Projet de session VirtuTuile

Livrable n^{o} 2 –

Modèle de conception et architecture logique

IFT-2007 : Analyse et conception des systèmes orientés objets

Automne 2019

Travail présenté à

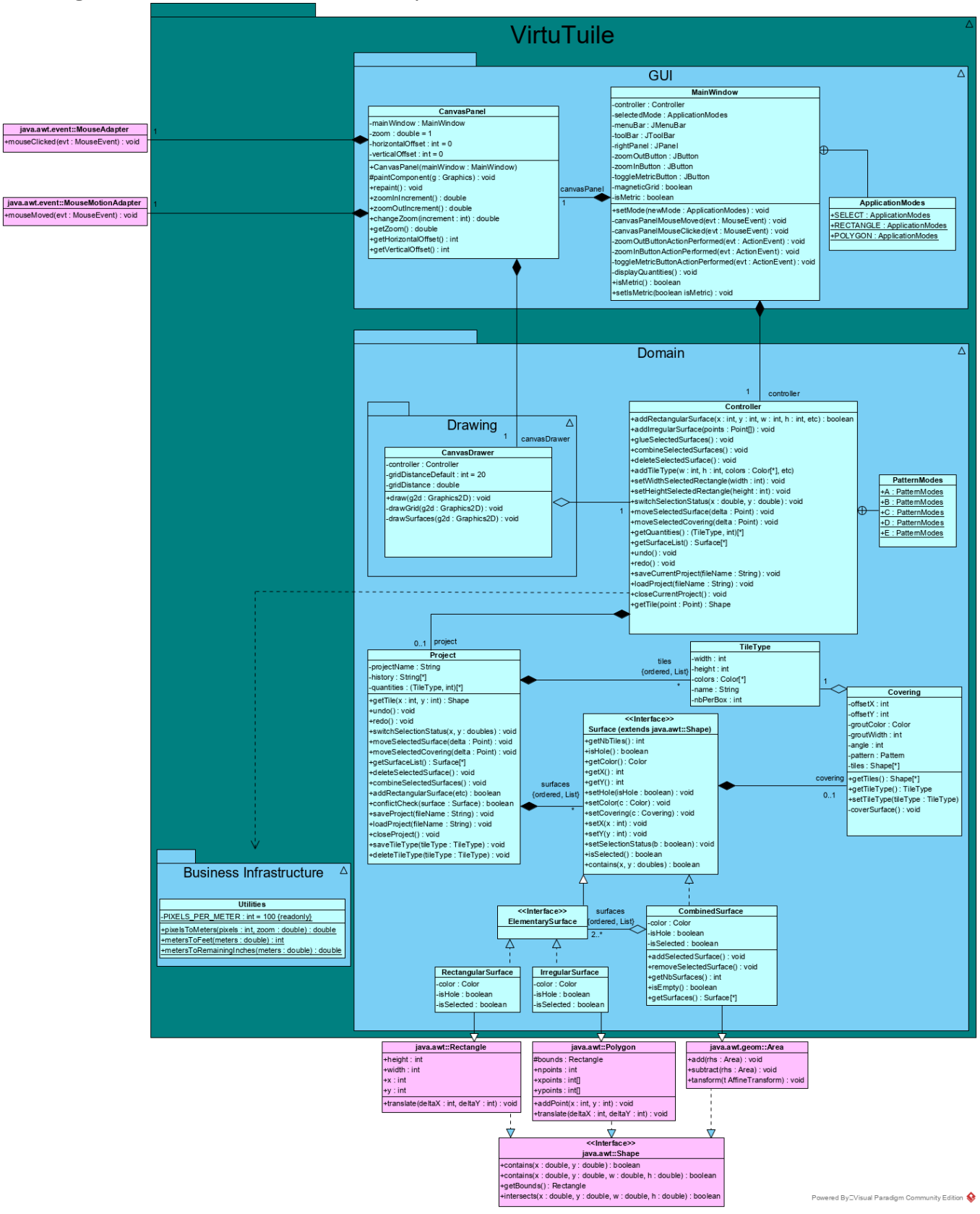
Marc Philippe Parent

Réalisé par

Équipe 8

Nathalie Ponton	nathalie.ponton.1@ulaval.ca	IFT	111 092 836
Gabriel Chevrette-Parrot	gabriel.chevrette-parrot.1@ulaval.ca	IFT	111 091 150
Petros Fytilis	petros.fytilis.1@ulaval.ca	IFT	111 074 144
Martin Sasseville	martin.sasseville.5@ulaval.ca	IFT	990 059 038

1. Diagramme de classe de conception



MainWindow: La fenêtre principale de l'application. Hérite de JFrame. C'est elle qui est responsable de créer tous les autres JComponents, en plus du Controller.

ApplicationModes: Une énumération (enum) de différents modes pour indiquer si l'utilisateur veut sélectionner une surface, créer un rectangle ou créer un polygone. Cette énumération est définie dans le MainWindow.

CanvasPanel: Le canevas de dessin où les surfaces sont dessinées. Hérite de JPanel et est instancié dans le MainWindow.

CanvasDrawer: Il est responsable d'afficher le domaine sur le CanvasPanel en utilisant Graphics et Graphics2D. Il est instancié dans le CanvasPanel et possède une référence au Controller.

Controller: Il contrôle tous les accès entre l'interface graphique et le domaine. C'est sa responsabilité d'appeler les fonctions respectives du domaine lorsque l'utilisateur fait une requête à partir du GUI.

PatternModes: Une énumération (enum) des cinq différents motifs de tuiles. Cette énumération est définie dans le Controller.

Project: Représente le projet courant. Il est composé d'une liste de TileType est d'une liste de Surface. Il est responsable de l'historique des opérations (pour le undo et redo), de la sauvegarde et du chargement de projet.

TileType: Représente un type (donc une description) de tuile.

Covering: Permet de couvrir de tuiles une surface. Il possède une référence à un TileType et est instancié dans une Surface. Après avoir couvert une surface, il enregistre les tuiles sous la forme d'une liste d'objets java.awt::Shape.

Surface: L'interface de toutes nos surfaces. Possède une liste de méthodes qui devront être implémentées dans toutes nos classes de surfaces. Hérite de java.awt::Shape.

ElementarySurface: L'interface d'une surface élémentaire. Elle hérite de Surface.

