VirtuTuile

Analyse - Livrable 2

présenté à

Jonathan Gaudreault

par

Équipe 32

matricule	*nom*
111 098 378	François Levasseur
111 161 341	Philippe Lafontaine
111 155 275	Arnaud Dorval-Leblanc
111 184 497	Vincent Marois-Boucher

Université Laval

15 2019

Table des matières

Table des figures ii

Liste des tableaux iii

1 Diagramme de classe de conception et architecture logique 1

2 Diagrammes de séquence de conception 5
2.1 Déterminer la surface sélectionnée lors d'un clic de souris dans la vue du plan 5
2.1.1 Perspective du UI . . . 5
2.1.2 Perspective du contrôleur de Larman . . . 6
2.2 Création d'une nouvelle surface rectangulaire . . . 7
2.3 Montrez comment l'affichage de la vue en plan sera réalisé. . . . 8

3 Pseudo-code d'un algorithme qui permet de déterminer si un point (x,y), en mètres, se trouve à l'intérieur d'une surface irrégulière 10
3.1 Description . . . 10

4 Diagramme de Gantt 14

5 Contribution de chacun des membres de l'équipe 15

6 Annexe 16
6.1 Énoncé de vision . . . 16
6.2 Modèle du domaine . . . 16
6.3 Modèle des cas d'utilisation . . . 18
6.3.1 Diagramme des cas d'utilisation . . . 18
6.3.2 Description des cas d'utilisation . . . 19
6.3.3 Diagramme de séquence système . . . 23

Bibliographie 29

Table des figures

1.1 Diagramme de classe de conception et architecture logique, $1^{ère}$ moitié 2
1.2 Diagramme de classe de conception et architecture logique, $2^{ème}$ moitiée . . . 3

2.1 Diagramme de séquence pour le point 2.1.1 6
2.2 Diagramme de séquence pour le point 3.1.2 7
2.3 Diagramme de séquence pour le point 3.2 8
2.4 Diagramme de séquence pour le point 3.3 9

4.1 Diagramme de Gantt 14

6.1 Modèle du domaine 17
6.2 Diagramme des cas d'utilisation 18
6.3 Créer un projet 23
6.4 Créer une surface 24
6.5 Ajouter des tuiles 24
6.6 Déplacer une surface 25
6.7 Sélectionner une surface 25
6.8 Soustraire une surface à une autre 26
6.9 Aligner des tuiles par rapport à une surface 26
6.10 Comptabiliser les tuiles et le coulis 27
6.11 Inspecter un patron de tuiles 27
6.12 Sauvegarder un projet 28

Liste des tableaux

Chapitre 1

Diagramme de classe de conception et architecture logique

Les figures 1.1 et 1.2 présentent de la diagramme de classe de conception. Ce diagramme est également remis dans un fichier Visual Paradigm.

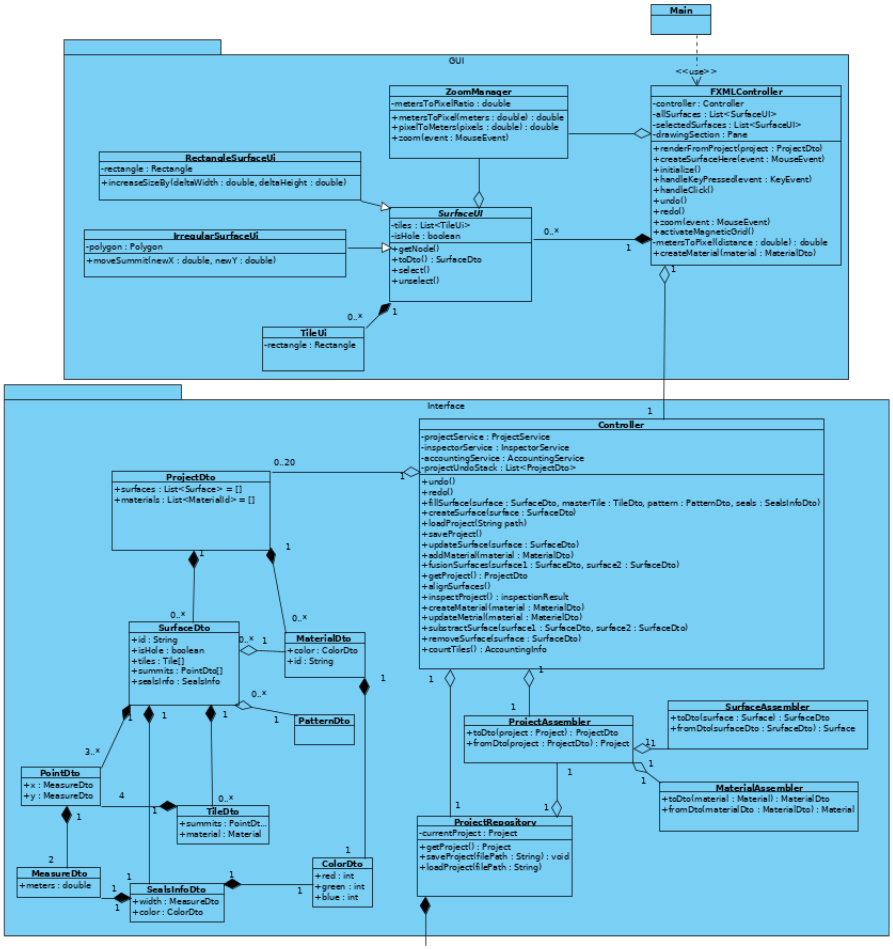

FIGURE 1.1 – Diagramme de classe de conception et architecture logique, $1^{ère}$ moitié

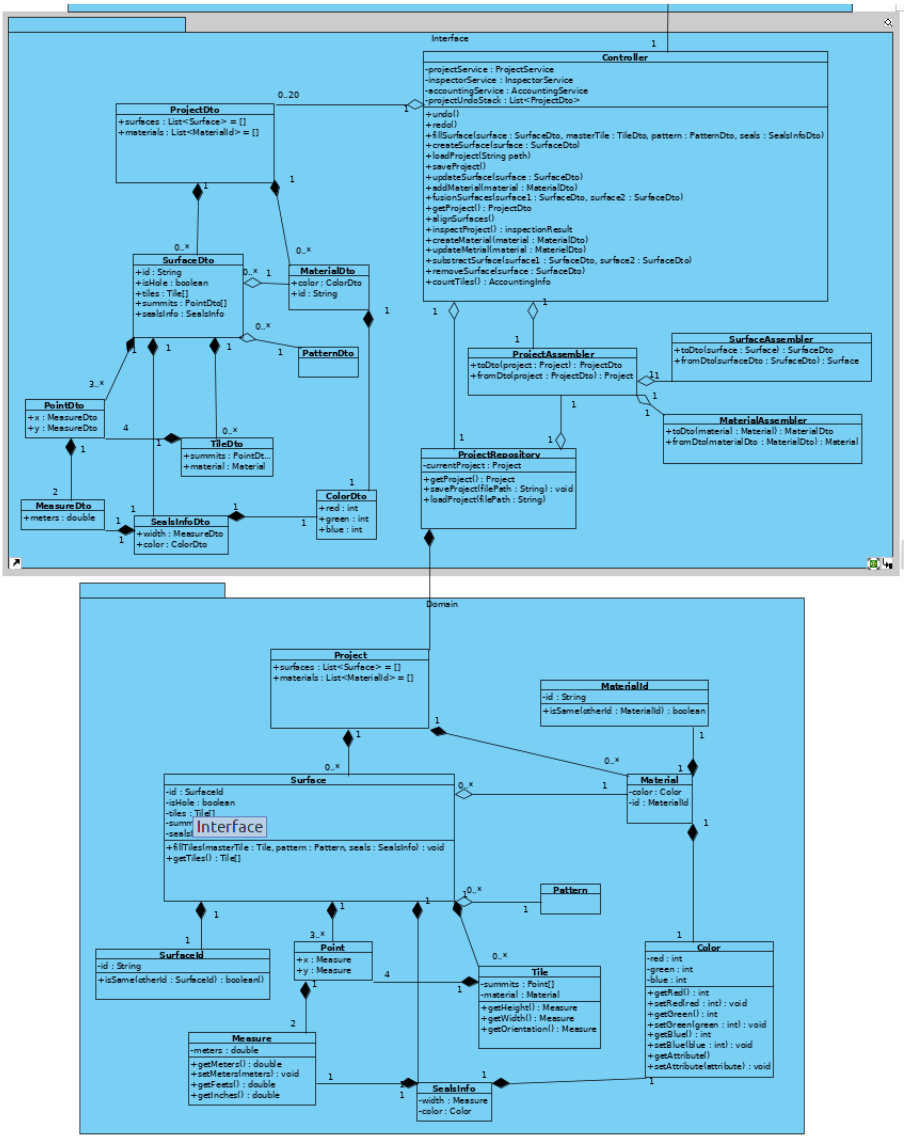

FIGURE 1.2 – Diagramme de classe de conception et architecture logique, $2^{ème}$ moitiée

Le diagramme se décompose en trois packages. Il y a d'abord le package *GUI* qui contient toute la logique concernant l'affichage. Ensuite, le package *Interface* contient principalement le contrôleur de l'application et fait le pont entre la vue et le domaine. Finalement, le package *Domaine* représente les objets du domaine.

