24/09/2019

# VirtuTuile

## Livrable 4

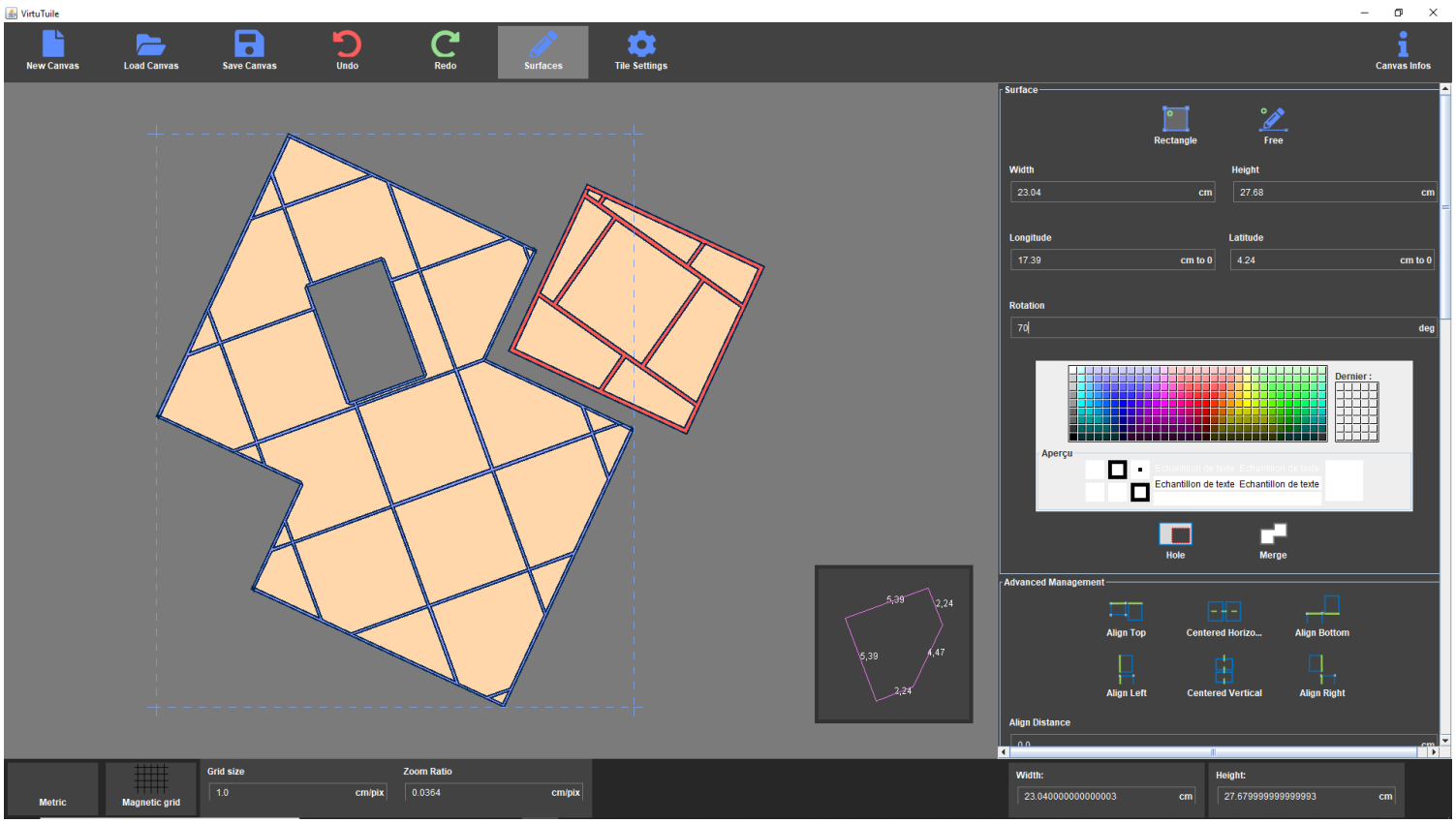
| Nom du rédacteur | Date | Changements |
|---|---|---|
| Martin Cotoni | 27 septembre 2019 | Création du document |
| Martin Cotoni<br>Thomas Lombard | 14 octobre 2019 | Refonte de / des / du:<br>- L'énoncé de vision<br>- Modèle du domaine<br>- Modèle des cas d'utilisation<br>Actualisation de / des / du :<br>- Diagramme de Gant<br>Ajout des demandes du Livrable 2 |
| Martin Cotoni | 24 novembre 2019 | Livrable 3 |
| Martin Cotoni | 16 décembre 2019 | Livrable 4 |

