Catherine Simard
111 234 718
Vincent Chavanel Jobin
111 091 286
Léo Guérin-Morneau
111 238 682
Jean-Sébastien Nantel
111 180 832

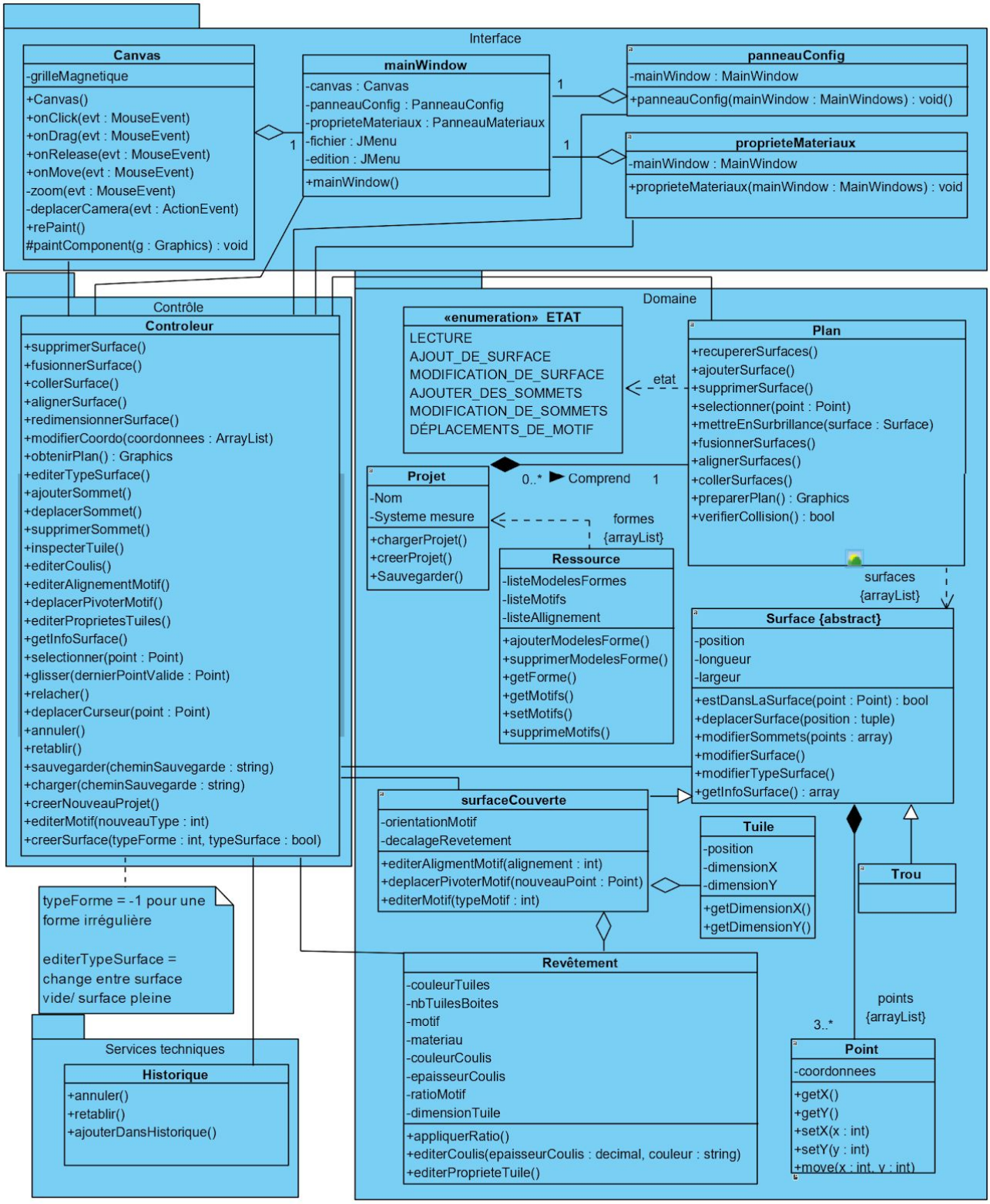
Génie logiciel orienté objet
GLO-2004

Livrable #2

Travail présenté à
Jonathan Gaudreault

Université Laval
Automne 2019

1. Diagramme de classe de conception



L'objet **mainWindow** sera composé du **panneauConfig** et du menu **proprieteMateriaux**. Il contiendra également l'objet **Canvas**, qui correspond à l'afficheur et qui affiche le plan.

