UNIVERSITÉ LAVAL

Muracle : Logicielle de conception de salles modulaires acoustiques en métal

Rapport de projet – livrable 2

Présenté dans le cadre du cours

GLO – 2004

Par

Équipe 33

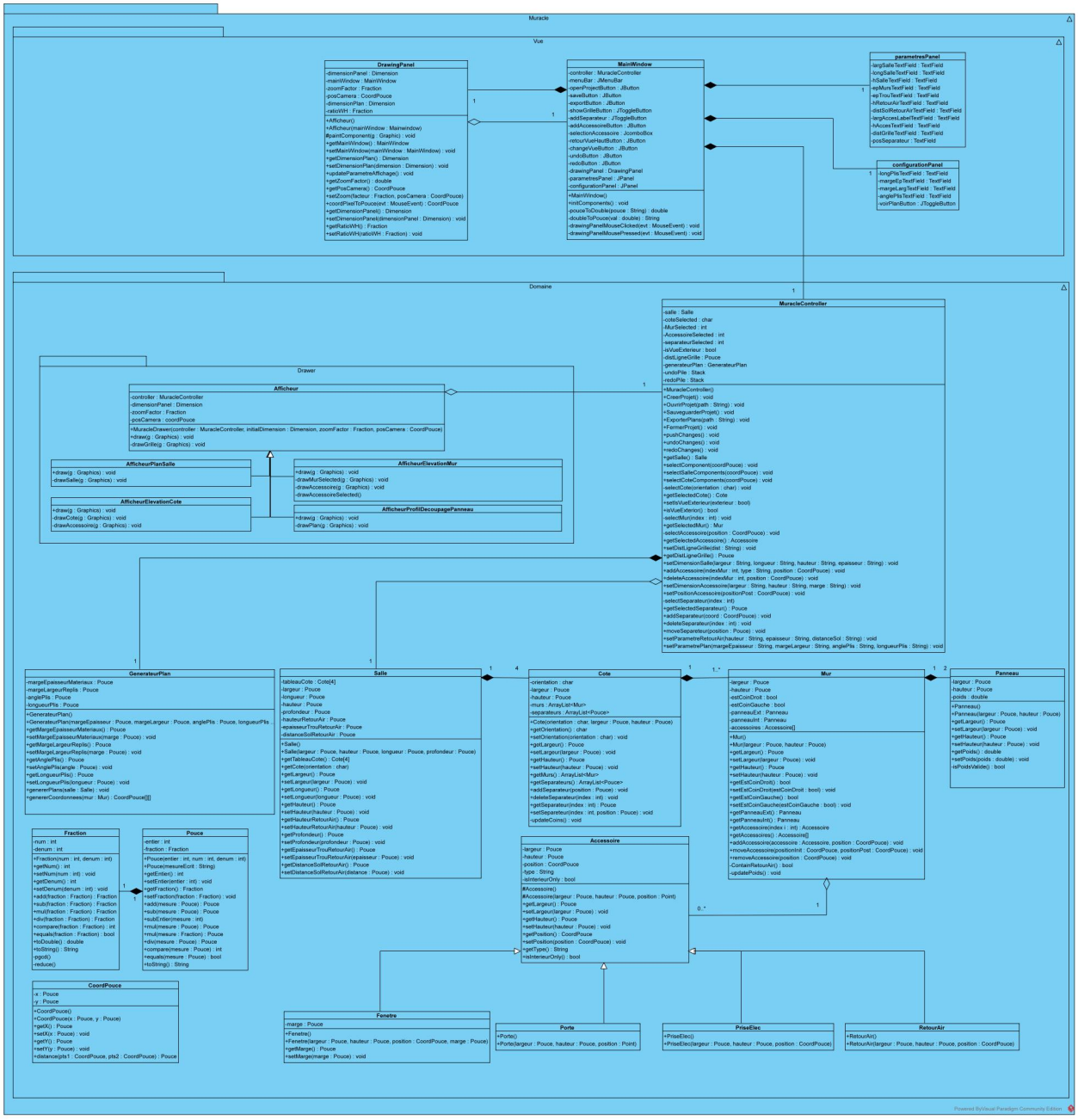
Matricule	*Nom*	*Signature*
536 890 723	Jérôme Lévesque	
536 896 717	Jérémi Girard	
111 261 390	Alex Prud'homme	
536 983 690	Maxime Bouchard	