EQUIPE 30
M.FOURRIER – A.PELLETANT – T.LOMBARD – M.COTONI

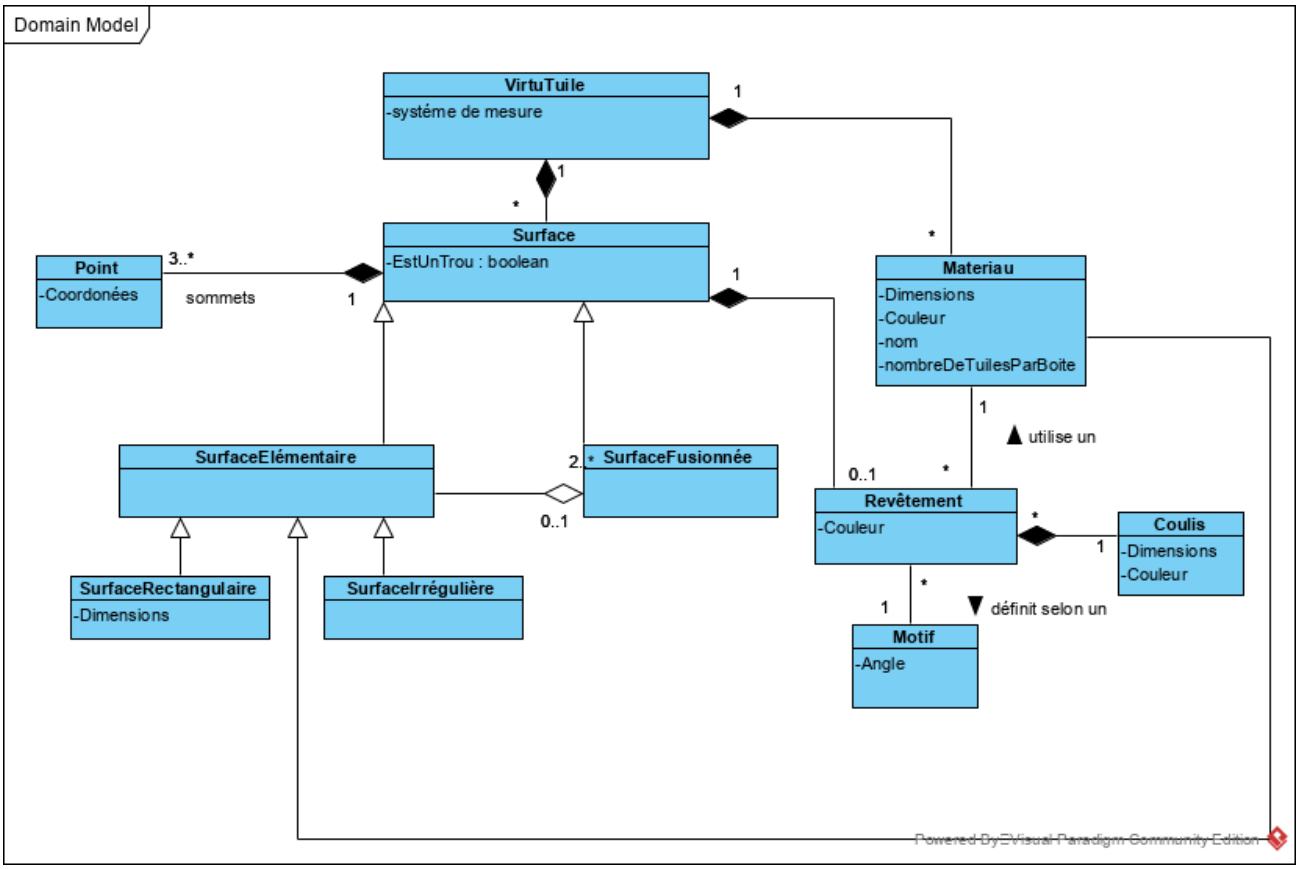
# Table des matières

# I. Enoncé de vision

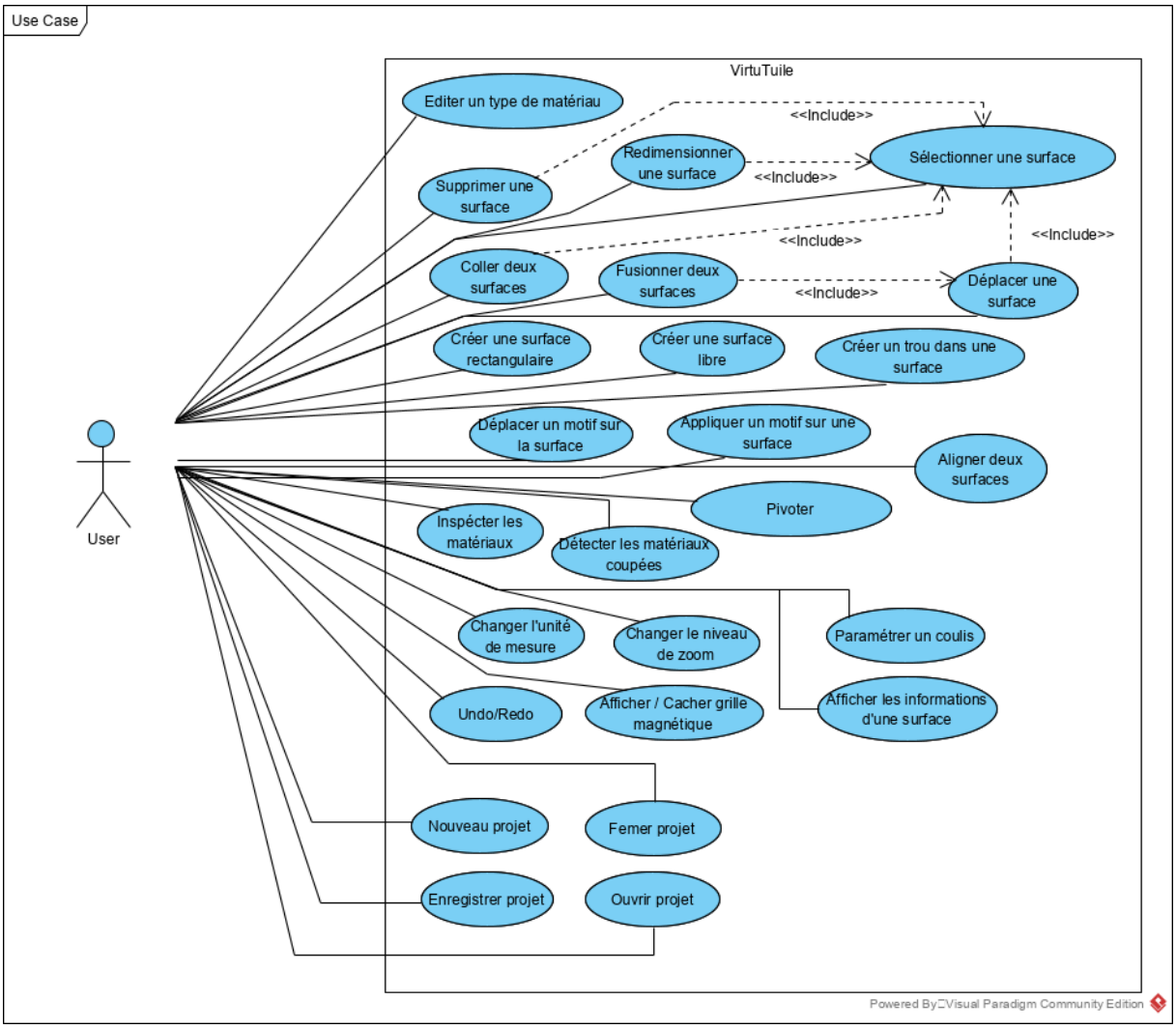
VirtuTuile a pour vocation de faciliter les travaux d'architecture d'une personne dans le monde du carrelage, qu'il soit un professionnel ou un particulier. Il est possible de reproduire en 2D, le plan d'une maison ou tout simplement une surface quelconque tel que par exemple, un mur. Une fois cette surface dessinée, il est possible d'éditer ses propriétés (comme sa couleur, ses dimensions ou encore sa position sur le plan de travail) et d'y associer un motif (motif configurable). Un large panel d'outil est disponible, afin de modeler et personnaliser les surfaces à volonté (fusion de deux surfaces, collage de deux surfaces, etc.).
Sur ces surfaces sont applicables des tuiles, disposer de manière à former un motif via une répétition paramétrable. Les tuiles qui composent le motif peuvent être celle que nous vous proposons par défaut ou encore celle que vous avez créé via le menu prévu à cet effet. Par ailleurs, de multiples motifs sont disponibles.
Au cours de votre création avec VirtuTuile, vous aurez accès via le menu « Canvas Infos » à toutes les informations relatives à votre projet. Ainsi, vous pourrez savoir combien de tuiles et de paquets sont utilisés ou encore combien de tuiles sont coupées.
Ce logiciel possède évidemment, des options de commodité tel que la possibilité de revenir en arrière sur vos actions ou encore la possibilité de sauvegarder votre travail et de le charger à volonté.