La classe la plus importante du package *GUI* est *FXMLController*. C'est à cette classe que revient le rôle d'intéragir avec les éléments du fichier de description *FXML*. *JavaFX* offre également des points d'injection de code pour répondre à des événements, comme un clic de souris ou l'appui d'une touche de clavier. Le *FXMLController* gère la majorité de

ces événements, mais le clic sur une surface est quant-à-lui géré par la classe *SurfaceUI*. Il s'agit d'une classe abstraite qui est étendue par deux enfants ; *RectangleSurfaceUI* et *IrregularSurfaceUI*. La classe *RectangleSurfaceUI* contient un objet *JavaFX Rectangle* qui lui même contient son gestionnaire (handler) de clic. Ainsi, lorsqu'un utilisateur clique sur une surface, c'est le gestionnaire de clic de cette instance *JavaFX Rectangle* qui est appelé. Le *FXMLController* a une méthode *renderFromProject()* qui s'occupe de générer correctement tous les objets *JavaFX* à partir d'un *ProjectDto*. Le *ProjectDto* lui est fourni par la classe *Controller* du package *Interface*.

La classe la plus importante du package *Interface* est le *Controller*. Il s'agit d'une classe qui fait le pont entre le package *GUI* et *Domaine*. Toutes les actions ou calculs qui concernent le domaine doivent passer par lui. Ce dernier ne contient pas une référence au projet, mais au *ProjectRepository*, qui lui est composé d'un projet. Le *ProjectRepository* peut également persister et lire des projets en format *JSON* dans des fichiers. La classe *Controller* est également responsable de la gestion des *undo* et *redo*. Pour ce faire, il possède une liste des états précédents d'un projet qu'il utilise comme pile. À chaque action réalisée sur le domaine, le *Controller* doit prendre une sauvegarde de l'état du projet dans la liste.

La tête d'aggrégat du domaine est le *Project*. Ce dernier contient des surfaces et matériaux, qui sont des instances de la classe *Surface* et *Material*. La surface contient une grande partie de la logique du domaine, qui est la méthode *fillTiles()*. À partir d'une tuile dite "maîtresse", d'un espacement de coulis et d'un mottif, la surface calcule les sommets de chacune des tuiles qui la couvre. La tuile maîtresse donne la dimension de toutes les tuiles de la surface, mais aussi la position de la première tuile du motif. L'action de déplacer un motif de tuile dans l'interface graphique passera par chacune des couches du logiciel jusqu'à cette méthode et aura pour effet de déplacer la tuile maitresse (la *masterTile*). La position des sommets de toutes les autres tuiles sera calculée à partir de celle-ci.

Chapitre 2

Diagrammes de séquence de conception

2.1 Déterminer la surface sélectionnée lors d'un clic de souris dans la vue du plan

2.1.1 Perspective du UI

On connaît la coordonnée du clic dans la méthode *handle* du gestionnaire d'événement du *JavaFX Node*. Cette dernière reçoit un *MouseEvent* en argument, qui lui possède les méthodes d'accès *getX()* et *getY()* qui retournent respectivement les coordonnées x et y du clic de souris. Un objet gestionnaire de zoom, le *zoomManager* garde en mémoire une constante de proportionnalité entre une grandeur en mètres et en pixels. Cette constante de proportionnalité permet le passage d'un système d'unités à l'autre. On sait que la correspondance entre les pixels et les mètres est linéaire car 0 pixels indique 0 mètres. La constante de proportionnalité par défaut, celle définit avant même qu'une action de zoom ou dézoom soit faite par l'utilisateur est arbitraire et définie dans le code (*hardcodé*).

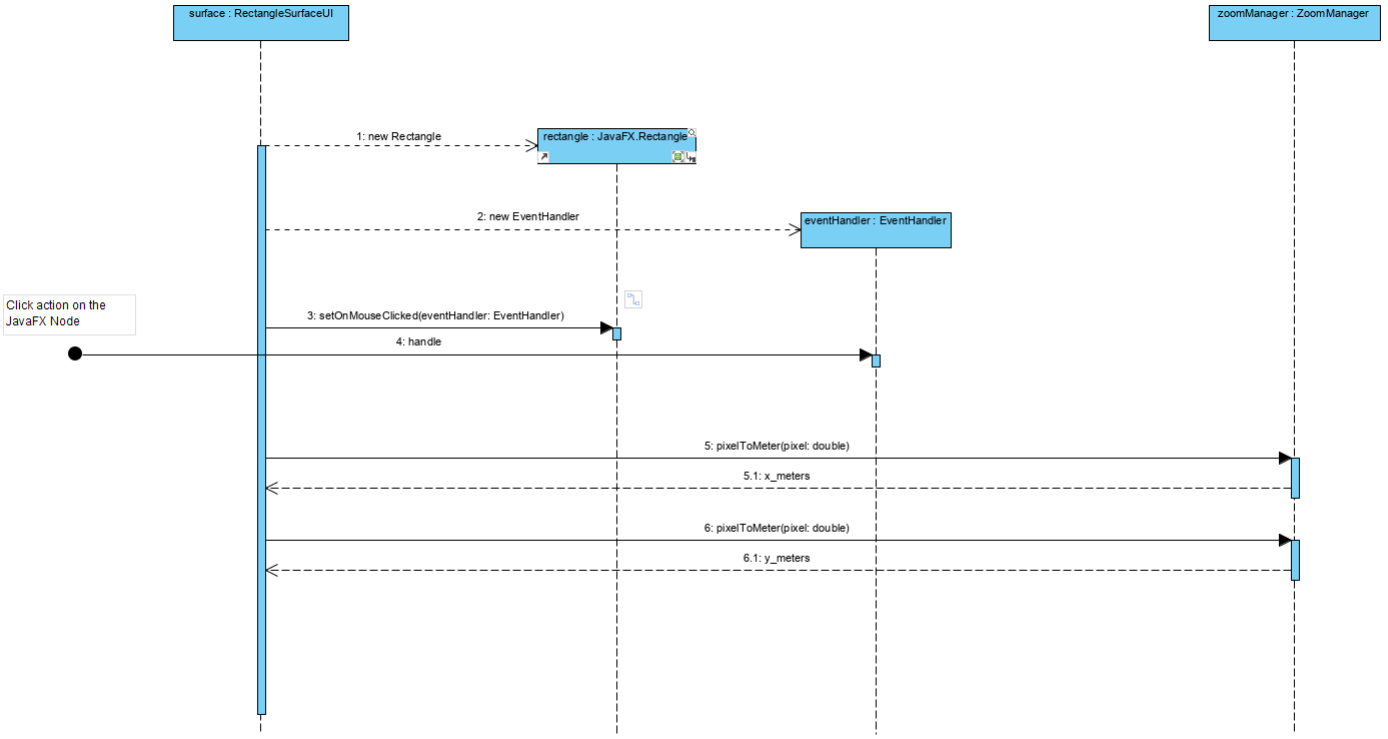

FIGURE 2.1 – Diagramme de séquence pour le point 2.1.1

2.1.2 Perspective du contrôleur de Larman

Dans *JavaFX*, tous les éléments de la vue sont des objets de classe *Nodes*. Pour chaque surface du projet, on crée une instance de la classe *RectangleSurfaceUi* (ou *IrregularSurfaceUi*), qui elle contient une instance de la classe *JavaFX Rectangle*. Le *JavaFX Rectangle* est un enfant de *JavaFX Node*. On ajoute ce rectangle aux enfants de l'instance de la classe *JavaFX Pane*. Ce rectangle possède un gestionnaire d'événement (*EventHandler*) qui gère l'événement où se fait le clic. Lorsque l'on clique sur une surface, c'est le gestionnaire d'événement de son rectangle qui est appelé. Il se fait appeler par la méthode *handle* et se fait passer un objet de type *MouseEvent*. On sait donc de quelle surface il s'agit.