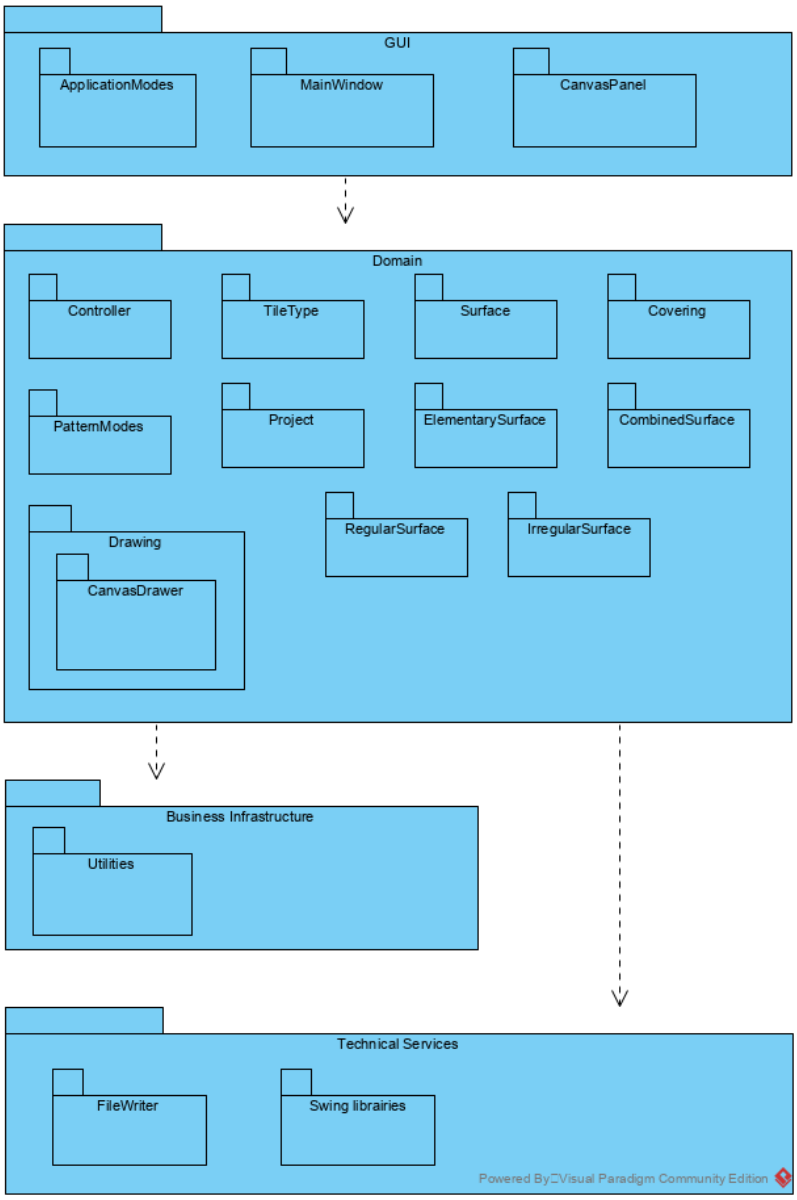
RectangularSurface: Une simple surface rectangulaire. Hérite de java.awt::Rectangle et implémente ElementarySurface.

IrregularSurface: Une surface irrégulière. Hérite de java.awt::Polygon et implémente ElementarySurface.

CombinedSurface: Une combinaison de surfaces élémentaires. Possède une liste de références de ElementarySurface. Hérite de java.awt.geom::Area et implémente Surface.

Utilities: Contient des fonctions statiques utilitaires pour la conversion entre les pixels, les mesures métriques et les mesures impériales.

2. Architecture logique



Le package **GUI** crée la fenêtre principale et les widgets de notre application. Il capture les événements en provenance de la souris et du clavier. Il est donc responsable des entrées et sorties de l'application. Il reçoit les demandes de l'utilisateur (au travers d'entrées utilisateurs) et fait des requêtes au Controller dans le Domain.

Le package **Domain** reçoit les requêtes du GUI à partir du contrôleur et exécute les opérations nécessaires. Il est responsable du projet et gère les surfaces de celui-ci, c'est-à-dire, les surfaces rectangulaire, irrégulière et celle qui sont combinées. Les tuiles disponibles pour le projet sont aussi gérées dans le domaine ainsi que le recouvrement

des surfaces par celle-ci. C'est aussi dans le domaine que sont codés les différents motifs pour la pose des tuiles du projet.

Le package **Drawing** est responsable d'afficher les éléments du domaine sur le GUI.

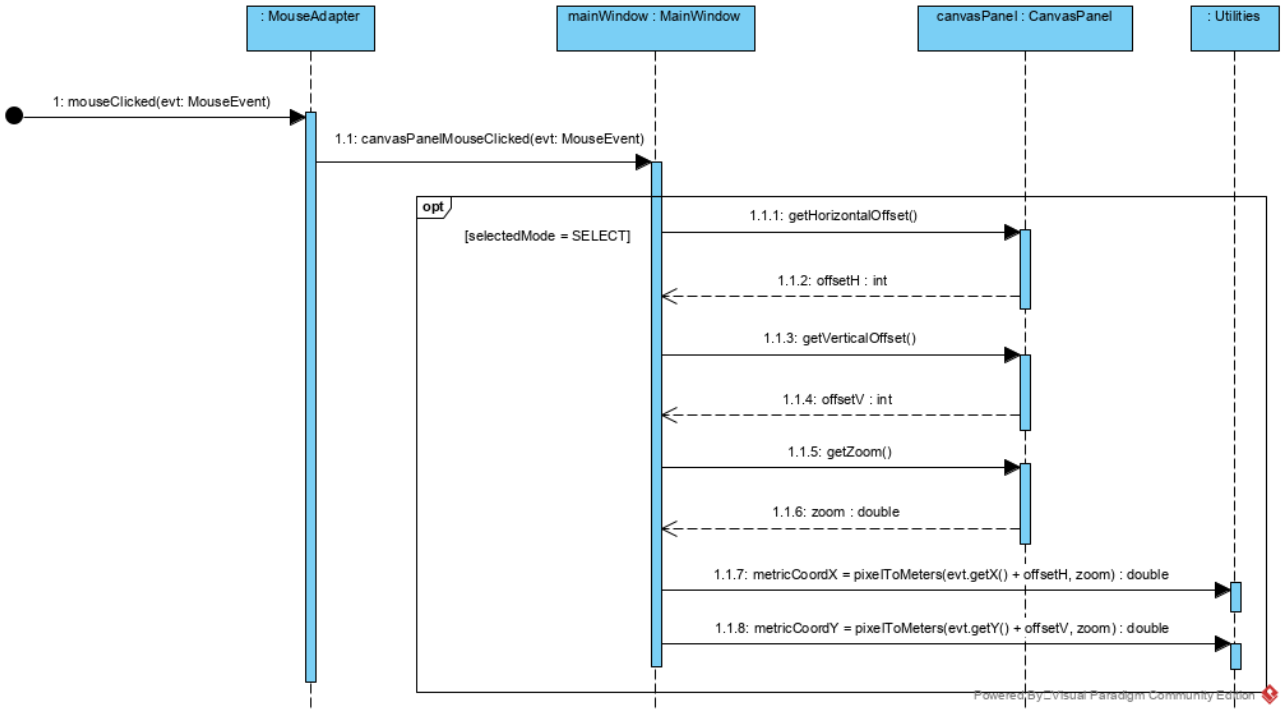
Le package **Business Infrastructure** possède des fonctions utilitaires statiques, soit des dépendances pour le reste de l'application. Par exemple, la formule pour convertir des millimètres en pouces se trouve dans ce package.