Lorsque l'utilisateur utilise un bouton du menu, la méthode correspondante est appelée dans le **contrôleur**, qui relaye l'information à la classe correspondante du domaine. Quand le canvas perçoit une action, comme un clic ou un glissement de la souris, les coordonnées décrivant cette action sont retransmises au contrôleur, par exemple avec un appel à la méthode sélectionner ou glisser. Le contrôleur appelle ensuite les méthodes du plan avec ces coordonnées, et c'est le **plan** qui détermine la façon dont il interprète ces actions pour modifier les **surfaces** qui le composent. Cette interprétation dépend principalement de **l'état** dans lequel le plan est, tel que le mode de création de surface ou de modification des sommets. Ce mode est habituellement changé par les boutons du menu.

Les **surfaces couvertes** et les **trous** hériteront tous deux de la classe abstraite **Surface**, et leurs dimensions seront définies par des objets **Point**.

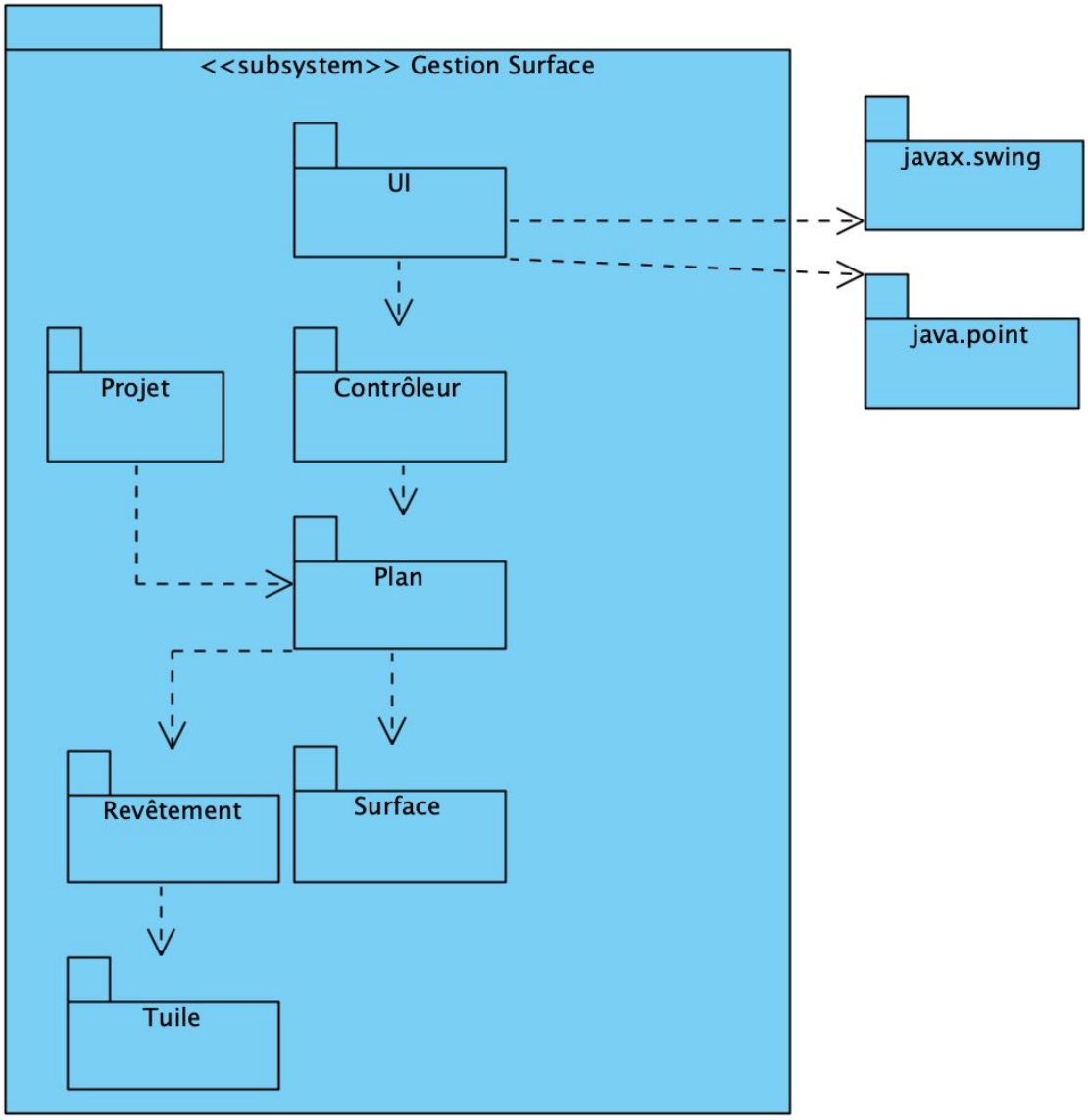
Le client a accès à une banque de **revêtements** pour recouvrir les surfaces. Un revêtement englobe un motif, les dimensions des tuiles, leur couleur, la couleur du coulis et l'épaisseur du coulis. L'utilisateur peut ajouter des revêtements à cette banque et en modifier. Si le client recouvre plusieurs surfaces avec le même revêtement, ces surfaces auront une référence au même objet revêtement dans leurs attributs.