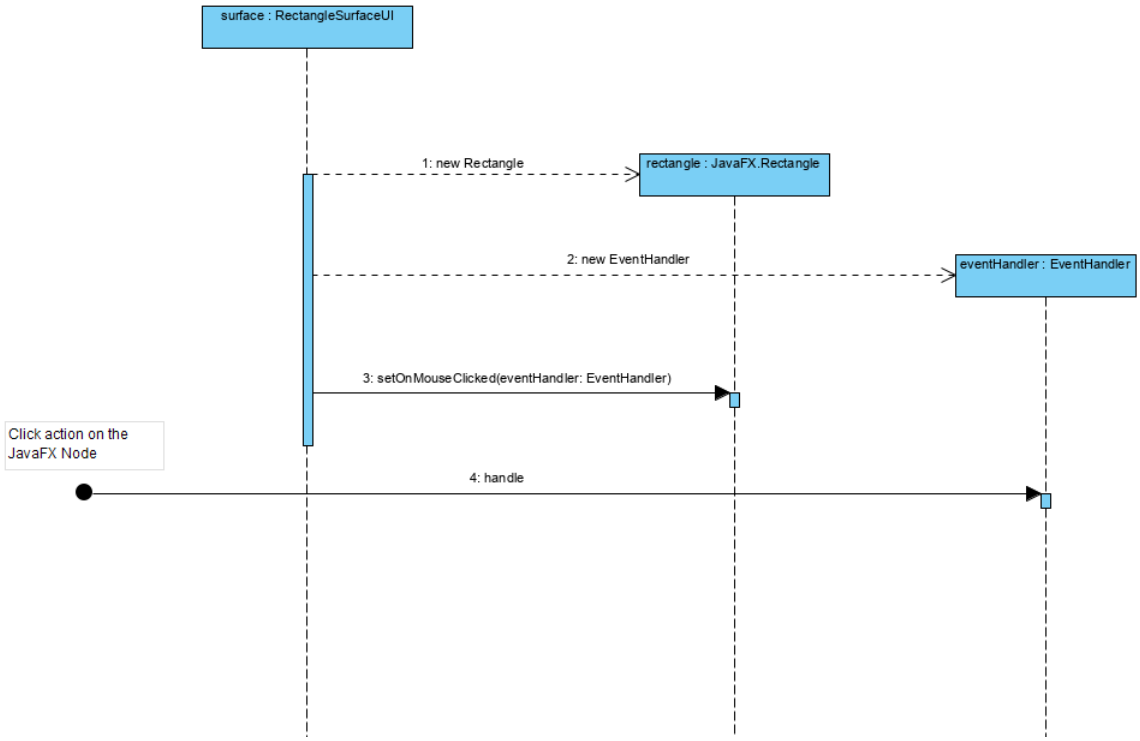

FIGURE 2.2 – Diagramme de séquence pour le point 3.1.2

2.2 Création d'une nouvelle surface rectangulaire

Le diagramme débute par l'appel du contrôleur par la méthode *createSurface()*. La méthode *fromDto(surfaceDto : SurfaceDto)* de *SurfaceAssembler* va créer un objet surface à partir du *DTO*. Il s'agit d'un traducteur entre le *DTO* et le véritable objet du domaine. Le contrôleur va aller chercher le projet avec la méthode *getProject()* de *ProjectRepository*. Le contrôleur de Larman va ensuite chercher les surfaces du projet avec la méthode *getSurfaces()* de *Project*. Comme *getSurfaces()* retourne une liste, on peut ajouter la surface dans la liste avec la méthode *add(surface : Surface)* du type *List<Surface>*.

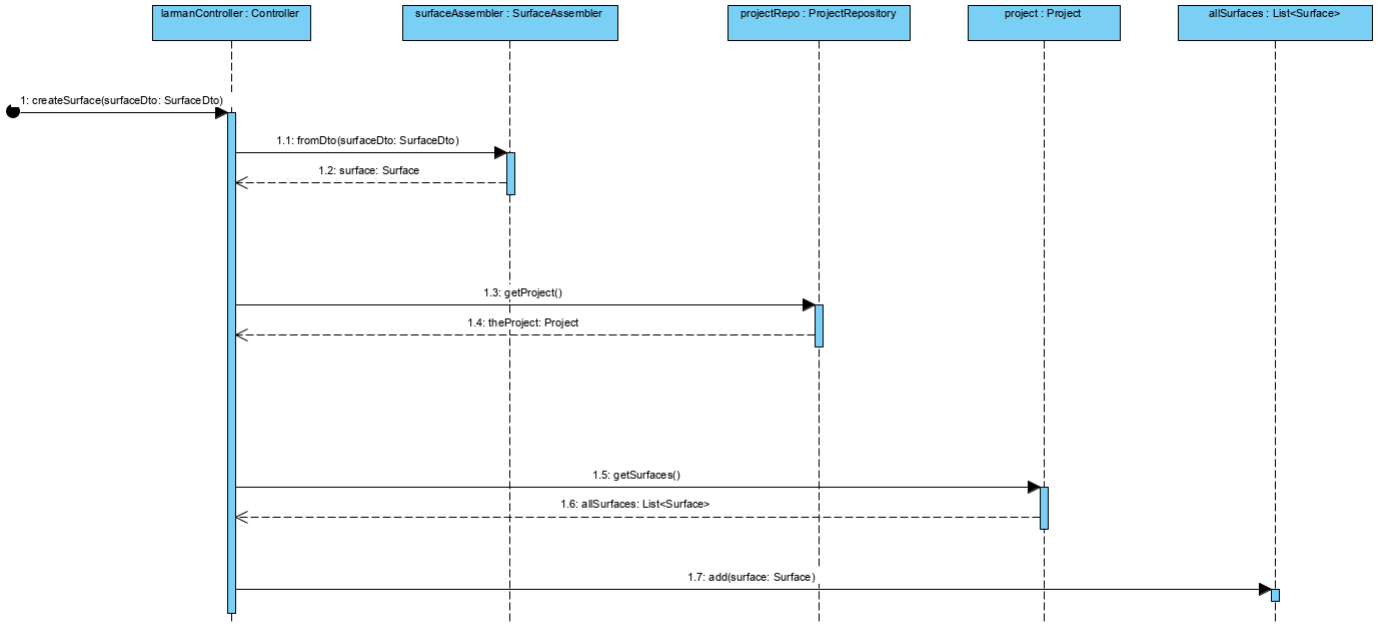

FIGURE 2.3 – Diagramme de séquence pour le point 3.2

2.3 Montrez comment l'affichage de la vue en plan sera réalisé.

À titre d'exemple, le diagramme séquence réalisé ci-dessous est celui de l'affichage d'une surface dans l'interface graphique suite à la création de celle-ci. Cela permet de montrer le mécanisme de synchronisation entre l'interface graphique et le domaine. *JavaFX* n'offre pas de point d'injection de code au moment de l'affichage. Les points d'injection offerts par *JavaFX* sont des gestionnaires d'événement. On peut interagir avec l'interface lors de la gestion de ces événements. C'est aussi dans ces méthodes que l'on fait des appels au contrôleur de Larman.