Université Laval

18 octobre 2022

Table Des Matières

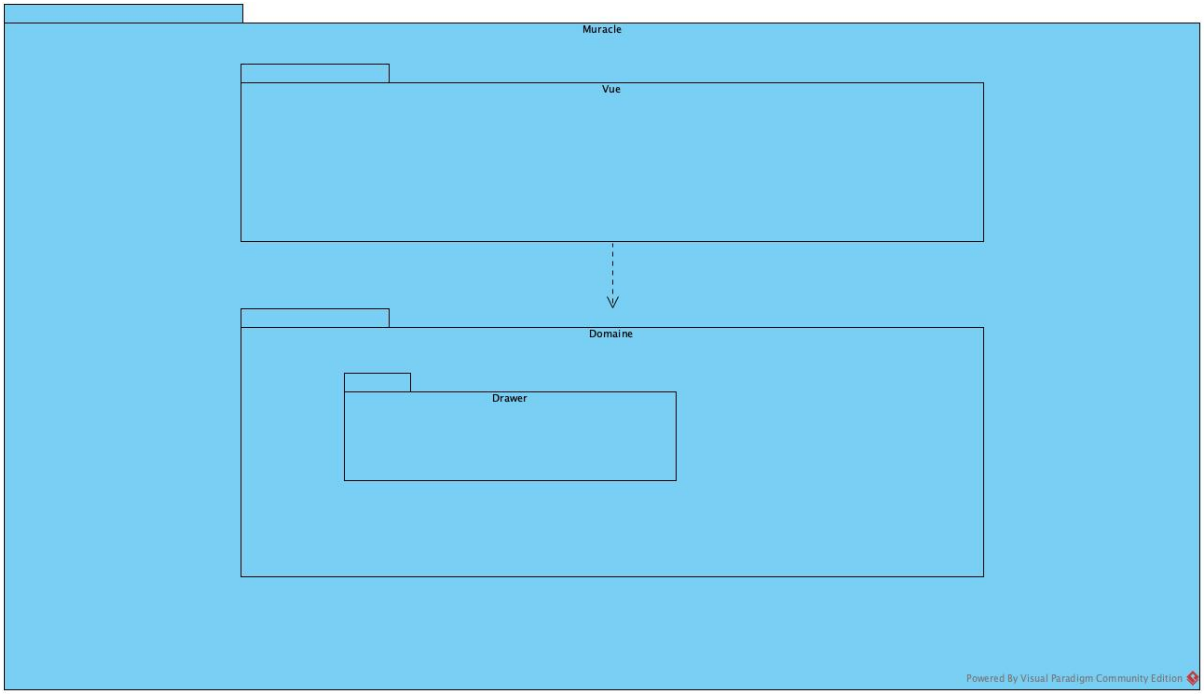
1. Diagramme de classes de conceptions



Notre diagramme de classes débute dans la vue avec la classe **MainWindow** qui sera l'interface logicielle avec laquelle communiquera l'utilisateur à l'aide de bouton et d'espace de texte (dans les objets **parametresPanel** et **configurationPanel** de la classe de base Panel) ainsi que la classe **DrawingPanel** qui elle s'occupera de l'interaction avec la souris et de gérer l'affichage en communiquant avec les différentes classes du drawer dérivées de la classe **Afficheur** (**AfficheurPlanSalle**, **AfficheurElevationMur**, **AfficheurElevationCote**, **AfficheurProfilDecoupagePanneau**). Ensuite, à l'entrée du domaine, nous avons notre classe **MuracleController** qui joue le rôle du contrôleur de Larman gérant donc toute l'information des classes contenue dans le domaine. Celui-ci contient toutes les fonctions

demander dans le projet à comme rôle primaire de manipuler un objet de la classe **Salle** et un objet de la classe **GenerateurPlan**. Le premier est une classe permettant de définir et manipuler les contraintes associées à la salle au complet telles que celles liées au retour d'air. Elle permet aussi de contenir quatre objets de la classe **Cote**. Le deuxième, de son côté, s'occupe des éléments concernant les plans de découpe et la génération de ceux-ci. Pour ce qui en est de la classe **Cote** mentionné plus tôt, celle-ci contient et manipule les séparateurs qui sont des valeurs et les objets de la classe **Mur** qui sont dépendants du premier. La classe **Mur** à la grande responsabilité de contenir et de manipuler un nombre arbitraire d'objets des dérivés de la classe **Accessoire** sans oublier deux objets de la classe **Panneau** représentant le panneau intérieur et le panneau extérieur. **Panneau** a quant à lui simplement le rôle de contenir et valider le poids des panneaux qui seront dessinés à la découpe. Pour en revenir à la classe **Accessoire**, cette classe définit les propriétés communes des différentes classes qui en sont dérivées (**Fenetre, Porte**, **PriseElec**, **RetourAir**). Ces classes se différencient avec leur type et si elles n'affectent que le panneau intérieur. La classe **Fenetre** ajoute également la propriété marge. Pour ce qui en est des classes utilisées par la majorité des classes du domaine, nous avons la classe **Pouce** qui permet de modéliser le format impérial et contient donc un entier et un objet de la classe **Fraction** (celle-ci est intuitive). Ces deux classes implémentent des méthodes pour les opérations mathématiques de base. Finalement, la classe **CoordPouce** permet de contenir la position des éléments en deux dimensions sous forme impériale et inclut une méthode de calcul de distance entre deux objets de cette classe.

