UNIVERSITÉ LAVAL

Faculté des sciences et de génie

Équipe N°09

Livrable 2

Ismaël Sdiri (111 271 868)
Moulay-Mostafa Filali (536 758 151)
Oumar Rafiou Barry (111 285 303)
Souleymane Kane (536 917 164)
Sabir, Sami (536 855 002)

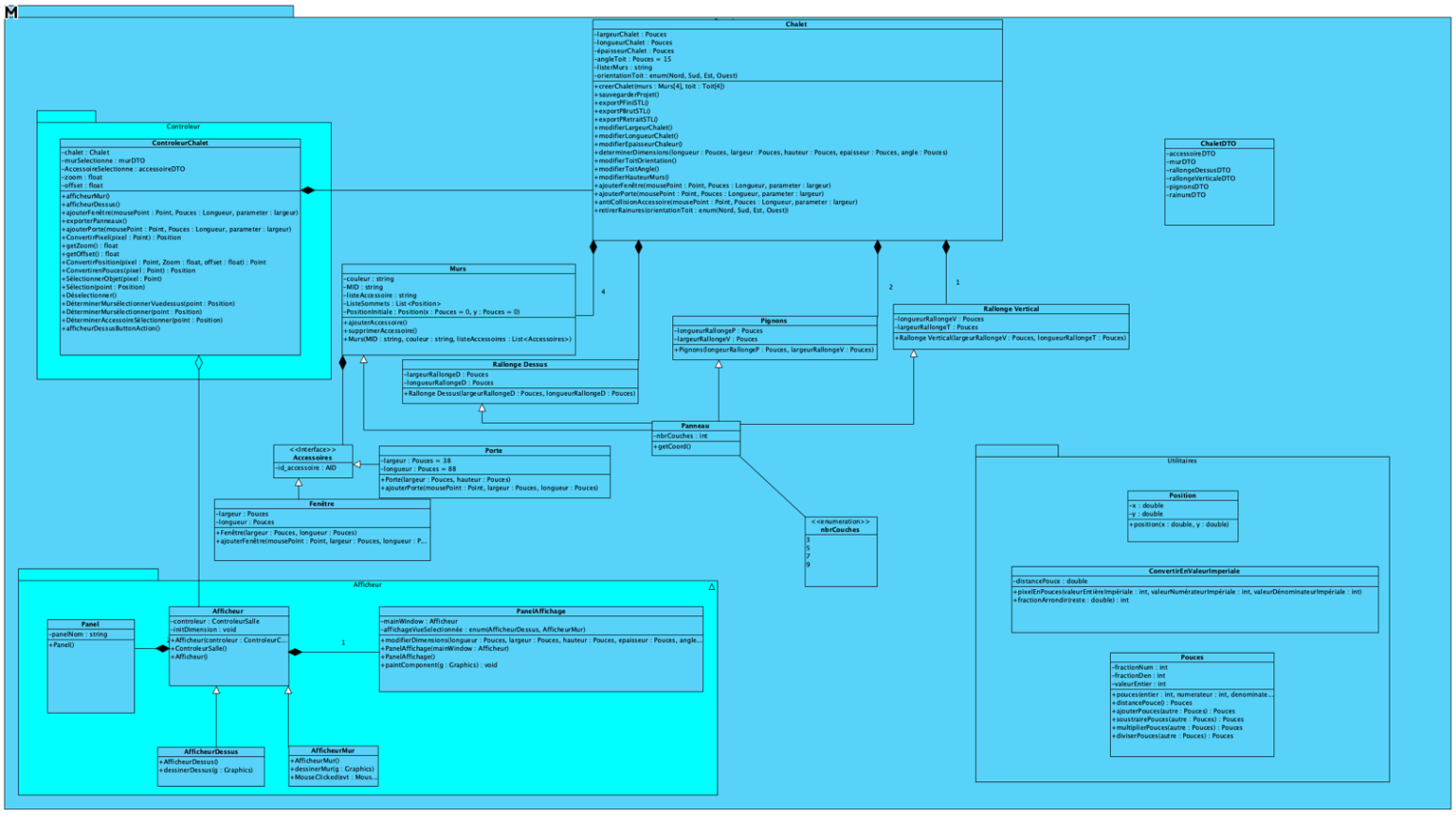
Génie logiciel orienté objet
GLO-2004
NRC: 85321

Travail présenté à
Marc-Philippe Parent

Automne 2023
17 octobre 2023

Table des matières

Diagramme de classes de conception



La conception, c'est avant tout une abstraction des mécanismes qui vont nous aider à résoudre le problème auquel nous sommes confrontés. Le diagramme des classes de conception présenté dans ce rapport fait gage de modèle pour nous permettre de concevoir une architecture robuste pour la mise au point de l'application ChalCLT. Le principe fondamental qui fût considérer dans la réalisation de ce dernier est celui du contrôleur de Larmann, principe qui veut à utiliser une classe **Contrôleur** permettant de faire le pont entre les différentes couches applicatives. Le projet est également séparé en dossiers, que l'on appelle « packages » ; **contrôleur**, **afficheur**, **utilitaire** et **Domaine.**

Dans **Domaine**, la modélisation des différentes classes permettant de générer un résultat est décrite. La classe **Chalet** est la classe qui représente l'objet principal du projet. Elle est composée de quatre classes : **Murs**, qui sont du nombre de quatre, **Rallonge Dessus**, **Pignons, du nombre de deux** et **Rallonge Vertical**, qui sont du nombre de deux. Ces classes sont toutes une