Par exemple, pour la création d'une surface, on appelle le contrôleur pour l'avertir de la création d'une surface en lui passant un objet *SurfaceDto* qui contient toutes les informations de la nouvelle surface. Ensuite, on lui demande la nouvelle version du projet par la méthode *getProject()*, qui retourne un *ProjectDto*. Le *FXMLController* est alors responsable de créer tous les objets graphiques à partir du *ProjectDto*. Pour chaque surface du projet y compris la dernière surface créée, on crée un objet *RectangleSurfaceUI*. Finalement, la classe *RectangleSurfaceUI* possède une méthode *getNode()*, qui retourne un *JavaFX Node*. On appelle cette méthode pour récupérer le rectangle représentant la surface et l'ajouter dans les enfants de la zone de dessin, qui est un *JavaFX Pane*. Le *FXMLController* est également responsable

d'appeler le contrôleur de Larman par la méthode *createSurface()* pour avertir le domaine qu'une nouvelle surface a été créée.

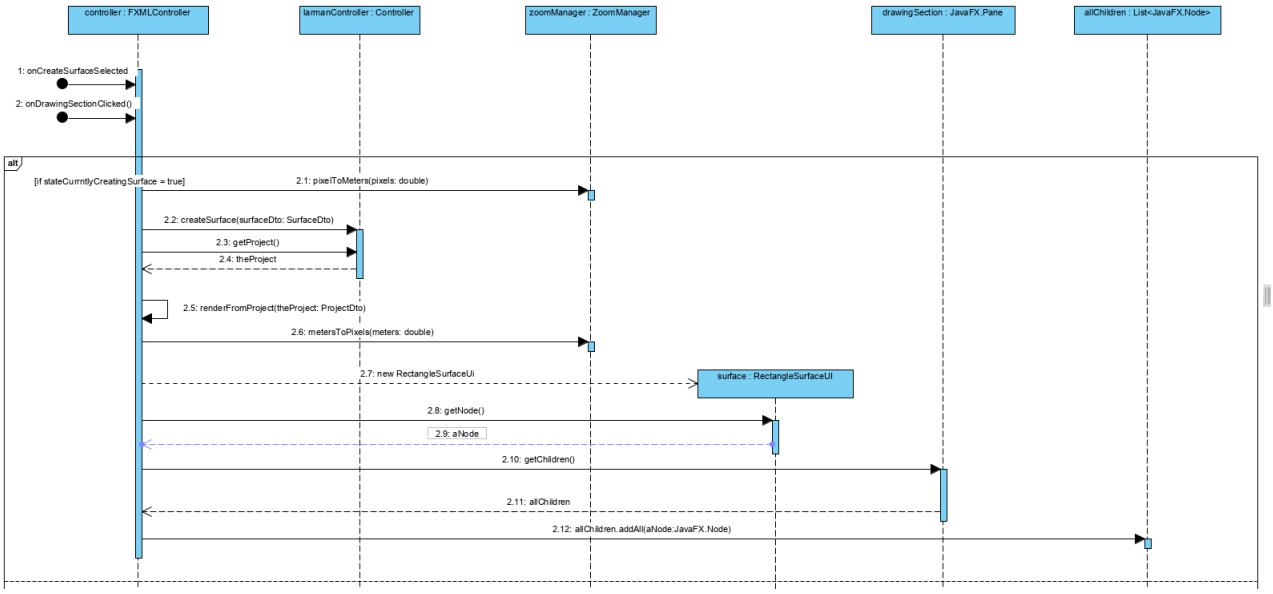

FIGURE 2.4 – Diagramme de séquence pour le point 3.3

Chapitre 3

Pseudo-code d'un algorithme qui permet de déterminer si un point (x,y), en mètres, se trouve à l'intérieur d'une surface irrégulière

3.1 Description

L'algorithme qui permet de déterminer si un point (x,y) se retrouve à l'intérieur d'une surface irrégulière ce base sur l'idée que si le point est transformé en une droite de longueur infinie, cette droite intersectera un nombre impair de fois la surface observée si elle est à l'intérieur de la surface. Il est aussi possible d'observer un nombre d'intersection pair si le point est à l'extérieur de la suface. Afin d'implanter la fonction *isInsideSurface*, trois sous fonctions sont nécessaire.

1. *ptsColineaire* permettant de détecter si le point est colinéaire avec un segment de la surface
2. *directionSegment* permettant de connaitre l'orientation d'un segment
3. *intersectionSegment* permettant de savoir si 2 segments s'intersectent.

```
1  package com.company;
2  import java.util.ArrayList;
3  import java.util.List;
4
5  public class Surface {
6
7      List<PointXY> sommet = new ArrayList<PointXY>();
8
9      public Surface(){
10     }
```

```
11
12      public Surface(List<PointXY> pList){
13          sommet = pList;
14      }
15
16      public int isMax(int pA, int pB){
17          if (pA < pB){
18              return pA;
19          }
20          return 0;
21      }
22
23      public int isMin(int pA, int pB){
24          if (pA > pB){
25              return pA;
26          }
27          return 10000;
28      }
29
30      public boolean ptsColineaire(PointXY pA, PointXY pB, PointXY pC)
            {
31          if (pB.valeurX <= isMax(pA.valeurX, pC.valeurX)){
32              if (pB.valeurX >= isMin(pA.valeurX, pC.valeurX)){
33                  if (pB.valeurY <= isMax(pA.valeurY, pC.valeurY)){
34                      if (pB.valeurY >= isMin(pA.valeurY, pC.valeurY))
                            {
35                          return true;
36                      }
37                  }
38              }
39          }
40
41          return false;
42      }
43
44      public int directionSegment(PointXY pA, PointXY pB, PointXY pC){
45          int valeur = (pB.valeurY - pA.valeurY) * (pC.valeurX - pA.
                valeurX) - (pB.valeurX - pA.valeurX) * (pC.valeurY - pB.
                valeurY);
46          if (valeur == 0) {
47              return 0;
48          }else{
49              return (valeur > 0)? 1:2;
50          }
51
```

```
52      }
53
54
55      public boolean intersectionSegment(PointXY pA, PointXY pB,
           PointXY pC, PointXY pD){
56          int direction1 = directionSegment(pA, pB, pC);
57          int direction2 = directionSegment(pA, pB, pD);
58          int direction3 = directionSegment(pC, pD, pA);
59          int direction4 = directionSegment(pC, pD, pB);
60
61          if (direction1 != direction2 && direction3 != direction4){
62              return true;
63          }else if (direction1 == 0 && ptsColineaire(pA, pC, pB)){
64              return true;
65          }else if (direction2 == 0 && ptsColineaire(pA, pD, pB)){
66              return true;
67          }else if (direction3 == 0 && ptsColineaire(pC, pA, pD)){
68              return true;
69          }else if (direction4 == 0 && ptsColineaire(pC, pB, pD)){
70              return true;
71          }
72          return false;
73      }
74
75      public boolean isInsideSurface(PointXY pPoint){
76          PointXY infini = new PointXY(Integer.MAX_VALUE, pPoint.
               valeurY);
77          int nbIntersection = 0;
78
79          for (PointXY i : sommet){
80
81              if (intersectionSegment(i, nextPoint(sommet.indexOf(i)),
                    pPoint, infini)){
82                  if (directionSegment(i, pPoint, nextPoint(sommet.
                       indexOf(i))) == 0){
83                      return ptsColineaire(i, pPoint, nextPoint(
                           sommet.indexOf(i)));
84                  }
85                  nbIntersection++;
86              }
87          }
88          return nbIntersection % 2 == 1;
89
90      }
91
```

```
92      public PointXY nextPoint(int pIndex){
93          if (pIndex == (sommet.size() - 1)){
94              return sommet.get(0);
95          }else {
96              return sommet.get(pIndex+1);
97          }
98      }
99
100 }
```