2. Architecture logique

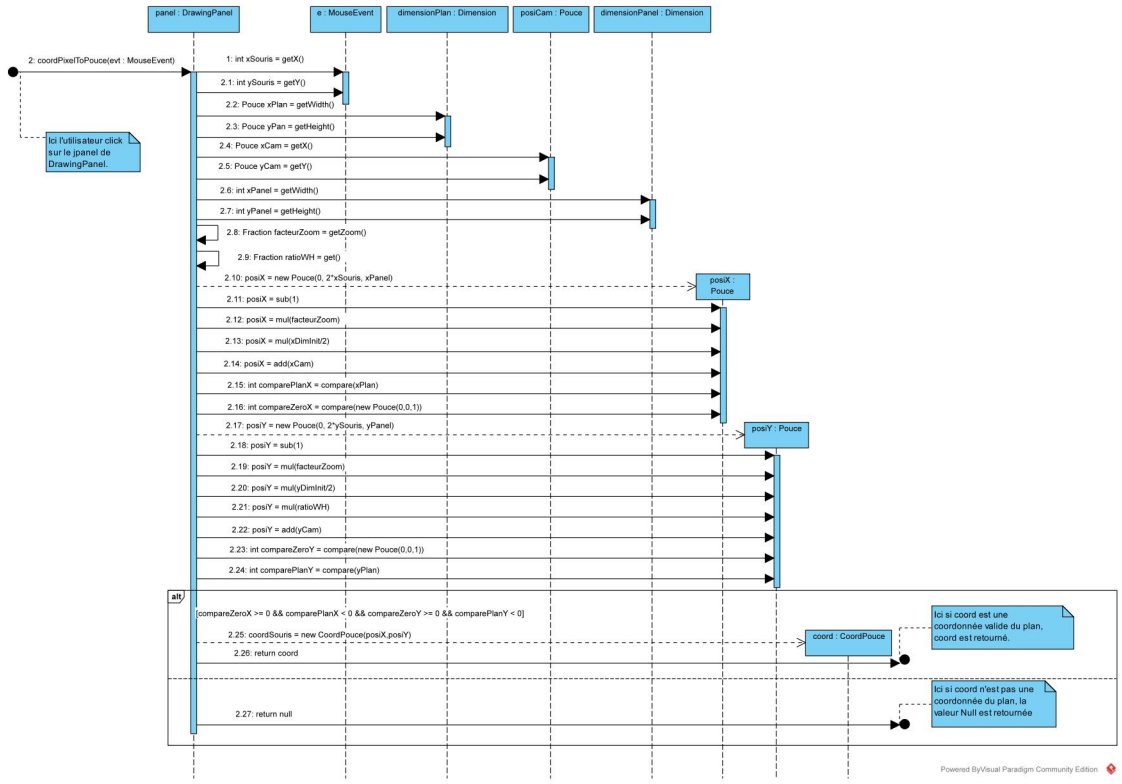


Notre diagramme de packages consiste en 4 packages différents. Le package Muracle est le package qui va contenir les packages Vue et Domaine et donc, l'entièreté du projet. Vue est le package qui va contenir les classes construisant l'interface. Ce package va contenir les classes comme le DrawingPanel et le MainWindow. Le Domaine va contenir la classe MuracleController qui aura comme rôle de contrôleur de Larman. Il va aussi contenir toutes les classes des différentes composantes comme Mur et Accessoire. Le Domaine va aussi contenir les différentes classes utilitaires comme Fraction et Pouce. Le dernier Package est le Drawer qui sera contenu dans Domaine. Celui-ci contient les classes servant à gérer les différentes classes d'affichage de la salle. C'est la classe Contrôleur qui relie les différents packages. La classe MainWindow du Package Vue va recevoir les indications du contrôleur pour ajuster l'interface.