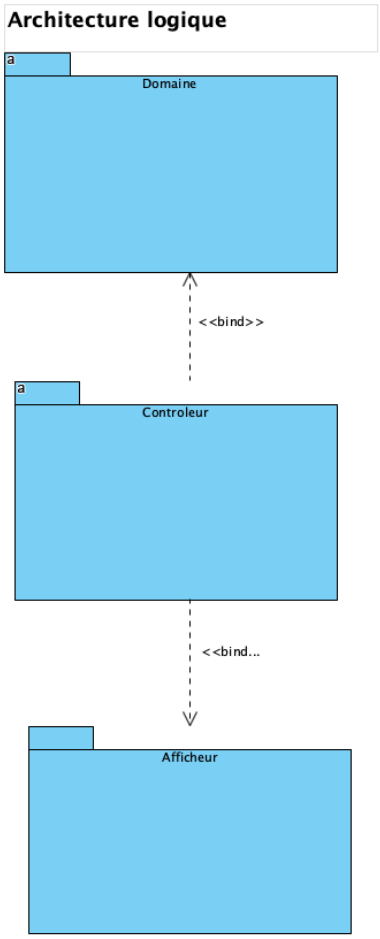
forme de **Panneau**, classe étant en relation avec un **nombre de couches**. On remarque que Mur à une particularité unique ; c'est qu'elle est composée d'**Accessoires** : des **Portes** et des **Fenêtres**.

Le dossier **Contrôleur**, lui, est composé d'une unique classe **ControleurChalet** qui permet de faire l'appel aux différentes fonctions faisant le pont entre l'interface utilisateur et la réalisation dorsale. Il permet d'utiliser les fonctions d'affichages, d'ajouts et d'exportation.

L'**Afficheur** est le dossier qui représente le développement d'interface de l'application. Il est composé d'une classe **Afficheur** qui elle est composé de deux **Panel**s qui serviront aux modifications et d'un **Panel Affichage** sur lequel il y aura l'affichage du chalet. Il pourra faire cela grâce aux deux affichages qui lui sont proposés qui eux sont des descendants d'**Afficheur** ; **AfficheurDessus** et **AfficheurMur**.

Le package utilitaire est composé d'une classe **Position** (en x et en y), d'une classe permettant de **Convertir en valeurs impériales** et d'une classe **Pouce**.

Architecture logique



Package Domaine :
Ce package contient toutes les classes nécessaires pour créer le Chalet. Elle contient la classe Chalet et ses composantes. Le lien entre les classes du domaine et l'exécution se fait par l'entremise du contrôleur.

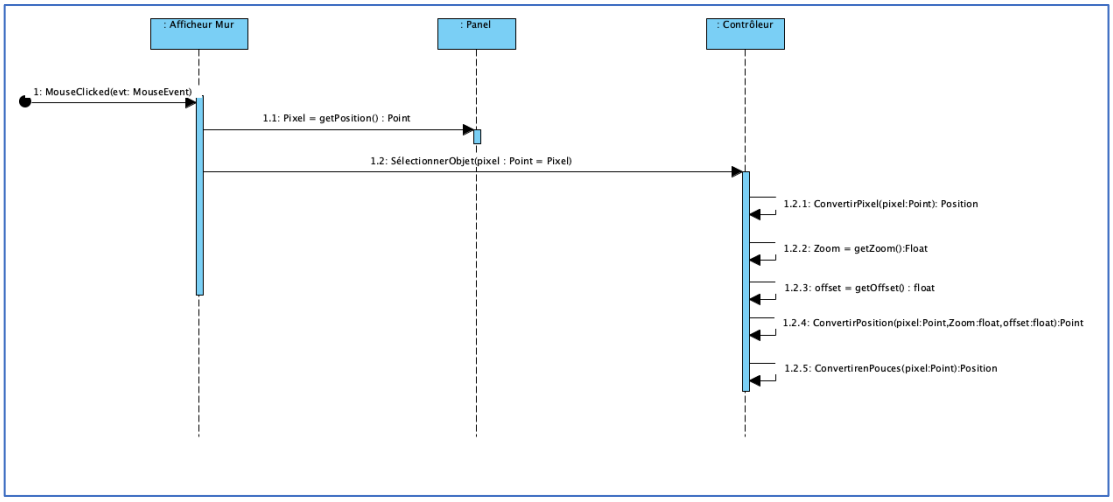
Package Contrôleur:
Ce package contient les méthodes nous permettant d'exécuter ce que l'utilisateur souhaite. C'est à dire soit ajouter un accessoire, exporter en STL, etc. Toute communication avec l'interface est gérée par la classe Contrôleur, qui est le contrôleur de Larmann.

Package Afficheur : Il s'agit de l'ensemble des classes qui seront responsable du dessin du Chalet dans le JPanel.