Une fois un revêtement appliqué à une surface à couvrir, celle-ci contiendra des objets **tuiles** qui pourront être consultés individuellement dans le détail par l'utilisateur.

Un objet **ressource** sera partagé entre chaque projet et chargé à partir d'un fichier. Celui-ci contiendra initialement les revêtements, motifs et formes par défaut. Y seront ajoutés ceux créés par l'utilisateur. L'objet **projet** contient quand à lui les informations pertinentes à la sauvegarde d'un projet individuel.

Les différentes actions de l'utilisateur seront sauvegardés dans un objet **historique**, qui permettra de revenir en arrière ou de rétablir ces actions.

2. Architecture logique



Au niveau de l'architecture logique de notre application, on retrouve un seul "subsystem" que nous avons appelé "gestion de surfaces" qui, comme son nom l'indique est responsable de l'ensemble de la gestion des surfaces (incluant les tuiles et les motifs qui sont contenus dans ceux-ci).

On commence par notre package "UI" qui consiste à des méthodes utiles à la gestion de l'interface et qui dépend des classes Swing et Point provenant de Java. Ces classes permettent non seulement l'affichage des surfaces et leur modification, mais la gestion

des divers paramètres du programme. Ce package contient les classes : Canvas, MainWindow, PanneauConfig et PanneauMateriaux.

Nous retrouvons ensuite le package “Contrôleur” qui contient seulement la classe “Contrôleur”. Ce package sera utilisé pour permettre la communication entre l’interface et le package “Plan”. Ce package sera également responsable de stocker les surfaces et leurs éléments.

Par la borne, on retrouve le package “Projet” qui contient les classes : Projet, Historique et Ressources. Ce package contient tous les éléments nécessaires à la bonne gestion d’un projet contenant des plans, mais aussi les outils pour manipuler certains aspects techniques du projet. Il contient également les ressources pouvant être utilisé par le plan (surfaces par défaut, motifs, etc …).

Ensuite, il y a le package “Surface” qui est contient les classes responsables de la gestion des différents types de surfaces. Ce package est également dépendant du package “Point” de Java pour les positions.