3. Diagrammes de séquence de conception

3.1 Déterminer l'élément sur lequel a lieu un clique de souris dans la vue d'élévation d'un côté

3.1.1 Évènement click



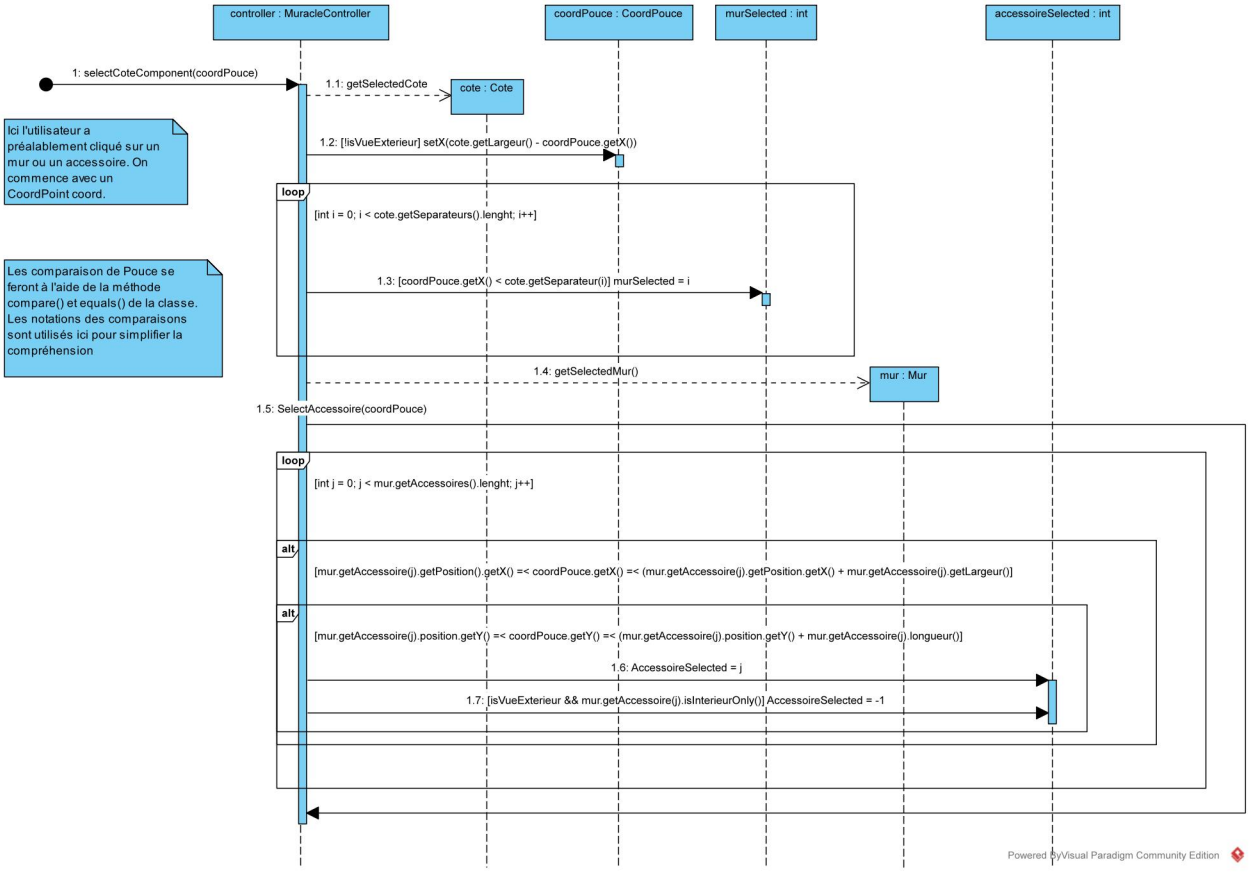
Lors de l'évènement click de la souris, DrawingPanel transfert le mouseEvent a sa fonction privée coordPixelToPouce qui fait un calcule pour trouver la position en pouce selon la position pixel de la souris. Le facteurZoom est le pourcentage de l'image de basse sur l'axe des x (exemple : 50/100 = 200% et 125/100 = 80%). Il faut inverser le dénominateur et numérateur pour obtenir le pourcentage de zoom. Le ratioWH est le ratio largeur/hauteur du plan. Il permet de trouver le facteurZoom sur l'axe des y, car les dimensions x et y du plan peuvent être de valeur différente. La position de la caméra est la coordonnée du plan attaché au centre du jPanel par rapport à la coordonnée (0,0) du plan. Les équations ci-dessous permettent de calculer la coordonnée de l'emplacement cliquer sur le jpanel en prenant le compte du facteurZoom et l'emplacement de la caméra. Les variables xPlan et yPlan sont les dimensions du plan, les variables xPanel et yPanel sont les dimensions du jpanel. xSouris et ySouris sont les coordonnées de l'emplacement clicker sur le jPanel.

posiX = xCam + xPlan/2 * facteurZoom * (2 * xSouris/xPanel-1)

posiY = yCam + yPlan/2 * facteurZoom * ratioWH * (2 * ySouris/yPanel-1)