Diagrammes de séquence de conception (DSC)

1. Déterminer l'élément sur lequel a lieu un clic de souris dans la vue d'un Mur :

1.1 : Clic de souris

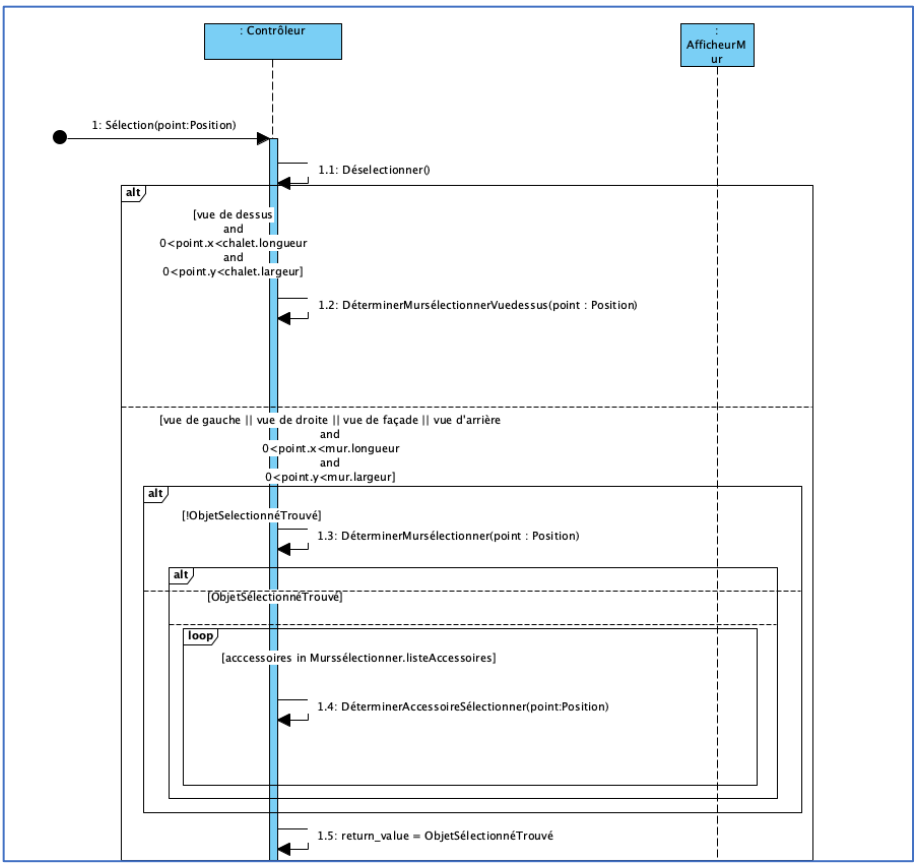


Le diagramme débute par un événement, soit le clic de souris dans l'AfficheurMur qui présente le chalet avec une vue de l'élévation de côté. Suite au clic dans le Jpanel, nous prenons sa position à l'aide de la méthode getPosition() de Java.
Les informations sont ainsi stockées dans une variable nommé Pixel sous le format Point de Java. Puis, la fonction ConvertirPixel() commence par récupérer les données du zoom en utilisant la méthode getZoom() ainsi que les données de décalage avec getOffset(). L'offset est déterminé en fonction d'un point de référence, ici ce sera **l'extrémité inférieure la plus à gauche de notre mur**. À l'aide de la fonction ConvertirPosition(), nous utilisons comme paramètres notre point, notre zoom et notre offset.

Calcul des coordonnées Post Zoom :
Coordonnée X Post Zoom = (Coordonnée écran X) ÷ Zoom + Offset X
Coordonnée Y Post Zoom = (Coordonnée écran Y) ÷ Zoom + Offset Y

Après ces calculs, nous obtenons un point. Ce point, qui est en réalité une coordonnée, est ensuite converti en pouces grâce à la fonction ConvertirEnPouces.

1.2 Sélection d'éléments



Pour sélectionner un objet, le contrôleur utilise la fonction **Sélection(),** renvoyant un booléen positif si la sélection réussit et un booléen négatif en cas d'échec. Cette fonction appelle Deselectionner(), réinitialisant l'attribut accessoiresSélectionné. Si on est dans la vue de dessus et que les points de la sélection respectent l'intervalle du chalet. Or, dans le cas échéant où l'on se trouve sur l'une des 4 vues et que le point se trouve dans les dimensions de la vue représenté, murSélectionné est activé en cas de clic sur un mur.

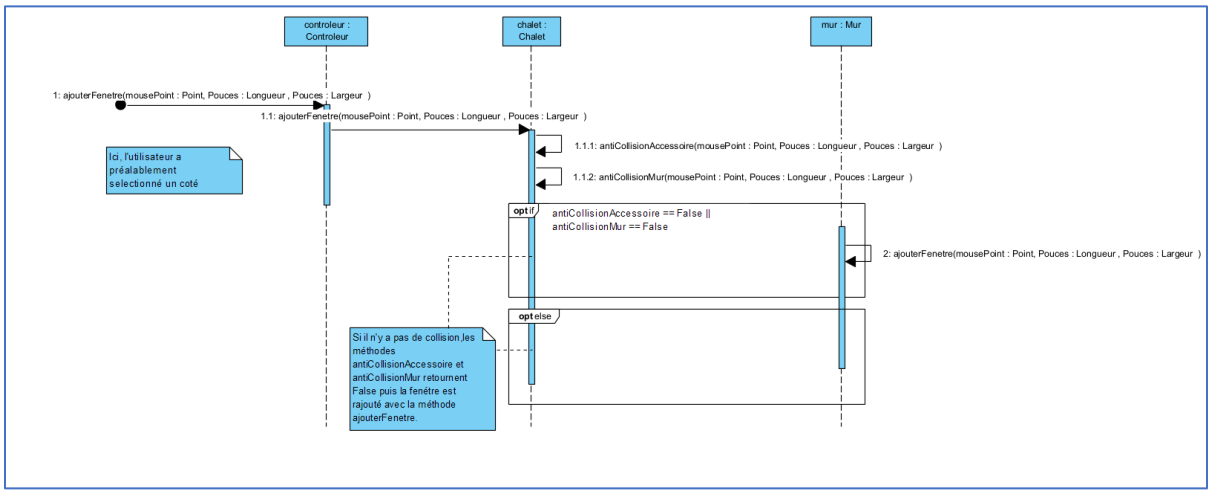
Lorsqu'un mur est sélectionné, **DéterminerAccessoireSélectionner** itère à travers les accessoires de listeAccessoires du mur sélectionné. Si un clic est détecté sur un accessoire, accessoireSélectionné est modifié. Enfin, la valeur ObjetSélectionTrouvé est renvoyée, étant False si aucun objet n'est sélectionné et True s'il y en a un. C'est une fonction de validation.