Chapitre 4

Diagramme de Gantt

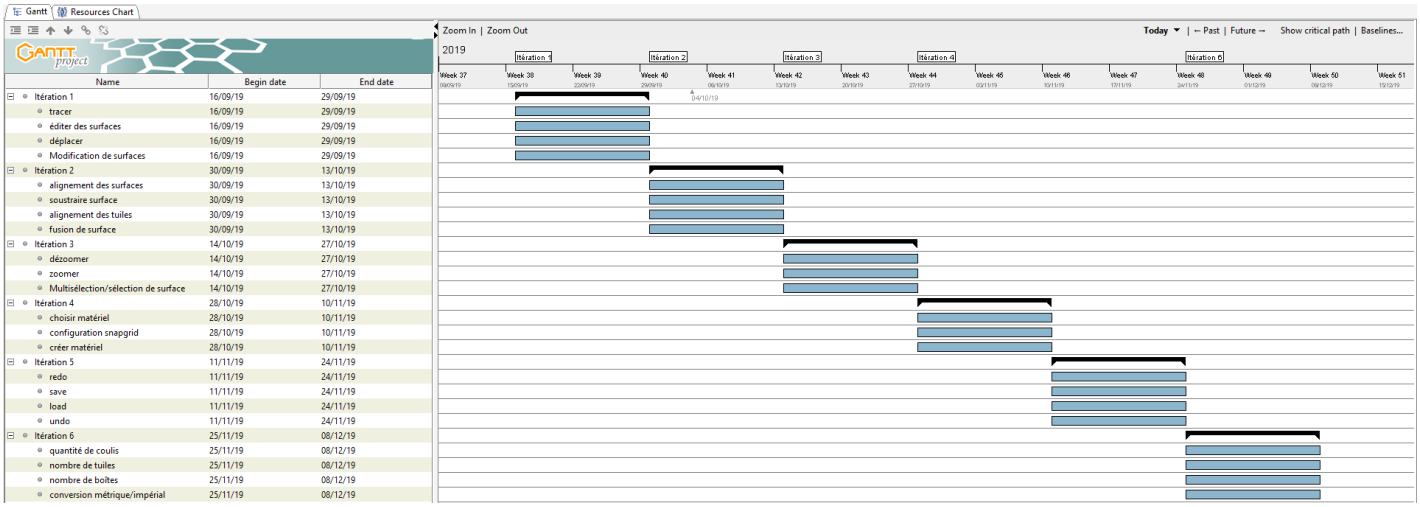

FIGURE 4.1 – Diagramme de Gantt

Chapitre 5

Contribution de chacun des membres de l'équipe

François Levasseur s'est occupé du diagramme de classe de conception ainsi que de l'architecture logique.

Philippe Lafontaine s'est occupé de faire les diagrammes de séquence de conception.

Arnaud Dorval-Leblanc a pris en charge le pseudo-code pour déterminer si un point (x,y), en mètres, se trouve à l'intérieur d'une surface irrégulière.

Vincent Marois-Boucher s'est occupé du diagramme de Gantt, ainsi que de la correction et de la mise en page de la version finale du rapport. Il s'est également chargé de faire la correction du livrable 1 et de l'ajouter en annexe au présent rapport.

Chaque partie a ensuite été revue et corrigée par tous les membres de l'équipe.

Chapitre 6

Annexe

6.1 Énoncé de vision

Le projet logiciel VirtuTuile est une application bureau dont l'objectif est de faciliter l'achat, la planification et la pose d'un motif de tuile. Le client cible de cette application peut être n'importe qui, que ce soit quelqu'un qui décide de poser de la tuile lui-même dans son domicile ou un entrepreneur mandaté pour le même travail.

La pose d'un motif de tuile peut rapidement devenir un casse-tête d'une certaine envergure. En effet, on peut être tenté de débuter un motif avec une tuile pleine à une extrémité du mur et se retrouver coincé de l'autre côté avec un trou trop petit pour accueillir une tuile complète. Un outil de visualisation permettrait d'éviter de devoir faire des coupes trop difficiles ou de se retrouver avec un motif asymétrique.

L'outil VirtuTuile permet cette visualisation. Dans VirtuTuile, on peut définir des surfaces à couvrir ou non, comme des murs ou des planchers et choisir parmi une sélection de motif de tuile. Une fois un motif appliqué, on ajuste la dimension des tuiles, la largeur du coulis de ciment ainsi que les matériaux utilisés. L'outil permet plusieurs autres fonctionnalités, comme faire l'inventaire des matériaux nécessaires pour effectuer un projet.

6.2 Modèle du domaine

Le modèle du domaine est présenté à la Figure 6.1. Un fichier de Visual Paradigms (*.vpp) est remis avec le rapport.

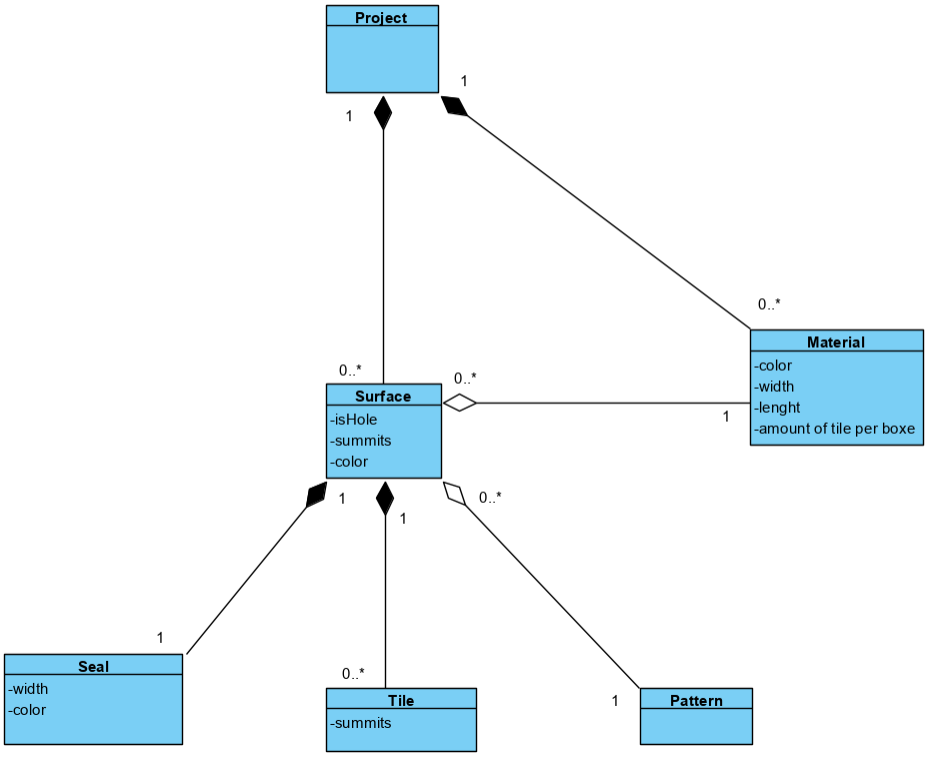

FIGURE 6.1 – Modèle du domaine

Les objets du domaine sont les suivants :

1. Le projet (Project) fait référence à un projet de pose de tuile. Si un utilisateur veux poser de la tuile dans deux résidences différentes, il fait référence à ces deux activités comme des projets différents. C'est la racine d'agrégat de tous les autres objets du domaine.
2. La surface (Surface) représente un mur, un planché ou tout autre surface qu'on veut couvrir de tuile. Toutes les surfaces ne doivent pas nécessairement être couvertes et sont alors définies comme des trous. Une surface est recouverte d'au plus un motif de tuile, toutes du même matériau. C'est pourquoi un mur peut être représenté par deux surfaces s'il est recouvert par plus d'un motif ou matériau. Si une surface n'est pas un trou alors elle doit être recouverte de une à plusieurs tuiles.
3. La tuile (Tile) représente une tuile, c'est à dire un morceau d'un matériau donné et également de coordonées données. Une tuile a un emplacement spécifique sur une surface.
4. Le matériau (Material) représente un matériau duquel est fabriqué une tuile.
5. Le joint (Seal) est un coulis de ciment qui marque la séparation entre les tuiles. Sa largeur peut varier et influencer l'emplacement de chacune des tuiles de la surface. Un joint plus large résulte en des tuiles plus espacées.