Le package **Technical Services** contient des librairies du JAVA API sur lesquels notre application dépend.

3. Diagrammes de séquence de conception

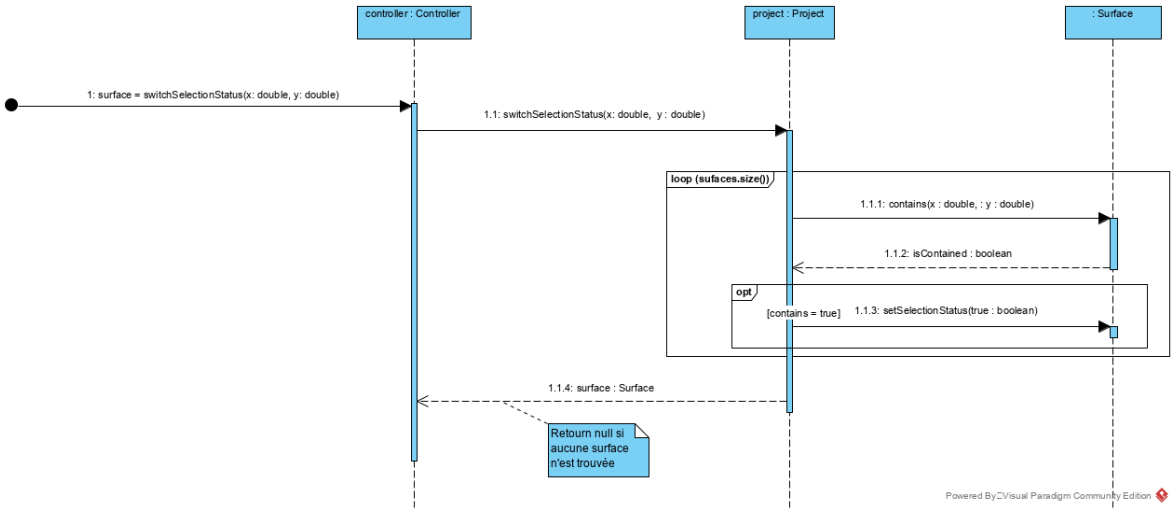
3.1. Déterminer la surface sélectionnée lors d'un clic de souris dans la vue en plan

3.1.1. Appel externe



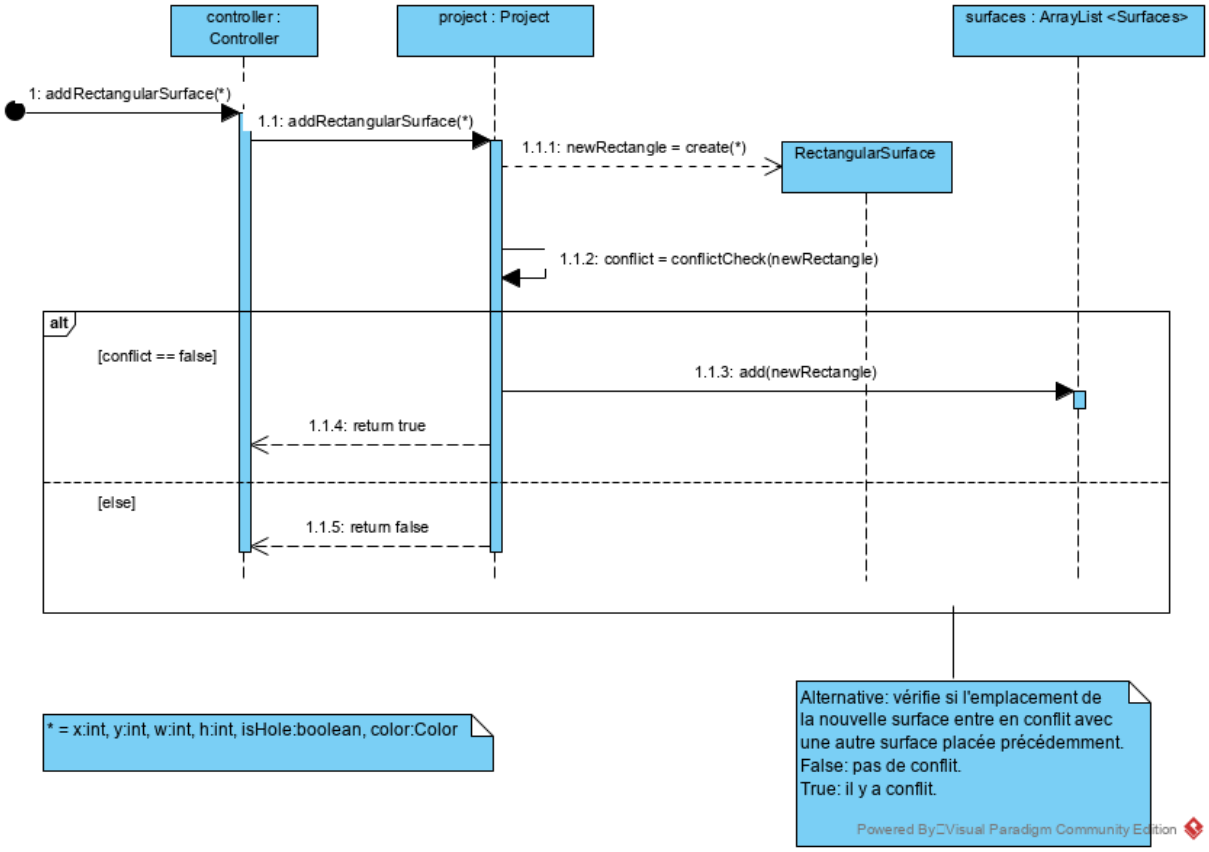
Le diagramme débute par un appel contenant un MouseEvent jusqu'au MouseAdapter. Le MouseAdapter envoie le MouseEvent au MainWindow. Si le mode de la fenêtre est en mode SELECT, le processus continue. Il fait des appels au CanvasPanel pour obtenir les offsets horizontal et vertical du canevas (reliés aux scrollbars), ainsi que le zoom actuel. Il utilise ces informations pour convertir les coordonnées du MouseEvent en mesures métriques, en appelant les fonctions statiques du module Utilities.