La fonction **DéterminerMursélectionnerVuedessus** sert à déterminer si le Point passé en argument est présent sur un des murs lors de la vue de dessus. La fonction examine la position en x et y pour vérifier si elles se situent dans l'intervalle d'un des murs. Si tel est le cas, elle effectue une vérification pour voir si le mur était déjà sélectionné. Si c'était le cas, cela signifie qu'il y a eu un double-clic, et le murSélectionné est alors réinitialiser.

La fonction **DéterminerAccessoireSélectionner** sert à vérifier si le point passé en argument se situe sur l'un des accessoires de la vue. Elle parcourt la liste d'accessoires et, pour chaque accessoire, examine sa position et ses dimensions. Ensuite, elle vérifie si le point se trouve dans le polygone de l'accessoire. Enfin, elle attribue l'accessoire à l'attribut accessoireSélectionné du chalet.

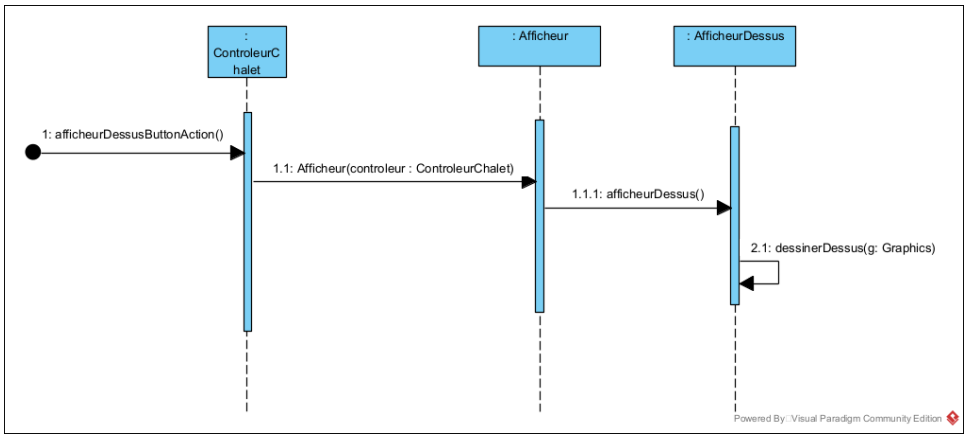
La fonction **DeselectionnerObjets** est chargée de réinitialiser accessoireSélectionné à une valeur nulle. Ceci est essentiel pour éviter d'avoir plusieurs objets sélectionnés en même temps. Il est important de noter que l'attribut mursSélectionné n'est pas inclus dans cette opération de désélection, car un double-clic sur un mur peut se produire et celui possède sa propre fonction pour gérer la sélection et la désélection de ce côté.

2. Créer une fenêtre :



Pour rajoute une fenêtre le contrôleur appelle la méthode **ajouterFenetre** en lui passant en paramètre les coordonnées du clic de souris c-a-d mousePoint, ainsi que les dimensions de la fenêtre en pouces. Cette méthode appelle une méthode de nom similaire dans salle en lui transmettant ses arguments qui a son tour va les transmettre aux méthodes **antiCollisionMur** et **antiCollisionFenêtres** qui retourne true s'il y'a collision entre différents accessoires ou que l'accessoire ne se situe pas dans un mur où false si tel n'est pas le cas. Sur la base de ce résultat la méthode **ajouterFenetre** de fenêtre est appelée ou non avec les mêmes arguments afin que l'objet fenêtre soit créé.