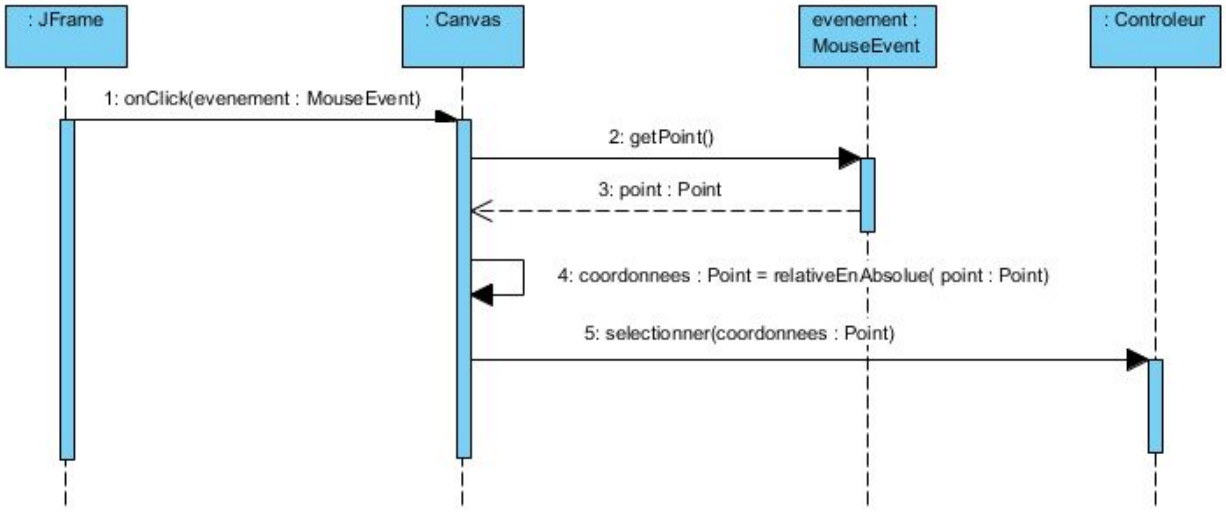
Par après, il y a le package “Revêtement” et “tuile”. Le package tuile, pour sa part, ne contient que le classe “Tuile”. Ce package fournit les méthodes pour la gestion des tuiles : dimension et positions.

Finalement, le package revêtement contient l’ensemble des méthodes nécessaires à la gestion des motifs, matériau, la quantité de tuiles et le coulis. À noter que le package “Revêtement” calcul le nombre de tuiles qu’en fonction de la taille déterminée par l’utilisateur : il ne contient pas de références aux tuiles elles-mêmes.

3. Diagramme de séquence de conception

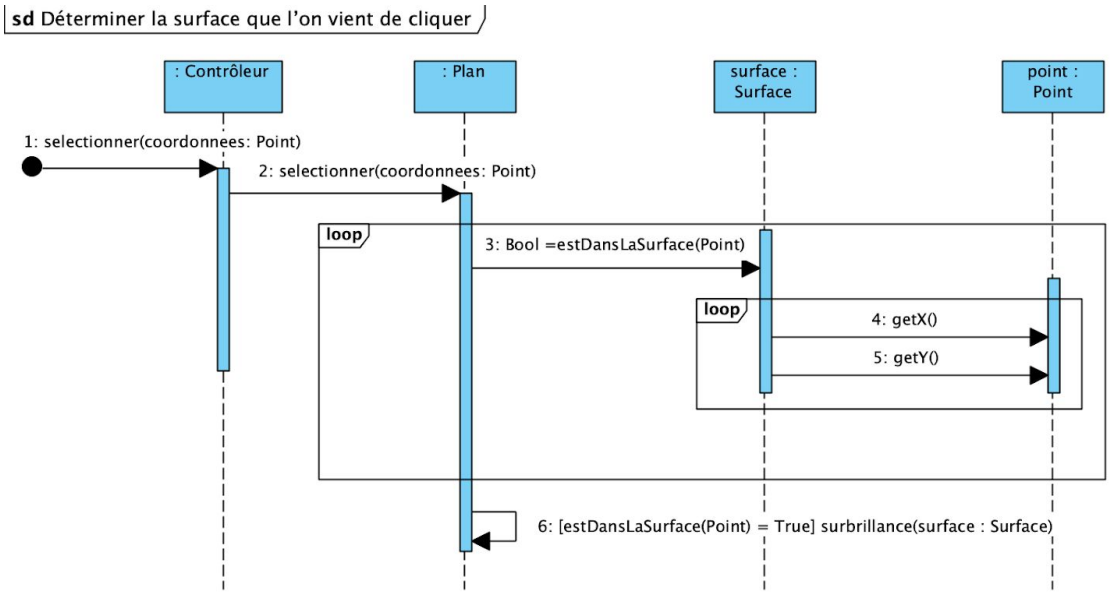
3.1 Déterminer la surface sélectionné lors d'un clic de la souris dans la vue en plan