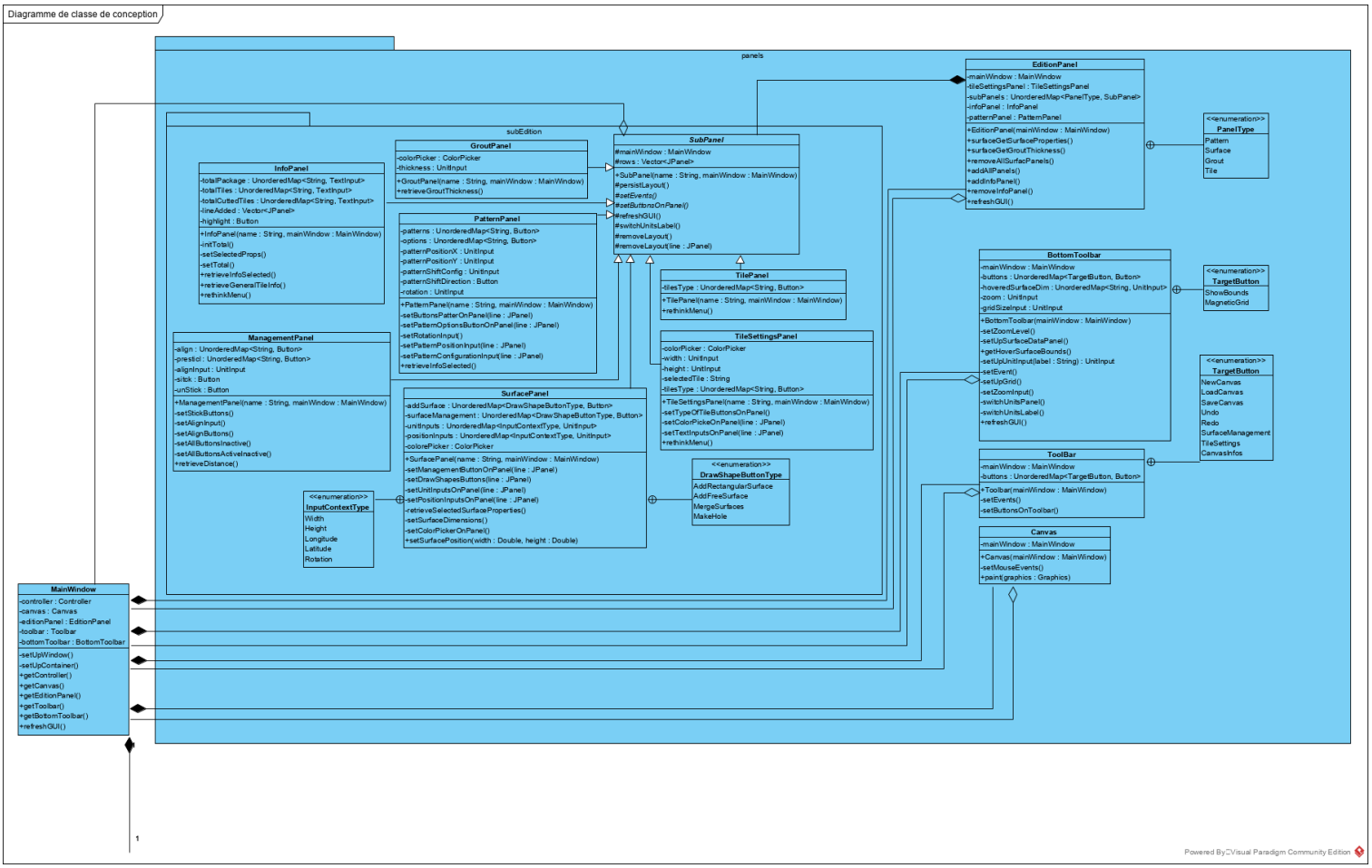
# II. Modèle du domaine



# III. Modèle des cas d'utilisation



# IV. Modèle de conception



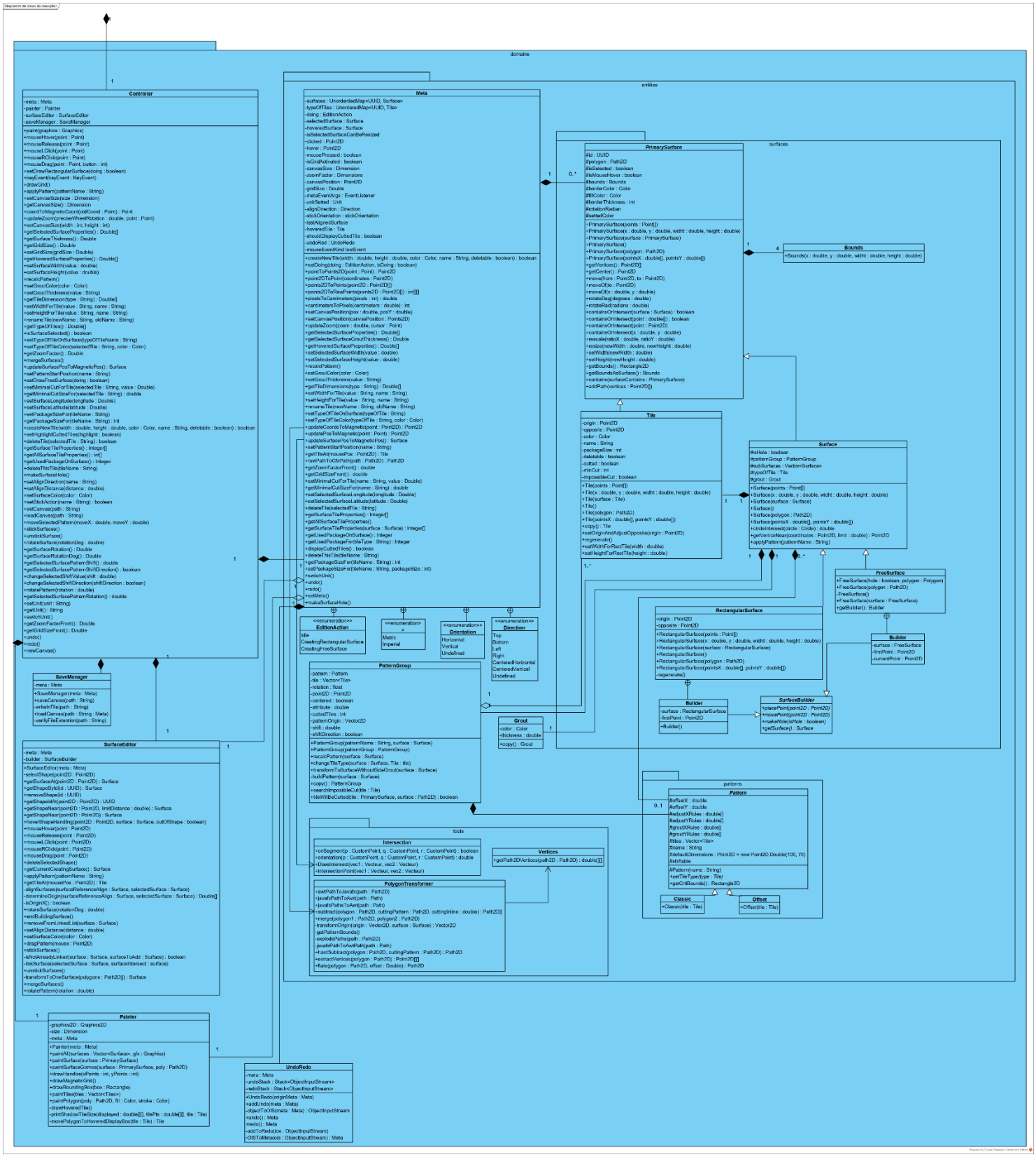


Diagramme de classe de conception
domaine
entities
surfaces
patterns
tools
Controller
-meta : Meta
-painter : Painter
-surfaceEditor : SurfaceEditor
-saveManager : SaveManager
+paint(graphics : Graphics)
+mouseHover(point : Point)
+mouseRelease(point : Point)
+mouseLClick(point : Point)
+mouseRClick(point : Point)
+mouseDrag(point : Point, button : int)
+setDrawRectangularSurface(doing : boolean)
+keyEvent(keyEvent : KeyEvent)
+drawGrid()
+applyPattern(patternName : String)
+setCanvasSize(size : Dimension)
+getCanvasSize() : Dimension
+coordToMagneticCoord(oldCoord : Point) : Point
+updateZoom(preciseWheelRotation : double, point : Point)
+setCanvasSize(width : int, height : int)
+getSelectedSurfaceProperties() : Double[]
+getSurfaceThickness() : Double
+getGridSize() : Double
+setGridSize(gridSize : Double)
+getHoveredSurfaceProperties() : Double[]
+setSurfaceWidth(value : double)
+setSurfaceHeight(value : double)
+recalcPattern()
+setGroutColor(color : Color)
+setGroutThickness(value : String)
+getTileDimension(type : String) : Double[]
+setWidthForTile(value : String, name : String)
+setHeightForTile(value : String, name : String)
+renameTile(newName : String, oldName : String)
+getTypeOfTiles() : Double[]
+isSurfaceSelected() : boolean
+setTypeOfTileOnSurface(typeOfTileName : String)
+setTypeOfTileColor(selectedTile : String, color : Color)
+getZoomFactor() : Double
+mergeSurfaces()
+updateSurfacePosToMagneticPos() : Surface
+setPatternStartPosition(name : String)
+setDrawFreeSurface(doing : boolean)
+setMinimalCutForTile(selectedTile : String, value : Double)
+getMinimalCutSizeFor(selectedTile : String) : double
+setSurfaceLongitude(longitude : Double)
+setSurfaceLatitude(latitude : Double)
+setPackageSizeFor(tileName : String)
+getPackageSizeFor(tileName : String) : int
+createNewTile(width : double, height : double, color : Color, name : String, deletable : boolean) : boolean
+setHighlightCuttedTiles(highlight : boolean)
+deleteTile(selectedTile : String) : boolean
+getSurfaceTileProperties() : Integer[]
+getAllSurfaceTileProperties() : int[]
+getUsedPackageOnSurface() : Integer
+deleteThisTile(tileName : String)
+makeSurface(-hole)
+setAlignDirection(name : String)
+setAlignDistance(distance : double)
+setSurfaceColor(color : Color)
+setStickAction(name : String) : boolean
+setCanvas(path : String)
+loadCanvas(path : String)
+moveSelectedPattern(moveX : double, moveY : double)
+stickSurfaces()
+unstickSurface()
+rotateSurface(rotationDeg : double)
+getSurfaceRotation() : Double
+getSurfaceRotationDeg() : Double
+getSelectedSurfacePatternShift() : double
+getSelectedSurfacePatternShiftDirection() : boolean
+changeSelectedShiftValue(shift : double)
+changeSelectedShiftDirection(shiftDirection : boolean)
+rotatePattern(rotation : double)
+getSelectedSurfacePatternRotation() : double
+setUnit(unit : String)
+getUnit() : String
+switchUnit()
+getZoomFactorFront() : Double
+getGridSizeFront() : Double
+undo()
+redo()
+newCanvas()
Meta
-surfaces : UnorderedMap<UUID, Surface>
-typeOfTiles : UnorderedMap<UUID, Tile>
-doing : EditionAction
-selectedSurface : Surface
-hoveredSurface : Surface
-isSelectedSurfaceCanBeResized
-clicked : Point2D