3.1.2. Appel au contrôleur



Le diagramme débute par un appel au Controller contenant les mesures métriques. Le Controller envoie ces mesures au Project. Le Project fait alors une boucle pour chaque surface de sa liste de surfaces. Si le point x, y est contenu dans la surface, il change l'état de sélection de la surface. Sinon, il ne fait rien. Au final il retourne la surface sélectionnée (s'il y a lieu) au Controller qui la retourne à l'appelant initial.

3.2. Création d'une nouvelle surface rectangulaire

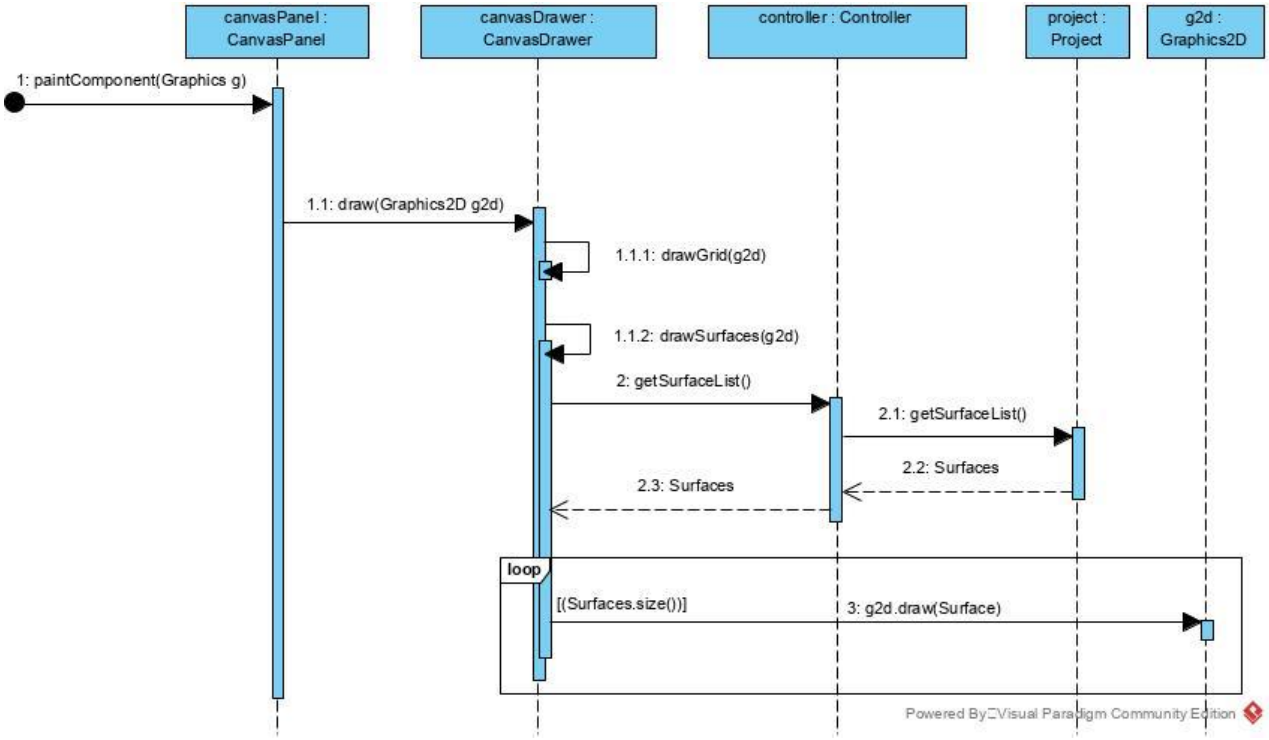


Les valeurs de tous les paramètres nécessaires à la création de la nouvelle surface rectangulaire ont été définies via des interactions au niveau de la couche interface-utilisateur.

L'entrée du diagramme se fait par la fonction **addRectangularSurface(x:int, y:int, w:int, h:int, isHole:boolean, color:Color)** du contrôleur. Le contrôleur envoie la demande au Project qui s'occupe de créer le nouveau rectangle.

Ensuite, il y a vérification pour s'assurer que cette nouvelle surface rectangulaire ne s'ajoute pas sur un emplacement où il y a déjà une surface. S'il n'y a pas de conflit, cette nouvelle surface sera ajoutée à la liste des surfaces de ce projet (une ArrayList) et un message de retour sera envoyé au contrôleur pour l'informer de la réussite. S'il y a conflit, une valeur false est retournée au contrôleur pour l'informer que la surface ne peut pas être créée et la surface cessera d'exister.

3.3. L'affichage de la vue en plan



La procédure d'affichage est déclenchée, suite à l'ajout/modification ou le déplacement d'un objet, par l'événement **paintComponent(),** de la classe CanvasPanel, qui appelle le CanvasDrawer pour commencer la procédure d'affichage. L'afficheur CanvasDrawer commence par dessiner la grille sur le canevas, puis dessine les surfaces via les fonctions **draw(Graphics2D g2d)**, **drawGrid(g2d)** et **drawSurfaces(g2d)**.

Pour obtenir la liste des surfaces du projet, il fait une requête (**getSurfaceList()**) au contrôleur qui est envoyée au Projet. Une boucle sera ainsi initiée jusqu'à l'obtention et à l'affichage de toutes les surfaces du projet.

4- Pseudo-code pour déterminer si un point est à l'intérieur d'une surface irrégulière

La méthode que nous avons choisie consiste à tirer une droite entre le point à vérifier et un point extérieur au polygone, puis à vérifier combien de fois cette droite croise les côtés du polygone. Si ce nombre est pair, le point est à l'extérieur. Si ce nombre est impair, le point est à l'intérieur du polygone.

Cette méthode est contains(x:double, y:double) : boolean, elle est définie dans l'interface Surface.