3.1.1 Déterminer les coordonnées correspondant au clic de souris



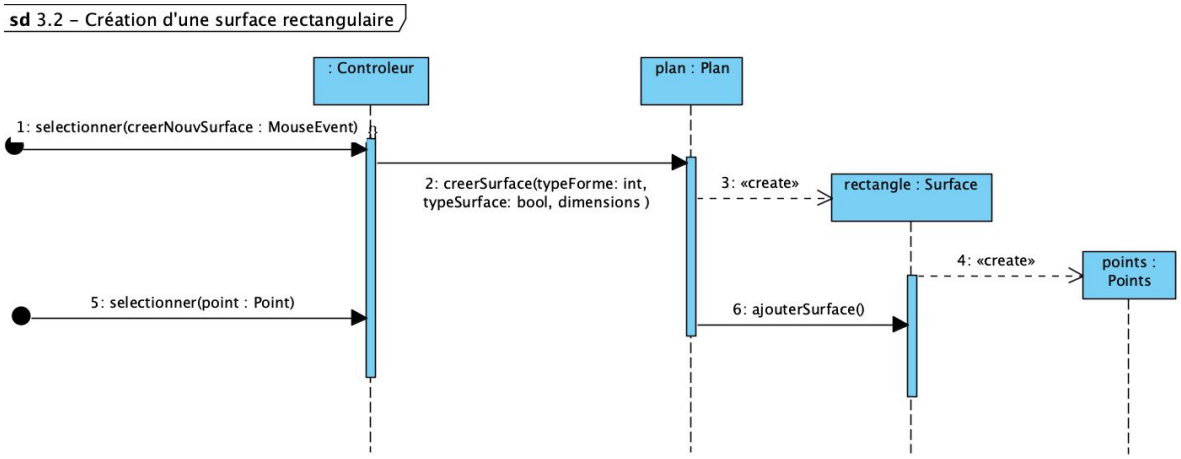
Lors d'un clic sur la vue en plan, un événement est envoyé à la classe Canvas sous la forme d'un MouseEvent. La classe canvas appelle la méthode getPoint de la classe MouseEvent afin de retourner les coordonnées du clic dans la fenêtre. Le point retourné est ensuite passé en paramètre à la méthode relativeEnAbsolue() qui convertit une coordonnée obtenue dans le plan en coordonnée absolue dans l'unité de mesure sélectionnée. La coordonnée obtenue est alors envoyé vers le contrôleur.