-hover : Point2D
-mousePressed : boolean
-isGridActivated : boolean
-canvasSize : Dimension
-zoomFactor : Dimensions
-canvasPosition : Point2D
-gridSize : Double
-metaEventArgs : EventListener
-unitSetted : Unit
-alignDirection : Direction
-stickOrientation : stickOrientation
-lastAlignedSurface
-hoveredTile : Tile
-shouldDisplayCuttedTile : boolean
-undoRed : UndoRedo
-mouseEventKind lastEvent
+createNewTile(width : double, height : double, color : Color, name : String, deletable : boolean) : boolean
+setDoing(doing : EditionAction, isDoing : boolean)
+pointToPoints2D(point : Point) : Point2D
+point2DToPoint(coordinates : Point2D)
+points2DToPoints(point2D : Point2D[])
+points2DToRawPoints(points2D : Point2D[]) : int[][]
+pixelsToCentimeters(pixels :int) : double
+centimetersToPixels(centimeters : double) : int
+setCanvasPosition(posx : double, posY : double)
+setCanvasPosition(canvasPosition : Point2D)
+updateZoom(zoom : double, cursor : Point)
+getSelectedSurfaceProperties() : Double[]
+getSelectedSurfaceGroutThickness() : Double
+getHoveredSurfaceProperties() : Double[]
+setSelectedSurfaceWidth(value : double)
+setSelectedSurfaceHeight(value : double)
+recalcPattern()
+setGroutColor(color : Color)
+setGroutThickness(value : String)
+getTileDimensions(type : String) : Double[]
+setWidthForTile(value : String, name : String)
+setHeightForTile(value : String, name : String)
+renameTile(newName : String, oldName : String)
+setTypeOfTileOnSurface(typeOfTile : String)
+setTypeOfTileColor(typeOfTile : String, color : Color)
+updateCoordsToMagnetic(point : Point2D) : Point2D
+updatePosToMagnetic(point : Point) : Point2D
+updateSurfacePosToMagneticPos() : Surface
+setPatternStartPosition(name : String)
+getTileAtMousePos : Point2D) : Tile
+newPathToGh?ath(path : Path2D) : Path2D
+getZoomFactorFront() : double
+getGridSizeFront() : double
+setMinimalCutForTile(name : String, value : Double)
+getMinimalCutSizeFor(name : String) : double
+setSelectedSurfaceLongitude(longitude : Double)
+setSelectedSurfaceLatitude(latitude : Double)
+deleteTile(selectedTile : String)
+getSurfaceTileProperties() : Integer[]
+getAllSurfaceTileProperties()
+getSurfaceTileProperties(surface : Surface) : Integer[]
+getUsedPackageOnSurface() : Integer
+getUsedPackageForTileType : String) : Integer
+displayCuttedTiles() : boolean
+deleteThisTile(tileName : String)
+setPackageSizeFor(tileName : String) : int
+setPackageSizeFor(tileName : String, packageSize : int)
+switchUnit()
+undo()
+redo()
+setMeta()
+makeSurfaceHole()
PrimarySurface
#id : UUID
#polygon : Path2D
#isSelected : boolean
#isMouseHover : boolean
#bounds : Bounds
#borderColor : Color
#fillColor : Color
#borderThickness : int
#rotationRadian
#setedColor
+PrimarySurface(points : Point[])
+PrimarySurface(x : double, y : double, width : double, height : double)
+PrimarySurface(surface : PrimarySurface)
+PrimarySurface()
+PrimarySurface(polygon : Path2D)
+PrimarySurface(pointsX : double[], pointsY : double[])
+getVertices() : Point2D[]
+getCenter() : Point2D
+moveFrom : Point2D, to : Point2D)
+moveOfIx : Point2D)
+moveOf(x : double, y : double)
+rotateDeg(degrees : double)
+rotateRad(radians : double)
+containsOrIntersect(surface : Surface) : boolean
+containsOrIntersect(point : double[]) : boolean
+containsOrIntersect(point : Point2D)
+containsOrIntersect(x : double, y : double)