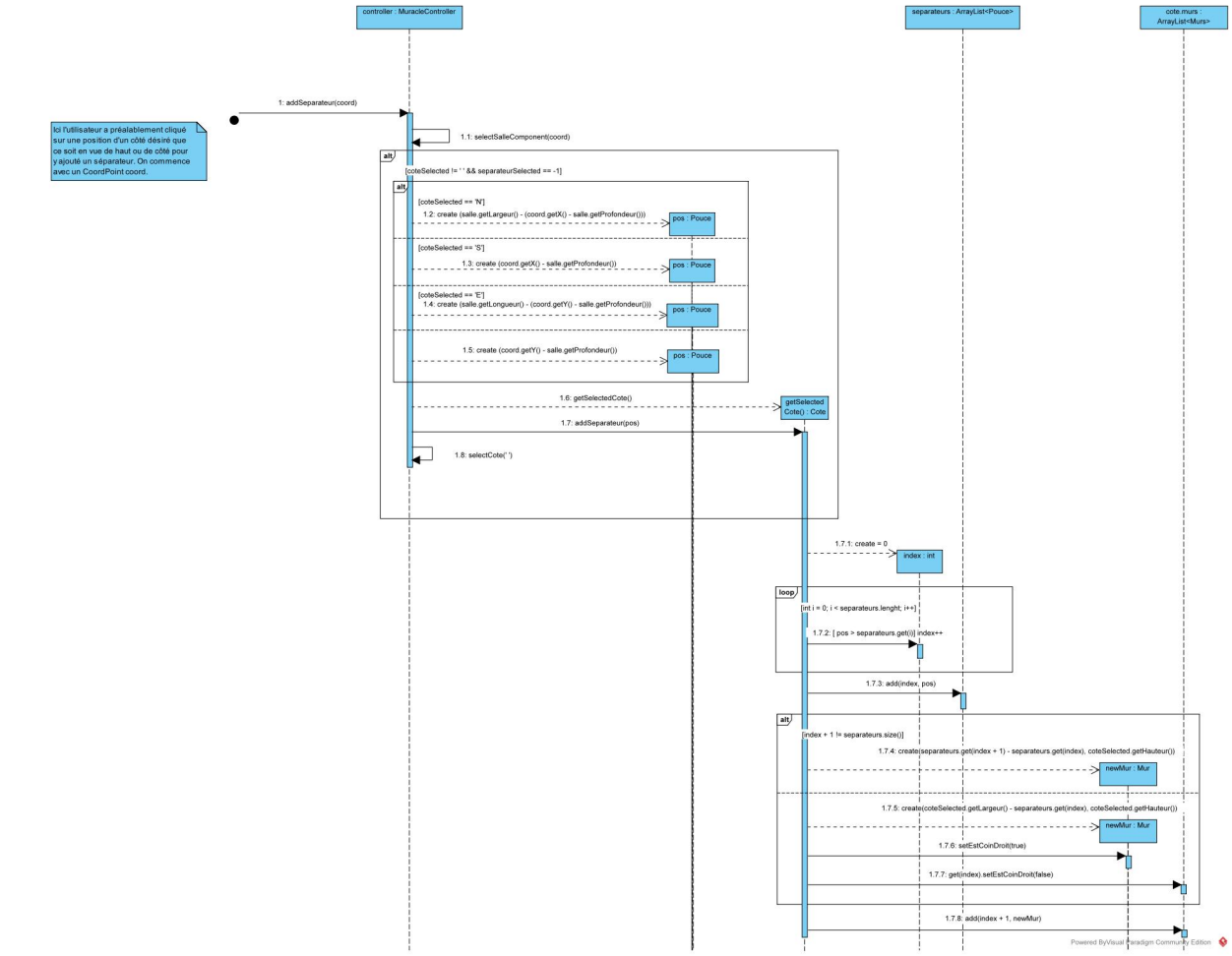
De plus, les variables compareZeroX_Y et zomparePlanX_Y sont initiés grâce à la comparaison entre la coordonnéeDuClick et x_yPlan et l'origine soit le point (0,0). La valeur du int représente un négatif pour "<", 0 pour "=" et positif pour ">". Si la coordonnée est à l'extérieur du plan, une valeur de null est retourné.

3.1.2 Selectionner Mur



Avec les coordonnées en pouce du clic, il va être possible de savoir sur quel élément (mur ou accessoire) on a cliqué avec la fonction SelectCoteComponent(). Elle débute en inversant sa position en x si on est en vue d'extérieur puisque les accessoires sont placés de gauche à droite de la vue extérieure en mémoire. Celle-ci contient ensuite une boucle qui sélectionne le bon mur grâce à la coordonnée X du point et à la valeur des séparateurs. Ensuite, on va effectuer la fonction SelectAccessoire() qui va vérifier si on a cliqué sur un accessoire en comparant la coordonnée x et y au domaine (position ainsi que les dimensions) de chacun des accessoires contenus dans le mur. Si un des accessoires a bel et bien rempli les conditions du clic, on lui donne le focus en le sélectionnant. Dans le cas où cette position est un accessoire d'intérieur et que nous sommes en vue extérieurs, on désélectionne l'accessoire puisqu'il n'est pas accessible.