6. Un motif (Pattern) représente une règle ou structure que doit respecter l'emplacement des tuiles sur une surface.

6.3 Modèle des cas d'utilisation

6.3.1 Diagramme des cas d'utilisation

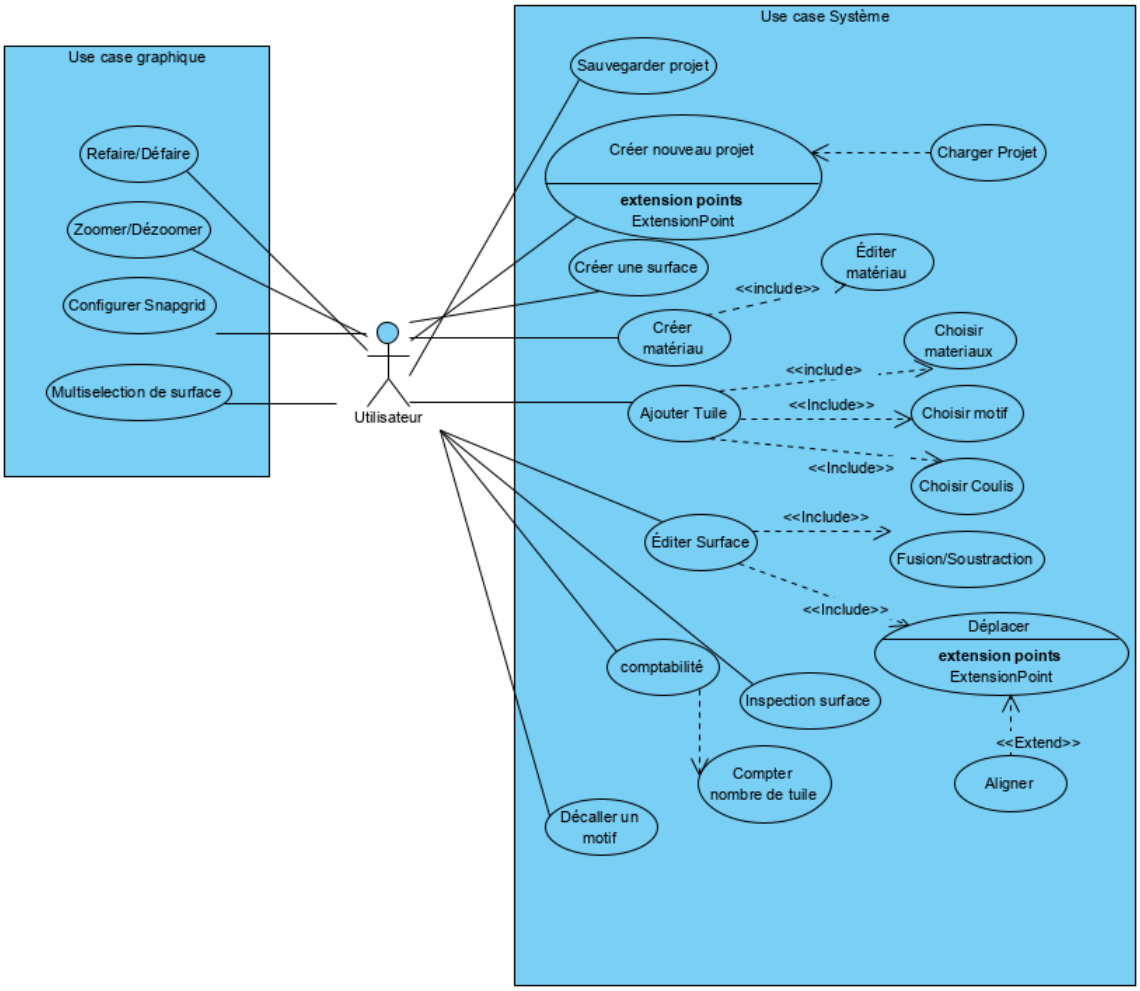

FIGURE 6.2 – Diagramme des cas d'utilisation

6.3.2 Description des cas d'utilisation

Créer un projet	
1. Un utilisateur ouvre VirtuTuile	2. Présentation des options
3. L'utilisateur choisi de créer un nouveau projet	
4. L'utilisateur travail sur le projet	
5. L'utilisateur quitte l'application	

Créer une surface	
1. Un utilisateur ouvre VirtuTuile	
2. L'utilisateur travaille sur le projet	
3. L'utilisateur indique les sommets d'une surface	4. Affichage de la surface tracée
5. L'utilisateur quitte l'application	

Ajouter des tuiles	
1. Un utilisateur ouvre VirtuTuile	
2. L'utilisateur travaille sur le projet	
3. L'utilisateur ajoute des tuiles sur une surface	4. Présentation des options
5. L'utilisateur choisi un motif	
6. L'utilisateur choisi un matériau	
7. L'utilisateur choisi un coulis	8. Insertion des tuiles dans le projet
9. Déplace le patron de tuile par rapport aux sommets de la surface	
10. L'utilisateur quitte l'application	

Déplacer une surface	
1. Un utilisateur ouvre VirtuTuile	
2. L'utilisateur travaille sur le projet	
3. L'utilisateur trace une surfaces	
4. L'utilisateur déplace la surface d'une position à une autre	5. Rafraîchissement de la position des surfaces

Multiselection de surface	
1. Un utilisateur ouvre VirtuTuile	
2. L'utilisateur travaille sur le projet	
3. L'utilisateur trace deux surfaces	
4. L'utilisateur sélectionne deux surfaces	5. Ajout d'un indicateur de sélection
6. L'utilisateur quitte l'application	

Soustraction de surface	
1. Un utilisateur ouvre VirtuTuile	
2. L'utilisateur travaille sur le projet	
3. L'utilisateur trace deux surfaces	
4. L'utilisateur sélectionne deux surfaces	
5. L'utilisateur soustrait une des surfaces à l'autre	6. Rafraîchissement des propriétés des surfaces
7. L'utilisateur quitte l'application	

Alignement des tuiles	
1. Un utilisateur ouvre VirtuTuile	
2. L'utilisateur travaille sur le projet	
3. L'utilisateur trace une surfaces	
4. L'utilisateur ajoute des tuiles	
5. L'utilisateur modifie l'alignement des tuiles	6. Rafraîchissement de l'affichage des tuiles sur la surface
7. L'utilisateur quitte l'application	

Comptabilité	
1. Un utilisateur ouvre VirtuTuile	
2. L'utilisateur travaille sur le projet	
3. L'utilisateur ouvre l'onglet de comptabilité	4. Calcul du nombre de tuile sur la surface
	5. Calcul de la quantité de coulis
	6. Calcul du nombre de boites
7. L'utilisateur quitte l'application	

Inspection	
1. Un utilisateur ouvre VirtuTuile	
2. L'utilisateur travaille sur le projet	
3. L'utilisateur modifie l'alignement des tuiles	
4. L'utilisateur inspecte la surface	5. Vérification du respect des contraintes
	6. Affichage des propriétés de la surface
7. L'utilisateur quitte l'application	

Sauvegarde	
1. Un utilisateur ouvre VirtuTuile	
2. L'utilisateur travaille sur le projet	
3. L'utilisateur sauvegarde le projet	4. Création d'un fichier de sauvegarde
5. L'utilisateur quitte l'application	

Cas d'utilisation :	Fusion de surface
Acteur :	Utilisateur
Type :	Primaire
Description :	L'utilisateur est dans un projet, sélectionne deux surfaces et les fusionne.

Cas d'utilisation :	Alignement de surface
Acteur :	Utilisateur
Type :	Primaire
Description :	L'utilisateur est dans un projet, sélectionne deux surfaces et les aligne.

Cas d'utilisation :	Charger un projet
Acteur :	Utilisateur
Type :	Primaire
Description :	L'utilisateur ouvre l'application, charge un projet et continu sont travail sur celui-ci.

Cas d'utilisation :	Redimensionner une surface
Acteur :	Utilisateur
Type :	Primaire
Description :	L'utilisateur est dans un projet, sélectionne une surface et change ces dimensions.

Cas d'utilisation :	Zoomer/Dé-zoomer
Acteur :	Utilisateur
Type :	Primaire
Description :	L'utilisateur est dans un projet, examine une surface de près puis regarde son projet dans l'ensemble.

Cas d'utilisation :	Faire/défaire une action
Acteur :	Utilisateur
Type :	Primaire
Description :	L'utilisateur est dans un projet, redimensionne une surface, retourne en arrière et décide de garder son changement.

Cas d'utilisation :	Changement d'unité
Acteur :	Utilisateur
Type :	Primaire
Description :	L'utilisateur est dans un projet, crée une surface avec des dimensions impériales, puis convertit ces dimensions en unités métriques.