+rescale(ratioX : double, ratioY : double)
+resize(newWidth : double, newHeight : double)
+setWidth(newWidth : double)
+setHeight(newHeight : double)
+getBounds() : Rectangle2D
+getBoundsAsSurface() : Bounds
+contains(surfaceContains : PrimarySurface)
+addPath(vertices : Point2D[])
Bounds
+Bounds(x : double, y : double, width : double, height : double)
Tile
-origin : Point2D
-opposite : Point2D
-color : Color
-name : String
-packageSize : int
-deletable : boolean
-cutted : boolean
-minCut : int
-impossibleCut : boolean
+Tile(points : Point[])
+Tile(x : double, y : double, width : double, height : double)
+Tile(surface : Tile)
+Tile()
+Tile(polygon : Path2D)
+Tile(pointsX : double[], pointsY : double[])
+copy() : Tile
+setOriginAndAdjustOpposite(origin : Point2D)
+regenerate()
+setWidthForRectTile(width : double)
+setHeightForRectTile(height : double)
Surface
#isHole : boolean
#patternGroup : PatternGroup
#subSurfaces : Vector<Surface>
#typeOfTile : Tile
#grout : Grout
+Surface(points : Point[])
+Surface(x : double, y : double, width : double, height : double)
+Surface(surface : Surface)
+Surface()
+Surface(polygon : Path2D)
+Surface(pointsX : double[], pointsY : double[])
+circleIntersect(circle : Circle) : double
+getVerticesNear(coordinates : Point2D, limit : double) : Point2D
+applyPattern(patternName : String)
FreeSurface
+FreeSurface(hole : boolean, polygon : Polygon)
+FreeSurface(polygon : Path2D)
-FreeSurface()
+FreeSurface(surface : FreeSurface)
+getBuilder() : Builder
Builder
-surface : FreeSurface
-firstPoint : Point2D
-currentPoint : Point2D
RectangularSurface
-origin : Point2D
-opposite : Point2D
+RectangularSurface(points : Point[])
+RectangularSurface(x : double, y : double, width : double, height : double)
+RectangularSurface(surface : RectangularSurface)
+RectangularSurface()
+RectangularSurface(polygon : Path2D)
+RectangularSurface(pointsX : double[], pointsY : double[])
-regenerate()
Builder
-surface : RectangularSurface
-firstPoint : Point2D
+Builder()
SurfaceBuilder
+pieceFrom(point2D : Point2D)
+moveFrom(point2D : Point2D)
+makeHole(aHole : boolean)
+getSurface() : Surface
<<enumeration>>
EditionAction
Idle
CreatingRectangularSurface
CreatingFreeSurface
<<enumeration>>
Metric
Imperial
<<enumeration>>
Orientation
Horizontal
Vertical
Undefined
<<enumeration>>
Direction
Top
Bottom
Left
Right
CenteredHorizontal
CenteredVertical
Undefined
PatternGroup
-pattern : Pattern
-tile : Vector<Tile>
-rotation : float
-point2D : Point2D
-centered : boolean
-shShift : double
-cuttedTiles : int
-patternOrigin : Vector2D
-shift : double
-shiftDirection : boolean
+PatternGroup(patternName : String, surface : Surface)
+PatternGroup(patternGroup : PatternGroup)
+recalcPattern(surface : Surface)
+changeTileType(surface : Surface, Tile : tile)
+transformToSurfaceWithoutSideGrout(surface : Surface)
-buildPattern(surface : Surface)
+copy() : PatternGroup
+searchImpossibleCut(tile : Tile)
+isItWillBeCutted(tile : PrimarySurface, surface : Path2D) : boolean
Grout
-color : Color
-thickness : double
+copy() : Grout
SaveManager
-meta : Meta
+SaveManager(meta : Meta)
+saveCanvas(path : String)
-writeInFile(path : String)
+loadCanvas(path : String : Meta)
-verifyFileExtention(path : String)
SurfaceEditor
-meta : Meta
-builder : SurfaceBuilder
+SurfaceEditor(meta : Meta)
+selectShape(point2D : Point2D)
+getSurfaceAt(point2D : Point2D) : Surface