3.1.2 Déterminer la surface sur laquelle on vient de cliquer



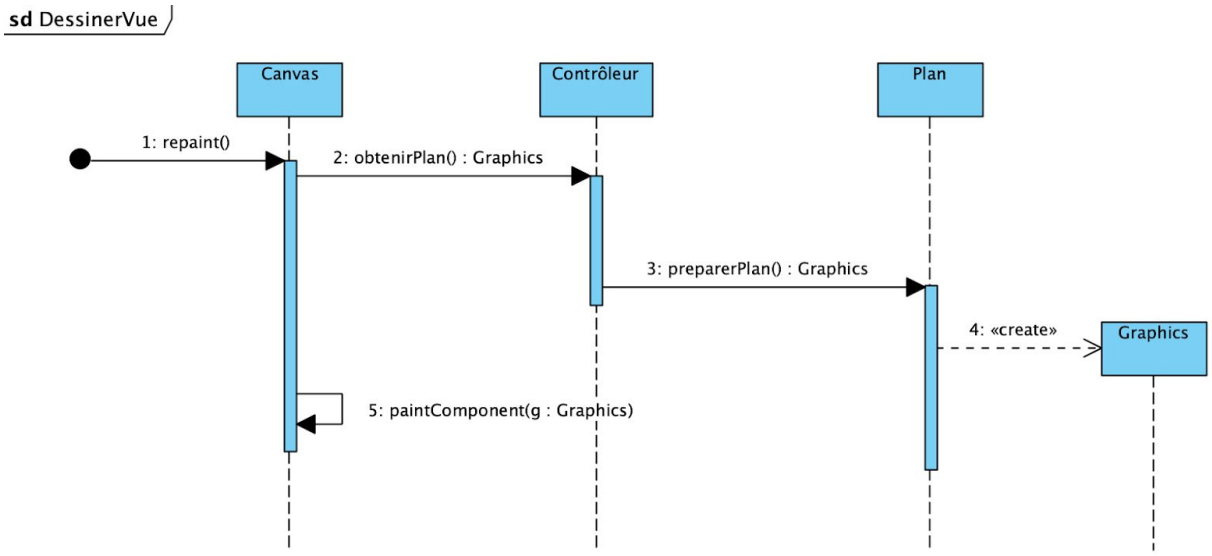
Après avoir fait appel au contrôleur avec la méthode selectionner() auquel on passe une les coordonnées en fonction de l'unité de mesure, le contrôleur fait appel à l'objet "plan"(qui contient les surfaces) avec la méthode selectionner() où on passe en coordonnées le point. Ensuite, le plan appelle la méthode estDansSurface() (où l'implémentation est fournie dans la section 4) qui appelle à son tour les méthodes getX() et getY() pour avoir tous les sommets de la surface. une fois que c'est le cas, on applique la méthode à la surface. Si le point est dans une des surfaces du plan, la boucle se termine et on retourne un booléen (vrai) associé à la surface en question. Si le point est dans l'une des surfaces, et donc que estDansLaSurface(Point) est vrai, on appelle la fonction mettreEnSurbrillance(surface : Surface) pour faire ressortir le polygone dans lequel le point est.

3.2 Créer une surface rectangulaire



L'activité débute lorsque l'utilisateur sélectionne le bouton "Créer une nouvelle surface" qui fait en sorte que l'évènement soit envoyé au contrôleur qui lui appelle la méthode creerSurface() où on lui passe toutes les valeurs nécessaires à sa création : ses dimensions (en X et Y), son typeForme (un rectangle: qui aura une clé (int) associée à celui-ci) et son typeSurface (trou ou pas). Une fois que toutes ces informations sont recueillies, la surface est créée à partir du plan avec la fonction ajouterSurface() qui crée l'objet surface et qui, à son tour, crée les objets points (qui assignent les sommets du rectangle à des coordonnées sur le graphique). La méthode ajouterSurface() permet d'ajouter la nouvelle forme ainsi que ses propriétés dans un conteneur au niveau de l'objet plan. Une fois que l'utilisateur clique à un endroit sur le canvas, l'objet est dessiné à cet endroit étant donné que la méthode selectionner() fait un appel à la méthode repaint() (défini dans le diagramme 3.3).

3.3 Montrez comment l’affichage de la vue en plan sera réalisé



Lorsqu’une modification est apportée au plan, a méthode repaint() est appelée sur le canvas (celle-ci contient la méthode paintComponent() qui sera appelée plus tard, une fois que toutes les informations relatives à l’affichage seront retournées). Par la suite, on appelle obtenirPlan() sur le contrôleur, où son but est de recueillir toutes les informations nécessaires (sous forme de Graphics) pour pouvoir effectuer le dessin. Par la suite, les informations au niveau du plan sont recueillies avec preparerPlan() qui, lui, permet de créer un objet Graphics. Une fois que l’objet est créé, le canvas appelle paintComponent() avec l’objet Graphics en paramètre afin d’afficher la modification.