3. Comment l'affichage de la vue de la vue du dessus sera réalisé :

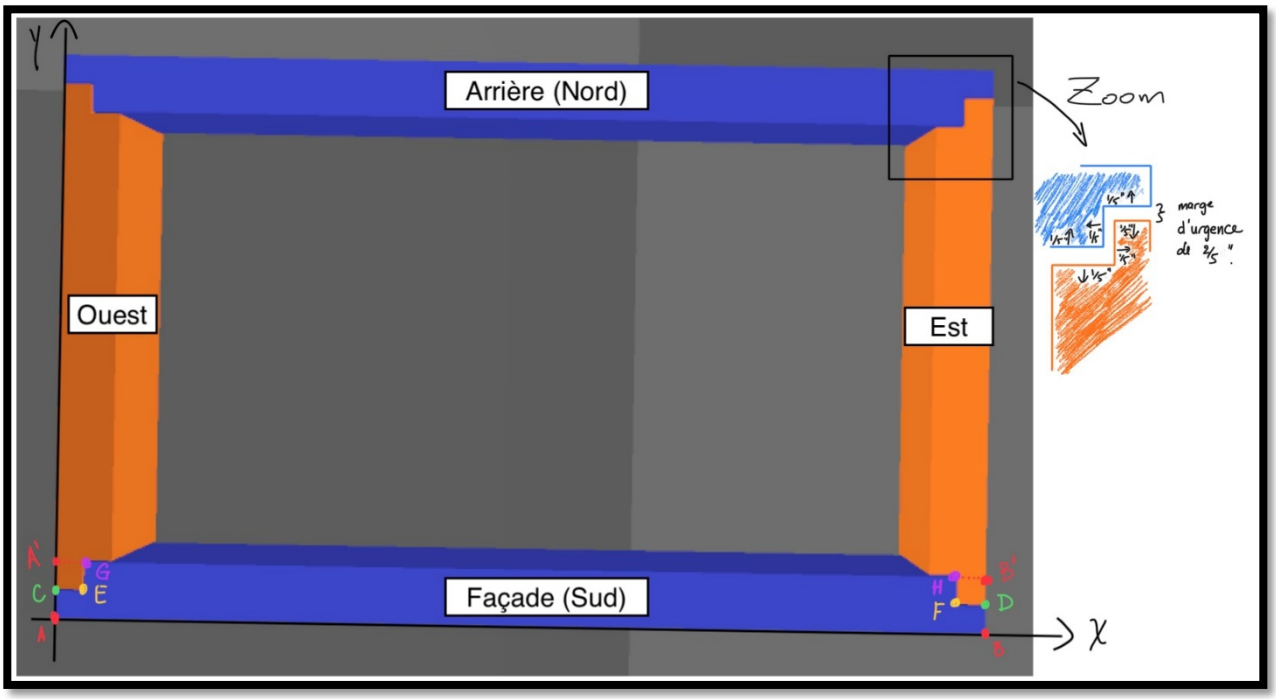


Pour avoir la vue de dessus, le contrôleur utilise la méthode afficheurDessusButtonAction(). Lorsque ce bouton est activé, la méthode Afficheur de la classe Afficheur est activé. À son tour, il appelle la méthode afficherDessus de la classe AfficheurDessus. C'est dans cette classe que la méthode dessinerDessus peut être utilisé, contenant l'objet Graphics.

Pseudo-code

Le pseudo-code ci-dessous représente un algorithme de retrait des rainures du mur. En effet, un chalet est composé de 4 murs et de 4 composantes de toit. Chacun des murs est, à la base, un simple prisme rectangulaire ayant une longueur, une largeur, une épaisseur et huit sommets. Toutefois, si nous procédons au découpage des rainures depuis la vue de haut, nous n'aurons à prendre en compte que 4 sommets (ceux du haut du mur), car ceux de la base du mur auront les mêmes coordonnées en x et en y, mais n'aurons qu'une hauteur différente.

La fonction de rainure permet de découper un petit prisme à base rectangulaire sur les côtés internes des murs afin de permettre aux différents murs du chalet de s'emboîter les uns dans les autres. Pour faciliter la compréhension de la fonction de rainure, voici une image des murs d'un chalet aux rainures découpées :



Vue de haut d'un chalet avec ses quatre murs

Comme il est possible de le voir, les sommets du dessus du mur de façade de ce chalet ont été dessinés de plusieurs couleurs et ont une lettre qui leur est attribuée. Puisque la hauteur est la même pour tous les murs, il nous est possible pour l'instant d'oublier le facteur hauteur et de voir l'esquisse qu'en 2D, d'où l'image ci-dessus.

Il faut s'imaginer qu'avant le découpage des rainures, le mur de façade n'est qu'un prisme à base rectangulaire ayant une longueur, une hauteur et une épaisseur. Depuis la vue de haut, l'axe des x représente la longueur du mur, alors que l'axe des y, ici, représente l'épaisseur du mur de façade. D'un point de vue de surplomb, ce ne serait, donc qu'un rectangle avec une longueur et une épaisseur et 4 sommets, dessinés ici en tant que A, B, A' et B' (tous en rouge).

Lors du découpage des rainures, il faudra considérer les deux points suivants :

- **L'orientation du toit**
  En effet, comme il est possible de le voir, les quatre murs ont une direction (N, S, E, O). Dans ce projet, nous allons prendre en compte que la rallonge verticale se situe sur le mur désigné par orientation du toit. Si l'orientation du toit est Nord par exemple, cela veut dire que la rallonge verticale du toit est sur le mur Nord et qu'elle a une pente descendante vers le mur opposé, soit de façade dans ce cas. Cela veut aussi dire que ce seront ces mêmes murs (d'arrière et de façade) qui déborderont, car le toit est dans cette direction. Dans le pseudo-code, cet aspect est pris en compte pour calculer les sommets d'un mur en fonction du fait qu'il déborde ou non.

- **Édition des sommets**
  Il est important de noter que certains sommets peuvent garder les mêmes coordonnées que celles initialisées pour le mur, notamment les points rouges A et B si le mur de façade reste celui qui déborde.

  Certains sommets peuvent être modifiés, notamment les points fictifs A' et B', qui représentaient les points intérieurs du mur de façade avant rainures, changeront de coordonnés et deviendront G et H. Bref, on peut considérer que A' et B', en changeant de place, deviennent les sommets intérieurs du mur rainuré, donc les sommets G et H.

  Finalement, certains sommets (les sommets C, D, E, F) seront rajoutés à la liste de sommets d'un mur lors du découpage des rainures, car le mur ne devient plus qu'un simple prisme à base rectangulaire, mais plutôt un polyèdre.

Il faut garder en tête quelques détails concernant ce pseudo-code afin de mieux le saisir:

- Une classe Chalet sera créé et elle contiendra une liste d'objets Murs, qui eux, seront créés dans la classe Mur. Elle contiendra aussi l'orientation du toit.

- La classe Point ou bien une classe Sommet sera créé et permettra d'instancier des sommets. Des méthodes permettront de changer leurs valeurs en X et en Y ou bien de rajouter/supprimer des sommets à un Mur quelconque.