Voici le pseudo-code :

Description: Vérifier si le point fait partie du polygone courant.

Entrée : les coordonnées du point à vérifier.

Sortie : true si le point est dans la surface, false sinon.

```
DÉBUT
    // Créer un compteur pour le calcul du nombre de croisements initialisé à 0.
    NombreDeCroisements <- 0
    // Créer une droite de départ entre le point à vérifier et le point extérieur
    // du polygone (-1, -1).
    Droite <- creerDroite entre (x, y) et (-1, -1)

    // Vérifier si le point est dans la liste des sommets de la surface. Si c'est
    // le cas, retourne true.
    RÉPÉTER pour chaque sommet de la surface
    DÉBUT
        SI sommet = (x, y)
```

ALORS

Retourne true

FIN

RÉPÉTER pour chaque arête de la surface

// Si le point se trouve sur un côté de la surface, il y aura croisement.

// Pour chaque paire de sommets se suivant dans la liste des sommets de la

// surface : Crée une droite entre ces sommets. Vérifier si celle-ci croise la droite

// de départ. Si c'est le cas, on incrémente le compteur.

// Créer une droite entre le premier et le dernier point de la liste des

// sommets. Vérifier si celle-ci croise la droite de départ. Si c'est le cas,

// on incrémente le compteur aussi.

DÉBUT

SI intersection entre arête et droite

ALORS

NombreDeCroisements <- NombreDeCroisements + 1

FIN

// Si le compteur est un nombre impair : Retourne true puisque le point est à

// l'intérieur de la surface.

// Si le compteur est un nombre pair : Retourne false puisque le point est à
// l'extérieur de la surface.
SI NombreDeCroisements est impair

ALORS

Retourne true

SINON

Retourne false

FIN

5. Diagramme de Gantt

Itération	1			2			3			4			5			
Créer/Éditer un type de tuile																
Créer surface rectangulaire																
Créer surface non rectangulaire																
Sélectionner/Désélectionner surface																
Courvrir surface																
Déplacer surface																
Supprimer surface																
Combiner des surfaces																
Coller deux surfaces																
Décaler le motif sur la surface																
Calculer les quantités requises																
Inspecter les tuiles																
Détecter petites tuiles																
Changer le système de mesure																
Undo/Redo																
Changer le niveau de zoom																
Activer/Désactiver grille magnétique																
Nouveau projet																
Ouvrir projet																
Fermer projet																
Enregistrer projet																
Semaine	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
				Liv. 1			Liv. 2						Liv. 3			Liv. 4

6. Justification de la contribution de chacun

Élément	Réalisé par	Révisé par
Diagramme de classe de conception	Petros	Toute l'équipe
Architecture logique	Nathalie	Toute l'équipe
DSC surface sélectionnée	Gabriel	Toute l'équipe
DSC nouvelle surface rectangulaire	Nathalie	Toute l'équipe
DSC affichage	Martin	Toute l'équipe
Pseudo-code	Nathalie	Toute l'équipe
Mise à jour diagrammes Livrable 1	Martin	Toute l'équipe
Programmation Java: création de l'interface, fonction zoom	Petros	Toute l'équipe
Programmation Java: création menus	Martin	Toute l'équipe
Programmation Java: classes domaine	Gabriel	
Diagramme de Gant	Martin	Toute l'équipe
Colliger les questions/réponses du forum	Martin	N/A

Annexes

Énoncé de vision

Nous voulons concevoir une application qui aidera les utilisateurs à planifier la pose de revêtement sur leurs planchers et murs. L'application offrira des outils aux utilisateurs qui leur permettra de modéliser ces surfaces en détail. L'application sera en mesure d'automatiquement simuler le recouvrement de ces surfaces par des tuiles dans différents motifs, puis de calculer le nombre de tuiles nécessaires à leur projet et, en fin de compte, le coût total associé à l'achat de ces tuiles.

Le logiciel VirtuTuile pourra être offert aux clients de détaillants de tuiles, ce qui leur permettra de commencer leur planification de rénovation de manière autonome, sans devoir ni rencontrer un professionnel ni se déplacer de leur chez-soi. L'application pourra également être disponible aux professionnels pour être utilisé dans leur travail.

Cette approche est fort intéressante pour deux raisons majeures :

• D'une première part, elle prend avantage de la tendance culturelle DIY («do-it-yourself»). De nos jours de plus en plus de gens désirent accomplir leurs projets de manière autonome pour 1) sauver de l'argent, 2) apprendre de nouvelles compétences et 3) en retirer plus de satisfaction personnelle. Nous avons ainsi une forte raison de croire que le produit sera populaire chez la clientèle des détaillants.

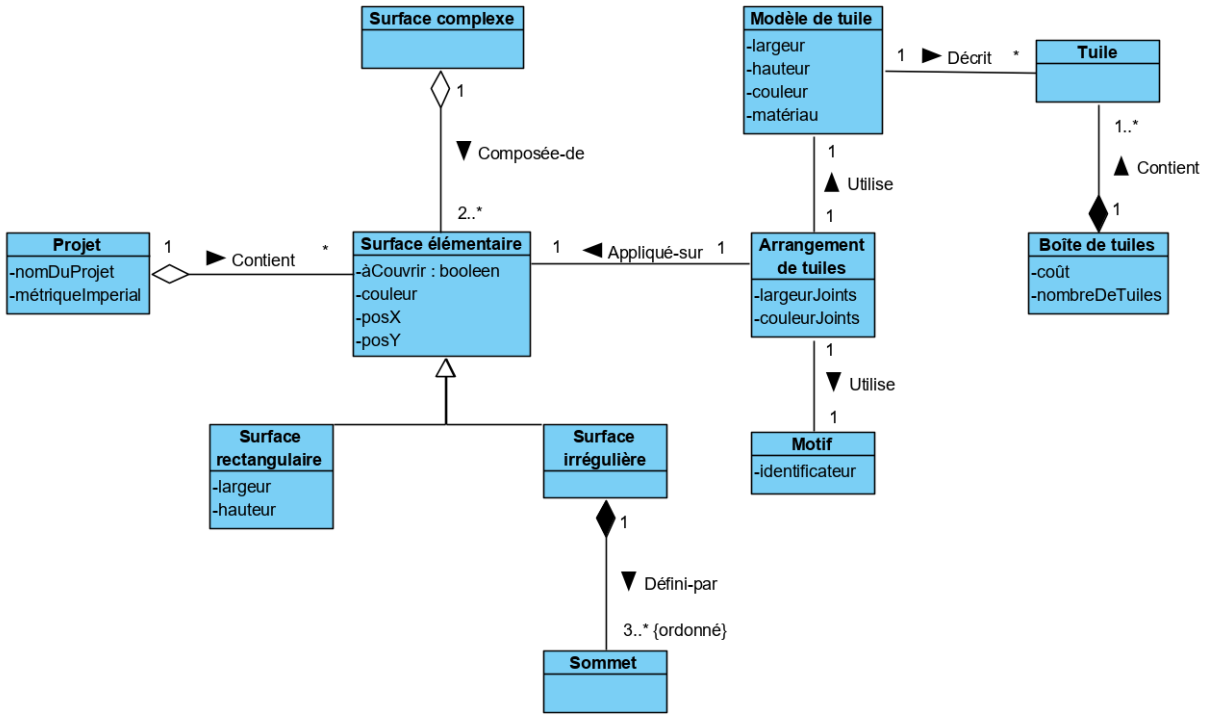
• D'autre part, le logiciel pourra ainsi se voir très utile aux détaillants en ligne, donc qui ne possèdent pas d'emplacement physique. Ces détaillants n'ont pas d'employés qui peuvent aider leurs clients sur place. Pouvoir offrir un tel logiciel à leur clientèle leur sera ainsi particulièrement avantageux.

