabriel
2. Diagrammes de séquence système : L'équipe
3. Texte des cas d'utilisation : L'équipe

Esquisse des interfaces utilisateur : Martine

Diagramme de Gantt : Yoan

Révision du document : L'équipe