- Dans le pseudo-code, nous considérons le point initial (de référence) de la vue de surplomb comme le coin inférieur gauche de n'importe quel mur qui dépasse l'autre. C'est, donc ce point qui a les coordonnées (0, 0), et on peut voir sur l'image que le point A est placé au croisement des axes. Si le mur de côté avait dépassé, ce serait son point qui serait placé à l'origine.

- Les distances de 0,2 (1/5 de pouces) pouces rajoutés dans les calculs des rainures, sont des distances arbitraires ajoutées au découpage de chaque rainure pour prévenir les erreurs d'usinage. Au total, la distance entre deux murs imbriqués sera de 0,4 pouces (2/5 de pouces), ce qui est négligeable et leur permettra, toutefois, de s'imbriquer sans problème. Le petit carré au coin supérieur droit dans l'image de surplomb fait un zoom sur l'agencement des rainures et expliques que la distance de 0,2 pouces sera prise en compte dans la direction de chaque flèche.

- Plusieurs lignes de commentaires ont été ajoutées au sein du pseudo-code et qui font référence à la figure du dessus permettent de faciliter la compréhension du pseudo-code et des sommets changés ou rajoutés.

Fonction retirerRainuresMurs

```
//Pour le bien de l'explication de ce pseudo-code, nous allons fournir une partie de la classe Mur pour
//illustrer qu'un mur sera initialisé et composé de sommets depuis le début, et donc, lui retirer des
//rainures revient à modifier certains de ces sommets et à en lui ajouter d'autres au besoin. Les sommets
//peuvent être de type Point ou d'une classe similaire crée à cet effet.
Class Mur {
    Constructeur Mur (nom, longueur, hauteur, epaisseur)
        this.nom = nom;
        this.longueur = longueur;
        this.hauteur = hauteur;
        this.epaisseur = epaisseur;
        this.sommets = ListeVide;
}


//La fonction retirerRainures permet de prendre un mur et de lui retirer ses rainures en fonction de son
//épaisseur et de la marge de sécurité établie.
//
//La fonction permet de retourner une Liste des sommets du mur en
//question. Certains sommets resteront pareils, d'autres changeront alors que d'autres seront crées et
//ajoutés à la liste de sommets du mur.
// On considère dans ce cas que à la base, si l'on regarde depuis le surplomb, le mur de façace est initialisé tel que:
// SommetA = (0, 0)
// SommetB = (longeur, 0)
// SommetG = (0, epaisseur)
// SommetH = (longueur, epaisseur)
Fonction retirerRainuresMurs(Mur, Orientation du Toit)
    if (Orientation du Toit = "Nord" ou Orientation du Toit = "Sud") //Veut dire que la rallonge verticale
                                                                    //est sur le mur Nord ou Sud, si on regarde en vue surplomb.
        if (Mur.nom == "Façade")
            sommets.get(2).setX((epaisseur/2) + 0.2);                                   //Modifier X du SommetG (Mauve)
            //Nous n'avons pas à modifier Y, car elle reste celle de l'épaisseur du mur.

            sommets.get(3).setX((longueur-(epaisseur/2) - 0.2));                        //Modifier SommetH (Mauve)
            //Nous n'avons pas à modifier Y, car elle reste celle de l'épaisseur du mur.

            sommets.add(new Sommet(0, (epaisseur/2)-0.2);                               //Ajout SommetC (Vert)
            sommets.add(new Sommet(longeur, (epaisseur/2)-0.2));                        //Ajout SommetD (Vert)

            sommets.add(new Sommet((epaisseur/2) + 0.2, (epaisseur/2) - 0.2));          //Ajout SommetE (Jaune)
            sommets.add(new Sommet(longeur - (epaisseur/2) - 0.2, (epaisseur/2) - 0.2));//Ajout SommetF(Jaune)

        if (Mur.nom == "Arrière") // va avoir les mêmes points en X, mais pas en Y.
            sommets.get(2).setX((epaisseur/2) + 0.2);                                   //Modifier SommetE (Bleu)
            .
            .
            .
    if (Orientation du Toit = "Est" ou Orientation du Toit = "Ouest")
        if (Mur.nom == "Façade")
            sommets.get(0).setX((epaisseur/2) + 0.2);                                   //Modifier X du SommetA (Rouge)
            sommets.get(1).setX(longueur-(epaisseur/2) - 0.2);                          //Modifier X du SommetB (Rouge)

            sommets.get(2).setX(epaisseur + 0.2);                                       //Modifier X du SommetG (Mauve)
            sommets.get(3).setX(longueur - epaisseur - 0.2);                            //Modifier X du SommetH (Mauve)

            sommets.add(new Sommet((epaisseur/2) + 0.2, (epaisseur/2) - 0.2);           //Ajout SommetC (Vert)
            sommets.add(new Sommet(longueur - (epaisseur/2) - 0.2, (epaisseur/2) - 0.2));//Ajout SommetD (Vert)

            sommets.add(new Sommet(epaisseur + 0.2, (epaisseur/2) - 0.2));              //Ajout SommetE (Jaune)
            sommets.add(new Sommet(longueur - epaisseur - 0.2, (epaisseur/2) - 0.2));   //Ajout SommetF (Jaune)

        if(Mur.nom == "Arrière") // il va avoir les mêmes points en X mais pas en Y.
        .
        .
        .
    .
    .
    .

    return Mur.sommets;
```