4. Algorithme pour déterminer si un point est dans un polygone

Input polygone
Input pointATester

Entier x <-- valeur en x de pointATester
Entier y <-- valeur en y de pointATester
Entier n <-- 1
Entier m <-- nombre de sommets du polygone
Booléen estDedans <-- 0

Tant que n ne dépasse pas le nombre de sommets du polygone :

    Entier nx <-- valeur en x du n-ième sommet du polygone
    Entier ny <-- valeur en y du n-ième sommet du polygone
    Entier mx <-- valeur en x du m-ième sommet du polygone
    Entier my <-- valeur en y du m-ième sommet du polygone

    // On imagine une droite horizontale de hauteur y

    Si ny > y >= my ou my > y >= ny :
                    // Cela signifie que la droite croise l'arête reliant n à m

        Entier croisement <-- ((mx - nx) * (y - ny) / (my - ny)) + nx
                        // Valeur en x où la droite croise l'arête reliant n à m

        Si croisement < x :   // Si le croisement est à gauche du point à tester

            Inverser estDedans   // La droite entre dans le polygone ou en sort

    m <-- n
    n <-- n + 1

Output estDedans

5. Diagramme de Gantt

	i1	i2	i3	i4	i5	i6	i7	i8	i9
Créer un nouveau projet									
Enregistrer un projet									
Créer une surface									
Créer une surface pleine									
Créer un trou									
Créer une forme à partir de points									
Déplacer une surface									
Sélectionner/ Déselectionner surface									
Ajouter un sommet									
Supprimer un sommet									
Modifier un sommet									
Redimensionner une surface									
Aligner avec une autre surface									
Coller à une autre surface									
Fusionner à une autre surface									
Appliquer un motif									

Choisir l’épaisseur du coulis			■						
Editer la taille des tuiles			■						
Choisir l’alignement des tuiles			■						
Modifier informations tuiles				■					
Téléverser un motif									■
Choisir la couleur du coulis				■					
Modifier les propriétés des matériaux				■					
Consulter les quantités de tuiles et de boîtes nécessaires							■		
Positionner manuellement le motif des tuiles							■		
Charger un projet								■	
Undo/Redo					■				
Centrer une surface		■							
Zoom					■				
Surligner une forme							■		
Supprimer une forme		■							

6. Section justifiant la contribution de chacun des membres de l'équipe

Catherine Simard :

- Réalisation du diagramme de classe de conception
- Contribution à l'architecture logique
- Réalisation de deux diagrammes de séquence de conception
- Réalisation du diagramme de Gantt

Jean-Sébastien Nantel

- Réalisation du diagramme de classe de conception
- Contribution à l'architecture logique
- Réalisation d'un diagramme de séquence de conception
- Mise en place des éléments de l'interface utilisateur

Léo Guérin-Morneau

- Réalisation du diagramme de classe de conception
- Contribution à l'architecture logique
- Contribution aux diagrammes de séquence de conception
- Réalisation de l'algorithme pour déterminer si un point est dans un polygone

Vincent Chavanel Jobin

- Réalisation du diagramme de classe de conception
- Réalisation de l'architecture logique
- Réalisation d'un diagramme de séquence de conception