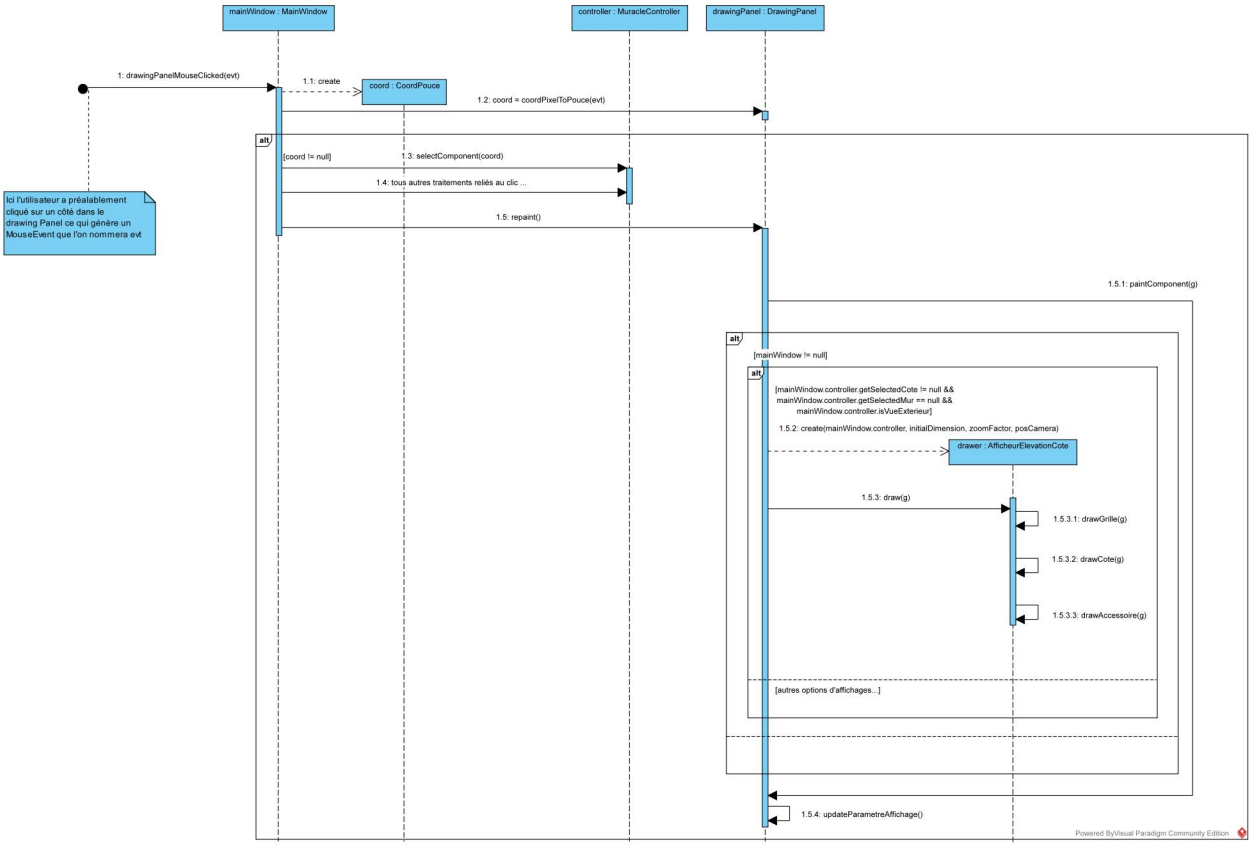
3.2 Créer un séparateur



On débute par l'utilisateur qui appuie sur le bouton afin d'activer le mode d'ajout de séparateur. Par la suite l'utilisateur place sont séparateur sur le côté désirer entre les murs déjà présents. Ceci aura pour effet d'appeler la fonction d'ajout du séparateur. Cette fonction obtient sa position par rapport au coin en haut à gauche du côté en pouce. On utilise la méthode selectSalleComponent() avec notre coordonnée pour déterminer sur quel côté de la salle le clic a été effectué. Si celui n'est sur aucun côté ou sur la position d'un séparateur déjà existant, la fonction s'arrête ici. Sinon, on initialise une variable nommée pos de la classe Pouce et lui affecte une valeur qui dépendent de la vue extérieure en parcourant de gauche à droite en n'oubliant pas de soustraire l'écart créer par les coins. Puis, on demande à notre côté sélectionné de rajouter le séparateur pos en désélectionnant le côté par la suite pour ne pas changer de vue et puisque que le contrôleur a plus besoin de côté ou le clic se trouvait. Du côté de la méthode addSeparateur(pos), on commence en créant un index de 0 pour parcourir les séparateurs du côté sélectionné en comparant chaque séparateur avec notre nouveau pour connaitre son emplacement en prenant compte du fait que notre Arraylist est ordonné. Une fois ajouté dans notre liste de séparateurs, on vérifie si notre séparateur n'est pas le dernier de la liste. SI ce n'est pas le cas, on utilise le séparateur suivant pour définir la largeur de notre nouveau mur. Sinon, on utilise la largeur du côté sélectionné, on l'affecte comme étant le coin droit (puisqu'il sera le dernier mur) et on désaffecte le