Le logiciel sera accompagné d'une banque de tuiles représentant les divers types de tuiles offertes par le détaillant (leurs dimensions, matériaux, couleurs, prix, etc.) Cette banque pourra être facilement mise-à-jour par les détaillants pour y ajouter leurs nouveaux produits. Les utilisateurs n'auront ainsi qu'à télécharger une version mise-à-jour du fichier en question. De plus, les utilisateurs pourront eux-mêmes y ajouter leurs propres modèles de tuiles, s'ils le désirent.

Cette banque facilement personnalisable nous permettra ainsi de vendre la même application à plusieurs détaillants avec un minimum de modifications de notre part. Il s'agit du même logiciel accompagné d'une banque personnalisée au détaillant.

Modèle du domaine

Nous avons ajouté de nouveaux attributs à la classe conceptuelle de Surface élémentaire, soit la couleur et la position X et Y.



Glossaire

Un **projet** englobe toute la planification de revêtement d'une ou de plusieurs surfaces. Il est identifiable par un nom de projet et peut se faire soit en système de mesure métrique ou impérial.

Un projet est ainsi une agrégation de **surfaces élémentaires**, soit des surfaces de plancher ou de mur. Chaque surface possède un attribut qui indique si elle doit être couverte ou pas.

Plus spécifiquement, une surface élémentaire peut soit être une **surface rectangulaire** ou une **surface irrégulière**. Une surface rectangulaire est définie par sa hauteur et sa largeur, tandis qu'une surface irrégulière est définie par ses sommets ordonnés.

Un **sommet** est un simple point cartésien.

Une **surface complexe** est une agrégation d'au moins deux surfaces élémentaires qui ont été combinées ensemble.

Une **tuile** est un objet de forme rectangulaire qui sert à couvrir des surfaces de plancher ou de mur.

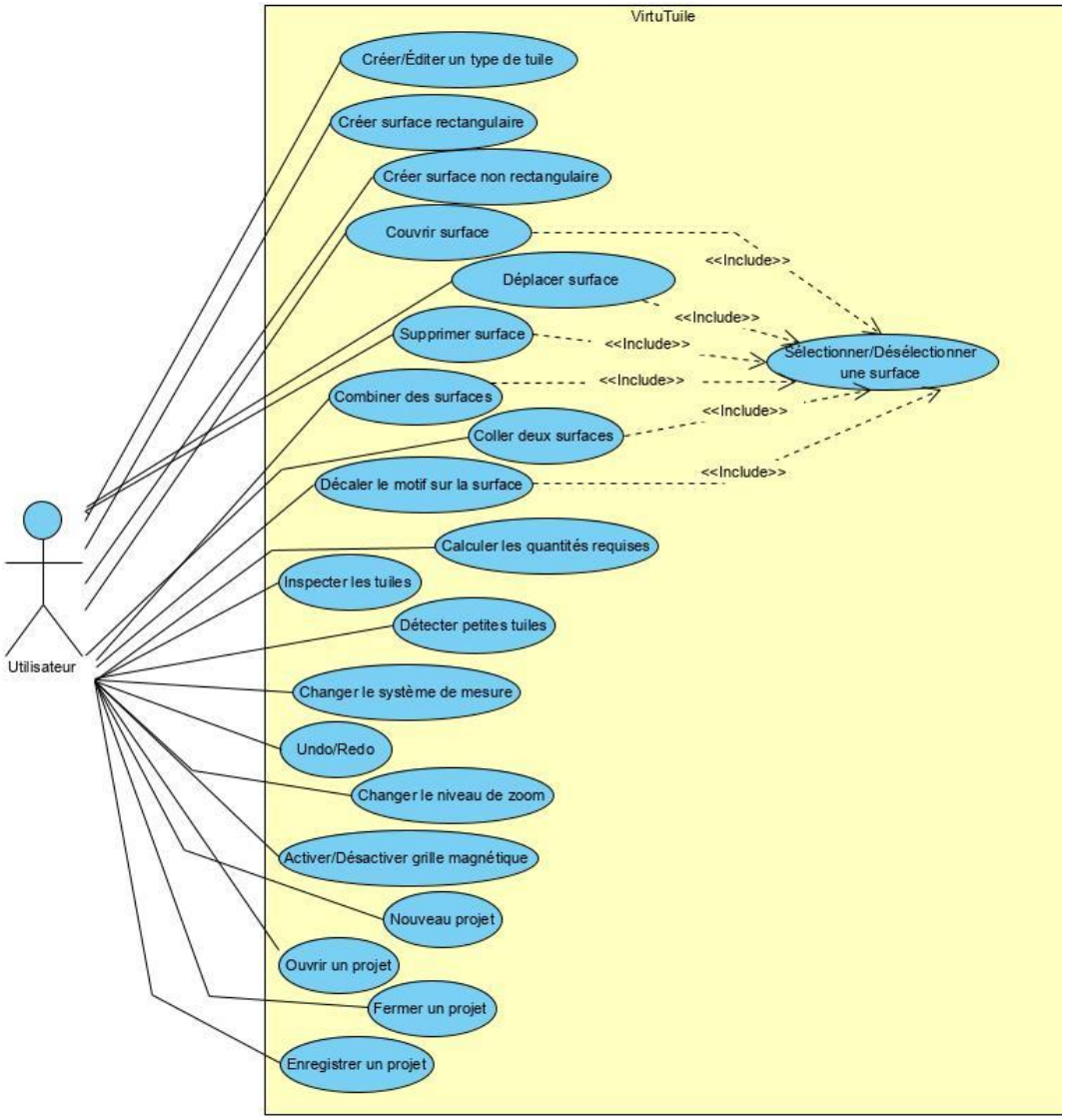
Une **boîte de tuiles** est un ensemble de tuiles qui peut être acheté à un certain prix. Elle contient un nombre prédéterminé de tuiles d'un modèle précis.