Pseudo-code qui illustre l'idée générale de la fonction de retrait des rainures

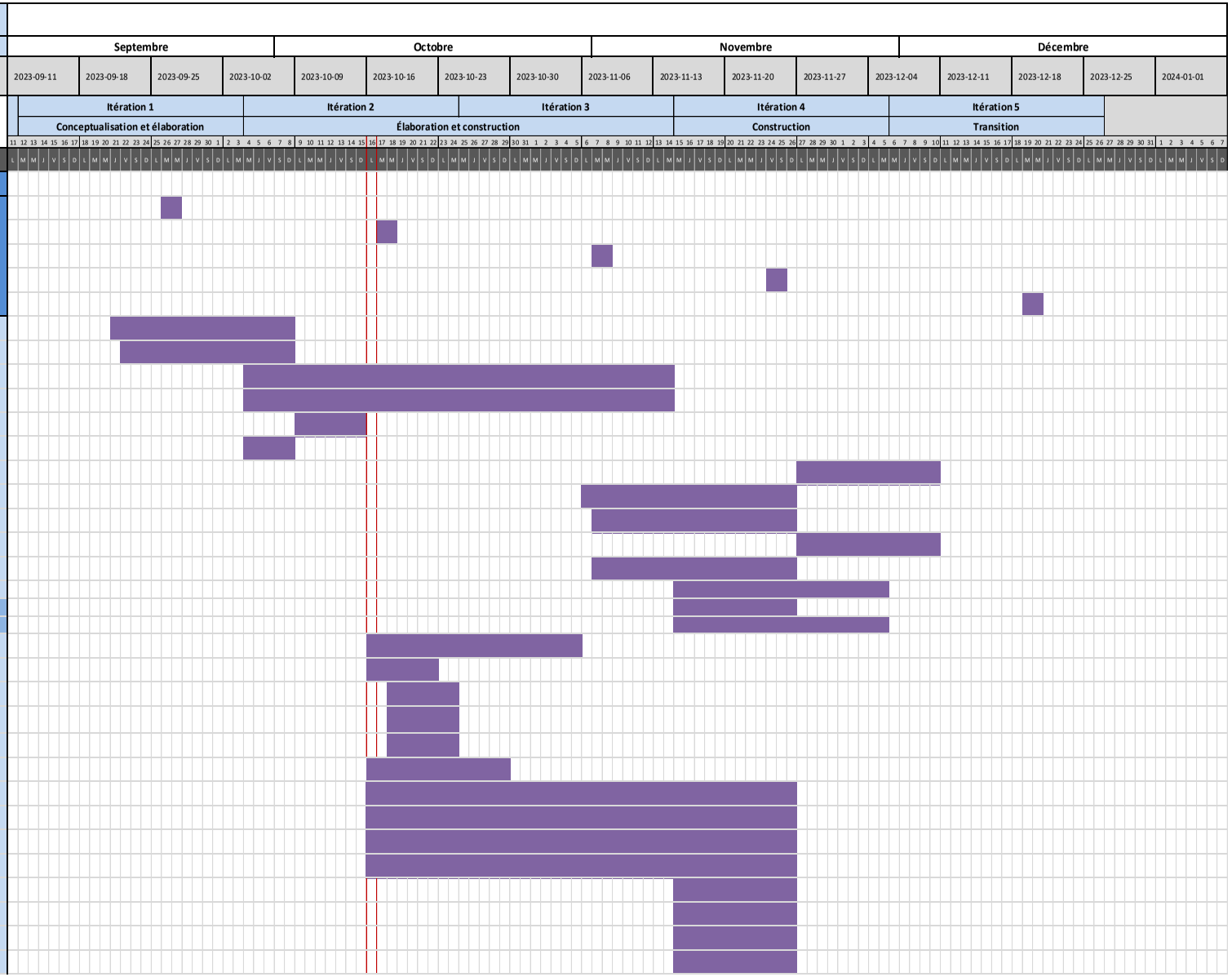
La liste de sommets retournée à la fin de la fonction **retirerRainuresMurs** permet de retourner la liste rafraîchie des sommets du Mur donné en paramètre de la fonction. L'autre paramètre, orientation du toit, quant à lui, sera trouvé dans la classe Chalet.

Plan de travail (Gantt)

ChalCLT

Projet de session GLO-2004 — 2023-09-13

Membres de l'équipe de projet : Sami Sabir, Souleymane Kane, Oumar Rafiou Barry , Ismaël Sdiri, Moulay-Mostafa Filali — 1