mur qui occupe cette position présentement (il ne sera plus le coin droit). Finalement, on ajoute le nouveau mur à la liste de mur à la position suivant le séparateur.

3.3 comment l'affichage de la vue d'élévation externe d'un côté sera réalisé



On débute avec un mouseEvent evt que mainWindow gère en appelant la méthode drawingPanelMouseClicked() avec l'évent en paramètre. On initialise ensuite une coordonnée en pouce appelé coord en utilisant coordPixelToPouce() du drawingPanel avec l'objet evt qui nous fournis la position en pouce par rapport au coin gauche de la salle ou le mur (dépends de la vue) ou un null dans le cas où l'utilisateur clic dans le vide que l'on filtre à l'étape suivante. Puis, on effectue les fonctions reliées au clic que ce soit de la vue de dessus ou de côté. Maintenant, on appelle la méthode repaint() de drawingPanel pour mettre à jour l'affichage. Celle-ci appelle sa propre méthode paintComponent avec l'objet g de la classe Graphic. Dans cette méthode, on vérifie d'abord si mainWindow n'est pas null et ensuite que les prérequis des différents affichages sont bel et bien remplis. Dans le cas présent, c'est l'affichage de vue d'élévation externe qui nous intéresse. On crée un objet drawer de la classe AfficheurElevationCote pour pouvoir appeler sa méthode draw() avec l'objet g. De son côté, drawer

appellera ses différentes méthodes de dessin avec l'objet g. On termine avec en mettant à jour les paramètres de l'affichage de drawingPanel (posCam, ratioWH, planDimension...).

4.Algorithme en pseudo-code

Ceci est un algorithme de pseudo-code pour déterminer les points X, Y afin de les envoyer dans le SVG et ainsi générer un plan à partir de ce dernier. Le pseudo-code ci-dessous effectue une addition et soustraction de plusieurs variables et constante déterminer ou non par l'utilisateur. Le point de départ est en fonction du sommet de la forme le plus haute, qui aura pour y = 0 ainsi que le sommet le plus à gauche de la forme qui aura pour x=0.<

- ListeDeSommets = Tous les sommets de la forme nécessaire à la création de ce dernier
- CordonnéesEnPouce = Est un objet qui représente une valeur en X et Y pour déterminer un point
- LonguerDePlis = la variable déterminer par l'utilisateur pour la longueur du haut du mur qui se plis pour en faire l'assemblage
- MargeEpaisseur = La marge qui est entré par l'utilisateur en paramètre en fonction de l'erreur de longueur lors du repli qui dépend du matériau utiliser.

ListeDeSommets (Cordonnées en pouce)

CordonnéesEnPouce sommet1 <- CordonnéesEnPouce(0, LongueurDePlis EpaisseurMur + MargeEpaisseur)

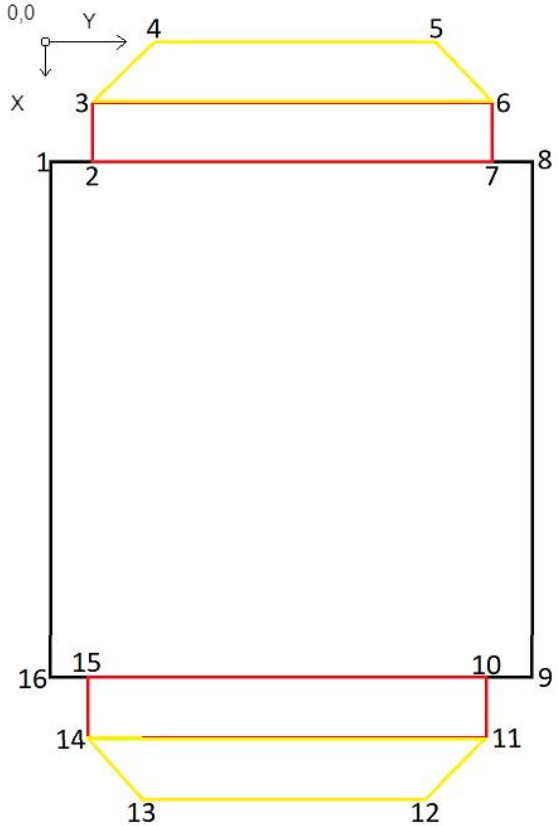
ListeDeSommets[0] <- (sommet1)