Cas d'utilisation :	Configuration de la grille magnétique
Acteur :	Utilisateur
Type :	Primaire
Description :	L'utilisateur est dans un projet, configure un quadrillé de référence puis trace des surfaces en relation avec ce quadrillé.

Cas d'utilisation :	Décaller motif
Acteur :	Utilisateur
Type :	Primaire
Description :	L'utilisateur est dans un projet, il veut déplacer le motif des tuile d'une surface .

Cas d'utilisation :	Éditer matériaux
Acteur :	Utilisateur
Type :	Primaire
Description :	L'utilisateur est dans un projet, ajoute un matériau au projet et veut modifier les parmètres d'un matériau.

6.3.3 Diagramme de séquence système

Les cas d'utilisation qui ont été présentés en format deux colonnes à la section 6.2 sont ici présentés sous forme de diagrammes séquence système.

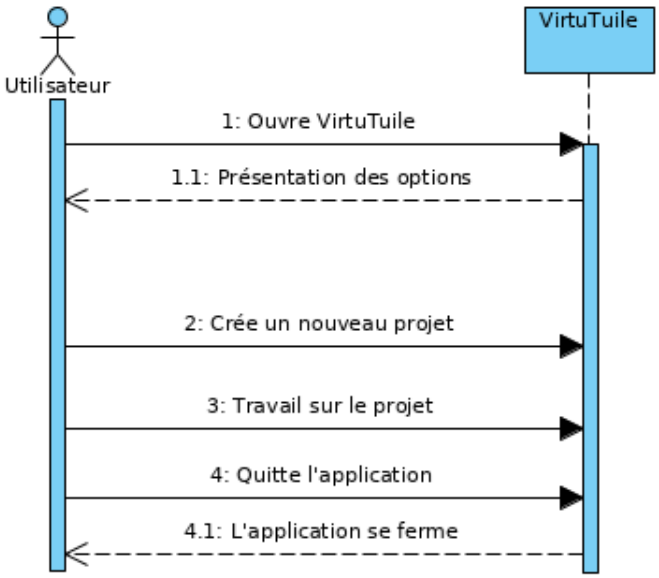