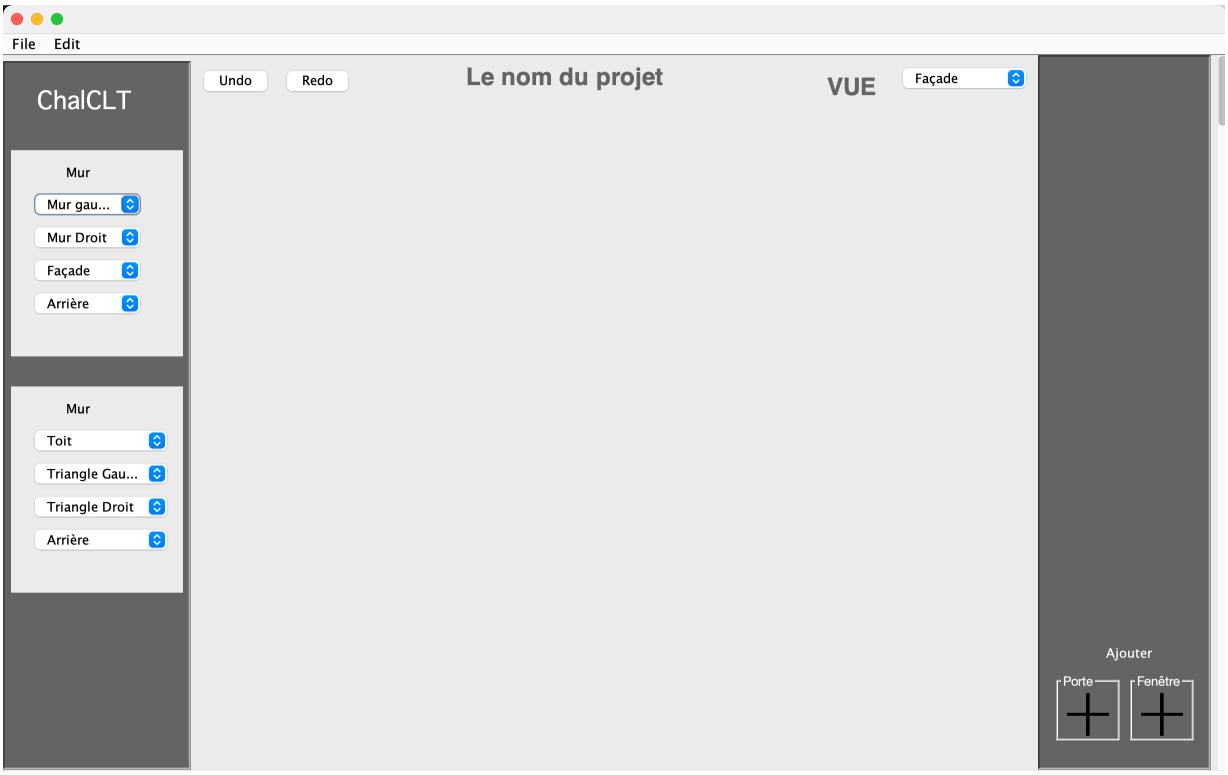
Tâches	Avancement	DÉBUT	FIN
Livrable du projet			
Remise livrable 1	0%	2023-09-26	2023-09-27
Remise livrable 2	0%	2023-10-17	2023-10-18
Remise livrable 3	0%	2023-11-07	2023-11-08
Remise livrable 4	0%	2023-11-24	2023-11-25
Remise livrable 5	0%	2023-12-19	2023-12-20
Ouvrir un projet	0%	2023-09-21	2023-10-08
Créer un projet	0%	2023-09-22	2023-10-08
Sauvegarder un projet	0%	2023-10-04	2023-11-14
Sauvegarder paramètres du chalet	0%	2023-10-04	2023-11-14
Proposer une sauvegarde	0%	2023-10-09	2023-10-15
Fermer un projet	0%	2023-10-04	2023-10-08
Exporter en STL les plans de coupe	0%	2023-11-27	2023-12-10
Ajouter des accessoires	0%	2023-11-06	2023-11-26
Déplacer des accessoires	0%	2023-11-07	2023-11-26
Redimensionner les murs	0%	2023-11-27	2023-12-10
Redimensionner les accessoires	0%	2023-11-07	2023-11-26
Sélectionner une vue	0%	2023-11-15	2023-12-05
Visualisation d'un chalet de dessus.	0%	2023-11-15	2023-11-26
Visualisation la vue des 4 murs.	0%	2023-11-15	2023-12-05
Modifier dimensions du chalet	0%	2023-10-16	2023-11-05
Modifier l'angle du toit	0%	2023-10-16	2023-10-22
Modifier l'épaisseur des murs	0%	2023-10-18	2023-10-24
Modifier l'épaisseur des matérieux utilisés	0%	2023-10-18	2023-10-24
Modifier la longueur du pli	0%	2023-10-18	2023-10-24
Modifier les propriétés d'un objet sélectioné dans le panneau	0%	2023-10-16	2023-10-29
Zoom in sur un objet	0%	2023-10-16	2023-11-26
Zoom out sur un objet	0%	2023-10-16	2023-11-26
Afficher/Cacher un objet	0%	2023-10-16	2023-11-26
Sélectionner un objet	0%	2023-10-16	2023-11-26
Copier un objet	0%	2023-11-15	2023-11-26
Supprimer un objet	0%	2023-11-15	2023-11-26
Valider les positions et afficher positions invalide	0%	2023-11-15	2023-11-26
Coller un objet	0%	2023-11-15	2023-11-26

Contribution des membres de l'équipe

Nom de l'individu	**Tâches**
Ismaël Sdiri	- Participation à la conception du diagramme de classes de conception - Texte explicatif - Conception de l'interface
Moulay-Mostafa Filali	- Participation à la conception du diagramme de classes de conception - DSC de l'afficheur de la vue de dessus - Texte explicatif - Diagramme de Gantt
Oumar Rafiou Barry	- Participation à la conception du diagramme de classes de conception - Architecture Logique - DSC du clic de souris
Souleymane Kane	- Participation à la conception du diagramme de classes de conception - Conception du DSC ajoutFenetre - Texte Explicatif
Sami Sabir	- Participation à la conception du diagramme de classes de conception - Conception du pseudo-code - Texte explicatif du pseudo-code

Version fonctionnelle du projet

Voici une capture d'écran du squelette de l'interface utilisateur de notre projet ChalCLT. Malgré le fait qu'elle reste simple, elle nous permet d'avoir une vue d'ensemble sur le placement des composantes et nous permet de commencer à penser à comment agencer les différentes fonctionnalités de l'application.



Capture d'écran du squelette de l'interface de ChalCLT conçue par le code