+getShapeById(id : UUID) : Surface
+removeShapeId : UUID)
+getShapeIdAt(point2D : Point2D) : UUID
+getShapeNear(point2D : Point2D, limitDistance : double) : Surface
+getShapeNear(point2D : Point2D) : Surface
+hoverShapeHandling(point2D : Point2D, surface : Surface, outOfShape : boolean)
+mouseHover(point : Point2D)
+mouseRelease(point : Point2D)
+mouseLClick(point : Point2D)
+mouseRClick(point : Point2D)
+mouseDrag(point : Point2D)
-deleteSelectedShape()
+getCurrentCreatingSurface() : Surface
+applyPattern(patternName : String)
+getTileAtmousePos : Point2D) : Tile
-alignSurfaces(surfaceReferenceAlign : Surface, selectedSurface : Surface)
-determineOrgin(surfaceReferenceAlign : Surface, selectedSurface : Surface) : Double[]
-isOriginX() : boolean
+rotateSurface(rotationDeg : double)
+endBuildingSurface()
+removeFromLinkedList(surface : Surface)
+setAlignDistance(distance : double)
+setSurfaceColor(color : Color)
+dragPattern(mouse : Point2D)
+stickSurfaces()
-isNotAlreadyLinked(surface : Surface, surfaceToAdd : Surface) : boolean
-linkSurface(selectedSurface : Surface, surfaceIntersect : surface)
+unstickSurfaces()
-transformToOneSurface(polygons : Path2D[]) : Surface
+mergeSurfaces()
+rotatePattern(rotation : double)
Painter
-graphics2D : Graphics2D
-size : Dimension
-meta : Meta
+Painter(meta : Meta)
+paintAll(surfaces : Vector<Surface>, gfx : Graphics)
+paintSurface(surface : PrimarySurface)
+paintSurfaceGizmos(surface : PrimarySurface, poly : Path2D)
+drawHandles(xPoints : int, yPoints : int)
+drawMagneticGrid()
+drawBoundingBox(box : Rectangle)
+paintTiles(tiles : Vector<Tile>)
+paintPolygon(poly : Path2D, fill : Color, stroke : Color)
-drawHoveredTile()
-printShadowTileSize(displayed : double[][], tilePts : double[][], tile : Tile)
-movePolygonToHoveredDisplayBox(tile : Tile) : Tile
UndoRedo
-meta : Meta
-undoStack : Stack<ObjectInputStream>
-redoStack : Stack<ObjectInputStream>
+UndoRedo(originMeta : Meta)
+addUndo(meta : Meta)
-objectToOS(meta : Meta) : ObjectInputStream
+undo() : Meta
+redo() : Meta
-addToRedo(ios : ObjectInputStream)
-OISToMetaIois : ObjectInputStream) : Meta
Pattern
#offsetX : double
#offsetY : double
#adjustXRules : double[]
#adjustYRules : double[]
#groutXRules : double[]
#groutYRules : double[]
#tiles : Vector<Tile>
#name : String
#defaultDimensions : Point2D = new Point2D.Double(100, 70)
#shiftable
#Pattern(name : String)
+setTileType(type : Tile)
+getCritBounds() : Rectangle2D
Classic
+Classic(tile : Tile)
Offset
+Offset(tile : Tile)
Intersection
+orSegmentIp : CustomPoint, q : CustomPoint, r : CustomPoint) : boolean
+orientation(p : CustomPoint, q : CustomPoint, r : CustomPoint) : double
+DoesIntersect(vec1 : Vecteur, vec2 : Vecteur)
+intersectionPoint(vec1 : Vecteur, vec2 : Vecteur)
Vertices
+getPath2DVertices(path2D : Path2D) : double[][]
PolygonTransformer
+awtPathToJavafx(path : Path2D)
+javafxPathToAwt(path : Path)
+javafxPathToAwtPath(path : Path)
+subtract(polygon : Path2D, cuttingPattern : Path2D, cuttingInline : double) : Path2D[]
+merge(polygon1 : Path2D, polygon2 : Path2D)
+transformOrigin(origin : Vector2D, surface : Surface) : Vector2D
-getPatternBounds()
-explodePaths(path : Path2D)
-javafxPathToAwtPath(path : Path)
+hardSubtract(polygon : Path2D, cuttingPattern : Path2D) : Path2D
+extractVertices(polygon : Path2D) : Point2D[][]
+flate(polygon : Path2D, offset : Double) : Path2D