CordonnéesEnPouce sommet2 <- CordonnéesEnPouce(La coordonné X du sommet 1 + MargeLargeurReplis,La coordonné Y du sommet 1)

ListeDeSommets[1] <- (sommet2)

CordonnéesEnPouce sommet3 <- CordonnéesEnPouce(La coordonné X du sommet 2,La coordonné Y du sommet 1 - (EpaisseurMur + MargeEpaisseur))

ListeDeSommets[2] <- (sommet3)



CordonnéesEnPouce sommet4 <- CordonnéesEnPouce(LongeurDePlis / tan(angle de replis),0)

ListeDeSommets[3] <- (sommet4)

CordonnéesEnPouce sommet5 <- CordonnéesEnPouce(Largeur - coordonné du sommet4 en X,0)

ListeDeSommets[4] <- (sommet5)

CordonnéesEnPouce sommet6 <- CordonnéesEnPouce(Largeur - coordonné du sommet3 en X,coordonné du sommet3 en Y)

ListeDeSommets[5] <- (sommet6)

CordonnéesEnPouce sommet7 <- CordonnéesEnPouce(Largeur - coordonné du sommet2 en X,La coordonné Y du sommet 1)

ListeDeSommets[6] <- (sommet7)

CordonnéesEnPouce sommet8 <- CordonnéesEnPouce(Largeur,La coordonné Y du sommet 1)

ListeDeSommets[7] <- (sommet8)

CordonnéesEnPouce sommet9 <- CordonnéesEnPouce(coordonné du sommet8 en X,La coordonné Y du sommet 1 + Hauteur)

ListeDeSommets[8] <- (sommet9)

CordonnéesEnPouce sommet10 <- CordonnéesEnPouce(coordonné du sommet7 en X,La coordonné Y du sommet 1 + Hauteur)

ListeDeSommets[9] <- (sommet10)

CordonnéesEnPouce sommet11 <- CordonnéesEnPouce(coordonné du sommet6 en X,La coordonné Y du sommet 1 + Hauteur + EpaisseurMur + MargeEpaisseur)

ListeDeSommets[10] <- (sommet11)

CordonnéesEnPouce sommet12 <- CordonnéesEnPouce(cordonné du sommet5 en X, coordonné du sommet11 en Y + LongeurDePlis)

ListeDeSommets[11] <- (sommet12)

CordonnéesEnPouce sommet13 <- CordonnéesEnPouce(cordonné du sommet4 en X, coordonné du sommet12 en Y)

ListeDeSommets[12] <- (sommet13)

CordonnéesEnPouce sommet14 <- CordonnéesEnPouce(cordonné du sommet3 en X, coordonné du sommet11 en Y)

ListeDeSommets[13] <- (sommet14)

CordonnéesEnPouce sommet15 <- CordonnéesEnPouce(cordonné du sommet2 en X, coordonné du sommet10 en Y)

ListeDeSommets[14] <- (sommet15)

CordonnéesEnPouce sommet16 <- CordonnéesEnPouce(cordonné du sommet1 en X, coordonné du sommet9 en Y)

ListeDeSommets[15] <- (sommet16)

5.Diagramme de Gantt mis à jour

Itérations (semaine)	5-6	7-8	9	10-11	12-13	14-15
-Créer nouvelle Salle						
-Modifier dimension de la salle						
-Gestion nombre Impériaux						
-Zoomer/Dézoomer						
-Changer de vue						
-Modifier configuration du plan						
-Ajouter et Supprimer séparateur						
-Modifier mur						
-Ajouter et Supprimer accessoire						
-Modifier accessoire						
-Déplacer accessoire						
-Déplacer séparateur						
-Voir Plan						
-Afficher grille d'aide au positionnement						
-Export SVG						
-Undo/Redo						
-Sauvegarder le projet						
-Ouvrir/Fermer un projet						

6.Contribution de chacun des membres de l’équipe

Jérémi Girard : Diagramme de Gantt, Diagramme de séquence de conception ClickSouris.

Jérôme Lévesque : Diagramme de classes de conceptions, diagramme de séquence de conception 3.3, version fonctionnelle du programme et exécutable .jar

Alex Prud’homme : Diagrammes de classes de conceptions, diagramme de package, diagramme de séquence de conception 3.1.2.

Maxime Bouchard : Diagramme de séquence de conception Ajouter séparateur, Pseudo-Code Coordonnées des points.