FIGURE 6.3 – Créer un projet

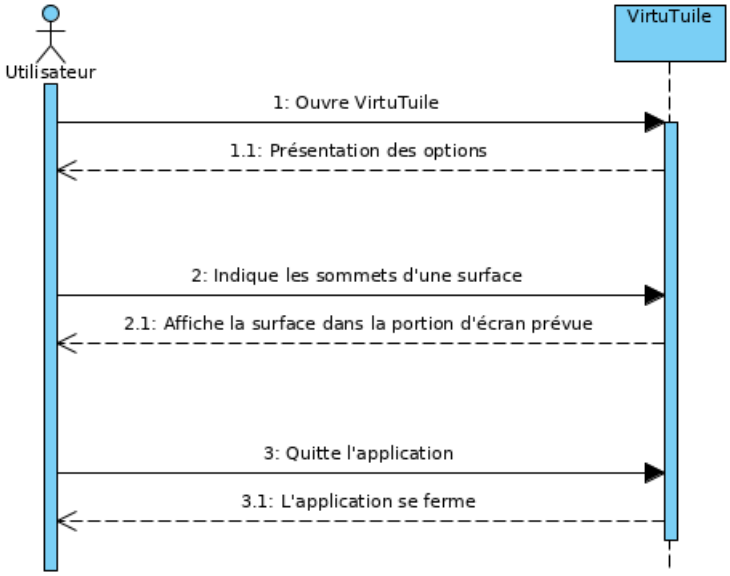

FIGURE 6.4 – Créer une surface

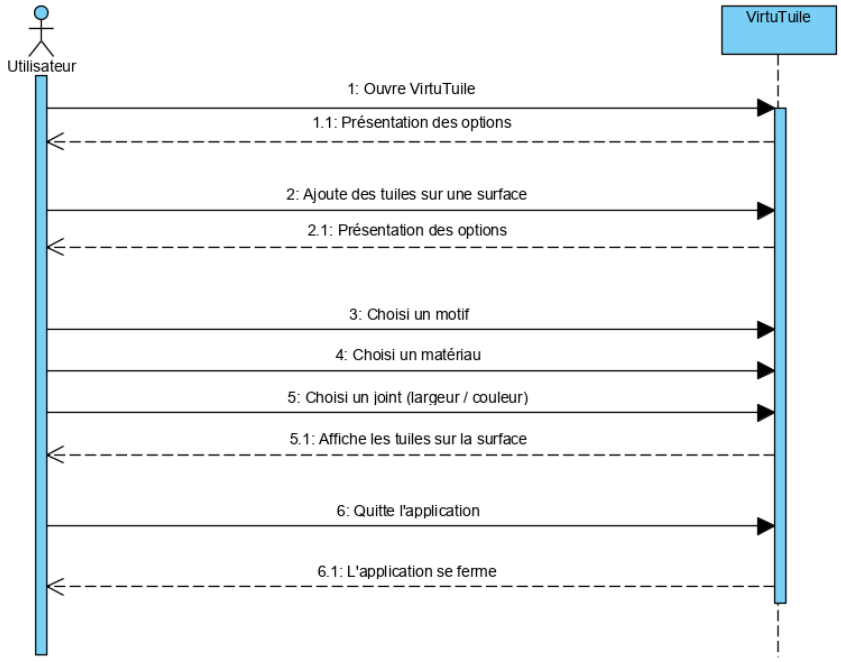

FIGURE 6.5 – Ajouter des tuiles

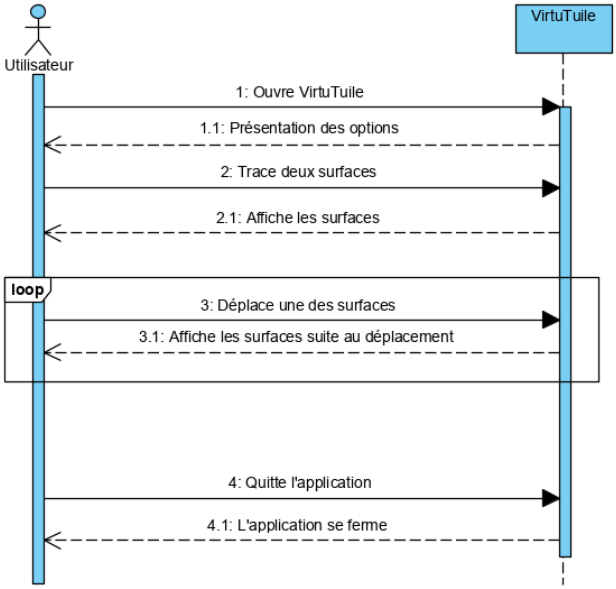

FIGURE 6.6 – Déplacer une surface

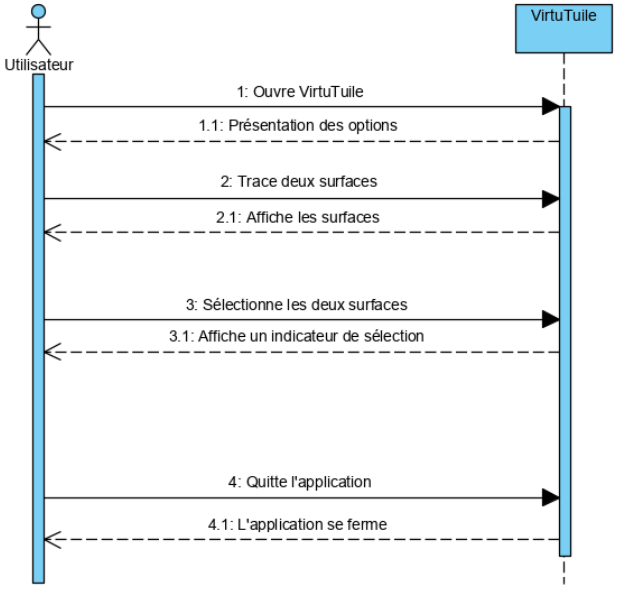

FIGURE 6.7 – Sélectionner une surface

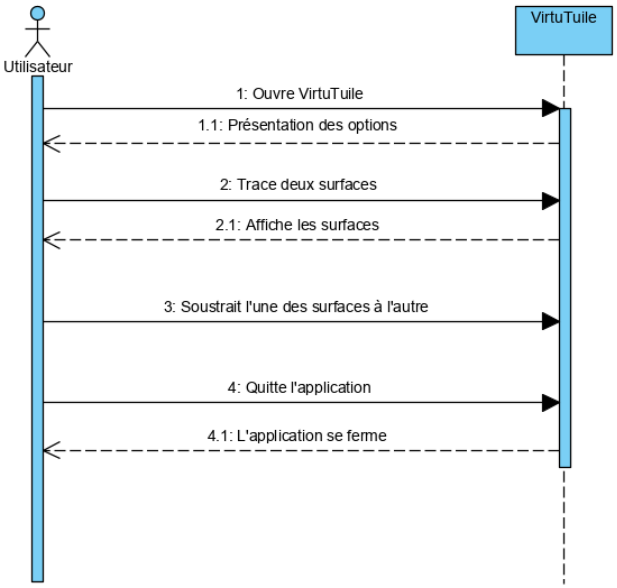

FIGURE 6.8 – Soustraire une surface à une autre

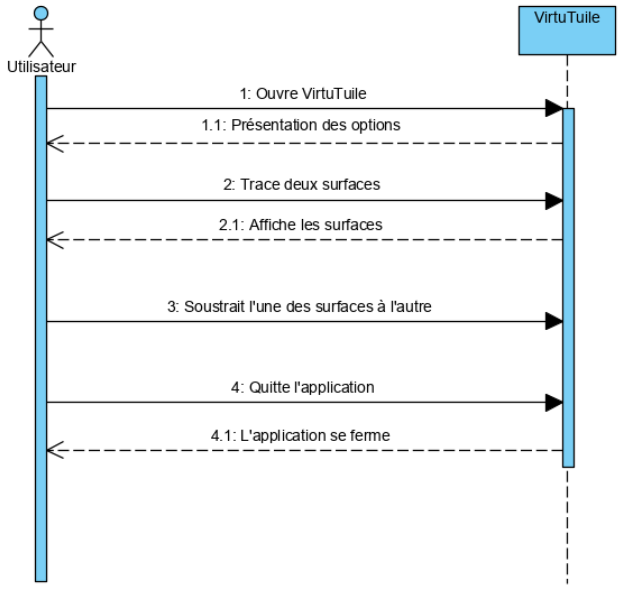

FIGURE 6.9 – Aligner des tuiles par rapport à une surface

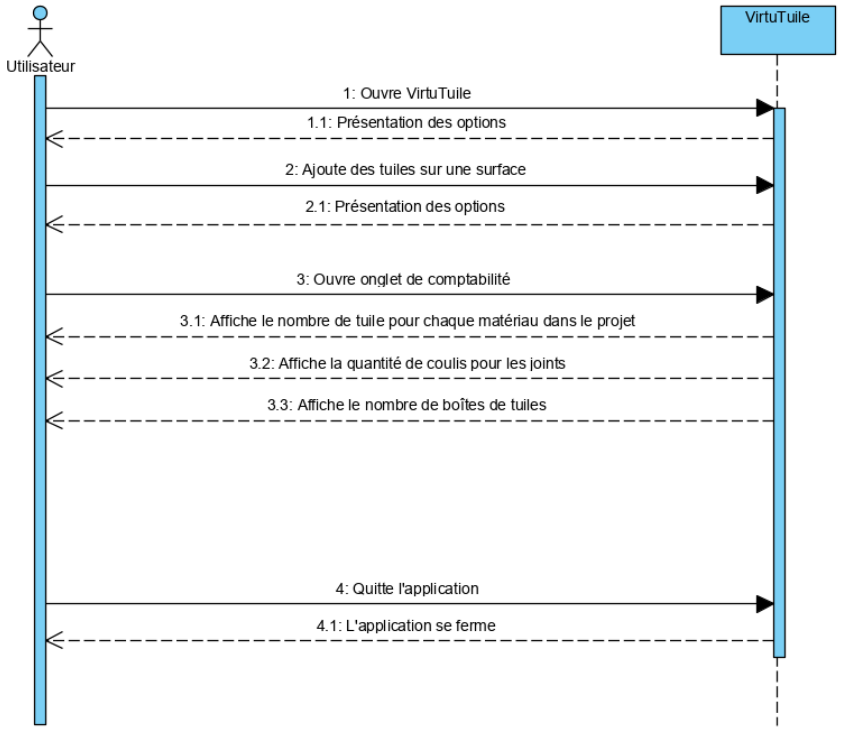

FIGURE 6.10 – Comptabiliser les tuiles et le coulis

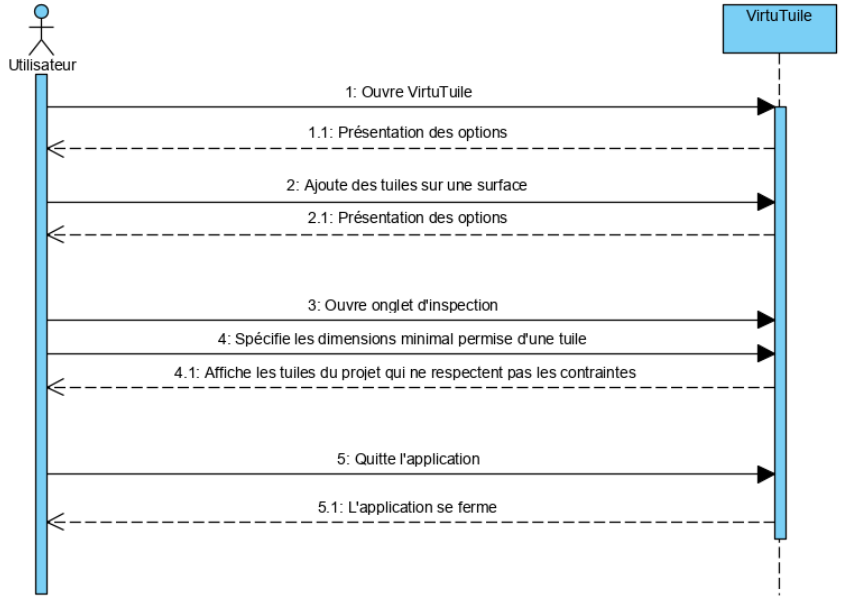

FIGURE 6.11 – Inspecter un patron de tuiles

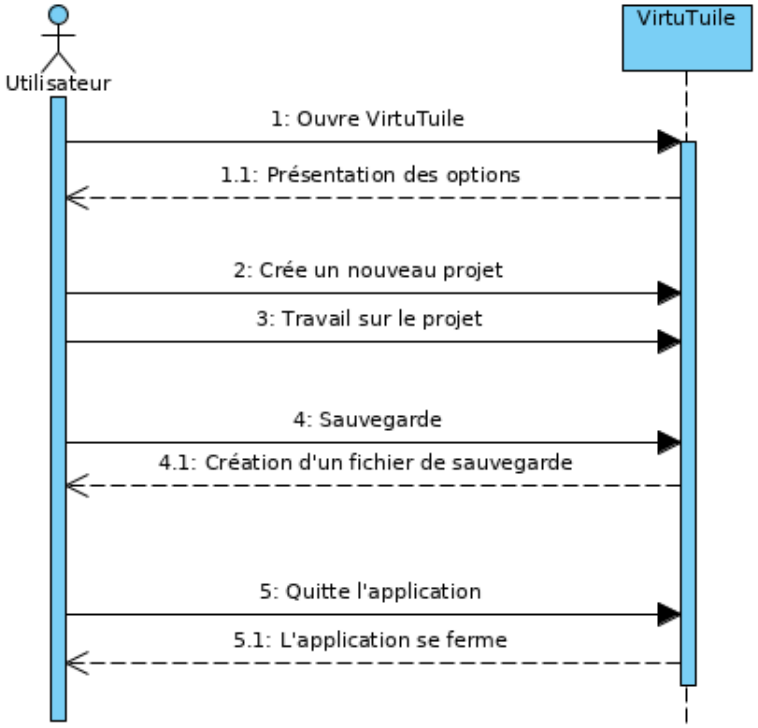

FIGURE 6.12 – Sauvegarder un projet

Bibliographie