# I. Afficheur

**MainWindow** : hérite de **JFrame**. Le rôle de cette classe est de gérer tout ce qui est relatif à la gestion de la fenêtre : que faire lorsqu'on clique sur la croix, ou la gestion de son contenu. Elle possède **Controller** et également tous les éléments de la fenêtre tel que **Canvas**, **EditionPanel**, **Toolbar** & **BottomToolbar**.
Cette classe, via des agrégations est possédés par **Canvas**, **EditionPanel**, **Toolbar** & **BottomToolbar**, afin que tout élément graphique puisse interagir entres elles et avec **Controller**. (Tout cela via des getters).

## 1. Panels

**Canvas** : Hérite de **BorderedEventPanel** (classe non présente dans le diagramme, surcharge d'un **JPanel** pour gérer des évenements). Cette classe contient des « Listeners » d'événements qui interagiront en fonction avec **Controller** via un getter de **MainWindow**.

**EditionPanel** : Hérite de **BorderedPanel** (classe non présente dans le diagramme, surcharge d'un **JPanel**). Cette classe contient et gère toutes les palettes d'outils contenues dans l'éditeur comme l'édition de formes, les paramètres du canvas, la gestion du coulis entre autres. Elle contient des **SubPanels**.

**Toolbar** : Hérite de **BorderedPanel** (classe non présente dans le diagramme, surcharge d'un **JPanel**). Cette classe gère et contient tous les boutons principaux. Créer un nouveau Canvas, Sauvegarder, Editer le contenu du Canvas, …

**BottomToolbar** : Hérite de **BorderedPanel** (classe non présente dans le diagramme, surcharge d'un **JPanel**). Cette classe gère et affiche des informations comme le système métrique choisi, le zoom …

## 2. SubEdition

**SubPanel** : Hérite de **PanelEvents** (classe non présente dans le diagramme, surcharge d'un **JPanel** pour gérer des évenements). Il s'agit d'une classe abstraite, utilisé par **GroutPanel**, **PatternPanel**, **SurfacePanel**, **TilePanel**, **TileSettingsPanel**. Elle contient, grâce a une référence, **MainWindow**

**GroutPanel** : Hérite de **SubPanel**. Cette classe gère les paramètres d'un coulis (couleur, taille) pour une surface sélectionnée.

**PatternPanel** : Hérite de **SubPanel**. Cette classe permet de choisir, pour une surface sélectionnée, quel type de motif doit être appliqué.

**SurfacePanel** : Hérite de **SubPanel**. Cette classe gère la création de **Surface**s ainsi que son édition (hauteur, largeur).

**TilePanel** : Hérite de **SubPanel**. Cette permet de choisir quel type de tuile appliqué pour une surface sélectionnée.

**TileSettingsPanel** : Hérite de **SubPanel**. Cette classe permet de configurer les différents types de tuile.

## II. Domaine

**Controller** : Cette classe représente l'unique point d'entrée de la couche Domaine. Elle contient toutes les méthodes qui peuvent être appelé par l'Afficheur ou Swing. (Controller de Larman)
Elle possède trois attributs, **Meta**, **SurfaceEditor** et **Painter**.

**Painter** : Cette classe, contenu dans **Controller**, permet d'afficher les éléments qui ont été créé par le programme et l'utilisateur (surfaces, motifs, etc.). Elle contient **Meta** en tant qu'attribut (référence).