Un **modèle de tuile** sert à décrire certaines tuiles. Il possède comme attributs une hauteur et une largeur (soit les dimensions de la tuile), une couleur et un matériau (e.g. céramique).

Un **motif** correspond à la façon d'arranger des tuiles ensemble. Nous travaillons avec 5 différents motifs, qui ont chacun un identificateur alphabétique.

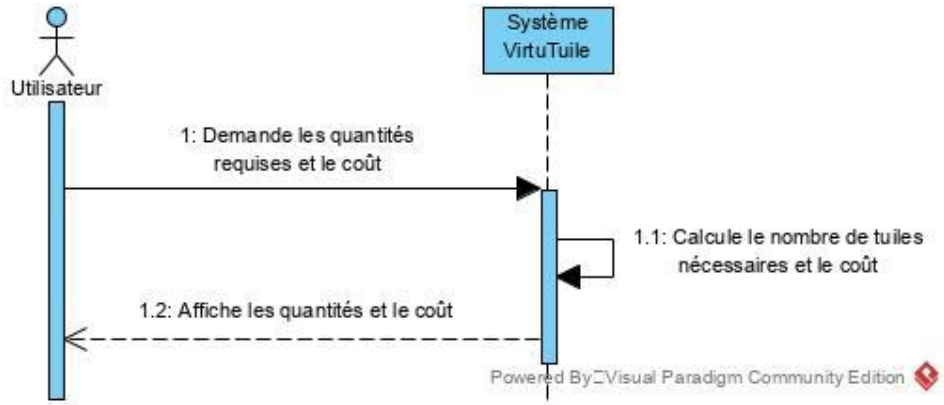
Un **arrangement de tuiles** représente la manière précise qu'une surface de travail est couverte. Il utilise un certain modèle de tuile et un certain motif, puis est appliqué sur une certaine surface. Il possède également, comme attributs, une largeur de joints (soit l'espace entre chaque tuile individuelle) et une couleur de coulis.

Diagramme des cas d'utilisation

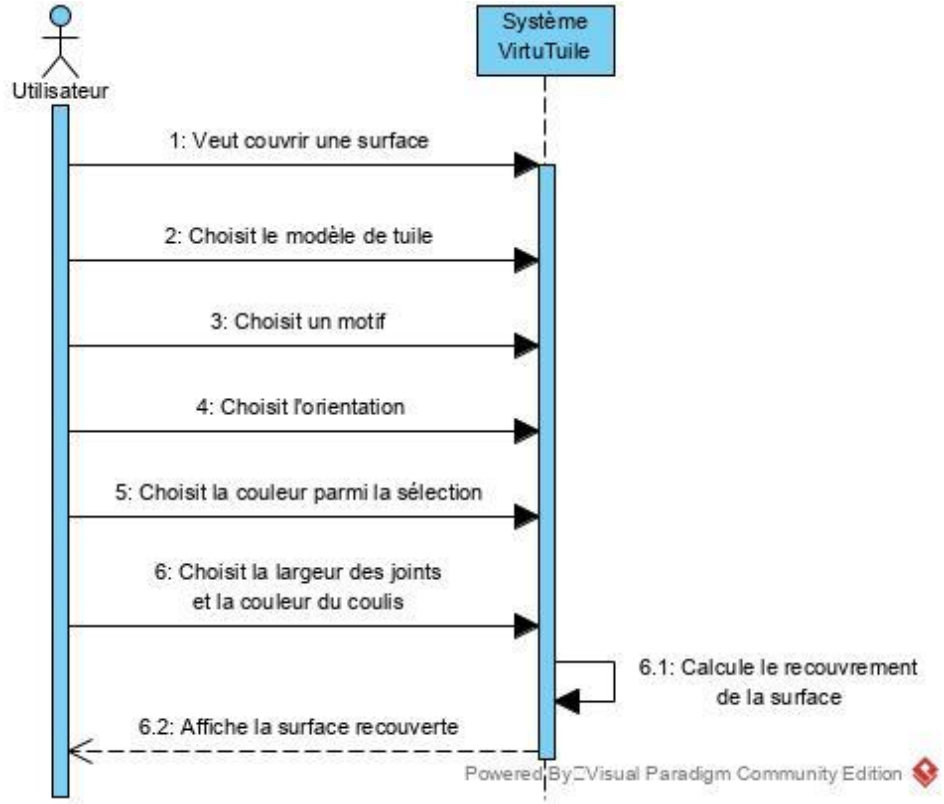


Mise à jour des diagrammes de séquence du livrable 1.
Changement de la flèche de retour du système pour une flèche pointillée.

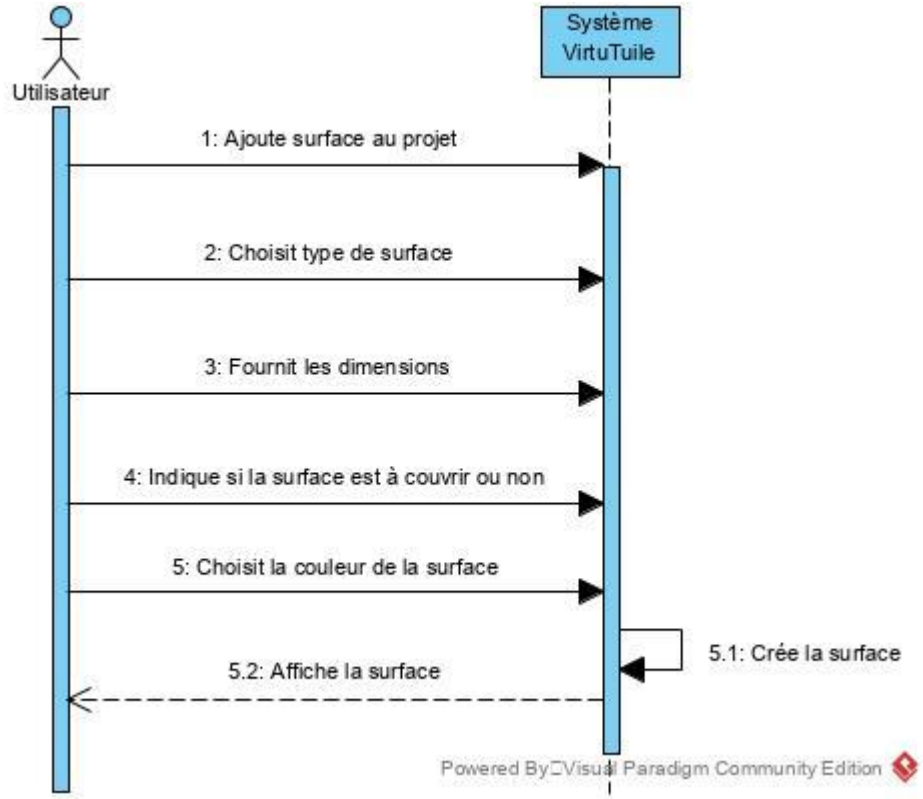
Calculer les quantités requises:



Couvrir une surface:



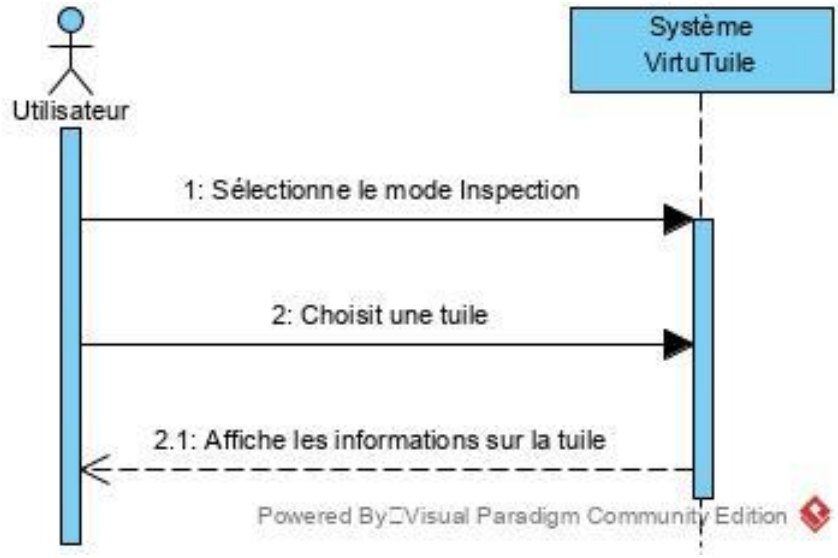
Créer une surface:



Ajout de 3 cas d'utilisation non présents dans le livrable 1.

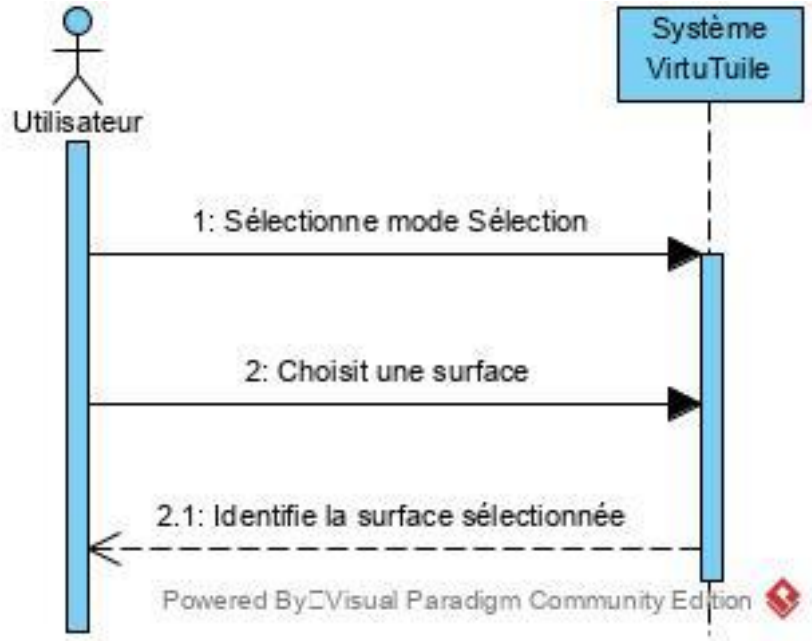
Cas d'utilisation:	Inspecter les tuiles
Système:	VirtuTuile
Acteur:	Utilisateur
Parties prenantes et intérêts:	Utilisateur: Il veut obtenir les informations sur une tuile en particulier.
Préconditions:	Un projet de travail est ouvert et contient des surfaces.
Garanties en cas de succès:	L'application affiche les informations de la tuile sélectionnée et indique les petites tuiles.
Scénario principal:	1. L'utilisateur sélectionne le mode Inspection. 2. L'utilisateur

	choisit une tuile sur la surface. 3. Les informations sur la tuile sont affichées.
Scénarios alternatifs:	N/A



Cas d'utilisation:	Changement du système de mesure
Système:	VirtuTuile
Acteur:	Utilisateur
Parties prenantes et intérêts:	Utilisateur: Il veut avoir les mesures en métrique ou en impériale.
Préconditions:	Un projet de travail est ouvert et contient des surfaces.
Garanties en cas de succès:	Les mesures sont affichées selon le mode choisi.
Scénario principal:	1. L'utilisateur sélectionne le mode Mesure métrique. 2. Les mesures sont affichées en mètre.
Scénarios alternatifs:	1a. L'utilisateur sélectionne le mode Mesure impériale.

Cas d'utilisation:	Sélectionner/Désélectionner une surface
Système:	VirtuTuile
Acteur:	Utilisateur
Parties prenantes et intérêts:	Utilisateur: Il veut sélectionner une surface pour pouvoir agir dessus ou désélectionner une surface.
Préconditions:	Un projet de travail est ouvert et contient des surfaces.
Garanties en cas de succès:	La surface voulue est sélectionnée et identifiée comme tel ou désélectionnée.
Scénario principal:	1. L'utilisateur sélectionne le mode Sélection. 2. L'utilisateur choisit une surface. 3. La surface est identifiée comme sélectionnée.
Scénarios alternatifs:	2a. L'utilisateur désélectionne la surface.


Utilisateur
Système VirtuTuile
1: Sélectionne mode Sélection
2: Choisit une surface
2.1: Identifie la surface sélectionnée
Powered By Visual Paradigm Community Edition