Annexes

1. Énoncé de vision

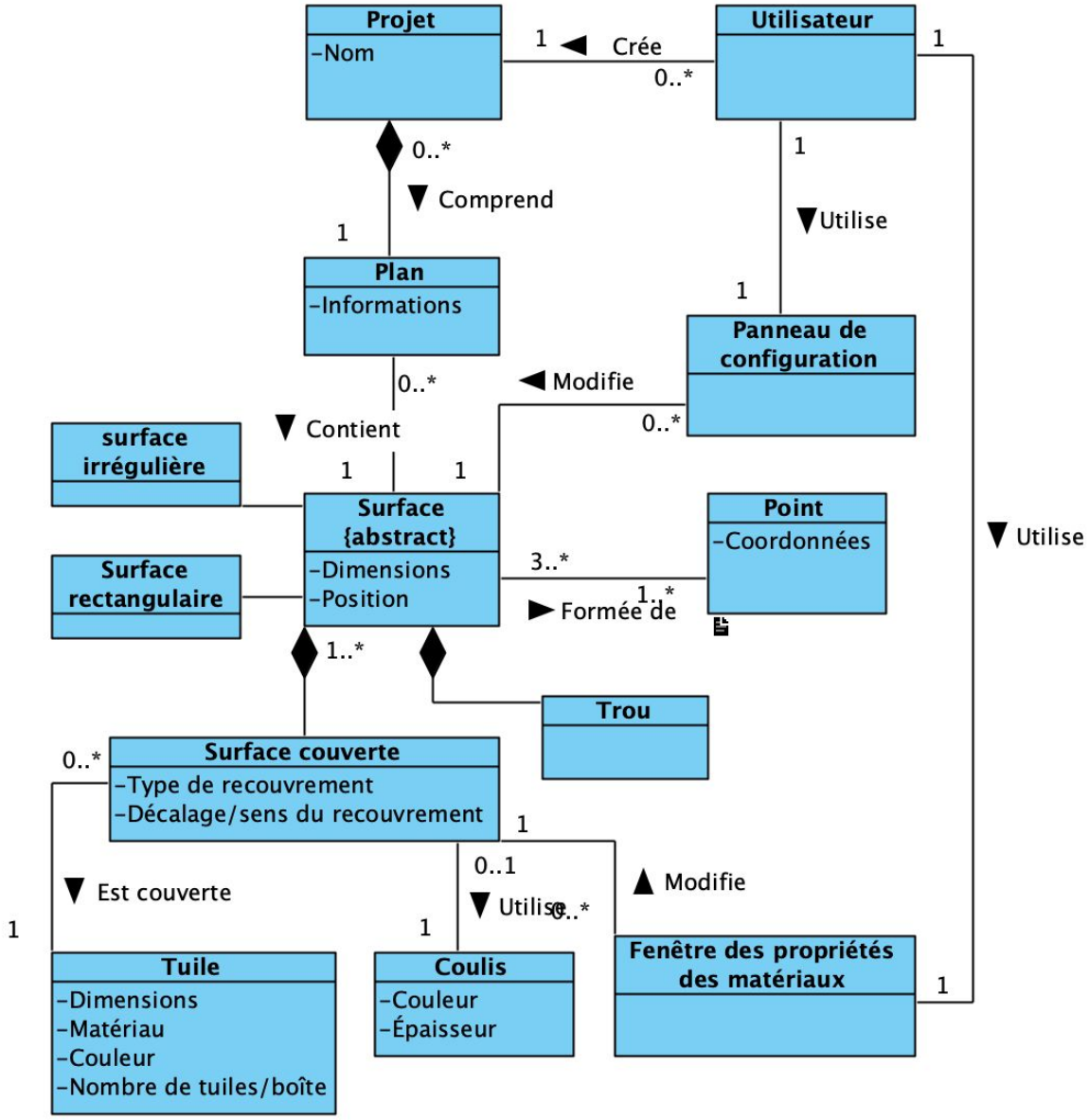
VirtuTuile est une application qui permet à l'utilisateur de planifier l'arrangement de motifs de tuiles désirés pour des planchers et des revêtements muraux..

VirtuTuile donne l'option au client de modéliser des formes personnalisées et de les redimensionner. Pour les créer, l'utilisateur peut choisir parmi une sélection de polygones de base, ou dessiner sa propre figure sommet par sommet. Ces formes peuvent être fusionnées ou collées afin de créer l'agencement désiré. Des trous peuvent également être insérés et modifiés afin de représenter l'étendue à couvrir.

Ensuite, l'utilisateur peut apposer des motifs sur ces surfaces en choisissant parmi plusieurs possibilités de modifications. Par exemple, il est possible de choisir un recouvrement parmi plusieurs matériaux différents, et de déterminer l'épaisseur de coulis entre les tuiles. L'application permet également au client de faire pivoter ces motifs, de les centrer, de les faire commencer par une tuile pleine ou de les décaler par un pourcentage donné.

Finalement, l'utilisateur peut obtenir des informations sur le plan construit ou pour une surface spécifique, comme l'aire couverte, le nombre de tuiles à utiliser et la quantité de boîtes à acheter. Il peut également éditer en tout temps les informations concernant les dimensions des surfaces et des tuiles, la couleur du coulis et des tuiles ainsi que l'alignement des joints.

2. Modèle du domaine



3. Modèle des cas d'utilisation