**SurfaceEditor :** Cette classe contient toute la méthode relative à la création d'une surface, ainsi que les méthodes qui vont être appelé par **Canvas** depuis **Controller**.

### 1. Entities

**Meta :** Cette classe va contenir toutes les informations relatives à la session de l'utilisateur (liste de surfaces, état de ce qu'il est en train de faire, surface survolée avec la souris, etc.). Elle contient également les méthodes qui vont être appelés par les menus d'éditions, par le biais de **Controller** (applyPattern, setHoveredSurface, etc.)
Par défaut, **Meta** instancie 3 types de Tuile

**Grout :** Cette classe, possédée par **Surface**, représente l'espacement, le coulis qui va être présent entre deux tuiles. Elle possède une couleur ainsi qu'une épaisseur.

**PatternGroup** : Cette classe, possédée par **Surface**, représente un groupe de **Pattern** dans une surface. Une de ses méthodes permet de construire le pattern sur une surface donnée.

#### *1.1. Pattern*

**Pattern :** Cette classe abstraite représente un motif.

**Classic** : Cette classe hérite de **Pattern.** Elle instancie simplement une tuile.

**Offset** : Cette classe hérite de **Pattern.** Elle instancie deux tuiles, de manière à avoir une tuile décalée par rapport à l'autre, sur un axe horizontal.

### *1.2. Surfaces*

**PrimarySurface :** Cette classe abstraite représente une surface primaire, dont **Bounds**, **FreeSurface, Rectangular Surface, Surface** et **Tile** vont hérités. Elle définit toutes les méthodes communes à une surface de base.

**Bounds :** Cette classe représente les limites d'une surface régulière et irrégulière, si cette surface était rectangulaire. Cette classe est contenu dans **Surface**.

**FreeSurface :** Cette classe hérite de **Surface** et implémente toutes les méthodes spécifiques à une surface irrégulière.

**RectangularSurface :** Cette classe hérité de **Surface.** Elle implémente toutes les méthodes spécifiques à une surface régulière. Elle contient également **Builder**, une classe statique qui permet de créer une surface de rectangulaire.

**Surface** : Cette classe hérite de **PrimarySurface**. Elle implémente toutes les méthodes et caractéristique d'une surface « Avancée », comme le fait d'être ou non, un trou. Elle possède également une liste de **Surface**, pouvant être une autre surface (comme un trou).

**SurfaceBuilder** : Cette classe abstraite permet de définir ce que les différents builders (pour le moment, il n'y a que **Builder**) doivent implémenter.

**Tile** : Cette classe, qui hérite directement de **PrimarySurface**, représente une tuile qui va être appliqué sur une surface via **PatternGrout**. Il s'agit plus exactement d'un « type » de tuile.

### *1.3. Tools*

**Intersection** : Cette classe permet grâce à des méthodes statiques, de connaitre le point d'intersection entre deux segments.

**PolygonTransorfmer** : Cette classe permet grâce à des méthodes statiques, d'exploiter des fonctionnalités mathématiques sur des Polygon.

**Vertices** : Cette classe contient une méthode statique permettant de récupérer les sommets d'un Path2D.

# V. Points fort & points faibles

## Points forts

Selon nous, notre projet se distingue par son côté « user friendly » avec son design directement inspiré de Material UI. Il n'est pas compliqué d'interagir avec
Le contrôleur de Larman quant à lui, permet aisément moyennant certaines modifications de remplacé l'interface utilisateur.

## Points faibles

- Optimisation : lors de l'utilisation de l'application, il n'est pas rare que de nombreuses entités soit recalculées ou redessinées inutilement. Les deux exemples les plus fréquents sont respectivement le recalcule des motifs lors d'actions qui ne le demande pas forcément ou encore le redessinement des lignes de la grille lors de chaque mouvement de souris.
- Cohésion : Certaines méthodes ne sont tout simplement pas bien placées. Des méthodes relatives à l'édition d'une surface peuvent être actuellement trouvée dans « Meta », classe destinée à être une sorte de « relais » intelligent du Contrôleur (elle représente à proprement parlé, le projet).
- Difficulté sur le zoom et la grille magnétique : pour une raison étrange, le zoom à un certain moment, agit avec un mouvement parabolique. Ce mouvement est à éviter.

## Améliorations

Afin de pouvoir utiliser notre logiciel, les points à corriger se dirigent évidemment vers les points faibles de l'application.
En plus de cela, il serait pratique d'avoir des fonctionnalités très avancées, tel que peut le proposer AutoCAD.
Nous pouvons donc imaginer, une multi-sélection, une liste de surface présente dans un menu afin de pouvoir éditer des surfaces qui ne sont pas présente dans la vue direct du plan de travail ou encore des raccourcis clavier pour chacune des actions pouvant être effectuer. Nous pourrions avoir des informations sur le type de mur, sur les côtes des différentes surfaces et formes et de manière générale avoir plus de retour visuel pour l'utilisateur.

# VI. Contribution des membres

Le travail pour ce dernier Livrable à était équitablement réparti et tout les membres se sont intéressés aux parties de chacun.

Thomas Lombard a réalisé :

- Création de Patterns (L)
- Décalage des tuiles entre les rangée paire et impaire
- Highlight des coupes impossibles
- Coulis en périphérie des motifs
- Créer une forme libre
- Montrer les tuiles coupées

Antoine Pelletant a réalisé :

- Undo
- Redo
- Sauvegarder projet
- Charger projet
- Nouveau projet
- Montrer les informations d'une tuile au survol

Martin Cotoni a réalisé :

- Alignement de deux surfaces
- Créer un trou
- Spécifier l'emplacement d'une surface via un input
- Informations générales sur le projet (nombre de tuiles etc.)
- Créer des tuiles
- Coller des surfaces
- Rotation d'une surface
- Rotation d'un motif
- Métrique et Impérial (décimal)